OpenTREP

0.6.0

Generated by Doxygen 1.8.6

Mon Feb 3 2014 14:17:53

Contents

1 OpenTrep Documentation 1

1.1 Getting Started . . . 1
1.2 OpenTrep at SourceForge . . . 1
1.3 OpenTrep Development . . . 1
1.4 External Libraries . . . 1
1.5 Support OpenTrep . . . 2
1.6 About OpenTrep . . . 2

2 OpenTrep People 2

2.1 Project Admins . . . 2
2.2 Developers . . . 2
2.3 Retired Developers . . . 2
2.4 Contributors . . . 2
2.5 Distribution Maintainers . . . 3

3 Coding Rules 3

3.1 Default Naming Rules for Variables . . . 3
3.2 Default Naming Rules for Functions . . . 3
3.3 Default Naming Rules for Classes and Structures . . . 3
3.4 Default Naming Rules for Files . . . 3
3.5 Default Functionality of Classes . . . 3

4 Copyright and License 4

4.1 GNU LESSER GENERAL PUBLIC LICENSE . . . 4
4.1.1 Version 2.1, February 1999 . . . 4
4.2 Preamble . . . 4
4.3 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION . . . 5
4.3.1 NO WARRANTY . . . 9
4.3.2 END OF TERMS AND CONDITIONS . . . 9
4.4 How to Apply These Terms to Your New Programs . . . 9

5 Documentation Rules 10

5.1 General Rules . . . 10
5.2 File Header . . . 11
5.3 Grouping Various Parts . . . 11

6 Main features 11

6.1 Standard Airline IT Business Object Model (BOM) . . . 12
6.2 Architecture of the OpenTrep library . . . 12

7 Make a Difference 12

8 Make a new release 13

8.1 Introduction . . . 13
8.2 Initialisation . . . 13
8.3 Branch creation . . . 13
8.4 Commit and publish the release branch . . . 13
8.5 Update the change-log in the trunk as well . . . 13
8.6 Create distribution packages . . . 14
8.7 Generation the RPM packages . . . 14
8.8 Update distributed change log . . . 14
8.9 Create the binary package, including the documentation . . . 14
8.10 Upload the files to SourceForge . . . 14
8.11 Upload the documentation to SourceForge . . . 15
8.12 Make a new post . . . 15
8.13 Send an email on the announcement mailing-list . . . 15

9 Installation 15

9.1 Table of Contents . . . 15
9.2 Fedora/RedHat Linux distributions . . . 16
9.3 OpenTrep Requirements . . . 16
9.4 Basic Installation . . . 16
9.5 Compilers and Options . . . 17
9.6 Compiling For Multiple Architectures . . . 17
9.7 Installation Names . . . 18
9.8 Optional Features . . . 19
9.9 Particular systems . . . 19
9.10 Specifying the System Type . . . 20
9.11 Sharing Defaults . . . 20
9.12 Defining Variables . . . 20
9.13 ‘cmake’ Invocation . . . 20

10 Linking with OpenTrep 24

10.1 Table of Contents . . . 24
10.2 Introdution . . . 24
10.3 Using the pkg-config command . . . 24
10.4 Using the opentrep-config script . . . 25
10.5 M4 macro for the GNU Autotools . . . 25
10.6 Using OpenTrep with dynamic linking . . . 25

11 Test Rules 25

11.1 The Test Source Files . . . 25
11.2 The Reference File . . . 26

11.3 Testing OpenTrep Library . . . 26

12 Users Guide **26**

12.1 Table of Contents . . . 26
12.2 Introduction . . . 26
12.3 Get Started . . . 26
12.3.1 Get the OpenTrep library . . . 26
12.3.2 Build the OpenTrep project . . . 27
12.3.3 Build and Run the Tests . . . 27
12.3.4 Install the OpenTrep Project (Binaries, Documentation) . . . 27
12.4 Exploring the Predefined BOM Tree . . . 27
12.4.1 Airline Distribution BOM Tree . . . 27
12.4.2 Airline Network BOM Tree . . . 27
12.4.3 Airline Inventory BOM Tree . . . 27
12.5 Extending the BOM Tree . . . 27

13 Supported Systems **27**

13.1 Table of Contents . . . 27
13.2 Introdution . . . 28
13.3 OpenTrep 3.10.x . . . 28
13.3.1 Linux Systems . . . 28
13.3.2 Windows Systems . . . 32
13.3.3 Unix Systems . . . 34

14 OpenTrep Supported Systems (Previous Releases) **34**

14.1 OpenTrep 3.9.1 . . . 34
14.2 OpenTrep 3.9.0 . . . 34
14.3 OpenTrep 3.8.1 . . . 34

15 Tutorials **35**

15.1 Table of Contents . . . 35
15.2 Introduction . . . 35
15.2.1 Preparing the StdAir Project for Development . . . 35
15.3 Build a Predefined BOM Tree . . . 35
15.3.1 Instanciate the BOM Root Object . . . 35
15.3.2 Instanciate the (Airline) Inventory Object . . . 36
15.3.3 Link the Inventory Object with the BOM Root . . . 36
15.3.4 Build Another Airline Inventory . . . 36
15.3.5 Dump The BOM Tree Content . . . 36
15.3.6 Result of the Tutorial Program . . . 36
15.4 Extend the Pre-Defined BOM Tree . . . 36

15.4.1 Extend an Airline Inventory Object . . . 37
15.4.2 Build the Specific BOM Objects . . . 37
15.4.3 Result of the Tutorial Program . . . 37

16 Command-Line Test to Demonstrate How To Test the OpenTREP Project 38

17 Command-Line Test to Demonstrate How To Test the OpenTREP Project 39

18 Namespace Index 40
18.1 Namespace List . . . 40

19 Hierarchical Index 40
19.1 Class Hierarchy . . . 40

20 Class Index 50
20.1 Class List . . . 50

21 File Index 57
21.1 File List . . . 57

22 Namespace Documentation 61
22.1 OPENTREP Namespace Reference . . . 61
22.1.1 Typedef Documentation . . . 69
22.1.2 Function Documentation . . . 75
22.1.3 Variable Documentation . . . 81
22.2 OPENTREP::LOG Namespace Reference . . . 84
22.2.1 Detailed Description . . . 85
22.2.2 Enumeration Type Documentation . . . 85
22.3 OPENTREP::PorParserHelper Namespace Reference . . . 85
22.3.1 Detailed Description . . . 86
22.3.2 Typedef Documentation . . . 87
22.3.3 Variable Documentation . . . 87
22.4 soci Namespace Reference . . . 88
22.5 treppb Namespace Reference . . . 88
22.6 Xapian Namespace Reference . . . 88

23 Class Documentation 88
23.1 OPENTREP::Admin1ASCIIName_T Struct Reference . . . 88
23.1.1 Detailed Description . . . 89
23.1.2 Constructor & Destructor Documentation . . . 89
23.2 OPENTREP::Admin1Code_T Struct Reference . . . 89
23.2.1 Detailed Description . . . 89
23.2.2 Constructor & Destructor Documentation . . . 90

23.3 OPENTREP::Admin1UTFName_T Struct Reference . . . 90
23.3.1 Detailed Description . . . 90
23.3.2 Constructor & Destructor Documentation . . . 90
23.4 OPENTREP::Admin2ASCIIName_T Struct Reference . . . 90
23.4.1 Detailed Description . . . 91
23.4.2 Constructor & Destructor Documentation . . . 91
23.5 OPENTREP::Admin2Code_T Struct Reference . . . 91
23.5.1 Detailed Description . . . 91
23.5.2 Constructor & Destructor Documentation . . . 92
23.6 OPENTREP::Admin2UTFName_T Struct Reference . . . 92
23.6.1 Detailed Description . . . 92
23.6.2 Constructor & Destructor Documentation . . . 92
23.7 OPENTREP::Admin3Code_T Struct Reference . . . 92
23.7.1 Detailed Description . . . 93
23.7.2 Constructor & Destructor Documentation . . . 93
23.8 OPENTREP::Admin4Code_T Struct Reference . . . 93
23.8.1 Detailed Description . . . 93
23.8.2 Constructor & Destructor Documentation . . . 94
23.9 OPENTREP::AltCountryCode_T Struct Reference . . . 94
23.9.1 Detailed Description . . . 94
23.9.2 Constructor & Destructor Documentation . . . 94
23.10 OPENTREP::AltNameShortListString_T Struct Reference . . . 94
23.10.1 Detailed Description . . . 95
23.10.2 Constructor & Destructor Documentation . . . 95
23.11 OPENTREP::ASCIIName_T Struct Reference . . . 95
23.11.1 Detailed Description . . . 96
23.11.2 Constructor & Destructor Documentation . . . 96
23.12 OPENTREP::BasChronometer Struct Reference . . . 96
23.12.1 Detailed Description . . . 96
23.12.2 Constructor & Destructor Documentation . . . 96
23.12.3 Member Function Documentation . . . 96
23.13 OPENTREP::BasFileMgr Struct Reference . . . 97
23.13.1 Detailed Description . . . 97
23.13.2 Member Function Documentation . . . 97
23.14 OPENTREP::BomAbstract Class Reference . . . 97
23.14.1 Detailed Description . . . 98
23.14.2 Constructor & Destructor Documentation . . . 98
23.14.3 Member Function Documentation . . . 99
23.14.4 Friends And Related Function Documentation . . . 99
23.15 OPENTREP::BomJSONExport Class Reference . . . 100

23.15.1 Detailed Description . . . 100
23.15.2 Member Function Documentation . . . 100
23.16 OPENTREP::BomType Struct Reference . . . 101
23.16.1 Detailed Description . . . 101
23.16.2 Member Enumeration Documentation . . . 101
23.16.3 Member Function Documentation . . . 101
23.17 OPENTREP::BuildIndexException Class Reference . . . 102
23.17.1 Detailed Description . . . 102
23.17.2 Constructor & Destructor Documentation . . . 102
23.17.3 Member Function Documentation . . . 102
23.17.4 Member Data Documentation . . . 103
23.18 OPENTREP::CityASCIIName_T Struct Reference . . . 103
23.18.1 Detailed Description . . . 103
23.18.2 Constructor & Destructor Documentation . . . 103
23.19 OPENTREP::CityCode_T Struct Reference . . . 103
23.19.1 Detailed Description . . . 104
23.19.2 Constructor & Destructor Documentation . . . 104
23.20 OPENTREP::CityUTFName_T Struct Reference . . . 104
23.20.1 Detailed Description . . . 104
23.20.2 Constructor & Destructor Documentation . . . 105
23.21 OPENTREP::CodeConversionException Class Reference . . . 105
23.21.1 Detailed Description . . . 105
23.21.2 Constructor & Destructor Documentation . . . 105
23.21.3 Member Function Documentation . . . 105
23.21.4 Member Data Documentation . . . 106
23.22 OPENTREP::CodeDuplicationException Class Reference . . . 106
23.22.1 Detailed Description . . . 106
23.22.2 Constructor & Destructor Documentation . . . 106
23.22.3 Member Function Documentation . . . 107
23.22.4 Member Data Documentation . . . 107
23.23 OPENTREP::Comment_T Struct Reference . . . 107
23.23.1 Detailed Description . . . 107
23.23.2 Constructor & Destructor Documentation . . . 107
23.24 OPENTREP::CommonName_T Struct Reference . . . 108
23.24.1 Detailed Description . . . 108
23.24.2 Constructor & Destructor Documentation . . . 108
23.25 OPENTREP::ContinentCode_T Struct Reference . . . 108
23.25.1 Detailed Description . . . 109
23.25.2 Constructor & Destructor Documentation . . . 109
23.26 OPENTREP::ContinentName_T Struct Reference . . . 109

23.26.1 Detailed Description . . . 109
23.26.2 Constructor & Destructor Documentation . . . 109
23.27 OPENTREP::CountryCode_T Struct Reference . . . 110
23.27.1 Detailed Description . . . 110
23.27.2 Constructor & Destructor Documentation . . . 110
23.28 OPENTREP::CountryName_T Struct Reference . . . 110
23.28.1 Detailed Description . . . 111
23.28.2 Constructor & Destructor Documentation . . . 111
23.29 OPENTREP::date_time_element< MIN, MAX > Struct Template Reference . . . 111
23.29.1 Detailed Description . . . 111
23.29.2 Constructor & Destructor Documentation . . . 111
23.29.3 Member Function Documentation . . . 112
23.29.4 Member Data Documentation . . . 112
23.30 OPENTREP::DbaAbstract Class Reference . . . 112
23.30.1 Detailed Description . . . 112
23.30.2 Constructor & Destructor Documentation . . . 112
23.30.3 Member Function Documentation . . . 113
23.31 OPENTREP::DBManager Class Reference . . . 113
23.31.1 Detailed Description . . . 113
23.31.2 Member Function Documentation . . . 114
23.32 OPENTREP::DBParams Struct Reference . . . 117
23.32.1 Detailed Description . . . 118
23.32.2 Constructor & Destructor Documentation . . . 118
23.32.3 Member Function Documentation . . . 118
23.33 OPENTREP::DBSessionManager Class Reference . . . 120
23.33.1 Detailed Description . . . 120
23.33.2 Friends And Related Function Documentation . . . 120
23.34 OPENTREP::DistanceErrorRule Struct Reference . . . 120
23.34.1 Detailed Description . . . 121
23.34.2 Constructor & Destructor Documentation . . . 121
23.34.3 Member Function Documentation . . . 121
23.35 OPENTREP::DocumentNotFoundException Class Reference . . . 122
23.35.1 Detailed Description . . . 123
23.35.2 Constructor & Destructor Documentation . . . 123
23.35.3 Member Function Documentation . . . 123
23.35.4 Member Data Documentation . . . 123
23.36 OPENTREP::PorParserHelper::doEndPor Struct Reference . . . 124
23.36.1 Detailed Description . . . 124
23.36.2 Constructor & Destructor Documentation . . . 124
23.36.3 Member Function Documentation . . . 124

23.36.4 Member Data Documentation . . . 124
23.37 OPENTREP::FAACode_T Struct Reference . . . 125
23.37.1 Detailed Description . . . 125
23.37.2 Constructor & Destructor Documentation . . . 126
23.38 OPENTREP::FacBomAbstract Class Reference . . . 126
23.38.1 Detailed Description . . . 126
23.38.2 Member Typedef Documentation . . . 127
23.38.3 Constructor & Destructor Documentation . . . 127
23.38.4 Member Function Documentation . . . 127
23.38.5 Friends And Related Function Documentation . . . 128
23.38.6 Member Data Documentation . . . 128
23.39 OPENTREP::FacOpenTrepServiceContext Class Reference . . . 128
23.39.1 Detailed Description . . . 129
23.39.2 Member Typedef Documentation . . . 129
23.39.3 Constructor & Destructor Documentation . . . 129
23.39.4 Member Function Documentation . . . 129
23.39.5 Member Data Documentation . . . 130
23.40 OPENTREP::FacPlace Class Reference . . . 130
23.40.1 Detailed Description . . . 131
23.40.2 Member Typedef Documentation . . . 131
23.40.3 Constructor & Destructor Documentation . . . 132
23.40.4 Member Function Documentation . . . 132
23.40.5 Member Data Documentation . . . 135
23.41 OPENTREP::FacPlaceHolder Class Reference . . . 135
23.41.1 Detailed Description . . . 136
23.41.2 Member Typedef Documentation . . . 136
23.41.3 Constructor & Destructor Documentation . . . 136
23.41.4 Member Function Documentation . . . 136
23.41.5 Member Data Documentation . . . 138
23.42 OPENTREP::FacResult Class Reference . . . 138
23.42.1 Detailed Description . . . 139
23.42.2 Member Typedef Documentation . . . 139
23.42.3 Constructor & Destructor Documentation . . . 139
23.42.4 Member Function Documentation . . . 139
23.42.5 Member Data Documentation . . . 140
23.43 OPENTREP::FacResultCombination Class Reference . . . 140
23.43.1 Detailed Description . . . 141
23.43.2 Member Typedef Documentation . . . 141
23.43.3 Constructor & Destructor Documentation . . . 141
23.43.4 Member Function Documentation . . . 141

23.43.5 Member Data Documentation . . . 143
23.44 OPENTREP::FacResultHolder Class Reference . . . 143
23.44.1 Detailed Description . . . 144
23.44.2 Member Typedef Documentation . . . 144
23.44.3 Constructor & Destructor Documentation . . . 144
23.44.4 Member Function Documentation . . . 144
23.44.5 Member Data Documentation . . . 146
23.45 OPENTREP::FacServiceAbstract Class Reference . . . 147
23.45.1 Detailed Description . . . 147
23.45.2 Member Typedef Documentation . . . 147
23.45.3 Constructor & Destructor Documentation . . . 147
23.45.4 Member Function Documentation . . . 148
23.45.5 Member Data Documentation . . . 148
23.46 OPENTREP::FacSupervisor Class Reference . . . 148
23.46.1 Detailed Description . . . 149
23.46.2 Member Typedef Documentation . . . 149
23.46.3 Constructor & Destructor Documentation . . . 149
23.46.4 Member Function Documentation . . . 149
23.47 OPENTREP::FacWorld Class Reference . . . 151
23.47.1 Detailed Description . . . 152
23.47.2 Member Typedef Documentation . . . 152
23.47.3 Constructor & Destructor Documentation . . . 152
23.47.4 Member Function Documentation . . . 152
23.47.5 Member Data Documentation . . . 154
23.48 OPENTREP::FeatureClass_T Struct Reference . . . 154
23.48.1 Detailed Description . . . 154
23.48.2 Constructor & Destructor Documentation . . . 154
23.49 OPENTREP::FeatureCode_T Struct Reference . . . 154
23.49.1 Detailed Description . . . 155
23.49.2 Constructor & Destructor Documentation . . . 155
23.50 OPENTREP::FeatureName_T Struct Reference . . . 155
23.50.1 Detailed Description . . . 155
23.50.2 Constructor & Destructor Documentation . . . 156
23.51 OPENTREP::FileException Class Reference . . . 156
23.51.1 Detailed Description . . . 156
23.51.2 Constructor & Destructor Documentation . . . 156
23.51.3 Member Function Documentation . . . 156
23.51.4 Member Data Documentation . . . 157
23.52 OPENTREP::FileNotFoundException Class Reference . . . 157
23.52.1 Detailed Description . . . 157

23.52.2 Constructor & Destructor Documentation . . . 157
23.52.3 Member Function Documentation . . . 158
23.52.4 Member Data Documentation . . . 158
23.53 OPENTREP::FilePath_T Struct Reference . . . 158
23.53.1 Detailed Description . . . 158
23.53.2 Constructor & Destructor Documentation . . . 158
23.54 OPENTREP::Filter Struct Reference . . . 159
23.54.1 Detailed Description . . . 159
23.54.2 Member Function Documentation . . . 159
23.55 FloatingPoint< RawType > Class Template Reference . . . 160
23.55.1 Detailed Description . . . 161
23.55.2 Member Typedef Documentation . . . 161
23.55.3 Constructor & Destructor Documentation . . . 161
23.55.4 Member Function Documentation . . . 161
23.55.5 Member Data Documentation . . . 162
23.56 OPENTREP::IATACode_T Struct Reference . . . 163
23.56.1 Detailed Description . . . 163
23.56.2 Constructor & Destructor Documentation . . . 163
23.57 OPENTREP::IATAType Struct Reference . . . 164
23.57.1 Detailed Description . . . 164
23.57.2 Member Enumeration Documentation . . . 164
23.57.3 Constructor & Destructor Documentation . . . 165
23.57.4 Member Function Documentation . . . 165
23.58 OPENTREP::ICAOCode_T Struct Reference . . . 167
23.58.1 Detailed Description . . . 167
23.58.2 Constructor & Destructor Documentation . . . 167
23.59 OPENTREP::IndexBuilder Class Reference . . . 167
23.59.1 Detailed Description . . . 168
23.59.2 Friends And Related Function Documentation . . . 168
23.60 OPENTREP::InterpreterUseCaseException Class Reference . . . 168
23.60.1 Detailed Description . . . 168
23.60.2 Constructor & Destructor Documentation . . . 169
23.60.3 Member Function Documentation . . . 169
23.60.4 Member Data Documentation . . . 169
23.61 OPENTREP::InterpreteTravelRequestException Class Reference . . . 169
23.61.1 Detailed Description . . . 170
23.61.2 Constructor & Destructor Documentation . . . 170
23.61.3 Member Function Documentation . . . 170
23.61.4 Member Data Documentation . . . 170
23.62 OPENTREP::KeyAbstract Class Reference . . . 170

23.62.1 Detailed Description . . . 171
23.62.2 Constructor & Destructor Documentation . . . 171
23.62.3 Member Function Documentation . . . 171
23.63OPENTREP::Language Struct Reference . . . 171
23.63.1 Detailed Description . . . 172
23.63.2 Member Enumeration Documentation . . . 172
23.63.3 Member Function Documentation . . . 173
23.64OPENTREP::LanguageCode_T Struct Reference . . . 174
23.64.1 Detailed Description . . . 174
23.64.2 Constructor & Destructor Documentation . . . 174
23.65OPENTREP::LanguageCodeNotDefinedInNameTableException Class Reference . . . 174
23.65.1 Detailed Description . . . 175
23.65.2 Constructor & Destructor Documentation . . . 175
23.65.3 Member Function Documentation . . . 175
23.65.4 Member Data Documentation . . . 175
23.66OPENTREP::Levenshtein Class Reference . . . 176
23.66.1 Detailed Description . . . 176
23.66.2 Member Function Documentation . . . 176
23.67OPENTREP::ListGenerator Struct Reference . . . 177
23.67.1 Detailed Description . . . 177
23.67.2 Member Function Documentation . . . 177
23.68OPENTREP::Location Struct Reference . . . 178
23.68.1 Detailed Description . . . 181
23.68.2 Constructor & Destructor Documentation . . . 181
23.68.3 Member Function Documentation . . . 182
23.68.4 Member Data Documentation . . . 199
23.69OPENTREP::LocationExchange Class Reference . . . 200
23.69.1 Detailed Description . . . 200
23.69.2 Member Function Documentation . . . 200
23.70OPENTREP::LocationKey Struct Reference . . . 201
23.70.1 Detailed Description . . . 202
23.70.2 Constructor & Destructor Documentation . . . 202
23.70.3 Member Function Documentation . . . 202
23.71OPENTREP::LocationName_T Struct Reference . . . 204
23.71.1 Detailed Description . . . 205
23.71.2 Constructor & Destructor Documentation . . . 205
23.72OPENTREP::PorParserHelper::LocationParser< Iterator > Struct Template Reference . . . 205
23.72.1 Detailed Description . . . 208
23.72.2 Constructor & Destructor Documentation . . . 209
23.72.3 Member Data Documentation . . . 210

23.73 OPENTREP::Logger Class Reference . . . 218
23.73.1 Detailed Description . . . 219
23.73.2 Member Function Documentation . . . 219
23.73.3 Friends And Related Function Documentation . . . 219
23.74 OPENTREP::MemoryAllocationException Class Reference . . . 220
23.74.1 Detailed Description . . . 220
23.74.2 Constructor & Destructor Documentation . . . 220
23.74.3 Member Function Documentation . . . 220
23.74.4 Member Data Documentation . . . 220
23.75 OPENTREP::MultipleRowsForASingleDocIDException Class Reference . . . 221
23.75.1 Detailed Description . . . 221
23.75.2 Constructor & Destructor Documentation . . . 221
23.75.3 Member Function Documentation . . . 221
23.75.4 Member Data Documentation . . . 222
23.76 OPENTREP::NameMatrix Struct Reference . . . 222
23.76.1 Detailed Description . . . 222
23.76.2 Constructor & Destructor Documentation . . . 222
23.76.3 Member Function Documentation . . . 223
23.77 OPENTREP::Names Struct Reference . . . 225
23.77.1 Detailed Description . . . 226
23.77.2 Constructor & Destructor Documentation . . . 226
23.77.3 Member Function Documentation . . . 226
23.78 OPENTREP::NoBestMatchingResultHolderException Class Reference . . . 228
23.78.1 Detailed Description . . . 229
23.78.2 Constructor & Destructor Documentation . . . 229
23.78.3 Member Function Documentation . . . 229
23.78.4 Member Data Documentation . . . 229
23.79 OPENTREP::NonInitialisedServiceException Class Reference . . . 229
23.79.1 Detailed Description . . . 230
23.79.2 Constructor & Destructor Documentation . . . 230
23.79.3 Member Function Documentation . . . 230
23.79.4 Member Data Documentation . . . 230
23.80 OPENTREP::ObjectNotFoundException Class Reference . . . 230
23.80.1 Detailed Description . . . 231
23.80.2 Constructor & Destructor Documentation . . . 231
23.80.3 Member Function Documentation . . . 231
23.80.4 Member Data Documentation . . . 231
23.81 OPENTREP::OPENTREP_Abstract Struct Reference . . . 232
23.81.1 Detailed Description . . . 232
23.81.2 Constructor & Destructor Documentation . . . 232

23.81.3 Member Function Documentation . . . 232
23.82 OPENTREP::OPENTREP_Service Class Reference . . . 233
23.82.1 Detailed Description . . . 234
23.82.2 Member Typedef Documentation . . . 234
23.82.3 Constructor & Destructor Documentation . . . 234
23.82.4 Member Function Documentation . . . 235
23.83 OPENTREP::OPENTREP_ServiceContext Class Reference . . . 236
23.83.1 Detailed Description . . . 237
23.83.2 Member Function Documentation . . . 237
23.83.3 Friends And Related Function Documentation . . . 239
23.84 OPENTREP::OpenTrepSearcher Struct Reference . . . 239
23.84.1 Detailed Description . . . 239
23.84.2 Constructor & Destructor Documentation . . . 240
23.84.3 Member Function Documentation . . . 240
23.85 OPENTREP::OTransliterator Class Reference . . . 241
23.85.1 Detailed Description . . . 241
23.85.2 Constructor & Destructor Documentation . . . 241
23.85.3 Member Function Documentation . . . 242
23.86 OPENTREP::OutputFormat Struct Reference . . . 243
23.86.1 Detailed Description . . . 243
23.86.2 Member Enumeration Documentation . . . 244
23.86.3 Constructor & Destructor Documentation . . . 244
23.86.4 Member Function Documentation . . . 244
23.87 OPENTREP::ParserException Class Reference . . . 245
23.87.1 Detailed Description . . . 246
23.87.2 Constructor & Destructor Documentation . . . 246
23.87.3 Member Function Documentation . . . 246
23.87.4 Member Data Documentation . . . 246
23.88 OPENTREP::PorParserHelper::ParserSemanticAction Struct Reference . . . 247
23.88.1 Detailed Description . . . 247
23.88.2 Constructor & Destructor Documentation . . . 247
23.88.3 Member Data Documentation . . . 248
23.89 OPENTREP::Place Class Reference . . . 248
23.89.1 Detailed Description . . . 251
23.89.2 Member Typedef Documentation . . . 251
23.89.3 Member Function Documentation . . . 252
23.89.4 Friends And Related Function Documentation . . . 270
23.90 OPENTREP::PlaceHolder Class Reference . . . 271
23.90.1 Detailed Description . . . 271
23.90.2 Member Function Documentation . . . 272

23.90.3 Friends And Related Function Documentation . . . 273
23.91 OPENTREP::PORFileParser Class Reference . . . 273
23.91.1 Detailed Description . . . 273
23.91.2 Constructor & Destructor Documentation . . . 273
23.91.3 Member Function Documentation . . . 273
23.92 OPENTREP::PorFileParsingException Class Reference . . . 274
23.92.1 Detailed Description . . . 274
23.92.2 Constructor & Destructor Documentation . . . 274
23.92.3 Member Function Documentation . . . 274
23.92.4 Member Data Documentation . . . 275
23.93 OPENTREP::PORFilePath_T Struct Reference . . . 275
23.93.1 Detailed Description . . . 275
23.93.2 Constructor & Destructor Documentation . . . 275
23.94 OPENTREP::PORParser Class Reference . . . 275
23.94.1 Detailed Description . . . 276
23.94.2 Member Function Documentation . . . 276
23.95 OPENTREP::PORStringParser Class Reference . . . 276
23.95.1 Detailed Description . . . 276
23.95.2 Constructor & Destructor Documentation . . . 276
23.95.3 Member Function Documentation . . . 277
23.96 OPENTREP::PRFileParsingException Class Reference . . . 277
23.96.1 Detailed Description . . . 277
23.96.2 Constructor & Destructor Documentation . . . 278
23.96.3 Member Function Documentation . . . 278
23.96.4 Member Data Documentation . . . 278
23.97 OPENTREP::QuerySlices Struct Reference . . . 278
23.97.1 Detailed Description . . . 279
23.97.2 Constructor & Destructor Documentation . . . 279
23.97.3 Member Function Documentation . . . 279
23.97.4 Member Data Documentation . . . 281
23.98 OPENTREP::RawDataString_T Struct Reference . . . 281
23.98.1 Detailed Description . . . 282
23.98.2 Constructor & Destructor Documentation . . . 282
23.99 OPENTREP::RequestInterpreter Class Reference . . . 282
23.99.1 Detailed Description . . . 282
23.99.2 Friends And Related Function Documentation . . . 282
23.100 OPENTREP::Result Class Reference . . . 282
23.100.1 Detailed Description . . . 284
23.100.2 Member Function Documentation . . . 284
23.100.3 Friends And Related Function Documentation . . . 292

23.101 OPENTREP::ResultCombination Class Reference . . . 292
23.101.1 Detailed Description . . . 293
23.101.2 Member Function Documentation . . . 293
23.101.3 Friends And Related Function Documentation . . . 295
23.102 OPENTREP::ResultHolder Class Reference . . . 296
23.102.1 Detailed Description . . . 296
23.102.2 Member Function Documentation . . . 296
23.102.3 Friends And Related Function Documentation . . . 299
23.103 OPENTREP::RootException Class Reference . . . 299
23.103.1 Detailed Description . . . 300
23.103.2 Constructor & Destructor Documentation . . . 300
23.103.3 Member Function Documentation . . . 301
23.103.4 Member Data Documentation . . . 301
23.104 OPENTREP::ScoreBoard Struct Reference . . . 301
23.104.1 Detailed Description . . . 302
23.104.2 Member Typedef Documentation . . . 302
23.104.3 Constructor & Destructor Documentation . . . 302
23.104.4 Member Function Documentation . . . 303
23.105 OPENTREP::ScoreType Struct Reference . . . 305
23.105.1 Detailed Description . . . 306
23.105.2 Member Enumeration Documentation . . . 306
23.105.3 Constructor & Destructor Documentation . . . 306
23.105.4 Member Function Documentation . . . 306
23.106 OPENTREP::ServiceAbstract Class Reference . . . 308
23.106.1 Detailed Description . . . 309
23.106.2 Constructor & Destructor Documentation . . . 309
23.106.3 Member Function Documentation . . . 309
23.107 OPENTREP::ServiceUtilities Class Reference . . . 309
23.107.1 Detailed Description . . . 310
23.107.2 Member Function Documentation . . . 310
23.108 OPENTREP::short_bool_policies Struct Reference . . . 310
23.108.1 Detailed Description . . . 310
23.108.2 Member Function Documentation . . . 311
23.109 OPENTREP::SQLDatabaseConnectionImpossibleException Class Reference . . . 311
23.109.1 Detailed Description . . . 311
23.109.2 Constructor & Destructor Documentation . . . 311
23.109.3 Member Function Documentation . . . 312
23.109.4 Member Data Documentation . . . 312
23.110 OPENTREP::SQLDatabaseException Class Reference . . . 312
23.110.1 Detailed Description . . . 312

23.110.2 Constructor & Destructor Documentation . . . 313
23.110.3 Member Function Documentation . . . 313
23.110.4 Member Data Documentation . . . 313
23.111 OPENTREP::SQLiteDBFilePath_T Struct Reference . . . 313
23.111.1 Detailed Description . . . 313
23.111.2 Constructor & Destructor Documentation . . . 314
23.112 OPENTREP::StateCode_T Struct Reference . . . 314
23.112.1 Detailed Description . . . 314
23.112.2 Constructor & Destructor Documentation . . . 314
23.113 OPENTREP::PorParserHelper::storeAdm1AsciiName Struct Reference . . . 314
23.113.1 Detailed Description . . . 315
23.113.2 Constructor & Destructor Documentation . . . 315
23.113.3 Member Function Documentation . . . 315
23.113.4 Member Data Documentation . . . 315
23.114 OPENTREP::PorParserHelper::storeAdm1Code Struct Reference . . . 316
23.114.1 Detailed Description . . . 316
23.114.2 Constructor & Destructor Documentation . . . 316
23.114.3 Member Function Documentation . . . 317
23.114.4 Member Data Documentation . . . 317
23.115 OPENTREP::PorParserHelper::storeAdm1UtfName Struct Reference . . . 317
23.115.1 Detailed Description . . . 318
23.115.2 Constructor & Destructor Documentation . . . 318
23.115.3 Member Function Documentation . . . 318
23.115.4 Member Data Documentation . . . 318
23.116 OPENTREP::PorParserHelper::storeAdm2AsciiName Struct Reference . . . 319
23.116.1 Detailed Description . . . 319
23.116.2 Constructor & Destructor Documentation . . . 319
23.116.3 Member Function Documentation . . . 320
23.116.4 Member Data Documentation . . . 320
23.117 OPENTREP::PorParserHelper::storeAdm2Code Struct Reference . . . 321
23.117.1 Detailed Description . . . 321
23.117.2 Constructor & Destructor Documentation . . . 321
23.117.3 Member Function Documentation . . . 321
23.117.4 Member Data Documentation . . . 321
23.118 OPENTREP::PorParserHelper::storeAdm2UtfName Struct Reference . . . 322
23.118.1 Detailed Description . . . 323
23.118.2 Constructor & Destructor Documentation . . . 323
23.118.3 Member Function Documentation . . . 323
23.118.4 Member Data Documentation . . . 323
23.119 OPENTREP::PorParserHelper::storeAdm3Code Struct Reference . . . 324

23.119.1 Detailed Description . . . 324
23.119.2 Constructor & Destructor Documentation . . . 324
23.119.3 Member Function Documentation . . . 324
23.119.4 Member Data Documentation . . . 325
23.120 OPENTREP::PorParserHelper::storeAdm4Code Struct Reference . . . 325
23.120.1 Detailed Description . . . 326
23.120.2 Constructor & Destructor Documentation . . . 326
23.120.3 Member Function Documentation . . . 326
23.120.4 Member Data Documentation . . . 326
23.121 OPENTREP::PorParserHelper::storeAltCountryCode Struct Reference . . . 327
23.121.1 Detailed Description . . . 327
23.121.2 Constructor & Destructor Documentation . . . 327
23.121.3 Member Function Documentation . . . 327
23.121.4 Member Data Documentation . . . 328
23.122 OPENTREP::PorParserHelper::storeAltLangCode2Char Struct Reference . . . 328
23.122.1 Detailed Description . . . 329
23.122.2 Constructor & Destructor Documentation . . . 329
23.122.3 Member Function Documentation . . . 329
23.122.4 Member Data Documentation . . . 329
23.123 OPENTREP::PorParserHelper::storeAltLangCodeExt Struct Reference . . . 330
23.123.1 Detailed Description . . . 330
23.123.2 Constructor & Destructor Documentation . . . 330
23.123.3 Member Function Documentation . . . 331
23.123.4 Member Data Documentation . . . 331
23.124 OPENTREP::PorParserHelper::storeAltLangCodeFull Struct Reference . . . 331
23.124.1 Detailed Description . . . 332
23.124.2 Constructor & Destructor Documentation . . . 332
23.124.3 Member Function Documentation . . . 332
23.124.4 Member Data Documentation . . . 332
23.125 OPENTREP::PorParserHelper::storeAltLangCodeHist Struct Reference . . . 333
23.125.1 Detailed Description . . . 333
23.125.2 Constructor & Destructor Documentation . . . 334
23.125.3 Member Function Documentation . . . 334
23.125.4 Member Data Documentation . . . 334
23.126 OPENTREP::PorParserHelper::storeAltName Struct Reference . . . 335
23.126.1 Detailed Description . . . 335
23.126.2 Constructor & Destructor Documentation . . . 335
23.126.3 Member Function Documentation . . . 335
23.126.4 Member Data Documentation . . . 335
23.127 OPENTREP::PorParserHelper::storeAltNameShort Struct Reference . . . 336

23.127.1 Detailed Description . . . 337
23.127.2 Constructor & Destructor Documentation . . . 337
23.127.3 Member Function Documentation . . . 337
23.127.4 Member Data Documentation . . . 337
23.128 OPENTREP::PorParserHelper::storeAltNameShortListString Struct Reference . . . 338
23.128.1 Detailed Description . . . 338
23.128.2 Constructor & Destructor Documentation . . . 338
23.128.3 Member Function Documentation . . . 338
23.128.4 Member Data Documentation . . . 339
23.129 OPENTREP::PorParserHelper::storeAsciiName Struct Reference . . . 339
23.129.1 Detailed Description . . . 340
23.129.2 Constructor & Destructor Documentation . . . 340
23.129.3 Member Function Documentation . . . 340
23.129.4 Member Data Documentation . . . 340
23.130 OPENTREP::PorParserHelper::storeCityAsciiName Struct Reference . . . 341
23.130.1 Detailed Description . . . 341
23.130.2 Constructor & Destructor Documentation . . . 341
23.130.3 Member Function Documentation . . . 341
23.130.4 Member Data Documentation . . . 342
23.131 OPENTREP::PorParserHelper::storeCityCode Struct Reference . . . 342
23.131.1 Detailed Description . . . 343
23.131.2 Constructor & Destructor Documentation . . . 343
23.131.3 Member Function Documentation . . . 343
23.131.4 Member Data Documentation . . . 343
23.132 OPENTREP::PorParserHelper::storeCityGeonamesID Struct Reference . . . 344
23.132.1 Detailed Description . . . 344
23.132.2 Constructor & Destructor Documentation . . . 344
23.132.3 Member Function Documentation . . . 345
23.132.4 Member Data Documentation . . . 345
23.133 OPENTREP::PorParserHelper::storeCityUtfName Struct Reference . . . 345
23.133.1 Detailed Description . . . 346
23.133.2 Constructor & Destructor Documentation . . . 346
23.133.3 Member Function Documentation . . . 346
23.133.4 Member Data Documentation . . . 346
23.134 OPENTREP::PorParserHelper::storeComments Struct Reference . . . 347
23.134.1 Detailed Description . . . 347
23.134.2 Constructor & Destructor Documentation . . . 348
23.134.3 Member Function Documentation . . . 348
23.134.4 Member Data Documentation . . . 348
23.135 OPENTREP::PorParserHelper::storeCommonName Struct Reference . . . 349

23.135.1 Detailed Description . . . 349
23.135.2 Constructor & Destructor Documentation . . . 349
23.135.3 Member Function Documentation . . . 349
23.135.4 Member Data Documentation . . . 349
23.136 OPENTREP::PorParserHelper::storeContinentName Struct Reference . . . 350
23.136.1 Detailed Description . . . 351
23.136.2 Constructor & Destructor Documentation . . . 351
23.136.3 Member Function Documentation . . . 351
23.136.4 Member Data Documentation . . . 351
23.137 OPENTREP::PorParserHelper::storeCountryCode Struct Reference . . . 352
23.137.1 Detailed Description . . . 352
23.137.2 Constructor & Destructor Documentation . . . 352
23.137.3 Member Function Documentation . . . 352
23.137.4 Member Data Documentation . . . 353
23.138 OPENTREP::PorParserHelper::storeCountryName Struct Reference . . . 353
23.138.1 Detailed Description . . . 354
23.138.2 Constructor & Destructor Documentation . . . 354
23.138.3 Member Function Documentation . . . 354
23.138.4 Member Data Documentation . . . 354
23.139 OPENTREP::PorParserHelper::storeDateFrom Struct Reference . . . 355
23.139.1 Detailed Description . . . 355
23.139.2 Constructor & Destructor Documentation . . . 355
23.139.3 Member Function Documentation . . . 355
23.139.4 Member Data Documentation . . . 356
23.140 OPENTREP::PorParserHelper::storeDateUntil Struct Reference . . . 356
23.140.1 Detailed Description . . . 357
23.140.2 Constructor & Destructor Documentation . . . 357
23.140.3 Member Function Documentation . . . 357
23.140.4 Member Data Documentation . . . 357
23.141 OPENTREP::PorParserHelper::storeDSTOffset Struct Reference . . . 358
23.141.1 Detailed Description . . . 358
23.141.2 Constructor & Destructor Documentation . . . 358
23.141.3 Member Function Documentation . . . 358
23.141.4 Member Data Documentation . . . 359
23.142 OPENTREP::PorParserHelper::storeElevation Struct Reference . . . 359
23.142.1 Detailed Description . . . 360
23.142.2 Constructor & Destructor Documentation . . . 360
23.142.3 Member Function Documentation . . . 360
23.142.4 Member Data Documentation . . . 360
23.143 OPENTREP::PorParserHelper::storeEnvelopeID Struct Reference . . . 361

23.143.1 Detailed Description . . . 361
23.143.2 Constructor & Destructor Documentation . . . 361
23.143.3 Member Function Documentation . . . 361
23.143.4 Member Data Documentation . . . 362
23.144 OPENTREP::PorParserHelper::storeFaaCode Struct Reference . . . 362
23.144.1 Detailed Description . . . 363
23.144.2 Constructor & Destructor Documentation . . . 363
23.144.3 Member Function Documentation . . . 363
23.144.4 Member Data Documentation . . . 363
23.145 OPENTREP::PorParserHelper::storeFeatureClass Struct Reference . . . 364
23.145.1 Detailed Description . . . 364
23.145.2 Constructor & Destructor Documentation . . . 364
23.145.3 Member Function Documentation . . . 365
23.145.4 Member Data Documentation . . . 365
23.146 OPENTREP::PorParserHelper::storeFeatureCode Struct Reference . . . 365
23.146.1 Detailed Description . . . 366
23.146.2 Constructor & Destructor Documentation . . . 366
23.146.3 Member Function Documentation . . . 366
23.146.4 Member Data Documentation . . . 366
23.147 OPENTREP::PorParserHelper::storeGeonamesID Struct Reference . . . 367
23.147.1 Detailed Description . . . 367
23.147.2 Constructor & Destructor Documentation . . . 367
23.147.3 Member Function Documentation . . . 368
23.147.4 Member Data Documentation . . . 368
23.148 OPENTREP::PorParserHelper::storeGMTOffset Struct Reference . . . 369
23.148.1 Detailed Description . . . 369
23.148.2 Constructor & Destructor Documentation . . . 369
23.148.3 Member Function Documentation . . . 369
23.148.4 Member Data Documentation . . . 369
23.149 OPENTREP::PorParserHelper::storeGTopo30 Struct Reference . . . 370
23.149.1 Detailed Description . . . 371
23.149.2 Constructor & Destructor Documentation . . . 371
23.149.3 Member Function Documentation . . . 371
23.149.4 Member Data Documentation . . . 371
23.150 OPENTREP::PorParserHelper::storeIataCode Struct Reference . . . 372
23.150.1 Detailed Description . . . 372
23.150.2 Constructor & Destructor Documentation . . . 372
23.150.3 Member Function Documentation . . . 372
23.150.4 Member Data Documentation . . . 373
23.151 OPENTREP::PorParserHelper::storeIcaoCode Struct Reference . . . 373

23.151.1 Detailed Description . . . 374
23.151.2 Constructor & Destructor Documentation . . . 374
23.151.3 Member Function Documentation . . . 374
23.151.4 Member Data Documentation . . . 374
23.152 OPENTREP::PorParserHelper::storeLatitude Struct Reference . . . 375
23.152.1 Detailed Description . . . 375
23.152.2 Constructor & Destructor Documentation . . . 375
23.152.3 Member Function Documentation . . . 375
23.152.4 Member Data Documentation . . . 376
23.153 OPENTREP::PorParserHelper::storeLongitude Struct Reference . . . 376
23.153.1 Detailed Description . . . 377
23.153.2 Constructor & Destructor Documentation . . . 377
23.153.3 Member Function Documentation . . . 377
23.153.4 Member Data Documentation . . . 377
23.154 OPENTREP::PorParserHelper::storeModDate Struct Reference . . . 378
23.154.1 Detailed Description . . . 378
23.154.2 Constructor & Destructor Documentation . . . 378
23.154.3 Member Function Documentation . . . 379
23.154.4 Member Data Documentation . . . 379
23.155 OPENTREP::PorParserHelper::storePageRank Struct Reference . . . 379
23.155.1 Detailed Description . . . 380
23.155.2 Constructor & Destructor Documentation . . . 380
23.155.3 Member Function Documentation . . . 380
23.155.4 Member Data Documentation . . . 380
23.156 OPENTREP::PorParserHelper::storePopulation Struct Reference . . . 381
23.156.1 Detailed Description . . . 381
23.156.2 Constructor & Destructor Documentation . . . 381
23.156.3 Member Function Documentation . . . 382
23.156.4 Member Data Documentation . . . 382
23.157 OPENTREP::PorParserHelper::storePORType Struct Reference . . . 383
23.157.1 Detailed Description . . . 383
23.157.2 Constructor & Destructor Documentation . . . 383
23.157.3 Member Function Documentation . . . 383
23.157.4 Member Data Documentation . . . 383
23.158 OPENTREP::PorParserHelper::storeRawOffset Struct Reference . . . 384
23.158.1 Detailed Description . . . 385
23.158.2 Constructor & Destructor Documentation . . . 385
23.158.3 Member Function Documentation . . . 385
23.158.4 Member Data Documentation . . . 385
23.159 OPENTREP::PorParserHelper::storeStateCode Struct Reference . . . 386

23.159.1 Detailed Description . . . 386
23.159.2 Constructor & Destructor Documentation . . . 386
23.159.3 Member Function Documentation . . . 386
23.159.4 Member Data Documentation . . . 387
23.160 OPENTREP::PorParserHelper::storeTimeZone Struct Reference . . . 387
23.160.1 Detailed Description . . . 388
23.160.2 Constructor & Destructor Documentation . . . 388
23.160.3 Member Function Documentation . . . 388
23.160.4 Member Data Documentation . . . 388
23.161 OPENTREP::PorParserHelper::storeTvlPORCode Struct Reference . . . 389
23.161.1 Detailed Description . . . 389
23.161.2 Constructor & Destructor Documentation . . . 389
23.161.3 Member Function Documentation . . . 389
23.161.4 Member Data Documentation . . . 390
23.162 OPENTREP::PorParserHelper::storeTvlPORListString Struct Reference . . . 390
23.162.1 Detailed Description . . . 391
23.162.2 Constructor & Destructor Documentation . . . 391
23.162.3 Member Function Documentation . . . 391
23.162.4 Member Data Documentation . . . 391
23.163 OPENTREP::PorParserHelper::storeWikiLink Struct Reference . . . 392
23.163.1 Detailed Description . . . 392
23.163.2 Constructor & Destructor Documentation . . . 392
23.163.3 Member Function Documentation . . . 393
23.163.4 Member Data Documentation . . . 393
23.164 OPENTREP::StringPartition Struct Reference . . . 393
23.164.1 Detailed Description . . . 394
23.164.2 Member Typedef Documentation . . . 394
23.164.3 Constructor & Destructor Documentation . . . 395
23.164.4 Member Function Documentation . . . 395
23.164.5 Member Data Documentation . . . 397
23.165 OPENTREP::StringSet Struct Reference . . . 397
23.165.1 Detailed Description . . . 398
23.165.2 Member Typedef Documentation . . . 398
23.165.3 Constructor & Destructor Documentation . . . 398
23.165.4 Member Function Documentation . . . 398
23.165.5 Member Data Documentation . . . 400
23.166 OPENTREP::StructAbstract Struct Reference . . . 400
23.166.1 Detailed Description . . . 401
23.166.2 Constructor & Destructor Documentation . . . 401
23.166.3 Member Function Documentation . . . 401

23.167 OPENTREP::TimeZone_T Struct Reference . . . 402
23.167.1 Detailed Description . . . 402
23.167.2 Constructor & Destructor Documentation . . . 402
23.168 OPENTREP::TravelDBFilePath_T Struct Reference . . . 402
23.168.1 Detailed Description . . . 403
23.168.2 Constructor & Destructor Documentation . . . 403
23.169 OPENTREP::TravelRequestEmptyException Class Reference . . . 403
23.169.1 Detailed Description . . . 403
23.169.2 Constructor & Destructor Documentation . . . 404
23.169.3 Member Function Documentation . . . 404
23.169.4 Member Data Documentation . . . 404
23.170 OPENTREP::TvlPORListString_T Struct Reference . . . 404
23.170.1 Detailed Description . . . 405
23.170.2 Constructor & Destructor Documentation . . . 405
23.171 soci::type_conversion< OPENTREP::Place > Struct Template Reference . . . 405
23.171.1 Detailed Description . . . 405
23.171.2 Member Typedef Documentation . . . 405
23.171.3 Member Function Documentation . . . 405
23.172 TypeWithSize< size > Class Template Reference . . . 406
23.172.1 Detailed Description . . . 406
23.172.2 Member Typedef Documentation . . . 406
23.173 TypeWithSize< 4 > Class Template Reference . . . 407
23.173.1 Detailed Description . . . 407
23.173.2 Member Typedef Documentation . . . 407
23.174 TypeWithSize< 8 > Class Template Reference . . . 407
23.174.1 Detailed Description . . . 407
23.174.2 Member Typedef Documentation . . . 407
23.175 OPENTREP::UnicodeException Class Reference . . . 408
23.175.1 Detailed Description . . . 408
23.175.2 Constructor & Destructor Documentation . . . 408
23.175.3 Member Function Documentation . . . 408
23.175.4 Member Data Documentation . . . 409
23.176 OPENTREP::UnicodeTransliteratorCreationException Class Reference . . . 409
23.176.1 Detailed Description . . . 409
23.176.2 Constructor & Destructor Documentation . . . 409
23.176.3 Member Function Documentation . . . 410
23.176.4 Member Data Documentation . . . 410
23.177 UnitTestConfig Struct Reference . . . 410
23.177.1 Detailed Description . . . 410
23.177.2 Constructor & Destructor Documentation . . . 410

23.178 OPENTREP::UTF8Handler Class Reference . . . 411
23.178.1 Detailed Description . . . 411
23.178.2 Member Function Documentation . . . 411
23.179 OPENTREP::WikiLink_T Struct Reference . . . 412
23.179.1 Detailed Description . . . 412
23.179.2 Constructor & Destructor Documentation . . . 412
23.180 OPENTREP::WordCombinationHolder Struct Reference . . . 413
23.180.1 Detailed Description . . . 413
23.180.2 Member Typedef Documentation . . . 414
23.180.3 Constructor & Destructor Documentation . . . 414
23.180.4 Member Function Documentation . . . 414
23.180.5 Member Data Documentation . . . 416
23.181 OPENTREP::WordHolder Class Reference . . . 416
23.181.1 Detailed Description . . . 417
23.181.2 Member Function Documentation . . . 417
23.181.3 Friends And Related Function Documentation . . . 418
23.182 OPENTREP::World Class Reference . . . 419
23.182.1 Detailed Description . . . 419
23.182.2 Member Function Documentation . . . 419
23.182.3 Friends And Related Function Documentation . . . 422
23.183 OPENTREP::XapianDatabaseFailureException Class Reference . . . 422
23.183.1 Detailed Description . . . 422
23.183.2 Constructor & Destructor Documentation . . . 422
23.183.3 Member Function Documentation . . . 423
23.183.4 Member Data Documentation . . . 423
23.184 OPENTREP::XapianException Class Reference . . . 423
23.184.1 Detailed Description . . . 423
23.184.2 Constructor & Destructor Documentation . . . 423
23.184.3 Member Function Documentation . . . 424
23.184.4 Member Data Documentation . . . 424
23.185 OPENTREP::XapianIndexManager Class Reference . . . 424
23.185.1 Detailed Description . . . 424
23.185.2 Friends And Related Function Documentation . . . 424
23.186 OPENTREP::XapianTravelDatabaseEmptyException Class Reference . . . 424
23.186.1 Detailed Description . . . 425
23.186.2 Constructor & Destructor Documentation . . . 425
23.186.3 Member Function Documentation . . . 425
23.186.4 Member Data Documentation . . . 425
23.187 OPENTREP::XapianTravelDatabaseWrongPathnameException Class Reference . . . 426
23.187.1 Detailed Description . . . 426

23.187.2 Constructor & Destructor Documentation . . . 426
23.187.3 Member Function Documentation . . . 426
23.187.4 Member Data Documentation . . . 427

24 File Documentation **427**

24.1 doc/local/authors.doc File Reference . . . 427
24.2 doc/local/codingrules.doc File Reference . . . 427
24.3 doc/local/copyright.doc File Reference . . . 427
24.4 doc/local/documentation.doc File Reference . . . 427
24.5 doc/local/features.doc File Reference . . . 427
24.6 doc/local/help_wanted.doc File Reference . . . 427
24.7 doc/local/howto_release.doc File Reference . . . 427
24.8 doc/local/index.doc File Reference . . . 427
24.9 doc/local/installation.doc File Reference . . . 427
24.10 doc/local/linking.doc File Reference . . . 427
24.11 doc/local/test.doc File Reference . . . 427
24.12 doc/local/users_guide.doc File Reference . . . 427
24.13 doc/local/verification.doc File Reference . . . 427
24.14 doc/tutorial/tutorial.doc File Reference . . . 427
24.15 opentrep/basic/BasChronometer.cpp File Reference . . . 427
24.16 BasChronometer.cpp . . . 428
24.17 opentrep/basic/BasChronometer.hpp File Reference . . . 428
24.18 BasChronometer.hpp . . . 428
24.19 opentrep/basic/BasConst.cpp File Reference . . . 429
24.20 BasConst.cpp . . . 430
24.21 opentrep/basic/BasConst_General.hpp File Reference . . . 431
24.22 BasConst_General.hpp . . . 432
24.23 opentrep/basic/BasConst_OPENTREP_Service.hpp File Reference . . . 432
24.24 BasConst_OPENTREP_Service.hpp . . . 433
24.25 opentrep/basic/BasConst_Unicode.hpp File Reference . . . 433
24.26 BasConst_Unicode.hpp . . . 433
24.27 opentrep/basic/BasFileMgr.cpp File Reference . . . 433
24.28 BasFileMgr.cpp . . . 434
24.29 opentrep/basic/BasFileMgr.hpp File Reference . . . 434
24.30 BasFileMgr.hpp . . . 434
24.31 opentrep/basic/BasParserTypes.hpp File Reference . . . 435
24.32 BasParserTypes.hpp . . . 436
24.33 opentrep/basic/float_utils.hpp File Reference . . . 437
24.34 float_utils.hpp . . . 437
24.35 opentrep/basic/float_utils_google.hpp File Reference . . . 438

24.36 float_utils_google.hpp . . . 438
24.37 opentrep/basic/icu_util.cpp File Reference . . . 441
24.38 icu_util.cpp . . . 441
24.39 opentrep/basic/icu_util.hpp File Reference . . . 442
24.40 icu_util.hpp . . . 443
24.41 opentrep/basic/Location.cpp File Reference . . . 443
24.42 Location.cpp . . . 443
24.43 opentrep/basic/LocationKey.cpp File Reference . . . 450
24.44 LocationKey.cpp . . . 450
24.45 opentrep/basic/NameMatrix.cpp File Reference . . . 451
24.46 NameMatrix.cpp . . . 451
24.47 opentrep/basic/Names.cpp File Reference . . . 453
24.48 Names.cpp . . . 453
24.49 opentrep/basic/OTransliterator.cpp File Reference . . . 454
24.50 OTransliterator.cpp . . . 454
24.51 opentrep/basic/OTransliterator.hpp File Reference . . . 456
24.52 OTransliterator.hpp . . . 457
24.53 opentrep/basic/OutputFormat.cpp File Reference . . . 457
24.54 OutputFormat.cpp . . . 457
24.55 opentrep/basic/ScoreType.cpp File Reference . . . 459
24.56 ScoreType.cpp . . . 459
24.57 opentrep/basic/ScoreType.hpp File Reference . . . 461
24.58 ScoreType.hpp . . . 461
24.59 opentrep/basic/StructAbstract.hpp File Reference . . . 462
24.59.1 Function Documentation . . . 462
24.60 StructAbstract.hpp . . . 463
24.61 opentrep/basic/UTF8Handler.cpp File Reference . . . 463
24.62 UTF8Handler.cpp . . . 464
24.63 opentrep/basic/UTF8Handler.hpp File Reference . . . 466
24.64 UTF8Handler.hpp . . . 466
24.65 opentrep/basic/Utilities.cpp File Reference . . . 467
24.66 Utilities.cpp . . . 467
24.67 opentrep/basic/Utilities.hpp File Reference . . . 469
24.68 Utilities.hpp . . . 469
24.69 opentrep/batches/opentrep-dbmgr.cpp File Reference . . . 469
24.69.1 Typedef Documentation . . . 470
24.69.2 Function Documentation . . . 470
24.69.3 Variable Documentation . . . 470
24.70 opentrep-dbmgr.cpp . . . 470
24.71 opentrep/batches/opentrep-indexer.cpp File Reference . . . 473

24.71.1 Typedef Documentation . . . 473
24.71.2 Function Documentation . . . 473
24.71.3 Variable Documentation . . . 474
24.72 opentrep-indexer.cpp . . . 474
24.73 opentrep/batches/opentrep-searcher.cpp File Reference . . . 476
24.73.1 Typedef Documentation . . . 477
24.73.2 Function Documentation . . . 477
24.73.3 Variable Documentation . . . 478
24.74 opentrep-searcher.cpp . . . 479
24.75 opentrep/bom/Airline.cpp File Reference . . . 482
24.76 Airline.cpp . . . 482
24.77 opentrep/bom/Airline.hpp File Reference . . . 483
24.78 Airline.hpp . . . 483
24.79 opentrep/bom/BomAbstract.hpp File Reference . . . 483
24.79.1 Function Documentation . . . 483
24.80 BomAbstract.hpp . . . 483
24.81 opentrep/bom/BomJSONExport.cpp File Reference . . . 484
24.82 BomJSONExport.cpp . . . 485
24.83 opentrep/bom/BomJSONExport.hpp File Reference . . . 486
24.84 BomJSONExport.hpp . . . 487
24.85 opentrep/bom/BomType.cpp File Reference . . . 487
24.86 BomType.cpp . . . 487
24.87 opentrep/bom/BomType.hpp File Reference . . . 488
24.88 BomType.hpp . . . 488
24.89 opentrep/bom/Country.cpp File Reference . . . 488
24.90 Country.cpp . . . 488
24.91 opentrep/bom/Country.hpp File Reference . . . 488
24.92 Country.hpp . . . 489
24.93 opentrep/bom/Day.cpp File Reference . . . 489
24.94 Day.cpp . . . 489
24.95 opentrep/bom/Day.hpp File Reference . . . 489
24.96 Day.hpp . . . 489
24.97 opentrep/bom/DocumentList.hpp File Reference . . . 489
24.98 DocumentList.hpp . . . 489
24.99 opentrep/bom/Filter.cpp File Reference . . . 490
24.100 Filter.cpp . . . 490
24.101 opentrep/bom/Filter.hpp File Reference . . . 492
24.102 Filter.hpp . . . 492
24.103 opentrep/bom/GenericBom.hpp File Reference . . . 493
24.104 GenericBom.hpp . . . 493

24.105 opentrep/bom/IATAType.cpp File Reference 493
24.106 IATAType.cpp 494
24.107 opentrep/bom/KeyAbstract.hpp File Reference 496
24.107.1 Function Documentation 496
24.108 KeyAbstract.hpp 497
24.109 opentrep/bom/Language.cpp File Reference 498
24.110 Language.cpp 498
24.111 opentrep/bom/Levenshtein.cpp File Reference 499
24.112 Levenshtein.cpp 499
24.113 opentrep/bom/Levenshtein.hpp File Reference 500
24.114 Levenshtein.hpp 500
24.115 opentrep/bom/LocationExchange.cpp File Reference 501
24.116 LocationExchange.cpp 501
24.117 opentrep/bom/LocationExchange.hpp File Reference 506
24.118 LocationExchange.hpp 506
24.119 opentrep/bom/Month.cpp File Reference 507
24.120 Month.cpp 507
24.121 opentrep/bom/Month.hpp File Reference 507
24.122 Month.hpp 507
24.123 opentrep/bom/Passenger.cpp File Reference 507
24.124 Passenger.cpp 507
24.125 opentrep/bom/Passenger.hpp File Reference 507
24.126 Passenger.hpp 507
24.127 opentrep/bom/Place.cpp File Reference 507
24.128 Place.cpp 507
24.129 opentrep/bom/Place.hpp File Reference 516
24.130 Place.hpp 516
24.131 opentrep/bom/PlaceHolder.cpp File Reference 523
24.132 PlaceHolder.cpp 524
24.133 opentrep/bom/PlaceHolder.hpp File Reference 525
24.134 PlaceHolder.hpp 525
24.135 opentrep/bom/PlaceList.hpp File Reference 526
24.136 PlaceList.hpp 526
24.137 opentrep/bom/PORParserHelper.cpp File Reference 527
24.137.1 Macro Definition Documentation 527
24.138 PORParserHelper.cpp 528
24.139 opentrep/bom/PORParserHelper.hpp File Reference 545
24.140 PORParserHelper.hpp 546
24.141 opentrep/bom/QuerySlices.cpp File Reference 552
24.142 QuerySlices.cpp 552

24.143 opentrep/bom/QuerySlices.hpp File Reference . . . 556
24.144 QuerySlices.hpp . . . 556
24.145 opentrep/bom/Result.cpp File Reference . . . 557
24.146 Result.cpp . . . 558
24.147 opentrep/bom/Result.hpp File Reference . . . 567
24.148 Result.hpp . . . 568
24.149 opentrep/bom/ResultCombination.cpp File Reference . . . 570
24.150 ResultCombination.cpp . . . 571
24.151 opentrep/bom/ResultCombination.hpp File Reference . . . 574
24.152 ResultCombination.hpp . . . 574
24.153 opentrep/bom/ResultHolder.cpp File Reference . . . 575
24.154 ResultHolder.cpp . . . 575
24.155 opentrep/bom/ResultHolder.hpp File Reference . . . 578
24.156 ResultHolder.hpp . . . 578
24.157 opentrep/bom/ResultHolderList.hpp File Reference . . . 579
24.158 ResultHolderList.hpp . . . 580
24.159 opentrep/bom/ResultList.hpp File Reference . . . 580
24.160 ResultList.hpp . . . 580
24.161 opentrep/bom/ScoreBoard.cpp File Reference . . . 580
24.162 ScoreBoard.cpp . . . 581
24.163 opentrep/bom/ScoreBoard.hpp File Reference . . . 583
24.164 ScoreBoard.hpp . . . 583
24.165 opentrep/bom/StringPartition.cpp File Reference . . . 584
24.166 StringPartition.cpp . . . 585
24.167 opentrep/bom/StringPartition.hpp File Reference . . . 587
24.168 StringPartition.hpp . . . 587
24.169 opentrep/bom/StringSet.cpp File Reference . . . 588
24.170 StringSet.cpp . . . 589
24.171 opentrep/bom/StringSet.hpp File Reference . . . 590
24.172 StringSet.hpp . . . 590
24.173 opentrep/bom/WordCombinationHolder.cpp File Reference . . . 591
24.174 WordCombinationHolder.cpp . . . 591
24.175 opentrep/bom/WordCombinationHolder.hpp File Reference . . . 594
24.176 WordCombinationHolder.hpp . . . 594
24.177 opentrep/bom/WordHolder.cpp File Reference . . . 595
24.178 WordHolder.cpp . . . 595
24.179 opentrep/bom/WordHolder.hpp File Reference . . . 596
24.180 WordHolder.hpp . . . 596
24.181 opentrep/bom/World.cpp File Reference . . . 597
24.182 World.cpp . . . 597

24.183 opentrep/bom/World.hpp File Reference . . . 598
24.184 World.hpp . . . 598
24.185 opentrep/command/DBManager.cpp File Reference . . . 599
24.186 DBManager.cpp . . . 600
24.187 opentrep/command/DBManager.hpp File Reference . . . 605
24.188 DBManager.hpp . . . 605
24.189 opentrep/command/DBSessionManager.cpp File Reference . . . 606
24.190 DBSessionManager.cpp . . . 606
24.191 opentrep/command/DBSessionManager.hpp File Reference . . . 607
24.192 DBSessionManager.hpp . . . 607
24.193 opentrep/command/IndexBuilder.cpp File Reference . . . 608
24.194 IndexBuilder.cpp . . . 609
24.195 opentrep/command/IndexBuilder.hpp File Reference . . . 611
24.196 IndexBuilder.hpp . . . 611
24.197 opentrep/command/PORParser.cpp File Reference . . . 612
24.198 PORParser.cpp . . . 612
24.199 opentrep/command/PORParser.hpp File Reference . . . 613
24.200 PORParser.hpp . . . 613
24.201 opentrep/command/RequestInterpreter.cpp File Reference . . . 614
24.202 RequestInterpreter.cpp . . . 614
24.203 opentrep/command/RequestInterpreter.hpp File Reference . . . 618
24.204 RequestInterpreter.hpp . . . 618
24.205 opentrep/command/XapianIndexManager.cpp File Reference . . . 619
24.206 XapianIndexManager.cpp . . . 619
24.207 opentrep/command/XapianIndexManager.hpp File Reference . . . 621
24.208 XapianIndexManager.hpp . . . 621
24.209 opentrep/dbadaptor/DbaAbstract.cpp File Reference . . . 622
24.210 DbaAbstract.cpp . . . 622
24.211 opentrep/dbadaptor/DbaAbstract.hpp File Reference . . . 622
24.211.1 Function Documentation . . . 622
24.212 DbaAbstract.hpp . . . 623
24.213 opentrep/dbadaptor/DbaPlace.cpp File Reference . . . 623
24.214 DbaPlace.cpp . . . 624
24.215 opentrep/dbadaptor/DbaPlace.hpp File Reference . . . 628
24.216 DbaPlace.hpp . . . 628
24.217 opentrep/DBParams.hpp File Reference . . . 629
24.218 DBParams.hpp . . . 629
24.219 opentrep/DistanceErrorRule.hpp File Reference . . . 631
24.220 DistanceErrorRule.hpp . . . 631
24.221 opentrep/factory/FacBomAbstract.cpp File Reference . . . 633

24.224FacBomAbstract.cpp . . . 633
24.225/home/user/dev/git/opentrep/opentrep/factory/FacBomAbstract.hpp File Reference . . . 634
24.226FacBomAbstract.hpp . . . 634
24.227/home/user/dev/git/opentrep/opentrep/factory/FacOpenTrepServiceContext.cpp File Reference . . . 635
24.228FacOpenTrepServiceContext.cpp . . . 635
24.229/home/user/dev/git/opentrep/opentrep/factory/FacOpenTrepServiceContext.hpp File Reference . . . 636
24.230FacOpenTrepServiceContext.hpp . . . 636
24.231/home/user/dev/git/opentrep/opentrep/factory/FacPlace.cpp File Reference . . . 637
24.232FacPlace.cpp . . . 637
24.233/home/user/dev/git/opentrep/opentrep/factory/FacPlace.hpp File Reference . . . 638
24.234FacPlace.hpp . . . 638
24.235/home/user/dev/git/opentrep/opentrep/factory/FacPlaceHolder.cpp File Reference . . . 639
24.236FacPlaceHolder.cpp . . . 639
24.237/home/user/dev/git/opentrep/opentrep/factory/FacPlaceHolder.hpp File Reference . . . 640
24.238FacPlaceHolder.hpp . . . 640
24.239/home/user/dev/git/opentrep/opentrep/factory/FacResult.cpp File Reference . . . 641
24.240FacResult.cpp . . . 641
24.241/home/user/dev/git/opentrep/opentrep/factory/FacResult.hpp File Reference . . . 642
24.242FacResult.hpp . . . 642
24.243/home/user/dev/git/opentrep/opentrep/factory/FacResultCombination.cpp File Reference . . . 643
24.244FacResultCombination.cpp . . . 643
24.245/home/user/dev/git/opentrep/opentrep/factory/FacResultCombination.hpp File Reference . . . 644
24.246FacResultCombination.hpp . . . 644
24.247/home/user/dev/git/opentrep/opentrep/factory/FacResultHolder.cpp File Reference . . . 645
24.248FacResultHolder.cpp . . . 645
24.249/home/user/dev/git/opentrep/opentrep/factory/FacResultHolder.hpp File Reference . . . 646
24.250FacResultHolder.hpp . . . 646
24.251/home/user/dev/git/opentrep/opentrep/factory/FacServiceAbstract.cpp File Reference . . . 647
24.252FacServiceAbstract.cpp . . . 647
24.253/home/user/dev/git/opentrep/opentrep/factory/FacServiceAbstract.hpp File Reference . . . 647
24.254FacServiceAbstract.hpp . . . 648
24.255/home/user/dev/git/opentrep/opentrep/factory/FacSupervisor.cpp File Reference . . . 648
24.256FacSupervisor.cpp . . . 648
24.257/home/user/dev/git/opentrep/opentrep/factory/FacSupervisor.hpp File Reference . . . 649
24.258FacSupervisor.hpp . . . 650
24.259/home/user/dev/git/opentrep/opentrep/factory/FacWorld.cpp File Reference . . . 650
24.260FacWorld.cpp . . . 651
24.261/home/user/dev/git/opentrep/opentrep/factory/FacWorld.hpp File Reference . . . 652
24.262FacWorld.hpp . . . 652
24.263/home/user/dev/git/opentrep/opentrep/IATAType.hpp File Reference . . . 653

24.262 ATAType.hpp . . . 653
24.263 stdair/stdair_basic_types.hpp File Reference . . . 654
24.264 stdair_basic_types.hpp . . . 654
24.265 stdair/stdair_date_time_types.hpp File Reference . . . 655
24.266 stdair_date_time_types.hpp . . . 655
24.267 stdair/stdair_exceptions.hpp File Reference . . . 663
24.268 stdair_exceptions.hpp . . . 664
24.269 stdair/stdair_inventory_types.hpp File Reference . . . 665
24.270 stdair_inventory_types.hpp . . . 665
24.271 stdair/stdair_maths_types.hpp File Reference . . . 665
24.272 stdair_maths_types.hpp . . . 666
24.273 stdair/stdair_service_types.hpp File Reference . . . 667
24.274 stdair_service_types.hpp . . . 667
24.275 stdair/stdair_types.hpp File Reference . . . 668
24.275.1 Function Documentation . . . 669
24.276 stdair_types.hpp . . . 669
24.277 stdair/stdair_ui_types.hpp File Reference . . . 670
24.278 stdair_ui_types.hpp . . . 670
24.279 stdair/stdair_utilities.hpp File Reference . . . 673
24.280 stdair_utilities.hpp . . . 673
24.281 stdair/stdair_version.hpp File Reference . . . 673
24.282 stdair_version.hpp . . . 674
24.283 stdair/STDAIR_Service.hpp File Reference . . . 675
24.284 STDAIR_Service.hpp . . . 675
24.285 stdair/STDAIR_Types.hpp File Reference . . . 676
24.286 STDAIR_Types.hpp . . . 678
24.287 stdair/stdair.hpp File Reference . . . 682
24.288 stdair.hpp . . . 682
24.289 stdair/python/pystdair.cpp File Reference . . . 683
24.289.1 Function Documentation . . . 683
24.290 pystdair.cpp . . . 683
24.291 stdair/service/Logger.cpp File Reference . . . 691
24.292 Logger.cpp . . . 691
24.293 stdair/service/Logger.hpp File Reference . . . 692
24.293.1 Macro Definition Documentation . . . 693
24.294 Logger.hpp . . . 694
24.295 stdair/service/STDAIR_Service.cpp File Reference . . . 695
24.296 STDAIR_Service.cpp . . . 696
24.297 stdair/service/STDAIR_ServiceContext.cpp File Reference . . . 699
24.298 STDAIR_ServiceContext.cpp . . . 700

24.299 opentrep/service/OPENTREP_ServiceContext.hpp File Reference . . . 701
24.300 OPENTREP_ServiceContext.hpp . . . 701
24.301 opentrep/service/ServiceAbstract.cpp File Reference . . . 702
24.302 ServiceAbstract.cpp . . . 702
24.303 opentrep/service/ServiceAbstract.hpp File Reference . . . 703
24.303.1 Function Documentation . . . 703
24.304 ServiceAbstract.hpp . . . 703
24.305 opentrep/service/ServiceUtilities.cpp File Reference . . . 704
24.306 ServiceUtilities.cpp . . . 704
24.307 opentrep/service/ServiceUtilities.hpp File Reference . . . 705
24.308 ServiceUtilities.hpp . . . 705
24.309 test/opentrep/IndexBuildingTestSuite.cpp File Reference . . . 705
24.310 IndexBuildingTestSuite.cpp . . . 705
24.311 test/opentrep/PartitionTestSuite.cpp File Reference . . . 707
24.311.1 Macro Definition Documentation . . . 707
24.311.2 Function Documentation . . . 707
24.312 PartitionTestSuite.cpp . . . 708
24.313 test/opentrep/SearchingTestSuite.cpp File Reference . . . 709
24.314 SearchingTestSuite.cpp . . . 709
24.315 test/opentrep/SliceTestSuite.cpp File Reference . . . 710
24.315.1 Macro Definition Documentation . . . 710
24.315.2 Function Documentation . . . 711
24.316 SliceTestSuite.cpp . . . 711
24.317 test/opentrep/UnicodeTestSuite.cpp File Reference . . . 713
24.317.1 Macro Definition Documentation . . . 713
24.317.2 Function Documentation . . . 713
24.318 UnicodeTestSuite.cpp . . . 714

1 OpenTrep Documentation

1.1 Getting Started

- Main features
- Installation
- Linking with OpenTrep
- Users Guide
- Tutorials
- Copyright and License
- Make a Difference
- Make a new release

- OpenTrep People

1.2 OpenTrep at SourceForge

- Project page
- Download OpenTrep
- Open a ticket fora bug or feature
- Mailing lists
- Forums
  - Discuss about Development issues
  - Ask for Help
  - Discuss OpenTrep

1.3 OpenTrep Development

- Git Repository (Subversion is deprecated)
- Coding Rules
- Documentation Rules
- Test Rules

1.4 External Libraries

- Boost (C++ STL extensions)
- Xapian (information retrieval made easy)
- Python
- MySQL client
- SOCI (C++ DB API)

1.5 Support OpenTrep

1.6 About OpenTrep

OpenTrep aims at providing a clean API, and the corresponding C++ implementation, for parsing travel-focused requests (e.g., "washington dc beijing monday a/r +aa -ua 1 week 2 adults 1 dog"). N

OpenTrep uses Xapian for the Information Retrieval part, on freely available travel-related data (e.g., country names and codes, city names and codes, airline names and codes, etc.).

OpenTrep exposes a simple, clean and object-oriented, API. For instance, the static Parse() method takes, as input, a string containing the travel request, and yields, as output, the list of the recognised terms as well as their corresponding types. As an example, the travel request "washington dc beijing monday a/r +aa -ua 1 week 2 adults 1 dog" would give the following list: Origin airport: Washington, DC, USA Destination airport: Beijing, China Date of travel: next Monday Date of return: 1 week after next Monday Preferred airline: American Airlines; non-preferred airline: United Airlines Number of travellers: 2 adults and a dog

The output can then be used by other systems, for instance to book the corresponding travel or to visualise it on a map and calendar and to share it with others.

OpenTrep makes an extensive use of existing open-source libraries for increased functionality, speed and accuracy. In particular the Boost (C++ Standard Extensions: http://www.boost.org) library is used.

The OpenTrep library originates from the department of Operational Research and Innovation at Amadeus, Sophia Antipolis, France. OpenTrep is released under the terms of the GNU Lesser General Public License (LGPLv2.1) for you to enjoy.

OpenTrep should work on GNU/Linux, Sun Solaris, Microsoft Windows (with Cygwin, MinGW/MSYS, or Microsoft Visual C++ .NET) and Mac OS X operating systems.

Note

(N) - The OpenTrep library is **NOT** intended, in any way, to be used for production systems. If you want to report issue, bug or feature request, or if you just want to give feedback, have a look on the right-hand side of this page for the preferred reporting methods. In any case, please do not contact Amadeus directly for any matter related to OpenTrep.

2 OpenTrep People

2.1 Project Admins

- Denis Arnaud denis_arnaud@users.sourceforge.net (N)
- Charles-Antoine Robelin carobelin@users.sourceforge.net (N)

2.2 Developers

- Patrick Grandjean pgrandjean@users.sourceforge.net (N)

2.3 Retired Developers

- Ngoc-Thach Hoang hoangngocthach@users.sourceforge.net (N)

2.4 Contributors

- Emmanuel Bastien ebastien@users.sourceforge.net (N)

2.5 Distribution Maintainers

- Denis Arnaud denis_arnaud@users.sourceforge.net (N)
- Emmanuel Bastien ebastien@users.sourceforge.net (N)

Note

(N) - Amadeus employees.

3 Coding Rules

In the following sections we describe the naming conventions which are used for files, classes, structures, local variables, and global variables.

3.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

3.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

- `int myFunctionName (const int& a, int b)`

3.3 Default Naming Rules for Classes and Structures

Each new word in a class or structure name should always start with a capital letter and the words should be separated with an under-score. Abbreviations are written with capital letters. Examples:

- `MyClassName`
- `MyStructName`

3.4 Default Naming Rules for Files

Files are named after the C++ class names.

Source files are named using `.cpp` suffix, whereas header files end with `.hpp` extension. Examples:

- `FlightDate.hpp`
- `SegmentDate.cpp`

3.5 Default Functionality of Classes

All classes that are configured by input parameters should include:

- default empty constructor
- one or more additional constructor(s) that takes input parameters and initializes the class instance
- setup function, preferably named `'setup'` or `'set_parameters'`

Explicit destructor functions are not required, unless they are needed. It shall not be possible to use any of the other member functions unless the class has been properly initiated with the input parameters.

4 Copyright and License

4.1 GNU LESSER GENERAL PUBLIC LICENSE

4.1.1 Version 2.1, February 1999


```
Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
```

```
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL.  It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]
```

4.2 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

4.3 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

1. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

```
a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices
stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no
charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a
table of data to be supplied by an application program that uses
the facility, other than as an argument passed when the facility
is invoked, then you must make a good faith effort to ensure that,
in the event an application does not supply such function or
table, the facility still operates, and performs whatever part of
its purpose remains meaningful.

(For example, a function in a library to compute square roots has
```

```
a purpose that is entirely well-defined independent of the
application.  Therefore, Subsection 2d requires that any
application-supplied function or table used by this function must
be optional: if the application does not supply it, the square
root function must still compute square roots.)
```

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

1. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

1. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

1. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

1. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

```
a) Accompany the work with the complete corresponding
machine-readable source code for the Library including whatever
changes were used in the work (which must be distributed under
Sections 1 and 2 above); and, if the work is an executable linked
with the Library, with the complete machine-readable "work that
uses the Library", as object code and/or source code, so that the
user can modify the Library and then relink to produce a modified
executable containing the modified Library.  (It is understood
that the user who changes the contents of definitions files in the
Library will not necessarily be able to recompile the application
to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the
Library.  A suitable mechanism is one that (1) uses at run time a
copy of the library already present on the user's computer system,
rather than copying library functions into the executable, and (2)
will operate properly with a modified version of the library, if
the user installs one, as long as the modified version is
interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at
least three years, to give the same user the materials
specified in Subsection 6a, above, for a charge no more
than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy
from a designated place, offer equivalent access to copy the above
specified materials from the same place.

e) Verify that the user has already received a copy of these
materials or that you have already sent this user a copy.
```

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

1. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   ```
   a) Accompany the combined library with a copy of the same work
   based on the Library, uncombined with any other library
   facilities.  This must be distributed under the terms of the
   Sections above.

   b) Give prominent notice with the combined library of the fact
   that part of it is a work based on the Library, and explaining
   where to find the accompanying uncombined form of the same work.
   ```

2. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and

will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

3. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

4. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

5. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

1. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

2. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

1. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

4.3.1 NO WARRANTY

1. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED

IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

2. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4.3.2 END OF TERMS AND CONDITIONS

4.4 How to Apply These Terms to Your New Programs

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!

Source

5 Documentation Rules

5.1 General Rules

All classes in StdAir should be properly documented with Doxygen comments in include (`.hpp`) files. Source (`.cpp`) files should be documented according to a normal standard for well documented C++ code.

An example of how the interface of a class shall be documented in StdAir is shown here:

```
/*!
 * \brief Brief description of MyClass here
 *
 * Detailed description of MyClass here. With example code if needed.
 */
class MyClass {
public:
  //! Default constructor
  MyClass(void) { setup_done = false; }

  /*!
   * \brief Constructor that initializes the class with parameters
   *
   * Detailed description of the constructor here if needed
   *
   * \param[in] param1 Description of \a param1 here
   * \param[in] param2 Description of \a param2 here
   */
  MyClass(TYPE1 param1, TYPE2 param2) { setup(param1, param2); }

  /*!
   * \brief Setup function for MyClass
   *
   * Detailed description of the setup function here if needed
   *
   * \param[in] param1 Description of \a param1 here
   * \param[in] param2 Description of \a param2 here
   */
  void setup(TYPE1 param1, TYPE2 param2);

  /*!
   * \brief Brief description of memberFunction1
   *
   * Detailed description of memberFunction1 here if needed
   *
   * \param[in]     param1 Description of \a param1 here
   * \param[in]     param2 Description of \a param2 here
   * \param[in,out] param3 Description of \a param3 here
   * \return Description of the return value here
   */
  TYPE4 memberFunction1(TYPE1 param1, TYPE2 param2, TYPE3 &param3);

private:

  bool _setupDone;         /*!< Variable that checks if the class is properly
                                initialized with parameters */
  TYPE1 _privateVariable1; //!< Short description of _privateVariable1 here
  TYPE2 _privateVariable2; //!< Short description of _privateVariable2 here
};
```

5.2 File Header

All files should start with the following header, which include Doxygen's `\file`, `\brief` and `\author` tags, `$Date$` and `$Revisions$` CVS tags, and a common copyright note:

```
/*!
 * \file
 * \brief Brief description of the file here
 * \author Names of the authors who contributed to this code
 * \date Date
 *
 * Detailed description of the file here if needed.
```

```
 *
 * ----------------------------------------------------------------------
 *
 * StdAir - C++ Standard Airline IT Object Library
 *
 * Copyright (C) 2009-2010  (\see authors file for a list of contributors)
 *
 * \see copyright file for license information
 *
 * ----------------------------------------------------------------------
 */
```

5.3 Grouping Various Parts

All functions must be added to a Doxygen group in order to appear in the documentation. The following code example defines the group `'my_group'`:

```
/*!
 * \defgroup my_group Brief description of the group here
 *
 * Detailed description of the group here
*/
```

The following example shows how to document the function `myFunction` and how to add it to the group `my_group`:

```
/*!
 * \brief Brief description of myFunction here
 * \ingroup my_group
 *
 * Detailed description of myFunction here
 *
 * \param[in] param1 Description of \a param1 here
 * \param[in] param2 Description of \a param2 here
 * \return Description of the return value here
 */
TYPE3 myFunction(TYPE1 param1, TYPE2 &param2);
```

6 Main features

A short list of the main features of OpenTrep is given below sorted in different categories. Many more features and functions exist and for these we refer to the reference documentation.

6.1 Standard Airline IT Business Object Model (BOM)

- (Airline) Network-related classes:
  - Network, ReachableUniverse
- (Air) Travel-related classes:
  - TravelSolution, OriginDestination,
- (Airline) Inventory-related classes:
  - Inventory, FlightDate, SegmentDate, SegmentCabin, BookingClass, LegDate, LegCabin, Bucket
- (Airline) Schedule-related classes:
  - FlightPeriod, SegmentPeriod, LegPeriod
- (Simulated) Passenger-related demand classes:

  - DemandStream, BookingRequest
- (Air) Price-related classes:
  - YieldStore

6.2 Architecture of the OpenTrep library

- Separate structure and content classes
- Boost.Fusion

7 Make a Difference

Do not ask what OpenTrep can do for you. Ask what you can do for OpenTrep.

You can help us to develop the OpenTrep library. There are always a lot of things you can do:

- Start using OpenTrep
- Tell your friends about OpenTrep and help them to get started using it
- If you find a bug, report it to us (on the dedicated Sourceforge's Trac Web site). Without your help we can never hope to produce a bug free code.
- Help us to improve the documentation by providing information about documentation bugs
- Answer support requests in the OpenTrep discussion forums on SourceForge. If you know the answer to a question, help others to overcome their OpenTrep problems.
- Help us to improve our algorithms. If you know of a better way (e.g., that is faster or requires less memory) to implement some of our algorithms, then let us know.
- Help to port OpenTrep to new platforms. If you manage to compile OpenTrep on a new platform, then tell how you did it.
- Send your code. If you have a good OpenTrep compatible code, which you can release under the LGPL, and you think it should be included in OpenTrep, then send it to the communauty.
- Become an OpenTrep developer. Send us (see the OpenTrep People page) an e-mail and tell what you can do for OpenTrep.

8 Make a new release

8.1 Introduction

This document describes briefly the recommended procedure of releasing a new version of OpenTrep using a Linux development machine and the SourceForge project site.

The following steps are required to make a release of the distribution package.

8.2 Initialisation

Clone locally the full Git project:

```
cd ~
mkdir -p dev/sim
cd ~/dev/sim
git clone git://opentrep.git.sourceforge.net/gitroot/opentrep/opentrep opentrepgit
cd opentrepgit
git checkout trunk
```

8.3 Branch creation

Create the branch, on your local clone, corresponding to the new release (say, 0.5.0):

```
cd ~/dev/sim/opentrepgit
git checkout trunk
git checkout -b 0.5.0
```

Update the version in the various build system files, replacing 99.99.99 by the correct version number:

```
vi CMakeLists.txt
vi autogen.sh
```

Update the version and add a change-log in the ChangeLog and in the RPM specification files:

```
vi ChangeLog
vi opentrep.spec
```

8.4 Commit and publish the release branch

Commit the new release:

```
cd ~/dev/sim/opentrepgit
git add -A
git commit -m "[Release 0.5.0] Release of version 0.5.0."
git push
```

8.5 Update the change-log in the trunk as well

Update the change-log in the ChangeLog and RPM specification files:

```
cd ~/dev/sim/opentrepgit
git checkout trunk
vi ChangeLog
vi opentrep.spec
```

Commit the change-logs and publish the trunk (main development branch):

```
git commit -m "[Doc] Integrated the change-log of the release 0.5.0."
git push
```

8.6 Create distribution packages

Create the distribution packages using the following command:

```
cd ~/dev/sim/opentrepgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/opentrep-0.5.0 \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
make check && make dist
```

This will configure, compile and check the package. The output packages will be named, for instance, `opentrep-0.5.0.tar.gz` and `opentrep-0.5.0.tar.bz2`.

8.7 Generation the RPM packages

Optionally, generate the RPM package (for instance, for Fedora/RedHat):

```
cd ~/dev/sim/opentrepgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/opentrep-99.99.99 \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
make dist
```

To perform this step, rpm-build, rpmlint and rpmdevtools have to be available on the system.

```
cp opentrep.spec ~/dev/packages/SPECS \
  && cp opentrep-0.5.0.tar.bz2 ~/dev/packages/SOURCES
cd ~/dev/packages/SPECS
rpmbuild -ba opentrep.spec
rpmlint -i ../SPECS/opentrep.spec ../SRPMS/opentrep-0.5.0-1.fc15.src.rpm \
  ../RPMS/noarch/opentrep-* ../RPMS/i686/opentrep-*
```

8.8 Update distributed change log

Update the `NEWS` and `ChangeLog` files with appropriate information, including what has changed since the previous release. Then commit and push the changes into the OpenTrep's Git repository.

8.9 Create the binary package, including the documentation

Create the binary package, which includes HTML and PDF documentation, using the following command:

```
make package
```

The output binary package will be named, for instance, `opentrep-0.5.0-Linux.tar.bz2`. That package contains both the HTML and PDF documentation. The binary package contains also the executables and shared libraries, as well as C++ header files, but all of those do not interest us for now.

8.10 Upload the files to SourceForge

Upload the distribution and documentation packages to the SourceForge server. Check SourceForge help page on uploading software.

8.11 Upload the documentation to SourceForge

In order to update the Web site files, either:

- synchronise them with rsync and SSH:

  ```
  cd ~/dev/sim/opentrepgit
  git checkout 0.5.0
  rsync -aiv doc/html/ doc/latex/refman.pdf joe,opentrep@web.sourceforge.net:htdocs/
  ```

  where `-aiv` options mean:

  - `-a`: archive/mirror mode; equals `-rlptgoD` (no `-H`, `-A`, `-X`)
  - `-v`: increase verbosity
  - `-i`: output a change-summary for all updates
  - Note the trailing slashes (/) at the end of both the source and target directories. It means that the content of the source directory (`doc/html`), rather than the directory itself, has to be copied into the content of the target directory.

- or use the SourceForge Shell service.

8.12 Make a new post

- submit a new entry in the SourceForge project-related news feed
- make a new post on the SourceForge hosted WordPress blog
- and update, if necessary, Trac tickets.

8.13 Send an email on the announcement mailing-list

Finally, you should send an announcement to opentrep-announce@lists.sourceforge.net (see https://lists.sourceforge.net/lists/listinfo/opentrep-announce for the archives)

9 Installation

9.1 Table of Contents

- Fedora/RedHat Linux distributions
- OpenTrep Requirements
- Basic Installation
- Compilers and Options
- Compiling For Multiple Architectures
- Installation Names
- Optional Features
- Particular systems
- Specifying the System Type
- Sharing Defaults
- Defining Variables
- 'cmake' Invocation

9.2 Fedora/RedHat Linux distributions

Note that on Fedora/RedHat Linux distributions, RPM packages are available and can be installed with your usual package manager. For instance:

```
yum -y install opentrep-devel opentrep-doc
```

RPM packages can also be available on the SourceForge download site.

9.3 OpenTrep Requirements

OpenTrep should compile without errors or warnings on most GNU/Linux systems, on UNIX systems like Solaris SunOS, and on POSIX based environments for Microsoft Windows like Cygwin or MinGW with MSYS. It can be also built on Microsoft Windows NT/2000/XP/Vista/7 using Microsoft's Visual C++ .NET, but our support for this compiler is limited. For GNU/Linux, SunOS, Cygwin and MinGW we assume that you have at least the following GNU software installed on your computer:

- GNU Autotools:

  - autoconf,
  - automake,
  - libtool,
  - make, version 3.72.1 or later (check version with 'make –version')

- GCC - GNU C++ Compiler (g++), version 4.3.x or later (check version with 'gcc –version')
- Boost - C++ STL extensions, version 1.35 or later (check version with 'grep "define BOOST_LIB_VERSION" /usr/include/boost/version.hpp')
- MySQL - Database client libraries, version 5.0 or later (check version with 'mysql –version')
- SOCI - C++ database client library wrapper, version 3.0.0 or later (check version with 'soci-config –version')

Optionally, you might need a few additional programs: Doxygen, LaTeX, Dvips and Ghostscript, to generate the HTML and PDF documentation.

We strongly recommend that you use recent stable releases of the GCC, if possible. We do not actively work on supporting older versions of the GCC, and they may therefore (without prior notice) become unsupported in future releases of OpenTrep.

9.4 Basic Installation

Briefly, the shell commands './cmake .. && make install' should configure, build and install this package. The following more-detailed instructions are generic; see the 'README' file for instructions specific to this package. Some packages provide this 'INSTALL' file but do not implement all of the features documented below. The lack of an optional feature in a given package is not necessarily a bug. More recommendations for GNU packages can be found in the info page corresponding to "Makefile Conventions: (standards)Makefile Conventions".

The 'cmake' shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a 'Makefile' in each directory of the package. It may also create one or more '.h' files containing system-dependent definitions. Finally, it creates a 'CMakeCache.txt' cache file that you can refer to in the future to recreate the current configuration, and files 'CMakeFiles' containing compiler output (useful mainly for debugging 'cmake').

It can also use an optional file (typically called 'config.cache' and enabled with '–cache-file=config.cache' or simply '-C') that saves the results of its tests to speed up reconfiguring. Caching is disabled by default to prevent problems with accidental use of stale cache files.

If you need to do unusual things to compile the package, please try to figure out how 'configure' could check whether to do them, and mail diffs or instructions to the address given in the 'README' so they can be considered for the next release. If you are using the cache, and at some point 'config.cache' contains results you don't want to keep, you may remove or edit it.

The file 'CMakeLists.txt' is used to create the 'Makefile' files.

The simplest way to compile this package is:

1. 'cd' to the directory containing the package's source code and type './cmake ..' to configure the package for your system. Running 'cmake' is generally fast. While running, it prints some messages telling which features it is checking for.
2. Type 'make' to compile the package.
3. Optionally, type 'make check'to run any self-tests that come with the package, generally using the just-built uninstalled binaries.
4. Type 'make install' to install the programs and any data files and documentation. When installing into a prefix owned by root, it is recommended that the package be configured and built as a regular user, and only the 'make install' phase executed with root privileges.

5. You can remove the program binaries and object files from the source code directory by typing ‘make clean’. To also remove the files that ‘configure’ created (so you can compile the package for a different kind of computer), type ‘make distclean’. There is also a ‘make maintainer-clean’ target, but that is intended mainly for the package’s developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.

6. Often, you can also type ‘make uninstall’ to remove the installed files again. In practice, not all packages have tested that uninstallation works correctly, even though it is required by the GNU Coding Standards.

9.5 Compilers and Options

Some systems require unusual options for compilation or linking that the ‘cmake’ script does not know about. Run ‘./cmake -help’ for details on some of the pertinent environment variables.

You can give ‘cmake’ initial values for configuration parameters by setting variables in the command line or in the environment. Here is an example:

```
./cmake CC=c99 CFLAGS=-g LIBS=-lposix
```

See Also

Defining Variables for more details.

9.6 Compiling For Multiple Architectures

You can compile the package for more than one kind of computer at the same time, by placing the object files for each architecture in their own directory. To do this, you can use GNU ‘make’. ‘cd’ to the directory where you want the object files and executables to go and run the ‘configure’ script. ‘configure’ automatically checks for the source code in the directory that ‘configure’ is in and in ‘..’. This is known as a "VPATH" build.

With a non-GNU ‘make’, it is safer to compile the package for one architecture at a time in the source code directory. After you have installed the package for one architecture, use ‘make distclean’ before reconfiguring for another architecture.

On MacOS X 10.5 and later systems, you can create libraries and executables that work on multiple system types-known as "fat" or "universal" binaries-by specifying multiple ‘-arch’ options to the compiler but only a single ‘-arch’ option to the preprocessor. Like this:

```
./configure CC="gcc -arch i386 -arch x86_64 -arch ppc -arch ppc64" \
            CXX="g++ -arch i386 -arch x86_64 -arch ppc -arch ppc64" \
            CPP="gcc -E" CXXCPP="g++ -E"
```

This is not guaranteed to produce working output in all cases, you may have to build one architecture at a time and combine the results using the ‘lipo’ tool if you have problems.

9.7 Installation Names

By default, ‘make install’ installs the package’s commands under ‘/usr/local/bin’, include files under ‘/usr/local/include’, etc. You can specify an installation

prefix other than '/usr/local' by giving 'configure' the option '-prefix=P-
REFIX', where PREFIX must be an absolute file name.

You can specify separate installation prefixes for architecture-specific
files and architecture-independent files. If you pass the option '-exec-prefix=P-
REFIX' to 'configure', the package uses PREFIX as the prefix for installing
programs and libraries. Documentation and other data files still use the
regular prefix.

In addition, if you use an unusual directory layout you can give options
like '-bindir=DIR' to specify different values for particular kinds of files.
Run 'configure -help' for a list of the directories you can set and what
kinds of files go in them. In general, the default for these options is
expressed in terms of '${prefix}', so that specifying just '-prefix' will
affect all of the other directory specifications that were not explicitly
provided.

The most portable way to affect installation locations is to pass the correct
locations to 'configure'; however, many packages provide one or both of the
following shortcuts of passing variable assignments to the 'make install'
command line to change installation locations without having to reconfigure
or recompile.

The first method involves providing an override variable for each affected
directory. For example, 'make install prefix=/alternate/directory' will
choose an alternate location for all directory configuration variables that
were expressed in terms of '${prefix}'. Any directories that were specified
during 'configure', but not in terms of '${prefix}', must each be overridden
at install time for the entire installation to be relocated. The approach
of makefile variable overrides for each directory variable is required by
the GNU Coding Standards, and ideally causes no recompilation. However,
some platforms have known limitations with the semantics of shared libraries
that end up requiring recompilation when using this method, particularly
noticeable in packages that use GNU Libtool.

The second method involves providing the 'DESTDIR' variable. For example,
'make install DESTDIR=/alternate/directory' will prepend '/alternate/directory'
before all installation names. The approach of 'DESTDIR' overrides is not
required by the GNU Coding Standards, and does not work on platforms that
have drive letters. On the other hand, it does better at avoiding recompilation
issues, and works well even when some directory options were not specified
in terms of '${prefix}' at 'configure' time.

9.8 Optional Features

If the package supports it, you can cause programs to be installed with an
extra prefix or suffix on their names by giving 'cmake' the option '-program-prefix=P-
REFIX' or '-program-suffix=SUFFIX'.

Some packages pay attention to '-enable-FEATURE' options to 'configure',
where FEATURE indicates an optional part of the package. They may also
pay attention to '-with-PACKAGE' options, where PACKAGE is something like
'gnu-as' or 'x' (for the X Window System). The 'README' should mention any
'-enable-' and '-with-' options that the package recognizes.

For packages that use the X Window System, 'configure' can usually find the
X include and library files automatically, but if it doesn't, you can use
the 'configure' options '-x-includes=DIR' and '-x-libraries=DIR' to specify
their locations.

Some packages offer the ability to configure how verbose the execution of
'make' will be. For these packages, running './configure -enable-silent-rules'

sets the default to minimal output, which can be overridden with ‘make V=1’; while running ‘./configure –disable-silent-rules’ sets the default to verbose, which can be overridden with ‘make V=0’.

9.9 Particular systems

On HP-UX, the default C compiler is not ANSI C compatible. If GNU CC is not installed, it is recommended to use the following options in order to use an ANSI C compiler:

```
./configure CC="cc -Ae -D_XOPEN_SOURCE=500"
```

and if that doesn’t work, install pre-built binaries of GCC for HP-UX.

On OSF/1 a.k.a. Tru64, some versions of the default C compiler cannot parse its ‘<wchar.h>’ header file. The option ‘-nodtk’ can be used as a workaround. If GNU CC is not installed, it is therefore recommended to try

```
./configure CC="cc"
```

and if that doesn’t work, try

```
./configure CC="cc -nodtk"
```

On Solaris, don’t put ‘/usr/ucb’ early in your ‘PATH’. This directory contains several dysfunctional programs; working variants of these programs are available in ‘/usr/bin’. So, if you need ‘/usr/ucb’ in your ‘PATH’, put it *after* ‘/usr/bin’.

On Haiku, software installed for all users goes in ‘/boot/common’, not ‘/usr/local’. It is recommended to use the following options:

```
./cmake -DCMAKE_INSTALL_PREFIX=/boot/common
```

9.10 Specifying the System Type

There may be some features ‘configure’ cannot figure out automatically, but needs to determine by the type of machine the package will run on. Usually, assuming the package is built to be run on the *same* architectures, ‘configure’ can figure that out, but if it prints a message saying it cannot guess the machine type, give it the ‘–build=TYPE’ option. TYPE can either be a short name for the system type, such as ‘sun4’, or a canonical name which has the form CPU-COMPANY-SYSTEM

where SYSTEM can have one of these forms:

- OS
- KERNEL-OS

  See the file ‘config.sub’ for the possible values of each field. If ‘config.sub’ isn’t included in this package, then this package doesn’t need to know the machine type.

  If you are *building* compiler tools for cross-compiling, you should use the option ‘–target=TYPE’ to select the type of system they will produce code for.

  If you want to *use* a cross compiler, that generates code for a platform different from the build platform, you should specify the "host" platform (i.e., that on which the generated programs will eventually be run) with ‘–host=TYPE’.

9.11 Sharing Defaults

If you want to set default values for 'configure' scripts to share, you can create a site shell script called 'config.site' that gives default values for variables like 'CC', 'cache_file', and 'prefix'. 'configure' looks for 'PREFIX/share/config.site' if it exists, then 'PREFIX/etc/config.site' if it exists. Or, you can set the 'CONFIG_SITE' environment variable to the location of the site script. A warning: not all 'configure' scripts look for a site script.

9.12 Defining Variables

Variables not defined in a site shell script can be set in the environment passed to 'configure'. However, some packages may run configure again during the build, and the customized values of these variables may be lost. In order to avoid this problem, you should set them in the 'configure' command line, using 'VAR=value'. For example:

```
./configure CC=/usr/local2/bin/gcc
```

causes the specified 'gcc' to be used as the C compiler (unless it is overridden in the site shell script).

Unfortunately, this technique does not work for 'CONFIG_SHELL' due to an Autoconf bug. Until the bug is fixed you can use this workaround:

```
CONFIG_SHELL=/bin/bash /bin/bash ./configure CONFIG_SHELL=/bin/bash
```

9.13 'cmake' Invocation

'cmake' recognizes the following options to control how it operates.

- '-help', '-h' print a summary of all of the options to 'configure', and exit.
- '-help=short', '-help=recursive' print a summary of the options unique to this package's 'configure', and exit. The 'short' variant lists options used only in the top level, while the 'recursive' variant lists options also present in any nested packages.
- '-version', '-V' print the version of Autoconf used to generate the 'configure' script, and exit.
- '-cache-file=FILE' enable the cache: use and save the results of the tests in FILE, traditionally 'config.cache'. FILE defaults to '/dev/null' to disable caching.
- '-config-cache', '-C' alias for '-cache-file=config.cache'.
- '-quiet', '-silent', '-q' do not print messages saying which checks are being made. To suppress all normal output, redirect it to '/dev/null' (any error messages will still be shown).
- '-srcdir=DIR' look for the package's source code in directory DIR. Usually 'configure' can determine that directory automatically.
- '-prefix=DIR' use DIR as the installation prefix.

See Also

Installation Names for more details, including other options available for fine-tuning the installation locations.

- ‘-no-create’, ‘-n’ run the configure checks, but stop before creating any output files.

‘cmake’ also accepts some other, not widely useful, options. Run ‘cmake -help’ for more details.

The ‘cmake’ script produces an ouput like this:

```
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/opentrep-0.50.0 \
 -DLIB_SUFFIX=64 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/lib64/ccache/gcc
-- Check for working C compiler: /usr/lib64/ccache/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/lib64/ccache/c++
-- Check for working CXX compiler: /usr/lib64/ccache/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Requires Git without specifying any version
-- Current Git revision name: e8beb4d11ff9b1af6b3f3e9ff1e92250aee0291a trunk
-- Requires PythonLibs-2.7
-- Found PythonLibs: /usr/lib64/libpython2.7.so (Required is at least version "2.7")
-- Found PythonLibs 2.7
-- Requires Boost-1.41
-- Boost version: 1.46.0
-- Found the following Boost libraries:
--   program_options
--   date_time
--   iostreams
--   serialization
--   filesystem
--   unit_test_framework
--   python
-- Found Boost version: 1.46.0
-- Found BoostWrapper: /usr/include (Required is at least version "1.41")
-- Requires ZeroMQ-2.0
-- Found ZeroMQ: /usr/lib64/libzmq.so (Required is at least version "2.0")
-- Found ZeroMQ version: 2.1
-- Requires MySQL-5.1
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL: /usr/lib64/mysql/libmysqlclient.so (Required is at least version "5.1")
-- Found MySQL version: 5.5.14
-- Requires SOCI-3.0
-- Using soci-config: /usr/bin/soci-config
-- SOCI headers are buried
-- Found SOCI: /usr/lib64/libsoci_core.so (Required is at least version "3.0")
-- Found SOCIMySQL: /usr/lib64/libsoci_mysql.so (Required is at least version "3.0")
-- Found SOCI with MySQL back-end support version: 3.0.0
-- Requires Doxygen-1.7
-- Found Doxygen: /usr/bin/doxygen
-- Found DoxygenWrapper: /usr/bin/doxygen (Required is at least version "1.7")
-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'opentreplib' to CXX
-- Had to set the linker language for 'opentrepuicllib' to CXX
-- Test 'OpenTrepTest' to be built with 'MPBomRoot.cpp;MPInventory.cpp;StandardAirlineITTestSuite.cpp'
--
-- ===========================================================
-- ----------------------------------
-- ---     Project Information    ---
-- ----------------------------------
-- PROJECT_NAME ................... : opentrep
-- PACKAGE_PRETTY_NAME ............ : OpenTrep
-- PACKAGE ........................ : opentrep
-- PACKAGE_NAME ................... : OPENTREP
-- PACKAGE_VERSION ................ : 0.50.0
```

```
-- GENERIC_LIB_VERSION ..... : 0.50.0
-- GENERIC_LIB_SOVERSION ..... : 99.99
--
-- ----------------------------------
-- ---     Build Configuration    ---
-- ----------------------------------
-- Modules to build ..... : opentrep
-- Libraries to build/install ..... : opentreplib;opentrepuicllib
-- Binaries to build/install ..... : opentrep
-- Modules to test ..... : opentrep
-- Binaries to test ..... : OpenTrepTesttst
--
-- * Module ..... : opentrep
--   + Layers to build ..... : .;basic;bom;factory;dbadaptor;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : opentreplib;opentrepuicllib
--   + Executables to build/install : opentrep
--   + Tests to perform ..... : OpenTrepTesttst
--
-- BUILD_SHARED_LIBS ..... : ON
-- CMAKE_BUILD_TYPE ..... : Debug
--  * CMAKE_C_FLAGS ..... :
--  * CMAKE_CXX_FLAGS ..... : -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protecto
--  * BUILD_FLAGS ..... :
--  * COMPILE_FLAGS ..... :
-- CMAKE_MODULE_PATH ..... : /home/user/dev/sim/opentrep/opentrepgithub/config/
-- CMAKE_INSTALL_PREFIX ..... : /home/user/dev/deliveries/opentrep-0.50.0
--
-- * Doxygen:
--   - DOXYGEN_VERSION ..... : 1.7.4
--   - DOXYGEN_EXECUTABLE ..... : /usr/bin/doxygen
--   - DOXYGEN_DOT_EXECUTABLE ..... : /usr/bin/dot
--   - DOXYGEN_DOT_PATH ..... : /usr/bin
--
-- ----------------------------------
-- --- Installation Configuration ---
-- ----------------------------------
-- INSTALL_LIB_DIR ..... : /home/user/dev/deliveries/opentrep-0.50.0/lib64
-- INSTALL_BIN_DIR ..... : /home/user/dev/deliveries/opentrep-0.50.0/bin
-- INSTALL_INCLUDE_DIR ..... : /home/user/dev/deliveries/opentrep-0.50.0/include
-- INSTALL_DATA_DIR ..... : /home/user/dev/deliveries/opentrep-0.50.0/share
-- INSTALL_SAMPLE_DIR ..... : /home/user/dev/deliveries/opentrep-0.50.0/share/opentrep/samples
-- INSTALL_DOC ..... : ON
--
-- ----------------------------------
-- ---   Packaging Configuration  ---
-- ----------------------------------
-- CPACK_PACKAGE_CONTACT ..... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot net>
-- CPACK_PACKAGE_VENDOR ..... : Denis Arnaud
-- CPACK_PACKAGE_VERSION ..... : 0.50.0
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/user/dev/sim/opentrep/opentrepgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/user/dev/sim/opentrep/opentrepgithub/COPYING
-- CPACK_GENERATOR ..... : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ..... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : opentrep-0.50.0
--
-- ----------------------------------
-- ---     External libraries    ---
-- ----------------------------------
--
-- * Python:
--   - PYTHONLIBS_VERSION ..... : 2.7
--   - PYTHON_LIBRARIES ..... : /usr/lib64/libpython2.7.so
--   - PYTHON_INCLUDE_PATH ..... : /usr/include/python2.7
--   - PYTHON_INCLUDE_DIRS ..... : /usr/include/python2.7
--   - PYTHON_DEBUG_LIBRARIES ..... :
--   - Python_ADDITIONAL_VERSIONS . :
--
-- * ZeroMQ:
--   - ZeroMQ_VERSION ..... : 2.1
--   - ZeroMQ_LIBRARIES ..... : /usr/lib64/libzmq.so
--   - ZeroMQ_INCLUDE_DIR ..... : /usr/include
```

```
--
-- * Boost:
--   - Boost_VERSION ..... : 104600
--   - Boost_LIB_VERSION ..... : 1_46
--   - Boost_HUMAN_VERSION ..... : 1.46.0
--   - Boost_INCLUDE_DIRS ..... : /usr/include
--   - Boost required components .. : program_options;date_time;iostreams;serialization;filesystem;unit_test_f
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_iostreams-mt.so;debug;/usr/lib64/libboost_
--
-- * MySQL:
--   - MYSQL_VERSION ..... : 5.5.14
--   - MYSQL_INCLUDE_DIR ..... : /usr/include/mysql
--   - MYSQL_LIBRARIES ..... : /usr/lib64/mysql/libmysqlclient.so
--
-- * SOCI:
--   - SOCI_VERSION ..... : 3.0.0
--   - SOCI_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCIMYSQL_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_LIBRARIES ..... : /usr/lib64/libsoci_core.so
--   - SOCIMYSQL_LIBRARIES ..... : /usr/lib64/libsoci_mysql.so
--
-- Change a value with: cmake -D<Variable>=<Value>
-- ------------------------------------------------------------
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/dev/sim/opentrep/opentrepgithub/build
```

It is recommended that you check if your library has been compiled and linked properly and works as expected. To do so, you should execute the testing process 'make check'. As a result, you should obtain a similar report:

```
[  0%] Built target hdr_cfg_opentrep
[ 97%] Built target opentreplib
[100%] Built target OpenTrepTesttst
Scanning dependencies of target check_opentreptst
Test project /home/user/dev/sim/opentrep/opentrepgithub/build/test/opentrep
    Start 1: OpenTrepTesttst
1/1 Test #1: OpenTrepTesttst ...................   Passed    0.02 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =   0.27 sec
[100%] Built target check_opentreptst
Scanning dependencies of target check
[100%] Built target check
```

Check if all the executed tests PASSED. If not, please contact us by filling a bug-report.

Finally, you should install the compiled and linked library, include files and (optionally) HTML and PDF documentation by typing:

```
make install
```

Depending on the PREFIX settings during configuration, you might need the root (administrator) access to perform this step.

Eventually, you might invoke the following command

```
make clean
```

to remove all files created during compilation process, or even

```
cd ~/dev/sim/opentrepgit
rm -rf build && mkdir build
cd build
```

to remove everything.

10 Linking with OpenTrep

10.1 Table of Contents

- Introdution
- Using the pkg-config command
- Using the opentrep-config script
- M4 macro for the GNU Autotools
- Using OpenTrep with dynamic linking

10.2 Introdution

There are two convenient methods of linking your programs with the OpenTrep library. The first one employs the 'pkg-config' command (see http://pkgconfig.freedesktop.org/), whereas the second one uses 'opentrep-config' script. These methods are shortly described below.

10.3 Using the pkg-config command

'pkg-config' is a helper tool used when compiling applications and libraries. It helps you insert the correct compiler and linker options. The syntax of the 'pkg-config' is as follows:

```
pkg-config <options> <library_name>
```

For instance, assuming that you need to compile an OpenTrep based program 'my_prog.cpp', you should use the following command:

```
g++ `pkg-config --cflags opentrep` -o my_prog my_prog.cpp \
 `pkg-config --libs opentrep`
```

For more information see the 'pkg-config' man pages.

10.4 Using the opentrep-config script

OpenTrep provides a shell script called opentrep-config, which is installed by default in '$prefix/bin' ('/usr/local/bin') directory. It can be used to simplify compilation and linking of OpenTrep based programs. The usage of this script is quite similar to the usage of the 'pkg-config' command.

Assuming that you need to compile the program 'my_prog.cpp' you can now do that with the following command:

```
g++ `opentrep-config --cflags` -o my_prog my_prog.cpp `opentrep-config --libs`
```

A list of 'opentrep-config' options can be obtained by typing:

```
opentrep-config --help
```

If the 'opentrep-config' command is not found by your shell, you should add its location '$prefix/bin' to the PATH environment variable, e.g.:

```
export PATH=/usr/local/bin:$PATH
```

10.5 M4 macro for the GNU Autotools

A M4 macro file is delivered with OpenTrep, namely 'opentrep.m4', which can be found in, e.g., '/usr/share/aclocal'. When used by a 'configure' script, thanks to he 'AM_PATH_OPENTREP' macro (specified in the M4 macro file), the following Makefile variables are then defined:

- 'OPENTREP_VERSION' (e.g., defined to 0.2.0)
- 'OPENTREP_CFLAGS' (e.g., defined to '-I${prefix}/include')
- 'OPENTREP_LIBS' (e.g., defined to '-L${prefix}/lib -lopentrep')

10.6 Using OpenTrep with dynamic linking

When using static linking some of the library routines in OpenTrep are copied into your executable program. This can lead to unnecessary large executables. To avoid having too large executable files you may use dynamic linking instead. Dynamic linking means that the actual linking is performed when the program is executed. This requires that the system is able to locate the shared OpenTrep library file during your program execution. If you install the OpenTrep library using a non-standard prefix, the 'LD_LIBRARY_PATH' environment variable might be used to inform the linker of the dynamic library location, e.g.:

```
export LD_LIBRARY_PATH=<OpenTrep installation prefix>/lib:$LD_LIBRARY_PATH
```

11 Test Rules

This section describes how the functionality of the OpenTrep library should be verified. In the 'test/opentrep' subdirectory, test source files are provided. All functionality should be tested using these test source files.

11.1 The Test Source Files

Each new OpenTrep module/class should be accompanied with a test source file. The test source file is an implementation in C++ that tests the functionality of a function/class or a group of functions/classes called test suites. The test source file should test relevant parameter settings and input/output relations to guarantee correct functionality of the corresponding classes/functions. The test source files should be maintained using version control and updated whenever new functionality is added to the OpenTrep library.

The test source file should print relevant data to a standard output that can be used to verify the functionality. All relevant parameter settings should be tested.

The test source file should be placed in the 'test/opentrep' subdirectory and should have a name ending with 'TestSuite.cpp'.

11.2 The Reference File

Consider a test source file named 'YieldTestSuite.cpp'. A reference file named 'YieldTestSuite.ref' should accompany the test source file. The reference file contains a reference printout of the standard output generated when running the test program. The reference file should be maintained using version control and updated according to the test source file.

11.3 Testing OpenTrep Library

One can compile and execute all test programs from the 'test/opentrep' sub-directory by typing:

```
% make check
```

after successful compilation of the OpenTrep library.

12 Users Guide

12.1 Table of Contents

- Introduction
- Get Started
  - Get the OpenTrep library
  - Build the OpenTrep project
  - Build and Run the Tests
  - Install the OpenTrep Project (Binaries, Documentation)
- Exploring the Predefined BOM Tree
  - Airline Distribution BOM Tree
  - Airline Network BOM Tree
  - Airline Inventory BOM Tree
- Extending the BOM Tree

12.2 Introduction

The `OpenTrep` library contains classes for airline business management. This document does not cover all the aspects of the `OpenTrep` library. It does however explain the most important things you need to know in order to start using `OpenTrep`.

12.3 Get Started

12.3.1 Get the OpenTrep library

12.3.2 Build the OpenTrep project

To run the configuration script the first time, go to the top directory (where the OpenTrep package has been unpacked), and issue either of the following two commands, depending on whether the OpenTrep project has been checked out from the Subversion repository or downloaded as a tar-ball package from the Sourceforge Web site:

- `./autogen.sh`
- `./configure`

12.3.3 Build and Run the Tests

12.3.4 Install the OpenTrep Project (Binaries, Documentation)

12.4 Exploring the Predefined BOM Tree

`OpenTrep` predefines a BOM (Business Object Model) tree specific to the airline IT arena.

12.4.1 Airline Distribution BOM Tree

- `opentrep::TravelSolutionStruct`

12.4.2 Airline Network BOM Tree

- `opentrep::FlightPeriod`

12.4.3 Airline Inventory BOM Tree

- `opentrep::Inventory`
- `opentrep::FlightDate`

12.4.3.1 Airline Inventory Marketing BOM Tree

- `opentrep::SegmentDate`
- `opentrep::SegmentCabin`
- `opentrep::FareFamily`
- `opentrep::BookingClass`

12.4.3.2 Airline Inventory Operating BOM Tree

- `opentrep::LegDate`
- `opentrep::LegCabin`
- `opentrep::Bucket`

12.5 Extending the BOM Tree

13 Supported Systems

13.1 Table of Contents

- Introdution
- OpenTrep 3.10.x
  - Linux Systems
    - Fedora Core 4 with ATLAS
    - Gentoo Linux with ACML
    - Gentoo Linux with ATLAS
    - Gentoo Linux with MKL
    - Gentoo Linux with NetLib's BLAS and LAPACK
    - Red Hat Enterprise Linux with OpenTrep External
    - SUSE Linux 10.0 with NetLib's BLAS and LAPACK
    - SUSE Linux 10.0 with MKL
  - Windows Systems
    - Microsoft Windows XP with Cygwin
    - Microsoft Windows XP with Cygwin and ATLAS
    - Microsoft Windows XP with Cygwin and ACML
    - Microsoft Windows XP with MinGW, MSYS and ACML
    - Microsoft Windows XP with MinGW, MSYS and OpenTrep External
    - Microsoft Windows XP with MS Visual C++ and Intel MKL
  - Unix Systems

        * SunOS 5.9 with OpenTrep External

- OpenTrep 3.9.1
- OpenTrep 3.9.0
- OpenTrep 3.8.1

13.2 Introdution

This page is intended to provide a list of OpenTrep supported systems, i.e. the systems on which configuration, installation and testing process of the OpenTrep library has been sucessful. Results are grouped based on minor release number. Therefore, only the latest tests for bug-fix releases are included. Besides, the information on this page is divided into sections dependent on the operating system.

Where necessary, some extra information is given for each tested configuration, e.g. external libraries installed, configuration commands used, etc.

If you manage to compile, install and test the OpenTrep library on a system not mentioned below, please let us know, so we could update this database.

13.3 OpenTrep 3.10.x

13.3.1 Linux Systems

13.3.1.1 Fedora Core 4 with ATLAS

- **Platform**: Intel Pentium 4
- **Operating System**: Fedora Core 4 (x86)
- **Compiler**: g++ (GCC) 4.0.2 20051125
- **OpenTrep release**: 3.10.0
- **External Libraries**: From FC4 distribution:
  - `fftw3.i386-3.0.1-3`
  - `fftw3-devel.i386-3.0.1-3`
  - `atlas-sse2.i386-3.6.0-8.fc4`
  - `atlas-sse2-devel.i386-3.6.0-8.fc4`
  - `blas.i386-3.0-35.fc4`
  - `lapack.i386-3.0-35.fc4`
- **Tests Status**: All tests PASSED
- **Comments**: OpenTrep configured with:

```
% CXXFLAGS="-O3 -pipe -march=pentium4" ./configure
```

- **Date**: March 7, 2006
- **Tester**: Tony Ottosson

13.3.1.2 Gentoo Linux with ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Gentoo Linux 2006.0 (x86 arch)
- **Compiler(s)**: g++ (GCC) 3.4.5
- **OpenTrep release**: 3.10.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/acml-3.0.0`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ACML
% eselect lapack set ACML
```

  OpenTrep configured with:

```
% export CPPFLAGS="-I/usr/include/acml"
% ./configure --with-blas="-lblas"
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.1.3 Gentoo Linux with ATLAS

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86)
- **Compiler**: g++ (GCC) 3.4.5
- **OpenTrep release**: 3.10.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/fftw-3.1`
  - `sci-libs/blas-atlas-3.6.0-r1`
  - `sci-libs/lapack-atlas-3.6.0`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ATLAS
% eselect lapack set ATLAS
```

  OpenTrep configured with:

```
% ./configure --with-blas="-lblas"
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.1.4 Gentoo Linux with MKL

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86 arch)
- **Compiler**: g++ (GCC) 3.4.5
- **OpenTrep release**: 3.10.0
- **External Libraries**: Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: `/opt/intel/mkl/8.0.1`
- **Tests Status**: All tests PASSED
- **Comments**: OpenTrep configured using the following commands:

```
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/32"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```

- **Date**: February 28, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.1.5 Gentoo Linux with NetLib's BLAS and LAPACK

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86)
- **Compiler**: g++ (GCC) 3.4.5
- **OpenTrep release**: 3.10.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/fftw-3.1`
  - `sci-libs/blas-reference-19940131-r2`
  - `sci-libs/cblas-reference-20030223`
  - `sci-libs/lapack-reference-3.0-r2`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

```
% blas-config reference
% lapack-config reference
```

OpenTrep configured with:

```
% ./configure --with-blas="-lblas"
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.1.6 Red Hat Enterprise Linux with OpenTrep External

- **Platform**: Intel Pentium 4
- **Operating System**: Red Hat Enterprise Linux AS release 4 (Nahant Update 2)
- **Compiler**: g++ (GCC) 3.4.4 20050721 (Red Hat 3.4.4-2)
- **OpenTrep release**: 3.10.0
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from OpenTrep External 2.1.1 package
- **Tests Status**: All tests PASSED
- **Date**: March 7, 2006
- **Tester**: Erik G. Larsson

13.3.1.7 SUSE Linux 10.0 with NetLib's BLAS and LAPACK

- **Platform**: Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System**: SUSE Linux 10.0 (x86_64)
- **Compiler(s)**: g++ (GCC) 4.0.2
- **OpenTrep release**: 3.10.0
- **External Libraries**: BLAS, LAPACK and FFTW libraries installed from OpenSuse 10.0 RPM repository:
  - `blas-3.0-926`
  - `lapack-3.0-926`
  - `fftw3-3.0.1-114`
  - `fftw3-threads-3.0.1-114`
  - `fftw3-devel-3.0.1-114`
- **Tests Status**: All tests PASSED
- **Comments**: OpenTrep configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% ./configure --with-lapack="/usr/lib64/liblapack.so.3"
```

- **Date**: March 1, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.1.8 SUSE Linux 10.0 with MKL

- **Platform**: Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System**: SUSE Linux 10.0 (x86_64)
- **Compiler(s)**: g++ (GCC) 4.0.2
- **OpenTrep release**: 3.10.0
- **External Libraries**: Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: `/opt/intel/mkl/8.0.1`
- **Tests Status**: All tests PASSED
- **Comments**: OpenTrep configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/em64t"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```

- **Date**: March 1, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.2 Windows Systems

13.3.2.1 Microsoft Windows XP with Cygwin

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **OpenTrep release**: 3.10.1
- **External Libraries**: Installed from Cygwin's repository:
  - `fftw-3.0.1-2`
  - `fftw-dev-3.0.1-1`
  - `lapack-3.0-4`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. OpenTrep configured with:

```
% ./configure
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.2.2 Microsoft Windows XP with Cygwin and ATLAS

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **OpenTrep release**: 3.10.1
- **External Libraries**: Installed from Cygwin's repository:
  - `fftw-3.0.1-2`
  - `fftw-dev-3.0.1-1`

  ATLAS BLAS and LAPACK libraries from OpenTrep External 2.1.1 package configured using:

```
% ./configure --enable-atlas --disable-fftw
```

- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. OpenTrep configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% ./configure
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.2.3 Microsoft Windows XP with Cygwin and ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **OpenTrep release**: 3.10.2
- **External Libraries**: ACML version 3.1.0 (`acml3.1.0-32-win32-g77.exe`) installed into a default directory, i.e. `"c:\Program Files\AMD\acml3.1.0"`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. OpenTrep configured with:

```
% export LDFLAGS="-L/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

- **Date**: May 15, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.2.4 Microsoft Windows XP with MinGW, MSYS and ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s)**: g++ (GCC) 3.4.4 (mingw special)
- **OpenTrep release**: 3.10.2
- **External Libraries**: ACML version 3.1.0 (`acml3.1.0-32-win32-g77.exe`) installed into a default directory, i.e. `"c:\Program Files\AMD\acml3.1.0"`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. OpenTrep configured with:

```
% export LDFLAGS="-L/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

- **Date**: May 15, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.2.5 Microsoft Windows XP with MinGW, MSYS and OpenTrep External

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s)**: g++ (GCC) 3.4.4 (mingw special)
- **OpenTrep release**: 3.10.5
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from OpenTrep External 2.2.0 package
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. OpenTrep configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-Wall -O3 -march=athlon-tbird -pipe"
% ./configure --disable-html-doc
```

- **Date**: August 11, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.2.6 Microsoft Windows XP with MS Visual C++ and Intel MKL

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2
- **Compiler(s)**: Microsoft Visual C++ 2005 .NET
- **OpenTrep release**: 3.10.5
- **External Libraries**: Intel Math Kernel Library (MKL) 8.1 installed manually in the following directory: "`C:\-Program Files\Intel\MKL\8.1`"
- **Tests Status**: Not fully tested. Some OpenTrep based programs compiled and run with success.
- **Comments**: Only static library can be built. OpenTrep built by opening the "`win32\opentrep.vcproj`" project file in MSVC++ and executing "`Build -> Build Solution`" command from menu.
- **Date**: August 11, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.3 Unix Systems

13.3.3.1 SunOS 5.9 with OpenTrep External

- **Platform**: SUNW, Sun-Blade-100 (SPARC)
- **Operating System**: SunOS 5.9 Generic_112233-10
- **Compiler(s)**: g++ (GCC) 3.4.5
- **OpenTrep release**: 3.10.2
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from OpenTrep External 2.1.1 package. The following configuration command has been used:

```
% export CFLAGS="-mcpu=ultrasparc -O2 -pipe -funroll-all-loops"
% ./configure
```

- **Tests Status**: All tests PASSED
- **Comments**: OpenTrep configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-mcpu=ultrasparc -O2 -pipe"
% ./configure --enable-debug
```

- **Date**: May 15, 2006
- **Tester**: Adam Piatyszek (ediap)

14 OpenTrep Supported Systems (Previous Releases)

14.1 OpenTrep 3.9.1

14.2 OpenTrep 3.9.0

14.3 OpenTrep 3.8.1

15 Tutorials

15.1 Table of Contents

- Introduction
  - Preparing the StdAir Project for Development
- Build a Predefined BOM Tree
  - Instanciate the BOM Root Object
  - Instanciate the (Airline) Inventory Object
  - Link the Inventory Object with the BOM Root
  - Build Another Airline Inventory
  - Dump The BOM Tree Content
  - Result of the Tutorial Program
- Extend the Pre-Defined BOM Tree
  - Extend an Airline Inventory Object
  - Build the Specific BOM Objects
  - Result of the Tutorial Program

15.2 Introduction

This page contains some tutorial examples that will help you getting started using StdAir. Most examples show how to construct some simple business objects, i.e., instances of the so-named Business Object Model (BOM).

15.2.1 Preparing the StdAir Project for Development

The source code for these examples can be found in the `batches` and `test/stdair` directories. They are compiled along with the rest of the `StdAir` project. See the User Guide (Users Guide) for more details on how to build the `StdAir` project.

15.3 Build a Predefined BOM Tree

A few steps:

- Instanciate the BOM Root Object
- Instanciate the (Airline) Inventory Object
- Link the Inventory Object with the BOM Root

15.3.1 Instanciate the BOM Root Object

First, a BOM root object (i.e., a root for all the classes in the project) is instantiated by the `stdair::STDAIR_ServiceContext` context object, when the `stdair::STDAIR_Service` is itself instantiated. The corresponding `StdAir` type (class) is `stdair::BomRoot`.

In the following sample, that object is named `ioBomRoot`, and is given as input/output parameter of the `stdair::CmdBomManager::buildSampleBom()` method:

15.3.2 Instanciate the (Airline) Inventory Object

An airline inventory object can then be instantiated. Let us give it the "BA" airline code (corresponding to `British Airways`) as the object key. That is, an object (let us name it `lBAKey`) of type (class) `stdair::InventoryKey` has first to be instantiated.

Thanks to that key, an airline inventory object, i.e. of type (class) `stdair::Inventory`, can be instantiated. Let us name that airline inventory object `lBAInv`.

15.3.3 Link the Inventory Object with the BOM Root

Then, both objects have to be linked: the airline inventory object (`stdair::Inventory`) has to be linked with the root of the BOM tree (`stdair::BomRoot`). That operation is as simple as using the `stdair::FacBomManager::addToListAndMap()` method:

15.3.4 Build Another Airline Inventory

Another airline inventory object, corresponding to the Air France (`Air France`) company, is instantiated the same way:

See the corresponding full program (cmd_bom_manager_cpp) for more details.

15.3.5 Dump The BOM Tree Content

From the `BomRoot` (of type `stdair::BomRoot`) object instance, the list of airline inventories (of type `stdair::Inventory`) can then be retrieved...

... and browsed:

See the corresponding full program (bom_display_cpp) for more details.

15.3.6 Result of the Tutorial Program

When the `stdair.cpp` program is run (with the `-b` option), the output should look like:

See the corresponding full program (batch_stdair_cpp) for more details.

15.4 Extend the Pre-Defined BOM Tree

Now that we master how to instantiate the pre-defined StdAir classes, let us see how to extend that BOM.

15.4.1 Extend an Airline Inventory Object

For instance, let us assume that some (IT) provider (e.g., you) would like to have a specific implementation of the `Inventory` object. The corresponding class has just to extend the `stdair::Inventory` class:

The STL containers have to be defined accordingly too:

See the full class definition (test_archi_inv_hpp) and implementation (test_archi_inv_cpp) for more details.

15.4.2 Build the Specific BOM Objects

The BOM root object (`stdair::BomRoot`) is instantiated the classical way:

Then, the specific implementation of the airline inventory object (`myprovider::Inventory`) can be instantiated the same way as a standard Inventory (`stdair::Inventory`) would be:

Then, the specific implementation of the airline inventory object (`myprovider::Inventory`) is linked to the root of the BOM tree (`stdair::BomRoot`) the same way as the standard Inventory (`stdair::Inventory`) would be:

Another specific airline inventory object is instantiated the same way:

From the `BomRoot` (of type `stdair::BomRoot`) object instance, the list of specific airline inventories (of type `stdair::Inventory`) can then be retrieved...

... and browsed:

15.4.3 Result of the Tutorial Program

When this program is run, the output should look like:

See the corresponding full program (StandardAirlineITTestSuite_cpp) for more details.

16 Command-Line Test to Demonstrate How To Test the OpenTREP Project

```
  /
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE IndexBuildingTestSuite
#include <boost/test/unit_test.hpp>
// OpenTrep
#include <opentrep/OPENTREP_Service.hpp>
#include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
#include <opentrep/Location.hpp>
#include <opentrep/config/opentrep-paths.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("IndexBuildingTestSuite_utfresults.xml");

struct UnitTestConfig {
  UnitTestConfig() {
    boost_utf::unit_test_log.set_stream (utfReportStream);
    boost_utf::unit_test_log.set_format (boost_utf::XML);
    boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
    //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
  }

  ~UnitTestConfig() {
  }
};


// ////////////// Constants for the tests ///////////////
const std::string K_POR_FILEPATH (OPENTREP_POR_DATA_DIR
                                  "/test_ori_por_public.csv");

const std::string X_XAPIAN_DB_FP ("/tmp/opentrep/test_traveldb");

const std::string X_SQLITE_DB_FP ("/tmp/opentrep/test_traveldb/ori_por_public.csv");


// ///////////////// Main: Unit Test Suite //////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestConfig);

// Start the test suite
BOOST_AUTO_TEST_SUITE (master_test_suite)


BOOST_AUTO_TEST_CASE (opentrep_simple_index) {

  // Output log File
  std::string lLogFilename ("IndexBuildingTestSuite.log");

  // Set the log parameters
  std::ofstream logOutputFile;
  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the context
  const OPENTREP::PORFilePath_T lPORFilePath (K_POR_FILEPATH);
  const OPENTREP::TravelDBFilePath_T lTravelDBFilePath (
      X_XAPIAN_DB_FP);
  const OPENTREP::SQLiteDBFilePath_T lSQLiteDBFilePath (
      X_SQLITE_DB_FP);
  OPENTREP::OPENTREP_Service opentrepService  (logOutputFile, lPORFilePath,
                                               lTravelDBFilePath,
                                               lSQLiteDBFilePath);

  // Query the Xapian database (index)
  OPENTREP::WordList_T lNonMatchedWordList;
  OPENTREP::LocationList_T lLocationList;
  // Launch the indexation
  const OPENTREP::NbOfDBEntries_T nbOfEntries =
    opentrepService.buildSearchIndex();
```

```
  BOOST_CHECK_MESSAGE (nbOfEntries == 9,
                       "The Xapian index ('" << lTravelDBFilePath
                       << "') contains " << nbOfEntries
                       << " entries, where as 9 are expected.");

  // Close the Log outputFile
  logOutputFile.close();
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()
```

17 Command-Line Test to Demonstrate How To Test the OpenTREP Project

```
  /
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE SearchingTestSuite
#include <boost/test/unit_test.hpp>
// OpenTrep
#include <opentrep/OPENTREP_Service.hpp>
#include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
#include <opentrep/Location.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("SearchingTestSuite_utfresults.xml");

struct UnitTestConfig {
  UnitTestConfig() {
    boost_utf::unit_test_log.set_stream (utfReportStream);
    boost_utf::unit_test_log.set_format (boost_utf::XML);
    boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
    //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
  }

  ~UnitTestConfig() {
  }
};


// ////////////// Constants for the tests ////////////////
const std::string X_XAPIAN_DB_FP ("/tmp/opentrep/test_traveldb");

const std::string X_SQLITE_DB_FP ("/tmp/opentrep/test_traveldb/ori_por_public.csv");


// /////////////// Main: Unit Test Suite //////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestConfig);

// Start the test suite
BOOST_AUTO_TEST_SUITE (master_test_suite)


BOOST_AUTO_TEST_CASE (opentrep_simple_search) {

  // Output log File
  std::string lLogFilename ("SearchingTestSuite.log");

  // Travel query
  std::string lTravelQuery ("nce");

  // Set the log parameters
  std::ofstream logOutputFile;
  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the context
```

```
  const OPENTREP::TravelDBFilePath_T lTravelDBFilePath (
      X_XAPIAN_DB_FP);
  const OPENTREP::SQLiteDBFilePath_T lSQLiteDBFilePath (
      X_SQLITE_DB_FP);
  OPENTREP::OPENTREP_Service opentrepService (logOutputFile, lTravelDBFilePath,
                                              lSQLiteDBFilePath);

  // Query the Xapian database (index)
  OPENTREP::WordList_T lNonMatchedWordList;
  OPENTREP::LocationList_T lLocationList;
  const OPENTREP::NbOfMatches_T nbOfMatches =
    opentrepService.interpretTravelRequest (lTravelQuery, lLocationList,
                                            lNonMatchedWordList);
  BOOST_CHECK_MESSAGE (nbOfMatches == 1,
                       "The travel query ('" << lTravelQuery
                       << "') matches with " << nbOfMatches
                       << " key-words, whereas 1 is expected.");

  // Close the Log outputFile
  logOutputFile.close();
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()
```

18 Namespace Index

18.1 Namespace List

Here is a list of all namespaces with brief descriptions:

OPENTREP 61

OPENTREP::LOG 84

OPENTREP::PorParserHelper 85

soci 88

treppb 88

Xapian 88

19 Hierarchical Index

19.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

std::allocator< T >
std::array< T >
std::auto_ptr< T >

OPENTREP::BasChronometer 96

OPENTREP::BasFileMgr 97
std::basic_string< Char >
    std::string

OPENTREP::Admin1ASCIIName_T 88

OPENTREP::Admin1Code_T 89

OPENTREP::Admin1UTFName_T 90

OPENTREP::Admin2ASCIIName_T 90

OPENTREP::Admin2Code_T 91

OPENTREP::Admin2UTFName_T 92

OPENTREP::Admin3Code_T 92

OPENTREP::Admin4Code_T 93

OPENTREP::AltCountryCode_T 94

OPENTREP::CityASCIIName_T 103

OPENTREP::CityUTFName_T 104

OPENTREP::Comment_T 107

OPENTREP::ContinentCode_T 108

OPENTREP::ContinentName_T 109

OPENTREP::CountryCode_T 110

OPENTREP::CountryName_T 110

OPENTREP::FAACode_T 125

OPENTREP::FeatureClass_T 154

OPENTREP::FeatureCode_T 154

OPENTREP::FeatureName_T 155

OPENTREP::FilePath_T 158

OPENTREP::PORFilePath_T 275

OPENTREP::IATACode_T 163

OPENTREP::CityCode_T 103

OPENTREP::TvlPORListString_T 404

OPENTREP::ICAOCode_T 167

OPENTREP::LanguageCode_T 174

OPENTREP::LocationName_T 204

OPENTREP::AltNameShortListString_T 94

OPENTREP::ASCIIName_T 95

OPENTREP::CommonName_T 108

OPENTREP::RawDataString_T 281

OPENTREP::SQLiteDBFilePath_T 313

OPENTREP::StateCode_T 314

OPENTREP::TimeZone_T 402

OPENTREP::TravelDBFilePath_T 402

OPENTREP::WikiLink_T 412

std::wstring
std::basic_string< char >
std::basic_string< wchar_t >
std::bitset< Bits >

OPENTREP::BomAbstract 97

OPENTREP::Levenshtein 176

OPENTREP::Place 248

OPENTREP::PlaceHolder 271

OPENTREP::Result 282

OPENTREP::ResultCombination 292

OPENTREP::ResultHolder 296

OPENTREP::WordHolder 416

OPENTREP::World 419

OPENTREP::BomJSONExport 100

OPENTREP::BomType 101
bool_policies

OPENTREP::short_bool_policies 310
std::complex
std::list< T >::const_iterator
std::forward_list< T >::const_iterator
std::map< K, T >::const_iterator
std::unordered_map< K, T >::const_iterator
std::multimap< K, T >::const_iterator
std::basic_string< Char >::const_iterator
std::unordered_multimap< K, T >::const_iterator
std::set< K >::const_iterator
std::string::const_iterator
std::unordered_set< K >::const_iterator
std::wstring::const_iterator
std::multiset< K >::const_iterator
std::unordered_multiset< K >::const_iterator
std::vector< T >::const_iterator
std::deque< T >::const_iterator
std::list< T >::const_reverse_iterator
std::forward_list< T >::const_reverse_iterator
std::map< K, T >::const_reverse_iterator
std::unordered_map< K, T >::const_reverse_iterator
std::multimap< K, T >::const_reverse_iterator
std::basic_string< Char >::const_reverse_iterator
std::unordered_multimap< K, T >::const_reverse_iterator
std::set< K >::const_reverse_iterator
std::string::const_reverse_iterator
std::unordered_set< K >::const_reverse_iterator
std::multiset< K >::const_reverse_iterator

std::wstring::const_reverse_iterator
std::unordered_multiset< K >::const_reverse_iterator
std::vector< T >::const_reverse_iterator
std::deque< T >::const_reverse_iterator

OPENTREP::date_time_element< MIN, MAX > 111

OPENTREP::date_time_element< 1, 12 > 111

OPENTREP::date_time_element< 1, 31 > 111

OPENTREP::date_time_element< 1900, 2100 > 111

OPENTREP::DbaAbstract 112

OPENTREP::DBManager 113

OPENTREP::DBSessionManager 120
std::deque< T >
std::error_category
std::error_code
std::error_condition
std::exception

OPENTREP::RootException 299

OPENTREP::BuildIndexException 102

OPENTREP::FileException 156

OPENTREP::FileNotFoundException 157

OPENTREP::InterpreterUseCaseException 168

OPENTREP::InterpreteTravelRequestException 169

OPENTREP::TravelRequestEmptyException 403

OPENTREP::LanguageCodeNotDefinedInNameTableException 174

OPENTREP::MemoryAllocationException 220

OPENTREP::MultipleRowsForASingleDocIDException 221

OPENTREP::NoBestMatchingResultHolderException 228

OPENTREP::NonInitialisedServiceException 229

OPENTREP::ObjectNotFoundException 230

OPENTREP::ParserException 245

OPENTREP::CodeConversionException 105

OPENTREP::CodeDuplicationException 106

OPENTREP::PorFileParsingException 274

OPENTREP::PRFileParsingException 277

OPENTREP::SQLDatabaseException 312

OPENTREP::SQLDatabaseConnectionImpossibleException 311

OPENTREP::UnicodeException 408

OPENTREP::UnicodeTransliteratorCreationException 409

OPENTREP::XapianException 423

OPENTREP::DocumentNotFoundException 122

OPENTREP::XapianDatabaseFailureException 422

OPENTREP::XapianTravelDatabaseEmptyException 424

OPENTREP::XapianTravelDatabaseWrongPathnameException 426

std::bad_alloc
std::bad_cast
std::bad_exception
std::bad_typeid
std::ios_base::failure
std::logic_error
std::domain_error
std::invalid_argument
std::length_error
std::out_of_range
std::runtime_error
std::overflow_error
std::range_error
std::underflow_error

OPENTREP::FacBomAbstract 126

OPENTREP::FacPlace 130

OPENTREP::FacPlaceHolder 135

OPENTREP::FacResult 138

OPENTREP::FacResultCombination 140

OPENTREP::FacResultHolder 143

OPENTREP::FacWorld 151

OPENTREP::FacServiceAbstract 147

OPENTREP::FacOpenTrepServiceContext 128

OPENTREP::FacSupervisor 148

OPENTREP::Filter 159

FloatingPoint< **RawType** > 160
std::forward_list< T >
grammar

OPENTREP::PorParserHelper::LocationParser< **Iterator** > 205

OPENTREP::IATAType 164

OPENTREP::IndexBuilder 167
std::ios_base
basic_ios< char >
basic_ios< wchar_t >

std::basic_ios
basic_istream< char >
basic_istream< wchar_t >
basic_ostream< char >
basic_ostream< wchar_t >
std::basic_istream
basic_ifstream< char >
basic_ifstream< wchar_t >
basic_iostream< char >
basic_iostream< wchar_t >
basic_istringstream< char >
basic_istringstream< wchar_t >
std::basic_ifstream
std::ifstream
std::wifstream
std::basic_iostream
basic_fstream< char >
basic_fstream< wchar_t >
basic_stringstream< char >
basic_stringstream< wchar_t >
std::basic_fstream
std::fstream
std::wfstream
std::basic_stringstream
std::stringstream
std::wstringstream
std::basic_istringstream
std::istringstream
std::wistringstream
std::istream
std::wistream
std::basic_ostream
basic_iostream< char >
basic_iostream< wchar_t >
basic_ofstream< char >
basic_ofstream< wchar_t >
basic_ostringstream< char >
basic_ostringstream< wchar_t >
std::basic_iostream
std::basic_ofstream
std::ofstream
std::wofstream
std::basic_ostringstream
std::ostringstream
std::wostringstream
std::ostream
std::wostream
std::ios
std::wios
std::list< T >::iterator
std::forward_list< T >::iterator
std::map< K, T >::iterator
std::unordered_map< K, T >::iterator
std::multimap< K, T >::iterator
std::basic_string< Char >::iterator
std::unordered_multimap< K, T >::iterator
std::set< K >::iterator
std::string::iterator

std::unordered_set< K >::iterator
std::wstring::iterator
std::multiset< K >::iterator
std::unordered_multiset< K >::iterator
std::vector< T >::iterator
std::deque< T >::iterator

OPENTREP::KeyAbstract 170

OPENTREP::Language 171
std::list< T >
std::list< AltNameShortListString_T >
std::list< Location >
std::list< MatchingDocuments >
std::list< Place * >
std::list< Result * >
std::list< ResultHolder * >
std::list< std::string >
std::list< StringPartition >
std::list< StringSet >
std::list< TvlPORListString_T >

OPENTREP::ListGenerator 177

OPENTREP::LocationExchange 200

OPENTREP::Logger 218
std::map< K, T >
std::map< const Weight_T, StringSet_T >
std::map< LanguageCode_T, Names >
std::map< NbOfLetters_T, NbOfErrors_T >
std::map< ScoreType::EN_ScoreType, Score_T >
std::map< std::size_t, GenericBom_T >
std::map< Xapian::docid, XapianDocumentPair_T >
std::multimap< K, T >
std::multimap< std::string, Place * >
std::multiset< K >

OPENTREP::OPENTREP_Abstract 232

OPENTREP::DBParams 117

OPENTREP::DistanceErrorRule 120

OPENTREP::Location 178

OPENTREP::LocationKey 201

OPENTREP::NameMatrix 222

OPENTREP::Names 225

OPENTREP::OPENTREP_Service 233

OPENTREP::OpenTrepSearcher 239

OPENTREP::OTransliterator 241

OPENTREP::OutputFormat 243

OPENTREP::PorParserHelper::ParserSemanticAction 247

OPENTREP::PorParserHelper::doEndPor 124

OPENTREP::PorParserHelper::storeAdm1AsciiName 314

OPENTREP::PorParserHelper::storeAdm1Code 316

OPENTREP::PorParserHelper::storeAdm1UtfName 317

OPENTREP::PorParserHelper::storeAdm2AsciiName 319

OPENTREP::PorParserHelper::storeAdm2Code 321

OPENTREP::PorParserHelper::storeAdm2UtfName 322

OPENTREP::PorParserHelper::storeAdm3Code 324

OPENTREP::PorParserHelper::storeAdm4Code 325

OPENTREP::PorParserHelper::storeAltCountryCode 327

OPENTREP::PorParserHelper::storeAltLangCode2Char 328

OPENTREP::PorParserHelper::storeAltLangCodeExt 330

OPENTREP::PorParserHelper::storeAltLangCodeFull 331

OPENTREP::PorParserHelper::storeAltLangCodeHist 333

OPENTREP::PorParserHelper::storeAltName 335

OPENTREP::PorParserHelper::storeAltNameShort 336

OPENTREP::PorParserHelper::storeAltNameShortListString 338

OPENTREP::PorParserHelper::storeAsciiName 339

OPENTREP::PorParserHelper::storeCityAsciiName 341

OPENTREP::PorParserHelper::storeCityCode 342

OPENTREP::PorParserHelper::storeCityGeonamesID 344

OPENTREP::PorParserHelper::storeCityUtfName 345

OPENTREP::PorParserHelper::storeComments 347

OPENTREP::PorParserHelper::storeCommonName 349

OPENTREP::PorParserHelper::storeContinentName 350

OPENTREP::PorParserHelper::storeCountryCode 352

OPENTREP::PorParserHelper::storeCountryName 353

OPENTREP::PorParserHelper::storeDateFrom 355

OPENTREP::PorParserHelper::storeDateUntil 356

OPENTREP::PorParserHelper::storeDSTOffset 358

OPENTREP::PorParserHelper::storeElevation 359

OPENTREP::PorParserHelper::storeEnvelopeID 361

OPENTREP::PorParserHelper::storeFaaCode 362

OPENTREP::PorParserHelper::storeFeatureClass 364

OPENTREP::PorParserHelper::storeFeatureCode 365

OPENTREP::PorParserHelper::storeGeonamesID 367

OPENTREP::PorParserHelper::storeGMTOffset 369

OPENTREP::PorParserHelper::storeGTopo30 370

OPENTREP::PorParserHelper::storeIataCode 372

OPENTREP::PorParserHelper::storeIcaoCode 373

OPENTREP::PorParserHelper::storeLatitude 375

OPENTREP::PorParserHelper::storeLongitude 376

OPENTREP::PorParserHelper::storeModDate 378

OPENTREP::PorParserHelper::storePageRank 379

OPENTREP::PorParserHelper::storePopulation 381

OPENTREP::PorParserHelper::storePORType 383

OPENTREP::PorParserHelper::storeRawOffset 384

OPENTREP::PorParserHelper::storeStateCode 386

OPENTREP::PorParserHelper::storeTimeZone 387

OPENTREP::PorParserHelper::storeTvlPORCode 389

OPENTREP::PorParserHelper::storeTvlPORListString 390

OPENTREP::PorParserHelper::storeWikiLink 392

OPENTREP::PORFileParser 273

OPENTREP::PORParser 275

OPENTREP::PORStringParser 276
std::priority_queue< T >
std::queue< T >

OPENTREP::RequestInterpreter 282
std::deque< T >::reverse_iterator
std::forward_list< T >::reverse_iterator
std::unordered_multimap< K, T >::reverse_iterator
std::unordered_map< K, T >::reverse_iterator
std::multimap< K, T >::reverse_iterator
std::map< K, T >::reverse_iterator
std::basic_string< Char >::reverse_iterator
std::set< K >::reverse_iterator
std::string::reverse_iterator
std::unordered_set< K >::reverse_iterator
std::multiset< K >::reverse_iterator
std::wstring::reverse_iterator
std::unordered_multiset< K >::reverse_iterator

std::vector< T >::reverse_iterator
std::list< T >::reverse_iterator

OPENTREP::ServiceAbstract 308

OPENTREP::OPENTREP_ServiceContext 236

OPENTREP::ServiceUtilities 309
std::set< K >
std::set< std::string >
std::smart_ptr< T >
std::stack< T >

OPENTREP::StructAbstract 400

OPENTREP::QuerySlices 278

OPENTREP::ScoreBoard 301

OPENTREP::ScoreType 305

OPENTREP::StringPartition 393

OPENTREP::StringSet 397

OPENTREP::WordCombinationHolder 413
std::system_error
std::thread

soci::type_conversion< OPENTREP::Place > 405

TypeWithSize< size > 406

TypeWithSize< 4 > 407

TypeWithSize< 8 > 407

TypeWithSize< sizeof(RawType)> 406
std::unique_ptr< T >

UnitTestConfig 410
std::unordered_map< K, T >
std::unordered_multimap< K, T >
std::unordered_multiset< K >
std::unordered_set< K >

OPENTREP::UTF8Handler 411
std::valarray< T >
std::vector< T >
std::vector< BomAbstract * >
std::vector< FacBomAbstract * >
std::vector< FacServiceAbstract * >
std::vector< ServiceAbstract * >
std::weak_ptr< T >

OPENTREP::XapianIndexManager 424
bool
const Weight_T
Database &
date
docid

double
EN_IATAType
EN_Language
EN_LogLevel
EN_OutputFormat
EN_ScoreType
EN_ScoreType
float
GenericBom_T
int
K
MatchingDocuments
NbOfLetters_T
ptime
RawType
rule< Iterator, bsu::blank_type >
Score_T
session *
size_t
static const Bits
static const char
T
Transliterator *
unsigned short
XapianDocumentPair_T

20 Class Index

20.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

OPENTREP::Admin1ASCIIName_T 88

OPENTREP::Admin1Code_T 89

OPENTREP::Admin1UTFName_T 90

OPENTREP::Admin2ASCIIName_T 90

OPENTREP::Admin2Code_T 91

OPENTREP::Admin2UTFName_T 92

OPENTREP::Admin3Code_T 92

OPENTREP::Admin4Code_T 93

OPENTREP::AltCountryCode_T 94

OPENTREP::AltNameShortListString_T 94

OPENTREP::ASCIIName_T 95

OPENTREP::BasChronometer
Structure allowing measuring the time elapsed between two events 96

OPENTREP::BasFileMgr
Helper class for operations on files and on the file-system 97

OPENTREP::BomAbstract
Base class for the Business Object Model (BOM) layer 97

OPENTREP::BomJSONExport
Utility class to export Opentrep structures in a JSON format 100

OPENTREP::BomType 101

OPENTREP::BuildIndexException 102

OPENTREP::CityASCIIName_T 103

OPENTREP::CityCode_T 103

OPENTREP::CityUTFName_T 104

OPENTREP::CodeConversionException 105

OPENTREP::CodeDuplicationException 106

OPENTREP::Comment_T 107

OPENTREP::CommonName_T 108

OPENTREP::ContinentCode_T 108

OPENTREP::ContinentName_T 109

OPENTREP::CountryCode_T 110

OPENTREP::CountryName_T 110

OPENTREP::date_time_element< MIN, MAX > 111

OPENTREP::DbaAbstract 112

OPENTREP::DBManager
Class building the Business Object Model (BOM) from data retrieved from the database 113

OPENTREP::DBParams
Structure modelling a database parameters 117

OPENTREP::DBSessionManager
Class handling the SOCI session 120

OPENTREP::DistanceErrorRule 120

OPENTREP::DocumentNotFoundException 122

OPENTREP::PorParserHelper::doEndPor 124

OPENTREP::FAACode_T 125

OPENTREP::FacBomAbstract 126

OPENTREP::FacOpenTrepServiceContext
Factory for Bucket 128

OPENTREP::FacPlace 130

OPENTREP::FacPlaceHolder 135

OPENTREP::FacResult 138

OPENTREP::FacResultCombination 140

OPENTREP::FacResultHolder 143

OPENTREP::FacServiceAbstract 147

OPENTREP::FacSupervisor 148

OPENTREP::FacWorld 151

OPENTREP::FeatureClass_T 154

OPENTREP::FeatureCode_T 154

OPENTREP::FeatureName_T 155

OPENTREP::FileException 156

OPENTREP::FileNotFoundException 157

OPENTREP::FilePath_T 158

OPENTREP::Filter
Class filtering out the words not suitable for indexing and/or searching, when part of greater strings. Hence, most of the methods take as parameter the "initial"/greater string 159

FloatingPoint< RawType > 160

OPENTREP::IATACode_T 163

OPENTREP::IATAType
Enumeration of output types 164

OPENTREP::ICAOCode_T 167

OPENTREP::IndexBuilder
Command wrapping the travel request process 167

OPENTREP::InterpreterUseCaseException 168

OPENTREP::InterpreteTravelRequestException 169

OPENTREP::KeyAbstract
Base class for the Business Object Model (KEY) layer 170

OPENTREP::Language 171

OPENTREP::LanguageCode_T 174

OPENTREP::LanguageCodeNotDefinedInNameTableException 174

OPENTREP::Levenshtein 176

OPENTREP::ListGenerator 177

OPENTREP::Location
Structure modelling a (geographical) location 178

OPENTREP::LocationExchange
Utility class to export Opentrep structures in a Protobuf format 200

OPENTREP::LocationKey
Class modelling the primary key of a location/POR (point of reference) 201

OPENTREP::LocationName_T 204

OPENTREP::PorParserHelper::LocationParser< Iterator > 205

OPENTREP::Logger 218

OPENTREP::MemoryAllocationException 220

OPENTREP::MultipleRowsForASingleDocIDException 221

OPENTREP::NameMatrix 222

OPENTREP::Names 225

OPENTREP::NoBestMatchingResultHolderException 228

OPENTREP::NonInitialisedServiceException 229

OPENTREP::ObjectNotFoundException 230

OPENTREP::OPENTREP_Abstract 232

OPENTREP::OPENTREP_Service
Interface for the OPENTREP Services 233

OPENTREP::OPENTREP_ServiceContext
Class holding the context of the OpenTrep services 236

OPENTREP::OpenTrepSearcher
API wrapper around the OpenTREP C++ API, so that Python scripts can use it seamlessly 239

OPENTREP::OTransliterator 241

OPENTREP::OutputFormat
Enumeration of output formats 243

OPENTREP::ParserException 245

OPENTREP::PorParserHelper::ParserSemanticAction 247

OPENTREP::Place
Class modelling a place/POR (point of reference) 248

OPENTREP::PlaceHolder 271

OPENTREP::PORFileParser 273

OPENTREP::PorFileParsingException 274

OPENTREP::PORFilePath_T 275

OPENTREP::PORParser 275

OPENTREP::PORStringParser 276

OPENTREP::PRFileParsingException 277

OPENTREP::QuerySlices 278

OPENTREP::RawDataString_T 281

OPENTREP::RequestInterpreter
Command wrapping the travel request process 282

OPENTREP::Result
Class wrapping a set of Xapian documents having matched a given query string 282

OPENTREP::ResultCombination
Class wrapping functions on a list of ResultHolder objects 292

OPENTREP::ResultHolder
Class wrapping functions on a list of Result objects 296

OPENTREP::RootException
Root of the OpenTREP exceptions 299

OPENTREP::ScoreBoard
Structure holding a board for all the types of score/matching having been performed 301

OPENTREP::ScoreType
Enumeration of score types 305

OPENTREP::ServiceAbstract 308

OPENTREP::ServiceUtilities 309

OPENTREP::short_bool_policies 310

OPENTREP::SQLDatabaseConnectionImpossibleException 311

OPENTREP::SQLDatabaseException 312

OPENTREP::SQLiteDBFilePath_T 313

OPENTREP::StateCode_T 314

OPENTREP::PorParserHelper::storeAdm1AsciiName 314

OPENTREP::PorParserHelper::storeAdm1Code 316

OPENTREP::PorParserHelper::storeAdm1UtfName 317

OPENTREP::PorParserHelper::storeAdm2AsciiName 319

OPENTREP::PorParserHelper::storeAdm2Code 321

OPENTREP::PorParserHelper::storeAdm2UtfName 322

OPENTREP::PorParserHelper::storeAdm3Code 324

OPENTREP::PorParserHelper::storeAdm4Code 325

OPENTREP::PorParserHelper::storeAltCountryCode 327

OPENTREP::PorParserHelper::storeAltLangCode2Char 328

OPENTREP::PorParserHelper::storeAltLangCodeExt 330

OPENTREP::PorParserHelper::storeAltLangCodeFull 331

OPENTREP::PorParserHelper::storeAltLangCodeHist 333

OPENTREP::PorParserHelper::storeAltName 335

OPENTREP::PorParserHelper::storeAltNameShort 336

OPENTREP::PorParserHelper::storeAltNameShortListString 338

OPENTREP::PorParserHelper::storeAsciiName 339

OPENTREP::PorParserHelper::storeCityAsciiName 341

OPENTREP::PorParserHelper::storeCityCode 342

OPENTREP::PorParserHelper::storeCityGeonamesID 344

OPENTREP::PorParserHelper::storeCityUtfName 345

OPENTREP::PorParserHelper::storeComments 347

OPENTREP::PorParserHelper::storeCommonName 349

OPENTREP::PorParserHelper::storeContinentName 350

OPENTREP::PorParserHelper::storeCountryCode 352

OPENTREP::PorParserHelper::storeCountryName 353

OPENTREP::PorParserHelper::storeDateFrom 355

OPENTREP::PorParserHelper::storeDateUntil 356

OPENTREP::PorParserHelper::storeDSTOffset 358

OPENTREP::PorParserHelper::storeElevation 359

OPENTREP::PorParserHelper::storeEnvelopeID 361

OPENTREP::PorParserHelper::storeFaaCode 362

OPENTREP::PorParserHelper::storeFeatureClass 364

OPENTREP::PorParserHelper::storeFeatureCode 365

OPENTREP::PorParserHelper::storeGeonamesID 367

OPENTREP::PorParserHelper::storeGMTOffset 369

OPENTREP::PorParserHelper::storeGTopo30 370

OPENTREP::PorParserHelper::storeIataCode 372

OPENTREP::PorParserHelper::storeIcaoCode 373

OPENTREP::PorParserHelper::storeLatitude 375

OPENTREP::PorParserHelper::storeLongitude 376

OPENTREP::PorParserHelper::storeModDate 378

OPENTREP::PorParserHelper::storePageRank 379

OPENTREP::PorParserHelper::storePopulation 381

OPENTREP::PorParserHelper::storePORType 383

OPENTREP::PorParserHelper::storeRawOffset 384

OPENTREP::PorParserHelper::storeStateCode 386

OPENTREP::PorParserHelper::storeTimeZone 387

OPENTREP::PorParserHelper::storeTvlPORCode 389

OPENTREP::PorParserHelper::storeTvlPORListString 390

OPENTREP::PorParserHelper::storeWikiLink 392

OPENTREP::StringPartition 393

OPENTREP::StringSet
Class holding a set of strings, e.g., {"rio", "de", "janeiro"} 397

OPENTREP::StructAbstract
Base class for the light structures 400

OPENTREP::TimeZone_T 402

OPENTREP::TravelDBFilePath_T 402

OPENTREP::TravelRequestEmptyException 403

OPENTREP::TvlPORListString_T 404

soci::type_conversion< OPENTREP::Place >
Specify how the Place class can be converted to (resp. from) values stored into (resp. retrieved from) database, using the SOCI framework 405

TypeWithSize< size > 406

TypeWithSize< 4 > 407

TypeWithSize< 8 > 407

OPENTREP::UnicodeException 408

OPENTREP::UnicodeTransliteratorCreationException 409

UnitTestConfig 410

OPENTREP::UTF8Handler
Utility class for basic handling of UTF-8 encoded strings 411

OPENTREP::WikiLink_T 412

OPENTREP::WordCombinationHolder 413

OPENTREP::WordHolder
Class wrapping utility functions to transform back and forth strings from and into a list of words 416

OPENTREP::World 419

OPENTREP::XapianDatabaseFailureException 422

OPENTREP::XapianException 423

OPENTREP::XapianIndexManager
Command wrapping utilities for the management of the Xapian (database) index 424

OPENTREP::XapianTravelDatabaseEmptyException 424

OPENTREP::XapianTravelDatabaseWrongPathnameException 426

21 File Index

21.1 File List

Here is a list of all files with brief descriptions:

opentrep/DBParams.hpp 629

opentrep/DistanceErrorRule.hpp 631

opentrep/IATAType.hpp 653

opentrep/Language.hpp 654

opentrep/Location.hpp 655

opentrep/LocationKey.hpp 664

opentrep/LocationList.hpp 665

opentrep/NameMatrix.hpp 666

opentrep/Names.hpp 667

opentrep/OPENTREP_Abstract.hpp 669

opentrep/OPENTREP_exceptions.hpp 670

opentrep/OPENTREP_log.hpp 673

opentrep/OPENTREP_ParserHelperTypes.hpp 674

opentrep/OPENTREP_Service.hpp 675

opentrep/OPENTREP_Types.hpp 678

opentrep/OutputFormat.hpp 682

opentrep/basic/BasChronometer.cpp 428

opentrep/basic/BasChronometer.hpp 428

opentrep/basic/BasConst.cpp 430

opentrep/basic/BasConst_General.hpp 432

opentrep/basic/BasConst_OPENTREP_Service.hpp 433

opentrep/basic/BasConst_Unicode.hpp 433

opentrep/basic/BasFileMgr.cpp 434

opentrep/basic/BasFileMgr.hpp 434

opentrep/basic/BasParserTypes.hpp 436

opentrep/basic/float_utils.hpp 437

opentrep/basic/float_utils_google.hpp 438

opentrep/basic/icu_util.cpp 441

opentrep/basic/icu_util.hpp 443

opentrep/basic/Location.cpp 443

opentrep/basic/LocationKey.cpp 450

opentrep/basic/NameMatrix.cpp 451

opentrep/basic/Names.cpp 453

opentrep/basic/OTransliterator.cpp 454

opentrep/basic/OTransliterator.hpp 457

opentrep/basic/OutputFormat.cpp 457

opentrep/basic/ScoreType.cpp 459

opentrep/basic/ScoreType.hpp 461

opentrep/basic/StructAbstract.hpp 463

opentrep/basic/UTF8Handler.cpp 464

opentrep/basic/UTF8Handler.hpp 466

opentrep/basic/Utilities.cpp 467

opentrep/basic/Utilities.hpp 469

opentrep/batches/opentrep-dbmgr.cpp 470

opentrep/batches/opentrep-indexer.cpp 474

opentrep/batches/opentrep-searcher.cpp 479

opentrep/bom/Airline.cpp 482

opentrep/bom/Airline.hpp 483

opentrep/bom/BomAbstract.hpp 483

opentrep/bom/BomJSONExport.cpp 485

opentrep/bom/BomJSONExport.hpp 487

opentrep/bom/BomType.cpp 487

opentrep/bom/BomType.hpp 488

opentrep/bom/Country.cpp 488

opentrep/bom/Country.hpp 489

opentrep/bom/Day.cpp 489

opentrep/bom/Day.hpp 489

opentrep/bom/DocumentList.hpp 489

opentrep/bom/Filter.cpp 490

opentrep/bom/Filter.hpp 492

opentrep/bom/GenericBom.hpp 493

opentrep/bom/IATAType.cpp 494

opentrep/bom/KeyAbstract.hpp 497

opentrep/bom/Language.cpp 498

opentrep/bom/Levenshtein.cpp 499

opentrep/bom/Levenshtein.hpp 500

opentrep/bom/LocationExchange.cpp 501

opentrep/bom/LocationExchange.hpp 506

opentrep/bom/Month.cpp 507

opentrep/bom/Month.hpp 507

opentrep/bom/Passenger.cpp 507

opentrep/bom/Passenger.hpp 507

opentrep/bom/Place.cpp 507

opentrep/bom/Place.hpp 516

opentrep/bom/PlaceHolder.cpp 524

opentrep/bom/PlaceHolder.hpp 525

opentrep/bom/PlaceList.hpp 526

opentrep/bom/PORParserHelper.cpp 528

opentrep/bom/PORParserHelper.hpp 546

opentrep/bom/QuerySlices.cpp 552

opentrep/bom/QuerySlices.hpp 556

opentrep/bom/Result.cpp 558

opentrep/bom/Result.hpp 568

opentrep/bom/ResultCombination.cpp 571

opentrep/bom/ResultCombination.hpp 574

opentrep/bom/ResultHolder.cpp 575

opentrep/bom/ResultHolder.hpp 578

opentrep/bom/ResultHolderList.hpp 580

opentrep/bom/ResultList.hpp 580

opentrep/bom/ScoreBoard.cpp 581

opentrep/bom/ScoreBoard.hpp 583

opentrep/bom/StringPartition.cpp 585

opentrep/bom/StringPartition.hpp 587

opentrep/bom/StringSet.cpp 589

opentrep/bom/StringSet.hpp 590

opentrep/bom/WordCombinationHolder.cpp 591

opentrep/bom/WordCombinationHolder.hpp 594

opentrep/bom/WordHolder.cpp 595

opentrep/bom/WordHolder.hpp 596

opentrep/bom/World.cpp 597

opentrep/bom/World.hpp 598

opentrep/command/DBManager.cpp 600

opentrep/command/DBManager.hpp 605

opentrep/command/DBSessionManager.cpp 606

opentrep/command/DBSessionManager.hpp 607

opentrep/command/IndexBuilder.cpp 609

opentrep/command/IndexBuilder.hpp 611

opentrep/command/PORParser.cpp 612

opentrep/command/PORParser.hpp 613

opentrep/command/RequestInterpreter.cpp 614

opentrep/command/RequestInterpreter.hpp 618

opentrep/command/XapianIndexManager.cpp 619

opentrep/command/XapianIndexManager.hpp 621

opentrep/dbadaptor/DbaAbstract.cpp 622

opentrep/dbadaptor/DbaAbstract.hpp 623

opentrep/dbadaptor/DbaPlace.cpp 624

opentrep/dbadaptor/DbaPlace.hpp 628

opentrep/factory/FacBomAbstract.cpp 633

opentrep/factory/FacBomAbstract.hpp 634

opentrep/factory/FacOpenTrepServiceContext.cpp 635

opentrep/factory/FacOpenTrepServiceContext.hpp 636

opentrep/factory/FacPlace.cpp 637

opentrep/factory/FacPlace.hpp 638

opentrep/factory/FacPlaceHolder.cpp 639

opentrep/factory/FacPlaceHolder.hpp 640

opentrep/factory/FacResult.cpp 641

opentrep/factory/FacResult.hpp 642

opentrep/factory/FacResultCombination.cpp 643

opentrep/factory/FacResultCombination.hpp 644

opentrep/factory/FacResultHolder.cpp 645

opentrep/factory/FacResultHolder.hpp 646

opentrep/factory/FacServiceAbstract.cpp 647

opentrep/factory/FacServiceAbstract.hpp 648

opentrep/factory/FacSupervisor.cpp 648

opentrep/factory/FacSupervisor.hpp 650

opentrep/factory/FacWorld.cpp 651

opentrep/factory/FacWorld.hpp 652

opentrep/python/pyopentrep.cpp 683

opentrep/service/Logger.cpp 691

opentrep/service/Logger.hpp 694

opentrep/service/OPENTREP_Service.cpp 696

opentrep/service/OPENTREP_ServiceContext.cpp 700

opentrep/service/OPENTREP_ServiceContext.hpp 701

opentrep/service/ServiceAbstract.cpp 702

opentrep/service/ServiceAbstract.hpp 703

opentrep/service/ServiceUtilities.cpp 704

opentrep/service/ServiceUtilities.hpp 705

test/opentrep/IndexBuildingTestSuite.cpp 705

test/opentrep/PartitionTestSuite.cpp 708

test/opentrep/SearchingTestSuite.cpp 709

test/opentrep/SliceTestSuite.cpp 711

test/opentrep/UnicodeTestSuite.cpp 714

22 Namespace Documentation

22.1 OPENTREP Namespace Reference

Namespaces

- PorParserHelper
- LOG

Classes

- struct BasChronometer

  Structure allowing measuring the time elapsed between two events.
- struct ListGenerator
- struct BasFileMgr

  Helper class for operations on files and on the file-system.
- struct short_bool_policies
- class OTransliterator
- struct ScoreType

  Enumeration of score types.
- struct StructAbstract

  Base class for the light structures.
- class UTF8Handler

  Utility class for basic handling of UTF-8 encoded strings.
- class BomAbstract

  Base class for the Business Object Model (BOM) layer.
- class BomJSONExport

  Utility class to export Opentrep structures in a JSON format.
- struct BomType
- struct Filter

  Class filtering out the words not suitable for indexing and/or searching, when part of greater strings. Hence, most of the methods take as parameter the "initial"/greater string.
- class KeyAbstract

  Base class for the Business Object Model (KEY) layer.
- class Levenshtein
- class LocationExchange

  Utility class to export Opentrep structures in a Protobuf format.
- class Place

  Class modelling a place/POR (point of reference).
- class PlaceHolder
- class PORStringParser
- class PORFileParser
- struct QuerySlices
- class Result

  Class wrapping a set of Xapian documents having matched a given query string.
- class ResultCombination

  Class wrapping functions on a list of ResultHolder objects.
- class ResultHolder

  Class wrapping functions on a list of Result objects.
- struct ScoreBoard

  Structure holding a board for all the types of score/matching having been performed.
- struct StringPartition
- struct StringSet

  Class holding a set of strings, e.g., {"rio", "de", "janeiro"}.
- struct WordCombinationHolder
- class WordHolder

Class wrapping utility functions to transform back and forth strings from and into a list of words.

- class World
- class DBManager

  Class building the Business Object Model (BOM) from data retrieved from the database.
- class DBSessionManager

  Class handling the SOCI session.
- class IndexBuilder

  Command wrapping the travel request process.
- class PORParser
- class RequestInterpreter

  Command wrapping the travel request process.
- class XapianIndexManager

  Command wrapping utilities for the management of the Xapian (database) index.
- class DbaAbstract
- struct DBParams

  Structure modelling a database parameters.
- struct DistanceErrorRule
- class FacBomAbstract
- class FacOpenTrepServiceContext

  Factory for Bucket.
- class FacPlace
- class FacPlaceHolder
- class FacResult
- class FacResultCombination
- class FacResultHolder
- class FacServiceAbstract
- class FacSupervisor
- class FacWorld
- struct IATAType

  Enumeration of output types.
- struct Language
- struct Location

  Structure modelling a (geographical) location.
- struct LocationKey

  Class modelling the primary key of a location/POR (point of reference).
- struct NameMatrix
- struct Names
- struct OPENTREP_Abstract
- class RootException

  Root of the OpenTREP exceptions.
- class NonInitialisedServiceException
- class UnicodeException
- class UnicodeTransliteratorCreationException
- class MultipleRowsForASingleDocIDException
- class LanguageCodeNotDefinedInNameTableException
- class MemoryAllocationException
- class ObjectNotFoundException
- class FileException
- class FileNotFoundException
- class ParserException
- class PorFileParsingException
- class PRFileParsingException
- class CodeConversionException

- class CodeDuplicationException
- class XapianException
- class DocumentNotFoundException
- class XapianDatabaseFailureException
- class XapianTravelDatabaseEmptyException
- class XapianTravelDatabaseWrongPathnameException
- class NoBestMatchingResultHolderException
- class SQLDatabaseException
- class SQLDatabaseConnectionImpossibleException
- class BuildIndexException
- class InterpreterUseCaseException
- class InterpreteTravelRequestException
- class TravelRequestEmptyException
- struct date_time_element
- class OPENTREP_Service

  Interface for the OPENTREP Services.
- struct FilePath_T
- struct PORFilePath_T
- struct TravelDBFilePath_T
- struct SQLiteDBFilePath_T
- struct RawDataString_T
- struct IATACode_T
- struct ICAOCode_T
- struct FAACode_T
- struct LocationName_T
- struct CommonName_T
- struct ASCIIName_T
- struct AltNameShortListString_T
- struct TvlPORListString_T
- struct LanguageCode_T
- struct Comment_T
- struct CityCode_T
- struct CityUTFName_T
- struct CityASCIIName_T
- struct StateCode_T
- struct CountryCode_T
- struct AltCountryCode_T
- struct CountryName_T
- struct ContinentCode_T
- struct ContinentName_T
- struct Admin1Code_T
- struct Admin1UTFName_T
- struct Admin1ASCIIName_T
- struct Admin2Code_T
- struct Admin2UTFName_T
- struct Admin2ASCIIName_T
- struct Admin3Code_T
- struct Admin4Code_T
- struct FeatureClass_T
- struct FeatureCode_T
- struct FeatureName_T
- struct WikiLink_T
- struct TimeZone_T
- struct OutputFormat

Enumeration of output formats.

- struct OpenTrepSearcher

  API wrapper around the OpenTREP C++ API, so that Python scripts can use it seamlessly.

- class Logger
- class OPENTREP_ServiceContext

  Class holding the context of the OpenTrep services.

- class ServiceAbstract
- class ServiceUtilities

Typedefs

- typedef
  std::istreambuf_iterator< char > base_iterator_t
- typedef
  boost::spirit::multi_pass
  < base_iterator_t > iterator_t
- typedef
  boost::spirit::classic::position_iterator2
  < iterator_t > pos_iterator_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 1, 1 > uint1_p_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 2, 2 > uint2_p_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 1, 2 > int1_2_p_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 4, 4 > uint4_p_t
- typedef
  boost::spirit::qi::int_parser
  < int, 10, 1, 5 > int1_5_p_t
- typedef
  boost::spirit::qi::int_parser
  < int, 10, 0, 5 > int0_5_p_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 1, 4 > uint1_4_p_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 1, 9 > uint1_9_p_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 0, 9 > uint0_9_p_t
- typedef
  boost::spirit::qi::uint_parser
  < hour_t, 10, 2, 2 > hour_p_t
- typedef
  boost::spirit::qi::uint_parser
  < minute_t, 10, 2, 2 > minute_p_t
- typedef
  boost::spirit::qi::uint_parser
  < second_t, 10, 2, 2 > second_p_t

- typedef boost::spirit::qi::uint_parser < year_t, 10, 4, 4 > year_p_t
- typedef boost::spirit::qi::uint_parser < month_t, 10, 2, 2 > month_p_t
- typedef boost::spirit::qi::uint_parser < day_t, 10, 2, 2 > day_p_t
- typedef boost::spirit::qi::bool_parser < bool, short_bool_policies > boolean_p_t
- typedef long unsigned int u_int32_t
- typedef std::list < MatchingDocuments > DocumentList_T
- typedef std::pair < BomType::EN_BomType, BomAbstract * > GenericBom_T
- typedef std::map< std::size_t, GenericBom_T > GenericBomList_T
- typedef std::multimap < std::string, Place * > PlaceList_T
- typedef std::list< Place * > PlaceOrderedList_T
- typedef std::pair < Xapian::Document, ScoreBoard > XapianDocumentPair_T
- typedef std::map < Xapian::docid, XapianDocumentPair_T > DocumentMap_T
- typedef std::list< ResultHolder * > ResultHolderList_T
- typedef std::list< Result * > ResultList_T
- typedef std::list < StringPartition > StringPartitionList_T
- typedef std::size_t WorldID_T
- typedef std::list< std::string > DBParamsNameList_T
- typedef std::list< Location > LocationList_T
- typedef std::list< std::string > NameList_T
- typedef std::map < LanguageCode_T, Names > NameMatrix_T
- typedef date_time_element< 0, 23 > hour_t
- typedef date_time_element< 0, 59 > minute_t
- typedef date_time_element< 0, 59 > second_t
- typedef date_time_element< 1900, 2100 > year_t
- typedef date_time_element< 1, 12 > month_t
- typedef date_time_element< 1, 31 > day_t
- typedef int GeonamesID_T
- typedef int EnvelopeID_T
- typedef std::list < AltNameShortListString_T > AltNameShortList_T
- typedef std::list < TvlPORListString_T > TvlPORList_T
- typedef std::list< FeatureName_T > FeatureNameList_T
- typedef double GeoCoord_T
- typedef GeoCoord_T Latitude_T
- typedef GeoCoord_T Longitude_T
- typedef boost::gregorian::date Date_T
- typedef unsigned int Population_T

- typedef int Elevation_T
- typedef int GTopo30_T
- typedef double PageRank_T
- typedef int XapianDocID_T
- typedef float GMTOffset_T
- typedef float DSTOffset_T
- typedef float RawOffset_T
- typedef bool IsGeonames_T
- typedef bool IsAirport_T
- typedef bool IsCommercial_T
- typedef unsigned short Weight_T
- typedef std::string TravelQuery_T
- typedef double MatchingPercentage_T
- typedef double Percentage_T
- typedef double Score_T
- typedef unsigned int NbOfDBEntries_T
- typedef std::string Word_T
- typedef std::list< Word_T > WordList_T
- typedef std::set< std::string > WordSet_T
- typedef std::set< std::string > BlackList_T
- typedef unsigned int NbOfLetters_T
- typedef unsigned short NbOfWords_T
- typedef unsigned short NbOfMatches_T
- typedef unsigned short NbOfErrors_T
- typedef std::map < NbOfLetters_T, NbOfErrors_T > DistanceErrorScale_T
- typedef boost::array < NbOfLetters_T, 5 > DistanceErrorScaleArray_T

Functions

- const std::string DEFAULT_OPENTREP_POR_FILEPATH (OPENTREP_POR_DATA_DIR"/ori_por_public.csv")
- const std::string DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH ("/tmp/opentrep/traveldb")
- const std::string DEFAULT_OPENTREP_MYSQL_DB_USER ("geo")
- const std::string DEFAULT_OPENTREP_MYSQL_DB_PASSWD ("geo")
- const std::string DEFAULT_OPENTREP_MYSQL_DB_DBNAME ("geo_ori")
- const std::string DEFAULT_OPENTREP_MYSQL_DB_HOST ("localhost")
- const std::string DEFAULT_OPENTREP_MYSQL_DB_PORT ("3306")
- const std::string DEFAULT_OPENTREP_SQLITE_DB_FILEPATH (OPENTREP_POR_DATA_DIR"/ori_por_public.db")
- const std::string DEFAULT_DATE_STR ("1970-01-01")
- const NbOfMatches_T K_DEFAULT_XAPIAN_MATCHING_SET_SIZE (30)
- const Weight_T K_DEFAULT_INDEXING_STD_WEIGHT (1)
- const Weight_T K_DEFAULT_INDEXING_EXTRA_WEIGHT (2)
- const Percentage_T K_DEFAULT_MODIFIED_MATCHING_PCT (100.0)
- const Percentage_T K_DEFAULT_PAGE_RANK (0.10)
- const Percentage_T K_DEFAULT_ENVELOPE_PCT (K_DEFAULT_PAGE_RANK)
- const Percentage_T K_DEFAULT_FULL_CODE_MATCH_PCT (11000/K_DEFAULT_PAGE_RANK)
- const Percentage_T K_DEFAULT_ATTENUATION_FCTR (34.0)
- const NbOfErrors_T K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT (4)
- bool check (const UErrorCode &iStatus, const char *iMsg)
- UnicodeString & appendHex (const uint32_t iNumber, int8_t ioDigits, UnicodeString &ioTarget)
- UnicodeString escape (const UnicodeString &iSource)

- std::string getUTF8 (const UnicodeString &iString)
- void tokeniseStringIntoWordList (const std::string &iPhrase, WordList_T &ioWordList)
- std::string createStringFromWordList (const WordList_T &iWordList, const unsigned short iSplitIdx, const bool iFromBeginningFlag)
- bool hasGoodSize (const std::string &iWord, const NbOfLetters_T &iMinWordLength)
- bool isBlackListed (const std::string &iWord)
- void rtrim (WordList_T &ioWordList, const NbOfLetters_T &iMinWordLength)
- void ltrim (WordList_T &ioWordList, const NbOfLetters_T &iMinWordLength)
- void trim (WordList_T &ioWordList, const NbOfLetters_T &iMinWordLength)
- static unsigned int calculateEditDistance (const TravelQuery_T &iPhrase)

  Helper function.
- bool doesMatch (const Xapian::Database &iDatabase, const std::string &iWord1, const std::string &iWord2)

  Helper function to query for a Xapian-based full text match.
- static unsigned int calculateEditDistance (const TravelQuery_T &iPhrase)

  Helper function.
- static const boost::char_separator< char > AllSepatorList (" .,;:|+-*/_=!@#$%‘~^&(){}[]?’<>\"")
- static const boost::char_separator< char > DocSepatorList (" ,-%")
- void baseTokeniseStringIntoWordList (const std::string &iPhrase, WordList_T &ioWordList, const boost::char_separator< char > &iSepatorList)
- void addToXapian (const Place &iPlace, Xapian::Document &ioDocument, Xapian::WritableDatabase &ioDatabase)
- void addUnmatchedWord (const TravelQuery_T &iQueryString, WordList_T &ioWordList, WordSet_T &ioWordSet)
- void createPlaces (const ResultCombination &iResultCombination, PlaceHolder &ioPlaceHolder)
- void searchString (const StringPartition &iStringPartition, const Xapian::Database &iDatabase, ResultCombination &ioResultCombination, WordList_T &ioWordList)
- void chooseBestMatchingResultHolder (ResultCombination &ioResultCombination)
- boost::filesystem::path checkTravelDBFilePath (const TravelDBFilePath_T &iTravelDBFilePath)
- template<int MIN, int MAX> date_time_element< MIN, MAX > operator* (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)
- template<int MIN, int MAX> date_time_element< MIN, MAX > operator+ (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)
- void logInit (const LOG::EN_LogLevel iLogLevel, std::ostream &ioLogOutputFile)

Variables

- const std::tm DEFAULT_DATE_TM
- const char * K_ICU_ACCENT_REMOVAL_RULE = "NFD; [:M:] Remove; NFC;"
- const char * K_ICU_QUOTATION_REMOVAL_RULE
- const char * K_ICU_PUNCTUATION_REMOVAL_RULE = "[:P:] Remove;"
- const char * K_ICU_GENERIC_TRANSLITERATOR_RULE
- const BlackList_T K_BLACK_LIST = ListGenerator::createBlackList()
- const NbOfMatches_T K_DEFAULT_XAPIAN_MATCHING_SET_SIZE
- const Weight_T K_DEFAULT_INDEXING_STD_WEIGHT
- const Weight_T K_DEFAULT_INDEXING_EXTRA_WEIGHT
- const Percentage_T K_DEFAULT_MODIFIED_MATCHING_PCT
- const Percentage_T K_DEFAULT_PAGE_RANK
- const Percentage_T K_DEFAULT_ENVELOPE_PCT
- const Percentage_T K_DEFAULT_FULL_CODE_MATCH_PCT
- const Percentage_T K_DEFAULT_ATTENUATION_FCTR

- const NbOfErrors_T K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT
- const std::string DEFAULT_DATE_STR
- const std::string DEFAULT_OPENTREP_POR_FILEPATH
- const std::string DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH
- const std::string DEFAULT_OPENTREP_MYSQL_DB_USER
- const std::string DEFAULT_OPENTREP_MYSQL_DB_PASSWD
- const std::string DEFAULT_OPENTREP_MYSQL_DB_DBNAME
- const std::string DEFAULT_OPENTREP_MYSQL_DB_HOST
- const std::string DEFAULT_OPENTREP_MYSQL_DB_PORT
- const std::string DEFAULT_OPENTREP_SQLITE_DB_FILEPATH
- static const u_int32_t offsetsFromUTF8 [6]
- static const char trailingBytesForUTF8 [256]
- static const DistanceErrorScaleArray_T K_DEFAULT_ERROR_SCALE

22.1.1 Typedef Documentation

22.1.1.1 typedef std::istreambuf_iterator<char> OPENTREP::base_iterator_t

Definition at line 27 of file BasParserTypes.hpp.

22.1.1.2 typedef boost::spirit::multi_pass<base_iterator_t> OPENTREP::iterator_t

Definition at line 28 of file BasParserTypes.hpp.

22.1.1.3 typedef boost::spirit::classic::position_iterator2<iterator_t> OPENTREP::pos_iterator_t

Definition at line 31 of file BasParserTypes.hpp.

22.1.1.4 typedef boost::spirit::qi::uint_parser<int, 10, 1, 1> OPENTREP::uint1_p_t

1-digit-integer parser

Definition at line 40 of file BasParserTypes.hpp.

22.1.1.5 typedef boost::spirit::qi::uint_parser<int, 10, 2, 2> OPENTREP::uint2_p_t

2-digit-integer parser

Definition at line 43 of file BasParserTypes.hpp.

22.1.1.6 typedef boost::spirit::qi::uint_parser<int, 10, 1, 2> OPENTREP::int1_2_p_t

Up-to-2-digit-integer parser

Definition at line 46 of file BasParserTypes.hpp.

22.1.1.7 typedef boost::spirit::qi::uint_parser<int, 10, 4, 4> OPENTREP::uint4_p_t

4-digit-integer parser

Definition at line 49 of file BasParserTypes.hpp.

22.1.1.8 typedef boost::spirit::qi::int_parser<int, 10, 1, 5> OPENTREP::int1_5_p_t

Up-to-5-digit-integer parser

Definition at line 52 of file BasParserTypes.hpp.

22.1.1.9 typedef boost::spirit::qi::int_parser<int, 10, 0, 5> OPENTREP::int0_5_p_t

Definition at line 53 of file BasParserTypes.hpp.

22.1.1.10 typedef boost::spirit::qi::uint_parser<int, 10, 1, 4> OPENTREP::uint1_4_p_t

Up-to-4-digit-integer parser

Definition at line 56 of file BasParserTypes.hpp.

22.1.1.11 typedef boost::spirit::qi::uint_parser<int, 10, 1, 9> OPENTREP::uint1_9_p_t

Up-to-9-digit-integer parser

Definition at line 59 of file BasParserTypes.hpp.

22.1.1.12 typedef boost::spirit::qi::uint_parser<int, 10, 0, 9> OPENTREP::uint0_9_p_t

Definition at line 60 of file BasParserTypes.hpp.

22.1.1.13 typedef boost::spirit::qi::uint_parser<hour_t, 10, 2, 2> OPENTREP::hour_p_t

Date & time element parsers

Definition at line 63 of file BasParserTypes.hpp.

22.1.1.14 typedef boost::spirit::qi::uint_parser<minute_t, 10, 2, 2> OPENTREP::minute_p_t

Definition at line 64 of file BasParserTypes.hpp.

22.1.1.15 typedef boost::spirit::qi::uint_parser<second_t, 10, 2, 2> OPENTREP::second_p_t

Definition at line 65 of file BasParserTypes.hpp.

22.1.1.16 typedef boost::spirit::qi::uint_parser<year_t, 10, 4, 4> OPENTREP::year_p_t

Definition at line 66 of file BasParserTypes.hpp.

22.1.1.17 typedef boost::spirit::qi::uint_parser<month_t, 10, 2, 2> OPENTREP::month_p_t

Definition at line 67 of file BasParserTypes.hpp.

22.1.1.18 typedef boost::spirit::qi::uint_parser<day_t, 10, 2, 2> OPENTREP::day_p_t

Definition at line 68 of file BasParserTypes.hpp.

22.1.1.19 typedef boost::spirit::qi::bool_parser<bool, short_bool_policies> OPENTREP::boolean_p_t

Definition at line 101 of file BasParserTypes.hpp.

22.1.1.20 typedef long unsigned int OPENTREP::u_int32_t

32-bit unsigned integer

Definition at line 16 of file UTF8Handler.cpp.

22.1.1.21 typedef std::list< XapianDocumentPair_T > OPENTREP::DocumentList_T

List of matching Xapian documents.

(STL) List of Xapian documents and their associated score board.

Definition at line 20 of file DocumentList.hpp.

22.1.1.22 typedef std::pair<BomType::EN_BomType, BomAbstract*> OPENTREP::GenericBom_T

A generic BOM is a pair of a BOM type (enumeration) and a BOM object pointer.

Definition at line 15 of file GenericBom.hpp.

22.1.1.23 typedef std::map<std::size_t, GenericBom_T> OPENTREP::GenericBomList_T

List of generic BOM objects.

Definition at line 22 of file GenericBom.hpp.

22.1.1.24 typedef std::multimap<std::string, Place*> OPENTREP::PlaceList_T

Definition at line 14 of file PlaceList.hpp.

22.1.1.25 typedef std::list<Place*> OPENTREP::PlaceOrderedList_T

Definition at line 21 of file PlaceList.hpp.

22.1.1.26 typedef std::pair<Xapian::Document, ScoreBoard> OPENTREP::XapianDocumentPair_T

Pair of a Xapian document and its associated score board.

Definition at line 23 of file Result.hpp.

22.1.1.27 typedef std::map<Xapian::docid, XapianDocumentPair_T> OPENTREP::DocumentMap_T

(STL) Map of Xapian documents and their associated score board.

Definition at line 40 of file Result.hpp.

22.1.1.28 typedef std::list<ResultHolder*> OPENTREP::ResultHolderList_T

List of ResultHolder objects.

Definition at line 13 of file ResultHolderList.hpp.

22.1.1.29 typedef std::list<Result*> OPENTREP::ResultList_T

List of Result objects.

Definition at line 13 of file ResultList.hpp.

22.1.1.30 typedef std::list<StringPartition> OPENTREP::StringPartitionList_T

Type gathering all the partitions of a string.

Definition at line 166 of file StringPartition.hpp.

22.1.1.31 typedef std::size_t OPENTREP::WorldID_T

Definition at line 92 of file World.hpp.

22.1.1.32 typedef std::list<std::string> OPENTREP::DBParamsNameList_T

List of names for a given (geographical) dbparams.

Definition at line 19 of file DBParams.hpp.

22.1.1.33 typedef std::list<Location> OPENTREP::LocationList_T

List of (geographical) location structures.

Definition at line 13 of file LocationList.hpp.

22.1.1.34 typedef std::list<std::string> OPENTREP::NameList_T

Definition at line 20 of file Names.hpp.

22.1.1.35 typedef std::map<LanguageCode_T, Names> OPENTREP::NameMatrix_T

Matrix of place names: for each of the language, the place gets a corresponding list of names.

For instance, MUC corresponds to Munich in English, München in German, Munique in French, in Russian, etc.

Definition at line 149 of file Names.hpp.

22.1.1.36 typedef date_time_element<0, 23> OPENTREP::hour_t

Type definitions for the date and time elements.

Definition at line 60 of file OPENTREP_ParserHelperTypes.hpp.

22.1.1.37 typedef date_time_element<0, 59> OPENTREP::minute_t

Definition at line 61 of file OPENTREP_ParserHelperTypes.hpp.

22.1.1.38 typedef date_time_element<0, 59> OPENTREP::second_t

Definition at line 62 of file OPENTREP_ParserHelperTypes.hpp.

22.1.1.39 typedef date_time_element<1900, 2100> OPENTREP::year_t

Definition at line 63 of file OPENTREP_ParserHelperTypes.hpp.

22.1.1.40 typedef date_time_element<1, 12> OPENTREP::month_t

Definition at line 64 of file OPENTREP_ParserHelperTypes.hpp.

22.1.1.41 typedef date_time_element<1, 31> OPENTREP::day_t

Definition at line 65 of file OPENTREP_ParserHelperTypes.hpp.

22.1.1.42 typedef int OPENTREP::GeonamesID_T

Geonames ID.

Definition at line 91 of file OPENTREP_Types.hpp.

22.1.1.43 typedef int OPENTREP::EnvelopeID_T

Envelope ID.

Definition at line 96 of file OPENTREP_Types.hpp.

22.1.1.44 typedef std::list<AltNameShortListString_T> OPENTREP::AltNameShortList_T

Definition at line 145 of file OPENTREP_Types.hpp.

22.1.1.45 typedef std::list<TvlPORListString_T> OPENTREP::TvlPORList_T

Definition at line 157 of file OPENTREP_Types.hpp.

22.1.1.46 typedef std::list<FeatureName_T> OPENTREP::FeatureNameList_T

Definition at line 349 of file OPENTREP_Types.hpp.

22.1.1.47 typedef double OPENTREP::GeoCoord_T

Geographical coordinate (e.g., -10.45 or 56.7).

Definition at line 354 of file OPENTREP_Types.hpp.

22.1.1.48 typedef GeoCoord_T OPENTREP::Latitude_T

Definition at line 355 of file OPENTREP_Types.hpp.

22.1.1.49 typedef GeoCoord_T OPENTREP::Longitude_T

Definition at line 356 of file OPENTREP_Types.hpp.

22.1.1.50 typedef boost::gregorian::date OPENTREP::Date_T

Date (e.g., modification date).

Definition at line 369 of file OPENTREP_Types.hpp.

22.1.1.51 typedef unsigned int OPENTREP::Population_T

Population (e.g., 400,000).

Definition at line 374 of file OPENTREP_Types.hpp.

22.1.1.52 typedef int OPENTREP::Elevation_T

Geographical elevation (e.g., 300).

Definition at line 379 of file OPENTREP_Types.hpp.

22.1.1.53 typedef int OPENTREP::GTopo30_T

Geographical topology 30.

Definition at line 384 of file OPENTREP_Types.hpp.

22.1.1.54 typedef double OPENTREP::PageRank_T

PageRank (e.g., 94.66%).

Definition at line 397 of file OPENTREP_Types.hpp.

22.1.1.55 typedef int OPENTREP::XapianDocID_T

Xapian document ID.

Definition at line 402 of file OPENTREP_Types.hpp.

22.1.1.56 typedef float OPENTREP::GMTOffset_T

GMT offset (e.g., 1)

Definition at line 407 of file OPENTREP_Types.hpp.

22.1.1.57 typedef float OPENTREP::DSTOffset_T

DST offset (e.g., 2)

Definition at line 412 of file OPENTREP_Types.hpp.

22.1.1.58 typedef float OPENTREP::RawOffset_T

Raw offset (e.g., 1)

Definition at line 417 of file OPENTREP_Types.hpp.

22.1.1.59 typedef bool OPENTREP::IsGeonames_T

Whether or not that POR is referenced by Geonames

Definition at line 422 of file OPENTREP_Types.hpp.

22.1.1.60 typedef bool OPENTREP::IsAirport_T

Whether or not that POR is an airport

Definition at line 427 of file OPENTREP_Types.hpp.

22.1.1.61 typedef bool OPENTREP::IsCommercial_T

Whether or not that POR is for commercial use

Definition at line 432 of file OPENTREP_Types.hpp.

22.1.1.62 typedef unsigned short OPENTREP::Weight_T

Weight when indexing terms of a Xapian document.

Definition at line 438 of file OPENTREP_Types.hpp.

22.1.1.63 typedef std::string OPENTREP::TravelQuery_T

Travel search query.

Definition at line 443 of file OPENTREP_Types.hpp.

22.1.1.64 typedef double OPENTREP::MatchingPercentage_T

Xapian percentage.

Definition at line 448 of file OPENTREP_Types.hpp.

22.1.1.65 typedef double OPENTREP::Percentage_T

Matching percentage for a Xapian document.

Definition at line 453 of file OPENTREP_Types.hpp.

22.1.1.66 typedef double OPENTREP::Score_T

Score (e.g., matching percentage) for a document/POR (point of reference).

Definition at line 458 of file OPENTREP_Types.hpp.

22.1.1.67 typedef unsigned int OPENTREP::NbOfDBEntries_T

Number of entries in the Xapian database.

Definition at line 463 of file OPENTREP_Types.hpp.

22.1.1.68 typedef std::string OPENTREP::Word_T

Word, which is the atomic element of a query string.

Definition at line 468 of file OPENTREP_Types.hpp.

22.1.1.69 typedef std::list<Word_T> OPENTREP::WordList_T

List of words.

Definition at line 473 of file OPENTREP_Types.hpp.

22.1.1.70 typedef std::set<std::string> OPENTREP::WordSet_T

Set of words.

Definition at line 478 of file OPENTREP_Types.hpp.

22.1.1.71 typedef std::set<std::string> OPENTREP::BlackList_T

List of (black-listed) words.

Definition at line 483 of file OPENTREP_Types.hpp.

22.1.1.72 typedef unsigned int OPENTREP::NbOfLetters_T

Number of letters in a word or phrase.

Definition at line 488 of file OPENTREP_Types.hpp.

22.1.1.73 typedef unsigned short OPENTREP::NbOfWords_T

Number of words in a list.

Definition at line 493 of file OPENTREP_Types.hpp.

22.1.1.74 typedef unsigned short OPENTREP::NbOfMatches_T

Number of matching documents.

Definition at line 498 of file OPENTREP_Types.hpp.

22.1.1.75 typedef unsigned short OPENTREP::NbOfErrors_T

Number of errors.

Definition at line 503 of file OPENTREP_Types.hpp.

22.1.1.76 typedef std::map<NbOfLetters_T, NbOfErrors_T> OPENTREP::DistanceErrorScale_T

Number of (distance) errors allowed for a given number of letters.

Definition at line 508 of file OPENTREP_Types.hpp.

22.1.1.77 typedef boost::array<NbOfLetters_T, 5> OPENTREP::DistanceErrorScaleArray_T

Number of (distance) errors allowed for a given number of letters.

Definition at line 513 of file OPENTREP_Types.hpp.

22.1.2 Function Documentation

22.1.2.1 const std::string OPENTREP::DEFAULT_OPENTREP_POR_FILEPATH (OPENTREP_POR_DATA_DIR"/ori_por_public.csv")

Default file-path for the ORI-maintained POR (points of reference) file.

22.1.2.2 const std::string OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH ("/tmp/opentrep/traveldb")

Default file-path for the Xapian database/index.

22.1.2.3 const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_USER ("geo")

Default name and location for the MySQL database (if existing).

22.1.2.4 const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_PASSWD ("geo")

22.1.2.5 const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_DBNAME ("geo_ori")

22.1.2.6 const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_HOST ("localhost")

22.1.2.7 const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_PORT ("3306")

22.1.2.8 const std::string OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH (OPENTREP_POR_DATA_DIR"/ori_por_public.db")

Default name and location for the SQLite3 database.

22.1.2.9 const std::string OPENTREP::DEFAULT_DATE_STR ("1970-01-01")

Default date-time with std::string as type.

22.1.2.10 const NbOfMatches_T OPENTREP::K_DEFAULT_XAPIAN_MATCHING_SET_SIZE (30)

Default size of matching set for Xapian (e.g., 30)

22.1.2.11 const Weight_T OPENTREP::K_DEFAULT_INDEXING_STD_WEIGHT (1)

Default indexing weight for standard terms (e.g., 1)

22.1.2.12 const Weight_T OPENTREP::K_DEFAULT_INDEXING_EXTRA_WEIGHT (2)

Default indexing weight for some terms like IATA/ICAO/FAA codes (e.g., 2)

22.1.2.13 const Percentage_T OPENTREP::K_DEFAULT_MODIFIED_MATCHING_PCT (100. *0*)

Default modified matching percentage (e.g., 99.999% == 0.99999)

22.1.2.14 const Percentage_T OPENTREP::K_DEFAULT_PAGE_RANK (0. *10*)

Default PageRank (e.g., 0.10% == 0.001)

22.1.2.15 const Percentage_T OPENTREP::K_DEFAULT_ENVELOPE_PCT (K_DEFAULT_PAGE_RANK)

Default envelope percentage (e.g., 0.10% == 0.001). Usually equal to K_DEFAULT_PAGE_RANK.

22.1.2.16 const Percentage_T OPENTREP::K_DEFAULT_FULL_CODE_MATCH_PCT (11000/ *K_DEFAULT_PAGE_RANK*)

Default full match percentage for a IATA/ICAO code (e.g., 11,000,000% == 110,000 == 110 / 0.001). Usually equal to 100 ∗ 1.1 / (K_DEFAULT_PAGE_RANK / 100).

22.1.2.17 const Percentage_T OPENTREP::K_DEFAULT_ATTENUATION_FCTR (34. *0*)

Default attenuation factor (e.g., 34.0)

22.1.2.18 const NbOfErrors_T OPENTREP::K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT (4)

Default number of characters by allowable spelling error unit

22.1.2.19 bool OPENTREP::check (const UErrorCode & *iStatus,* const char ∗ *iMsg*)

Verify that the given status corresponds to a successful operation.

Parameters

const	UErrorCode& The status of the last Unicode operation
const	char∗ Error message

Returns

bool Whether or not the operation was successful

Definition at line 18 of file icu_util.cpp.

References OPENTREP_LOG_ERROR.

22.1.2.20 UnicodeString& OPENTREP::appendHex (const uint32_t *iNumber,* int8_t *ioDigits,* UnicodeString & *ioTarget*)

Definition at line 34 of file icu_util.cpp.

Referenced by escape().

22.1.2.21 UnicodeString OPENTREP::escape (const UnicodeString &)

Replace non-printable characters with Unicode escapes.

Parameters

const	UnicodeString& ICU Unicode string

Returns

UnicodeString ICU Unicode string with escaped characters

Definition at line 45 of file icu_util.cpp.

References appendHex().

22.1.2.22 std::string OPENTREP::getUTF8 (const UnicodeString &)

Get a STL string from the given UnicodeString.

Parameters

const	UnicodeString& ICU Unicode string

Returns

std::string STL string

Definition at line 65 of file icu_util.cpp.

Referenced by OPENTREP::OTransliterator::normalise().

22.1.2.23 void OPENTREP::tokeniseStringIntoWordList (const std::string & *iPhrase,* WordList_T &)

Split a string into a list of tokens.

Definition at line 16 of file Utilities.cpp.

Referenced by OPENTREP::Place::addNameToXapianSets(), and OPENTREP::Filter::trim().

22.1.2.24 std::string OPENTREP::createStringFromWordList (const WordList_T & , const unsigned short *iSplitIdx* = 0, const bool *iFromBeginningFlag* = true)

Create a string from a list of words.

The string is made of the words up until a given index. The default value of that index (0) means that the string will be made by all the words of the list.

If the flag is set to false, the string is made of the words from the given index up until the end of the list.

Definition at line 38 of file Utilities.cpp.

Referenced by OPENTREP::Place::addNameToXapianSets(), and OPENTREP::Filter::trim().

22.1.2.25 bool OPENTREP::hasGoodSize (const std::string & *iWord,* const NbOfLetters_T & *iMinWordLength*)

Helper function to check whether the word should be kept.

If the word has no more than <iMinWordLength> letters (e.g., 'de', 'san'), it should be filtered out. Indeed, when 'san' is part of 'san francisco', for instance, it should not be indexed/searched alone (in a search, the resulting match score will be zero).

Definition at line 39 of file Filter.cpp.

Referenced by ltrim(), rtrim(), and OPENTREP::Filter::shouldKeep().

22.1.2.26 bool OPENTREP::isBlackListed (const std::string & *iWord*)

Helper function to check whether the given word is black-listed.

Definition at line 53 of file Filter.cpp.

References K_BLACK_LIST.

Referenced by ltrim(), rtrim(), and OPENTREP::Filter::shouldKeep().

22.1.2.27 void OPENTREP::rtrim (WordList_T & *ioWordList,* const NbOfLetters_T & *iMinWordLength*)

Helper function to trim the non-relevant right outer words.

Definition at line 72 of file Filter.cpp.

References hasGoodSize(), and isBlackListed().

Referenced by trim().

22.1.2.28 void OPENTREP::ltrim (WordList_T & *ioWordList,* const NbOfLetters_T & *iMinWordLength*)

Helper function to trim the non-relevant left outer words.

Definition at line 97 of file Filter.cpp.

References hasGoodSize(), and isBlackListed().

Referenced by trim().

22.1.2.29 void OPENTREP::trim (WordList_T & *ioWordList,* const NbOfLetters_T & *iMinWordLength*)

Helper function to trim the non-relevant left and right outer words.

Definition at line 122 of file Filter.cpp.

References ltrim(), and rtrim().

Referenced by OPENTREP::Filter::trim().

22.1.2.30 static unsigned int OPENTREP::calculateEditDistance (const TravelQuery_T & *iPhrase*) [static]

Helper function.

Given the size of the phrase, determine the allowed edit distance for spelling purpose. For instance, if K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT is equal to 4:

- An edit distance of 1 will be allowed on a 4-letter word
- While an edit distance of 3 will be allowed on an 11-letter word.

Definition at line 108 of file QuerySlices.cpp.

References K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT.

Referenced by doesMatch().

22.1.2.31 bool OPENTREP::doesMatch (const Xapian::Database & *iDatabase,* const std::string & *iWord1,* const std::string & *iWord2*)

Helper function to query for a Xapian-based full text match.

Definition at line 121 of file QuerySlices.cpp.

References calculateEditDistance(), OPENTREP::Levenshtein::getDistance(), and OPENTREP_LOG_ERROR.

22.1.2.32 static unsigned int OPENTREP::calculateEditDistance (const TravelQuery_T & *iPhrase*) [static]

Helper function.

Given the size of the phrase, determine the allowed edit distance for spelling purpose. For instance, if K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT is equal to 4:

- An edit distance of 1 will be allowed on a 4-letter word
- While an edit distance of 3 will be allowed on an 11-letter word.

Definition at line 242 of file Result.cpp.

References K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT.

22.1.2.33 static const boost::char_separator<char> OPENTREP::AllSepatorList (" ., ;:|+-*/ _ = !@#$%`~^&(){}[]?'<>\"") [static]

Referenced by OPENTREP::WordHolder::tokeniseStringIntoWordList().

22.1.2.34 static const boost::char_separator<char> OPENTREP::DocSepatorList (" , -%") [static]

Referenced by OPENTREP::WordHolder::tokeniseDocIntoWordList().

22.1.2.35 void OPENTREP::baseTokeniseStringIntoWordList (const std::string & *iPhrase,* WordList_T & *ioWordList,* const boost::char_separator< char > & *iSepatorList*)

Definition at line 18 of file WordHolder.cpp.

Referenced by OPENTREP::WordHolder::tokeniseDocIntoWordList(), and OPENTREP::WordHolder::tokeniseStringIntoWordList().

22.1.2.36 void OPENTREP::addToXapian (const Place & *iPlace,* Xapian::Document & *ioDocument,* Xapian::WritableDatabase & *ioDatabase*)

Definition at line 30 of file IndexBuilder.cpp.

References OPENTREP::Place::describeKey(), OPENTREP::Place::describeSets(), OPENTREP::Place::getSpellingSet(), OPENTREP::Place::getTermSetMap(), and OPENTREP_LOG_DEBUG.

22.1.2.37 void OPENTREP::addUnmatchedWord (const TravelQuery_T & *iQueryString,* WordList_T & *ioWordList,* WordSet_T & *ioWordSet*)

Helper function to add the given string to the list of unknown words, only when that given string is made of a single word. Otherwise, the given string is simply ignored.

Parameters

const	TravelQuery_T& The given string
WordList_T&	ioWordList The list to which the given single-word string should be added (if needed).
WordSet_T&	A (STL) set of words, allowing to keep track of the already added words.

Definition at line 44 of file RequestInterpreter.cpp.

References OPENTREP::Filter::shouldKeep(), and OPENTREP::WordHolder::tokeniseStringIntoWordList().

Referenced by searchString().

22.1.2.38 void OPENTREP::createPlaces (const ResultCombination & *iResultCombination,* PlaceHolder & *ioPlaceHolder*)

Definition at line 65 of file RequestInterpreter.cpp.

References OPENTREP::FacPlace::create(), OPENTREP::Result::fillPlace(), OPENTREP::Result::getBestDocData(), OPENTREP::ResultCombination::getBestMatchingResultHolder(), OPENTREP::ResultHolder::getResultList(), OPENTREP::Result::hasFullTextMatched(), OPENTREP::FacPlaceHolder::initLinkWithPlace(), OPENTRE-

P::FacPlace::instance(), OPENTREP_LOG_DEBUG, OPENTREP::Result::retrieveLocation(), and OPENTREP::-Place::toString().

22.1.2.39 void OPENTREP::searchString (const StringPartition & *iStringPartition,* const Xapian::Database & *iDatabase,* ResultCombination & *ioResultCombination,* WordList_T & *ioWordList*)

For all the elements (StringSet) of the string partitions, derived from the given travel query, perform a Xapian-based full-text match. Each Xapian-based full-text match gives (potentially) a full set of matches, some with the highest matching percentage and some with a lower percentage.

Parameters

TravelQuery_T&	The query string.
const	Xapian::Database& The Xapian index/database.
Result-Combination&	List of ResultHolder objects.
WordList_T&	List of non-matched words of the query string.

Definition at line 125 of file RequestInterpreter.cpp.

References OPENTREP::StringPartition::_partition, OPENTREP::StringSet::_set, addUnmatchedWord(), OPENTREP::FacResultHolder::create(), OPENTREP::FacResult::create(), OPENTREP::StringSet::describe(), OPENTREP::Result::fullTextMatch(), OPENTREP::FacResultHolder::initLinkWithResult(), OPENTREP::FacResultCombination::initLinkWithResultHolder(), OPENTREP::FacResult::instance(), OPENTREP::FacResultHolder::instance(), OPENTREP_LOG_DEBUG, OPENTREP_LOG_ERROR, and OPENTREP::ResultHolder::toString().

22.1.2.40 void OPENTREP::chooseBestMatchingResultHolder (ResultCombination & *ioResultCombination*)

Select the best matching string partition, based on the results of several rules, that are all materialised by weighting percentages:

- Xapian-based full-text match. Score type: XAPIAN_PCT
- Schedule-derived PageRank for the point of reference (POR). Score type: PAGE_RANK
- Heuristic. Score type: Heuristic

See Also

ScoreType.hpp for the various types of score.

The score for the combination (score type: COMBINATION) is simply the product of all the scores/weighting percentages.

Parameters

Result-Combination&	List of ResultHolder objects.

Definition at line 220 of file RequestInterpreter.cpp.

References OPENTREP::ResultCombination::chooseBestMatchingResultHolder(), OPENTREP::ResultHolder::describeShortKey(), OPENTREP::ResultCombination::describeShortKey(), OPENTREP::ResultCombination::getBestMatchingResultHolder(), OPENTREP::ResultCombination::getBestMatchingWeight(), OPENTREP::ResultCombination::getCorrectedStringSet(), and OPENTREP_LOG_DEBUG.

22.1.2.41 boost::filesystem::path OPENTREP::checkTravelDBFilePath (const TravelDBFilePath_T & *iTravelDBFilePath*)

Definition at line 26 of file XapianIndexManager.cpp.

References OPENTREP_LOG_ERROR.

22.1.2.42 template<int MIN, int MAX> date_time_element<MIN, MAX> OPENTREP::operator* (const date_time_element< MIN, MAX > & *o1,* const date_time_element< MIN, MAX > & *o2*) [inline]

Operator* overload.

Definition at line 46 of file OPENTREP_ParserHelperTypes.hpp.

References OPENTREP::date_time_element< MIN, MAX >::_value.

22.1.2.43 template<int MIN, int MAX> date_time_element<MIN, MAX> OPENTREP::operator+ (const date_time_element< MIN, MAX > & *o1,* const date_time_element< MIN, MAX > & *o2*) [inline]

Operator+ overload.

Definition at line 54 of file OPENTREP_ParserHelperTypes.hpp.

References OPENTREP::date_time_element< MIN, MAX >::_value.

22.1.2.44 void OPENTREP::logInit (const LOG::EN_LogLevel *iLogLevel,* std::ostream & *ioLogOutputFile*)

Definition at line 61 of file OPENTREP_Service.cpp.

References OPENTREP::Logger::instance(), and OPENTREP::Logger::setLogParameters().

22.1.3 Variable Documentation

22.1.3.1 const std::tm OPENTREP::DEFAULT_DATE_TM

Initial value:

```
=
    boost::gregorian::to_tm (Date_T (1970, boost::gregorian::Jan, 01))
```

Default date-time with std::tm structure as type.

Default std::tm (date-time) structure (e.g., 1970-JAN-01).

Definition at line 42 of file BasConst.cpp.

22.1.3.2 const char * OPENTREP::K_ICU_ACCENT_REMOVAL_RULE = "NFD; [:M:] Remove; NFC;"

Default Unicode transliterator rule (e.g., "NFD; [:M:] Remove; NFC;")

Default Unicode transliterator rule for the removal of accents (e.g., "NFD; [:M:] Remove; NFC;").

Definition at line 54 of file BasConst.cpp.

22.1.3.3 const char * OPENTREP::K_ICU_QUOTATION_REMOVAL_RULE

Initial value:

```
=
    "[\\u02B9] > \\u0027; [\\u002D] > \\u0020; [\\u0027] > \\u0020;"
```

Default Unicode transliterator rule for the removal of quotes (e.g., "[\\u02B9] > \\u0027; [\\u002D] > \\u0020;")

Definition at line 60 of file BasConst.cpp.

22.1.3.4 const char * OPENTREP::K_ICU_PUNCTUATION_REMOVAL_RULE = "[:P:] Remove;"

Default Unicode transliterator rule for the removal of punctuation (e.g., "[:P:] Remove;")

Definition at line 67 of file BasConst.cpp.

22.1.3.5 const char * OPENTREP::K_ICU_GENERIC_TRANSLITERATOR_RULE

Initial value:

```
=
    "Any-Latin; NFD; [:M:] Remove; NFC; Lower;"
```

Default Unicode transliterator rule for the removal of punctuation (e.g., "Any-Latin;")

Definition at line 73 of file BasConst.cpp.

22.1.3.6 const BlackList_T OPENTREP::K_BLACK_LIST = ListGenerator::createBlackList()

Black list, i.e., a list of words which should not be indexed and/or searched for (e.g., "airport", "international").

Definition at line 130 of file BasConst.cpp.

Referenced by isBlackListed().

22.1.3.7 const NbOfMatches_T OPENTREP::K_DEFAULT_XAPIAN_MATCHING_SET_SIZE

Default size of matching set for Xapian (e.g., 30)

22.1.3.8 const Weight_T OPENTREP::K_DEFAULT_INDEXING_STD_WEIGHT

Default indexing weight for standard terms (e.g., 1)

Referenced by OPENTREP::Place::buildIndexSets().

22.1.3.9 const Weight_T OPENTREP::K_DEFAULT_INDEXING_EXTRA_WEIGHT

Default indexing weight for some terms like IATA/ICAO/FAA codes (e.g., 2)

Referenced by OPENTREP::Place::buildIndexSets().

22.1.3.10 const Percentage_T OPENTREP::K_DEFAULT_MODIFIED_MATCHING_PCT

Default modified matching percentage (e.g., 99.999% == 0.99999)

Referenced by OPENTREP::Result::calculateCodeMatches(), and OPENTREP::ScoreBoard::setScore().

22.1.3.11 const Percentage_T OPENTREP::K_DEFAULT_PAGE_RANK

Default PageRank (e.g., 0.10% == 0.001)

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

22.1.3.12 const Percentage_T OPENTREP::K_DEFAULT_ENVELOPE_PCT

Default envelope percentage (e.g., 0.10% == 0.001). Usually equal to K_DEFAULT_PAGE_RANK.

Referenced by OPENTREP::ScoreBoard::setScore().

22.1.3.13 const Percentage_T OPENTREP::K_DEFAULT_FULL_CODE_MATCH_PCT

Default full match percentage for a IATA/ICAO code (e.g., 11,000,000% == 110,000 == 110 / 0.001). Usually equal to 100 * 1.1 / (K_DEFAULT_PAGE_RANK / 100).

Referenced by OPENTREP::Result::calculateCodeMatches(), and OPENTREP::ScoreBoard::setScore().

22.1.3.14 const Percentage_T OPENTREP::K_DEFAULT_ATTENUATION_FCTR

Default attenuation factor (e.g., 34.0)

Referenced by OPENTREP::ResultHolder::calculateCombinedWeights().

22.1.3.15 const NbOfErrors_T OPENTREP::K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT

Default number of characters by allowable spelling error unit

Referenced by calculateEditDistance().

22.1.3.16 const std::string OPENTREP::DEFAULT_DATE_STR

Default date-time with std::string as type (e.g., "1970-01-01").

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

22.1.3.17 const std::string OPENTREP::DEFAULT_OPENTREP_POR_FILEPATH

Default file-path for the ORI-maintained POR (points of reference) file.

Usually, the default file-path on Linux/Unix is `<installation directory>="">/share/opentrep/data/ori_por_public.csv` where `<installation directory>="">` may be:

- `/usr` when installed as root (e.g., with the RPM)
- `$HOME/dev/deliveries/opentrep-<version>`

Referenced by readConfiguration().

22.1.3.18 const std::string OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH

Default file-path for the Xapian database/index.

Usually, the default file-path on Linux/Unix is `/tmp/opentrep/traveldb`

Referenced by readConfiguration().

22.1.3.19 const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_USER

Default name and location for the MySQL database (if existing; currently, it does not).

Usually, the default MySQL database details are:

- User: geo
- Passwd: geo
- DB name: geo_ori
- Server hostname: localhost
- Server port: 3306

22.1.3.20 const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_PASSWD

22.1.3.21 const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_DBNAME

22.1.3.22 const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_HOST

22.1.3.23 const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_PORT

22.1.3.24 const std::string OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH

Default name and location for the SQLite3 database.

Usually, the default file-path on Linux/Unix is `<installation directory>="">/share/opentrep/data/ori_por_public.db` where `<installation directory>="">` may be:

- `/usr` when installed as root (e.g., with the RPM)
- `$HOME/dev/deliveries/opentrep-<version>`

Referenced by readConfiguration().

22.1.3.25 const u_int32_t OPENTREP::offsetsFromUTF8[6] [static]

Initial value:

```
= {
    0x00000000UL, 0x00003080UL, 0x000E2080UL,
    0x03C82080UL, 0xFA082080UL, 0x82082080UL
  }
```

Definition at line 19 of file UTF8Handler.cpp.

Referenced by OPENTREP::UTF8Handler::toWideString().

22.1.3.26 const char OPENTREP::trailingBytesForUTF8[256] [static]

Initial value:

```
= {
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
    2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2, 3,3,3,3,3,3,3,4,4,4,4,5,5,5,5
  }
```

Definition at line 25 of file UTF8Handler.cpp.

Referenced by OPENTREP::UTF8Handler::toWideString().

22.1.3.27 const DistanceErrorScaleArray_T OPENTREP::K_DEFAULT_ERROR_SCALE [static]

Initial value:

```
=
    { {3, 6, 9, 14, 19} }
```

Default distance error scale.

Allowed error for a given number of letters:

- < 4 => 0
- < 7 => 1
- < 10 => 2
- < 15 => 3
- < 20 => 4
- >= 20 => (nb of letters) / 5

Definition at line 35 of file DistanceErrorRule.hpp.

22.2 OPENTREP::LOG Namespace Reference

Enumerations

- enum EN_LogLevel {
  CRITICAL = 0, ERROR, NOTIFICATION, WARNING,
  DEBUG, VERBOSE, LAST_VALUE }

22.2.1 Detailed Description

Level of logs.

22.2.2 Enumeration Type Documentation

22.2.2.1 enum OPENTREP::LOG::EN_LogLevel

Enumerator

CRITICAL

ERROR

NOTIFICATION

WARNING

DEBUG

VERBOSE

LAST_VALUE

Definition at line 15 of file OPENTREP_log.hpp.

22.3 OPENTREP::PorParserHelper Namespace Reference

Classes

- struct LocationParser
- struct ParserSemanticAction
- struct storeIataCode
- struct storeIcaoCode
- struct storeFaaCode
- struct storeGeonamesID
- struct storeEnvelopeID
- struct storeCommonName
- struct storeAsciiName
- struct storeAltNameShort
- struct storeAltNameShortListString
- struct storeTvlPORCode
- struct storeTvlPORListString
- struct storeLatitude
- struct storeLongitude
- struct storeFeatureClass
- struct storeFeatureCode
- struct storePageRank
- struct storeDateFrom
- struct storeDateUntil
- struct storeComments
- struct storeCountryCode
- struct storeAltCountryCode
- struct storeCountryName
- struct storeContinentName
- struct storeAdm1Code
- struct storeAdm1UtfName
- struct storeAdm1AsciiName
- struct storeAdm2Code

- struct storeAdm2UtfName
- struct storeAdm2AsciiName
- struct storeAdm3Code
- struct storeAdm4Code
- struct storePopulation
- struct storeElevation
- struct storeGTopo30
- struct storeTimeZone
- struct storeGMTOffset
- struct storeDSTOffset
- struct storeRawOffset
- struct storeModDate
- struct storeCityCode
- struct storeCityUtfName
- struct storeCityAsciiName
- struct storeCityGeonamesID
- struct storeStateCode
- struct storePORType
- struct storeWikiLink
- struct storeAltLangCodeFull
- struct storeAltLangCode2Char
- struct storeAltLangCodeExt
- struct storeAltLangCodeHist
- struct storeAltName
- struct doEndPor

Typedefs

- typedef boost::uint32_t uchar_t

Variables

- OPENTREP::uint1_p_t uint1_p
- OPENTREP::uint2_p_t uint2_p
- OPENTREP::int1_2_p_t int1_2_p
- OPENTREP::uint4_p_t uint4_p
- OPENTREP::int1_5_p_t int1_5_p
- OPENTREP::int0_5_p_t int0_5_p
- OPENTREP::uint1_4_p_t uint1_4_p
- OPENTREP::uint1_9_p_t uint1_9_p
- OPENTREP::uint0_9_p_t uint0_9_p
- OPENTREP::hour_p_t hour_p
- OPENTREP::minute_p_t minute_p
- OPENTREP::second_p_t second_p
- OPENTREP::year_p_t year_p
- OPENTREP::month_p_t month_p
- OPENTREP::day_p_t day_p
- OPENTREP::boolean_p_t boolean_p

22.3.1 Detailed Description

Namespaces.

22.3.2 Typedef Documentation

22.3.2.1 typedef boost::uint32_t OPENTREP::PorParserHelper::uchar_t

Unicode character.

Definition at line 21 of file PORParserHelper.hpp.

22.3.3 Variable Documentation

22.3.3.1 OPENTREP::uint1_p_t OPENTREP::PorParserHelper::uint1_p

1-digit-integer parser

Definition at line 857 of file PORParserHelper.cpp.

22.3.3.2 OPENTREP::uint2_p_t OPENTREP::PorParserHelper::uint2_p

2-digit-integer parser

Definition at line 860 of file PORParserHelper.cpp.

22.3.3.3 OPENTREP::int1_2_p_t OPENTREP::PorParserHelper::int1_2_p

Up-to-2-digit-integer parser

Definition at line 863 of file PORParserHelper.cpp.

22.3.3.4 OPENTREP::uint4_p_t OPENTREP::PorParserHelper::uint4_p

4-digit-integer parser

Definition at line 866 of file PORParserHelper.cpp.

22.3.3.5 OPENTREP::int1_5_p_t OPENTREP::PorParserHelper::int1_5_p

Up-to-5-digit-integer parser

Definition at line 869 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

22.3.3.6 OPENTREP::int0_5_p_t OPENTREP::PorParserHelper::int0_5_p

Definition at line 870 of file PORParserHelper.cpp.

22.3.3.7 OPENTREP::uint1_4_p_t OPENTREP::PorParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 873 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

22.3.3.8 OPENTREP::uint1_9_p_t OPENTREP::PorParserHelper::uint1_9_p

Up-to-9-digit-integer parser

Definition at line 876 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

22.3.3.9 OPENTREP::uint0_9_p_t OPENTREP::PorParserHelper::uint0_9_p

Definition at line 877 of file PORParserHelper.cpp.

22.3.3.10 OPENTREP::hour_p_t OPENTREP::PorParserHelper::hour_p

Time element parsers.

Definition at line 880 of file PORParserHelper.cpp.

22.3.3.11 OPENTREP::minute_p_t OPENTREP::PorParserHelper::minute_p

Definition at line 881 of file PORParserHelper.cpp.

22.3.3.12 OPENTREP::second_p_t OPENTREP::PorParserHelper::second_p

Definition at line 882 of file PORParserHelper.cpp.

22.3.3.13 OPENTREP::year_p_t OPENTREP::PorParserHelper::year_p

Date element parsers.

Definition at line 885 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

22.3.3.14 OPENTREP::month_p_t OPENTREP::PorParserHelper::month_p

Definition at line 886 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

22.3.3.15 OPENTREP::day_p_t OPENTREP::PorParserHelper::day_p

Definition at line 887 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

22.3.3.16 OPENTREP::boolean_p_t OPENTREP::PorParserHelper::boolean_p

Boolean parser

Definition at line 890 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

22.4 soci Namespace Reference

Classes

- struct type_conversion< OPENTREP::Place >

  Specify how the Place class can be converted to (resp. from) values stored into (resp. retrieved from) database, using the SOCI framework.

22.5 treppb Namespace Reference

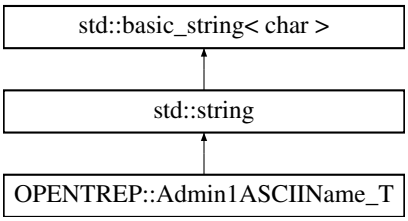
22.6 Xapian Namespace Reference

23 Class Documentation

23.1 OPENTREP::Admin1ASCIIName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::Admin1ASCIIName_T:



Public Member Functions

- Admin1ASCIIName_T (const std::string &iValue)

23.1.1 Detailed Description

Administrative ASCII name of rank 1 (e.g., Illinois).

Definition at line 277 of file OPENTREP_Types.hpp.

23.1.2 Constructor & Destructor Documentation

23.1.2.1 OPENTREP::Admin1ASCIIName_T::Admin1ASCIIName_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 279 of file OPENTREP_Types.hpp.

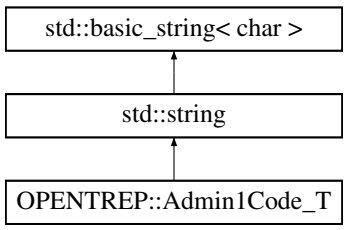
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.2 OPENTREP::Admin1Code_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::Admin1Code_T:



Public Member Functions

- Admin1Code_T (const std::string &iValue)

23.2.1 Detailed Description

Administrative code of rank 1 (e.g., IL for Illinois).

Definition at line 260 of file OPENTREP_Types.hpp.

23.2.2 Constructor & Destructor Documentation

23.2.2.1 OPENTREP::Admin1Code_T::Admin1Code_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 262 of file OPENTREP_Types.hpp.

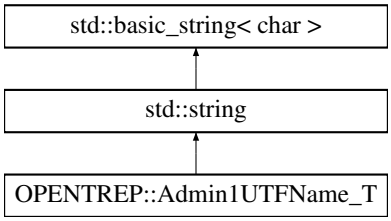
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.3 OPENTREP::Admin1UTFName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::Admin1UTFName_T:



Public Member Functions

- Admin1UTFName_T (const std::string &iValue)

23.3.1 Detailed Description

Administrative UTF8 name of rank 1 (e.g., Illinois).

Definition at line 268 of file OPENTREP_Types.hpp.

23.3.2 Constructor & Destructor Documentation

23.3.2.1 OPENTREP::Admin1UTFName_T::Admin1UTFName_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 270 of file OPENTREP_Types.hpp.

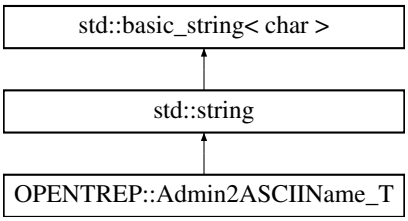
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.4 OPENTREP::Admin2ASCIIName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::Admin2ASCIIName_T:



Public Member Functions

- Admin2ASCIIName_T (const std::string &iValue)

23.4.1 Detailed Description

Administrative ASCII name of rank 2 (e.g., Alpes-Maritimes).

Definition at line 303 of file OPENTREP_Types.hpp.

23.4.2 Constructor & Destructor Documentation

23.4.2.1 OPENTREP::Admin2ASCIIName_T::Admin2ASCIIName_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 305 of file OPENTREP_Types.hpp.

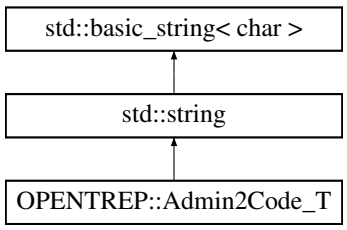
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.5 OPENTREP::Admin2Code_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::Admin2Code_T:



Public Member Functions

- Admin2Code_T (const std::string &iValue)

23.5.1 Detailed Description

Administrative code of rank 2.

Definition at line 286 of file OPENTREP_Types.hpp.

23.5.2 Constructor & Destructor Documentation

23.5.2.1 OPENTREP::Admin2Code_T::Admin2Code_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 288 of file OPENTREP_Types.hpp.

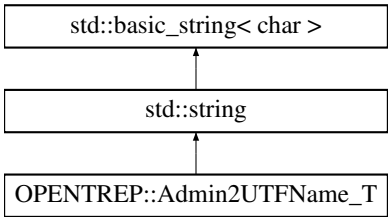
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.6 OPENTREP::Admin2UTFName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::Admin2UTFName_T:



Public Member Functions

- Admin2UTFName_T (const std::string &iValue)

23.6.1 Detailed Description

Administrative UTF8 name of rank 2 (e.g., Alpes-Maritimes).

Definition at line 294 of file OPENTREP_Types.hpp.

23.6.2 Constructor & Destructor Documentation

23.6.2.1 OPENTREP::Admin2UTFName_T::Admin2UTFName_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 296 of file OPENTREP_Types.hpp.

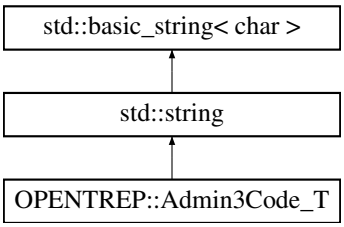
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.7 OPENTREP::Admin3Code_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::Admin3Code_T:



Public Member Functions

- Admin3Code_T (const std::string &iValue)

23.7.1 Detailed Description

Administrative code of rank 3.

Definition at line 312 of file OPENTREP_Types.hpp.

23.7.2 Constructor & Destructor Documentation

23.7.2.1 OPENTREP::Admin3Code_T::Admin3Code_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 314 of file OPENTREP_Types.hpp.

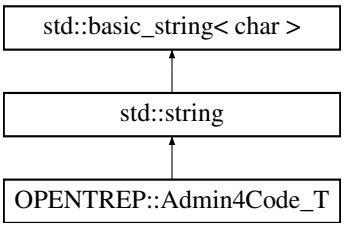
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.8 OPENTREP::Admin4Code_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::Admin4Code_T:



Public Member Functions

- Admin4Code_T (const std::string &iValue)

23.8.1 Detailed Description

Administrative code of rank 4.

Definition at line 320 of file OPENTREP_Types.hpp.

23.8.2 Constructor & Destructor Documentation

23.8.2.1 OPENTREP::Admin4Code_T::Admin4Code_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 322 of file OPENTREP_Types.hpp.

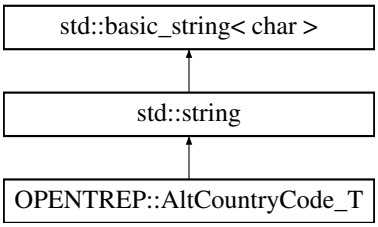
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.9 OPENTREP::AltCountryCode_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::AltCountryCode_T:



Public Member Functions

- AltCountryCode_T (const std::string &iValue)

23.9.1 Detailed Description

Alternative country code (e.g., US for USA).

Definition at line 224 of file OPENTREP_Types.hpp.

23.9.2 Constructor & Destructor Documentation

23.9.2.1 OPENTREP::AltCountryCode_T::AltCountryCode_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 226 of file OPENTREP_Types.hpp.

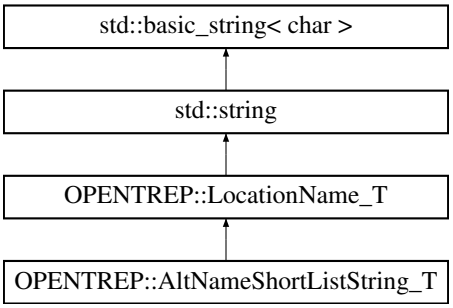
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.10 OPENTREP::AltNameShortListString_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::AltNameShortListString_T:



Public Member Functions

- AltNameShortListString_T (const std::string &iValue)

23.10.1 Detailed Description

Short list of alternate names, without the language codes (e.g., "Aéroport de Nice Côte d'Azur,Nice Airport,-Flughafen Nizza"), aggregated within a single string.

Definition at line 139 of file OPENTREP_Types.hpp.

23.10.2 Constructor & Destructor Documentation

23.10.2.1 OPENTREP::AltNameShortListString_T::AltNameShortListString_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 141 of file OPENTREP_Types.hpp.

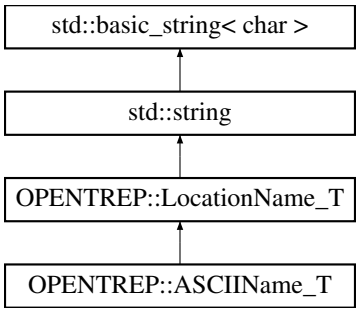
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.11 OPENTREP::ASCIIName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::ASCIIName_T:



Public Member Functions

- ASCIIName_T (const std::string &iValue)

23.11.1 Detailed Description

ASCII name (not necessarily in English, e.g., aéroport de Nice Cote d'Azur).

Definition at line 128 of file OPENTREP_Types.hpp.

23.11.2 Constructor & Destructor Documentation

23.11.2.1 OPENTREP::ASCIIName_T::ASCIIName_T (const std::string & *iValue*) [inline],[explicit]

Definition at line 130 of file OPENTREP_Types.hpp.

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.12 OPENTREP::BasChronometer Struct Reference

Structure allowing measuring the time elapsed between two events.

`#include <opentrep/basic/BasChronometer.hpp>`

Public Member Functions

- BasChronometer ()
- void start ()
- std::string getStart () const
- double elapsed () const

23.12.1 Detailed Description

Structure allowing measuring the time elapsed between two events.

Definition at line 15 of file BasChronometer.hpp.

23.12.2 Constructor & Destructor Documentation

23.12.2.1 OPENTREP::BasChronometer::BasChronometer ( )

Constructor.

Definition at line 13 of file BasChronometer.cpp.

23.12.3 Member Function Documentation

23.12.3.1 void OPENTREP::BasChronometer::start ( )

Start the chronometer from the local time.

The elapsed time given is the one elapsed since the start is launched.

Definition at line 17 of file BasChronometer.cpp.

Referenced by OPENTREP::OPENTREP_Service::buildSearchIndex(), OPENTREP::OPENTREP_Service::buildSQLDB(), OPENTREP::OPENTREP_Service::drawRandomLocations(), OPENTREP::OPENTREP_Service::getIndexSize(), and OPENTREP::OPENTREP_Service::interpretTravelRequest().

23.12.3.2 std::string OPENTREP::BasChronometer::getStart ( ) const [inline]

Get the start time.

Definition at line 31 of file BasChronometer.hpp.

23.12.3.3 double OPENTREP::BasChronometer::elapsed ( ) const

Return the time elapsed since the structure has been instanciated.

That elapsed time is expressed in seconds.

Definition at line 27 of file BasChronometer.cpp.

Referenced by OPENTREP::OPENTREP_Service::buildSearchIndex(), OPENTREP::OPENTREP_Service::buildSQLDB(), OPENTREP::OPENTREP_Service::drawRandomLocations(), OPENTREP::OPENTREP_Service::getIndexSize(), and OPENTREP::OPENTREP_Service::interpretTravelRequest().

The documentation for this struct was generated from the following files:

- opentrep/basic/BasChronometer.hpp
- opentrep/basic/BasChronometer.cpp

23.13 OPENTREP::BasFileMgr Struct Reference

Helper class for operations on files and on the file-system.

`#include <opentrep/basic/BasFileMgr.hpp>`

Static Public Member Functions

- static bool doesExistAndIsReadable (const std::string &iFilepath)

23.13.1 Detailed Description

Helper class for operations on files and on the file-system.

Definition at line 15 of file BasFileMgr.hpp.

23.13.2 Member Function Documentation

23.13.2.1 bool OPENTREP::BasFileMgr::doesExistAndIsReadable (const std::string & *iFilepath*) [static]

Definition at line 23 of file BasFileMgr.cpp.

Referenced by OPENTREP::PORParser::PORGeneration().

The documentation for this struct was generated from the following files:

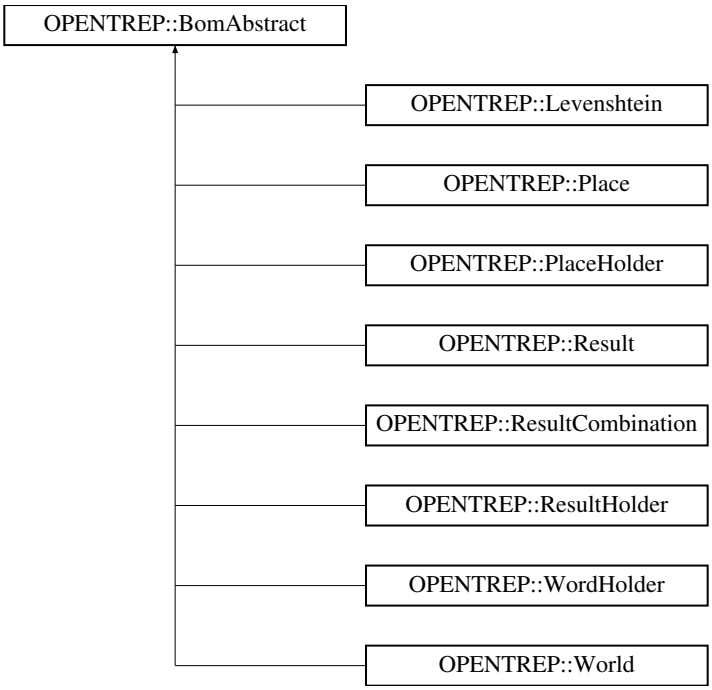
- opentrep/basic/BasFileMgr.hpp
- opentrep/basic/BasFileMgr.cpp

23.14 OPENTREP::BomAbstract Class Reference

Base class for the Business Object Model (BOM) layer.

`#include <opentrep/bom/BomAbstract.hpp>`

Inheritance diagram for OPENTREP::BomAbstract:



Public Member Functions

- virtual void toStream (std::ostream &) const =0
- virtual void fromStream (std::istream &)=0
- virtual std::string toString () const =0
- virtual std::string describeKey () const =0
- virtual std::string describeShortKey () const =0

Protected Member Functions

- BomAbstract ()
- BomAbstract (const BomAbstract &)
- virtual ~BomAbstract ()

Friends

- class FacBomAbstract

23.14.1 Detailed Description

Base class for the Business Object Model (BOM) layer.

Definition at line 17 of file BomAbstract.hpp.

23.14.2 Constructor & Destructor Documentation

23.14.2.1 OPENTREP::BomAbstract::BomAbstract ( ) [inline], [protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 57 of file BomAbstract.hpp.

23.14.2.2 OPENTREP::BomAbstract::BomAbstract (const BomAbstract &) [inline], [protected]

Definition at line 58 of file BomAbstract.hpp.

23.14.2.3 virtual OPENTREP::BomAbstract::~BomAbstract ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 63 of file BomAbstract.hpp.

23.14.3 Member Function Documentation

23.14.3.1 virtual void OPENTREP::BomAbstract::toStream (std::ostream &) const [pure virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::World, and OPENTREP::PlaceHolder.

23.14.3.2 virtual void OPENTREP::BomAbstract::fromStream (std::istream &) [pure virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::World, and OPENTREP::PlaceHolder.

Referenced by operator>>().

23.14.3.3 virtual std::string OPENTREP::BomAbstract::toString ( ) const [pure virtual]

Get the serialised version of the Business Object.

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::World, and OPENTREP::PlaceHolder.

23.14.3.4 virtual std::string OPENTREP::BomAbstract::describeKey ( ) const [pure virtual]

Get a string describing the whole key (differentiating two objects at any level).

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::PlaceHolder, and OPENTREP::World.

23.14.3.5 virtual std::string OPENTREP::BomAbstract::describeShortKey ( ) const [pure virtual]

Get a string describing the short key (differentiating two objects at the same level).

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::PlaceHolder, and OPENTREP::World.

23.14.4 Friends And Related Function Documentation

23.14.4.1 friend class FacBomAbstract [friend]

Definition at line 18 of file BomAbstract.hpp.

The documentation for this class was generated from the following file:

- opentrep/bom/BomAbstract.hpp

23.15 OPENTREP::BomJSONExport Class Reference

Utility class to export Opentrep structures in a JSON format.

`#include <opentrep/bom/BomJSONExport.hpp>`

Static Public Member Functions

- static void jsonExportLocationList (std::ostream &, const LocationList_T &)
- static void jsonExportLocation (bpt::ptree &, const Location &)

23.15.1 Detailed Description

Utility class to export Opentrep structures in a JSON format.

Definition at line 25 of file BomJSONExport.hpp.

23.15.2 Member Function Documentation

23.15.2.1 void OPENTREP::BomJSONExport::jsonExportLocationList (std::ostream & *oStream,* const LocationList_T & *iLocationList*) [static]

Export (dump in the underlying output log stream and in JSON format) a list of Location objects.

Parameters

std::ostream&	Output stream in which the Location objects should be logged/dumped.
const	LocationList_T& List of Location objects to be exported.

Definition at line 17 of file BomJSONExport.cpp.

References OPENTREP::Location::getAlternateLocationList(), OPENTREP::Location::getExtraLocationList(), and jsonExportLocation().

23.15.2.2 void OPENTREP::BomJSONExport::jsonExportLocation (bpt::ptree & *ioPTLocation,* const Location & *iLocation*) [static]

Export (dump in the underlying output log stream and in JSON format) a Location object.

Parameters

bpt::ptree&	Property tree in which the Location structure should be logged/dumped.
const	Location& Location object to be exported.

Definition at line 92 of file BomJSONExport.cpp.

References OPENTREP::Location::getAdmin1AsciiName(), OPENTREP::Location::getAdmin1Code(), OPENTREP::Location::getAdmin1UtfName(), OPENTREP::Location::getAdmin2AsciiName(), OPENTREP::Location::getAdmin2Code(), OPENTREP::Location::getAdmin2UtfName(), OPENTREP::Location::getAdmin3Code(), OPENTREP::Location::getAdmin4Code(), OPENTREP::Location::getAllowableEditDistance(), OPENTREP::Location::getCityAsciiName(), OPENTREP::Location::getCityCode(), OPENTREP::Location::getCityUtfName(), OPENTREP::Location::getContinentName(), OPENTREP::Location::getCorrectedKeywords(), OPENTREP::Location::getCountryCode(), OPENTREP::Location::getCountryName(), OPENTREP::Location::getEditDistance(), OPENTREP::Location::getFaaCode(), OPENTREP::Location::getGeonamesID(), OPENTREP::Location::getIataCode(), OPENTREP::Location::getIcaoCode(), OPENTREP::Location::getLatitude(), OPENTREP::Location::getLongitude(), OPENTREP::Names::getNameList(), OPENTREP::NameMatrix::getNameMatrix(), OPENTREP::Location::getNameMatrix(), OPENTREP::Location::getOriginalKeywords(), OPENTREP::Location::getPageRank(), OPENT-

REP::Location::getPercentage(), OPENTREP::Location::getStateCode(), OPENTREP::Location::getTimeZone(), OPENTREP::Location::getTvlPORListString(), and OPENTREP::Location::getWikiLink().

Referenced by jsonExportLocationList().

The documentation for this class was generated from the following files:

- opentrep/bom/BomJSONExport.hpp
- opentrep/bom/BomJSONExport.cpp

23.16 OPENTREP::BomType Struct Reference

`#include <opentrep/bom/BomType.hpp>`

Public Types

- enum EN_BomType {
  WORLD = 0, PLACE, AIRLINE, COUNTRY,
  DAY, MONTH, PASSENGER, LAST_VALUE }

Static Public Member Functions

- static const std::string & getLabel (const EN_BomType &)

23.16.1 Detailed Description

Enumeration of BOM types.

Definition at line 13 of file BomType.hpp.

23.16.2 Member Enumeration Documentation

23.16.2.1 enum OPENTREP::BomType::EN_BomType

Enumerator

WORLD
PLACE
AIRLINE
COUNTRY
DAY
MONTH
PASSENGER
LAST_VALUE

Definition at line 15 of file BomType.hpp.

23.16.3 Member Function Documentation

23.16.3.1 const std::string & OPENTREP::BomType::getLabel (const EN_BomType & *iBomType*) [static]

Get the label (BOM type) as a string.

Definition at line 16 of file BomType.cpp.

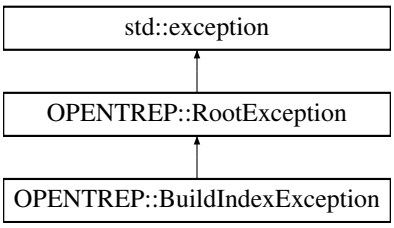
The documentation for this struct was generated from the following files:

- opentrep/bom/BomType.hpp
- opentrep/bom/BomType.cpp

23.17 OPENTREP::BuildIndexException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::BuildIndexException:



Public Member Functions

- BuildIndexException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.17.1 Detailed Description

Index issue.

Definition at line 309 of file OPENTREP_exceptions.hpp.

23.17.2 Constructor & Destructor Documentation

23.17.2.1 OPENTREP::BuildIndexException::BuildIndexException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 314 of file OPENTREP_exceptions.hpp.

23.17.3 Member Function Documentation

23.17.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.17.4 Member Data Documentation

23.17.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

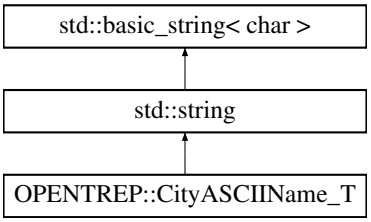
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.18 OPENTREP::CityASCIIName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::CityASCIIName_T:



Public Member Functions

- CityASCIIName_T (const std::string &iValue)

23.18.1 Detailed Description

City name in ASCII (but not necessarily in English, e.g., ville de Nice Cote d'Azur).

Definition at line 199 of file OPENTREP_Types.hpp.

23.18.2 Constructor & Destructor Documentation

23.18.2.1 OPENTREP::CityASCIIName_T::CityASCIIName_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 201 of file OPENTREP_Types.hpp.

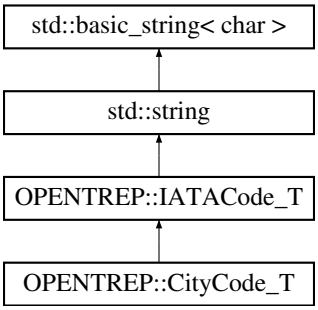
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.19 OPENTREP::CityCode_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::CityCode_T:



Public Member Functions

- CityCode_T (const std::string &iValue)

23.19.1 Detailed Description

IATA city code (e.g., CHI for Chicago, IL, USA).

Definition at line 180 of file OPENTREP_Types.hpp.

23.19.2 Constructor & Destructor Documentation

23.19.2.1 OPENTREP::CityCode_T::CityCode_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 182 of file OPENTREP_Types.hpp.

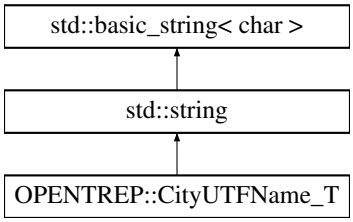
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.20 OPENTREP::CityUTFName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::CityUTFName_T:



Public Member Functions

- CityUTFName_T (const std::string &iValue)

23.20.1 Detailed Description

City name in UTF8 (e.g., Nice Côte d'Azur).

Definition at line 190 of file OPENTREP_Types.hpp.

23.20.2 Constructor & Destructor Documentation

23.20.2.1 OPENTREP::CityUTFName_T::CityUTFName_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 192 of file OPENTREP_Types.hpp.

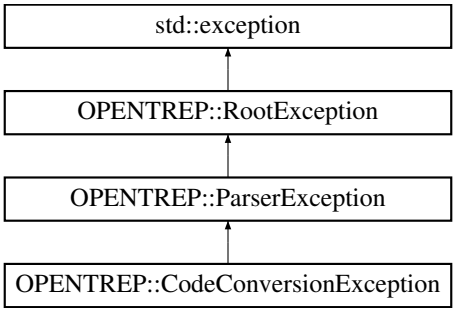
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.21 OPENTREP::CodeConversionException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::CodeConversionException:



Public Member Functions

- CodeConversionException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.21.1 Detailed Description

Code conversion.

Definition at line 191 of file OPENTREP_exceptions.hpp.

23.21.2 Constructor & Destructor Documentation

23.21.2.1 OPENTREP::CodeConversionException::CodeConversionException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 196 of file OPENTREP_exceptions.hpp.

23.21.3 Member Function Documentation

23.21.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.21.4 Member Data Documentation

23.21.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

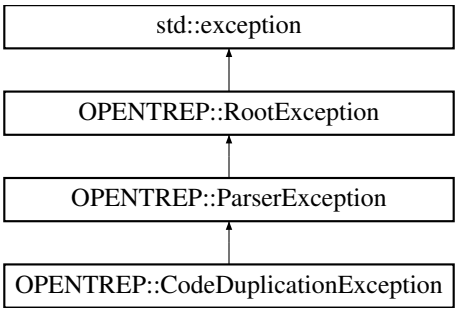
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.22 OPENTREP::CodeDuplicationException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::CodeDuplicationException:



Public Member Functions

- CodeDuplicationException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.22.1 Detailed Description

Code duplication.

Definition at line 203 of file OPENTREP_exceptions.hpp.

23.22.2 Constructor & Destructor Documentation

23.22.2.1 OPENTREP::CodeDuplicationException::CodeDuplicationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 208 of file OPENTREP_exceptions.hpp.

23.22.3 Member Function Documentation

23.22.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.22.4 Member Data Documentation

23.22.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

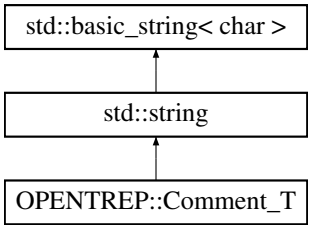
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.23 OPENTREP::Comment_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::Comment_T:



Public Member Functions

- Comment_T (const std::string &iValue)

23.23.1 Detailed Description

Comment (e.g., "Code claimed back by IATA in November 2012").

Definition at line 172 of file OPENTREP_Types.hpp.

23.23.2 Constructor & Destructor Documentation

23.23.2.1 OPENTREP::Comment_T::Comment_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 174 of file OPENTREP_Types.hpp.

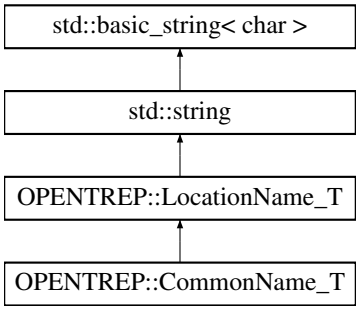
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.24 OPENTREP::CommonName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::CommonName_T:



Public Member Functions

- CommonName_T (const std::string &iValue)

23.24.1 Detailed Description

Common name (usually in American English, but not necessarily in ASCII, e.g., Nice Côte d'Azur).

Definition at line 119 of file OPENTREP_Types.hpp.

23.24.2 Constructor & Destructor Documentation

23.24.2.1 OPENTREP::CommonName_T::CommonName_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 121 of file OPENTREP_Types.hpp.

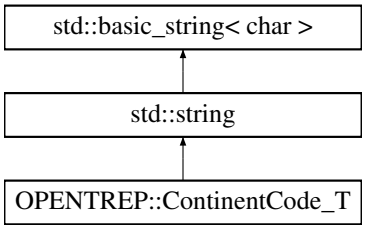
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.25 OPENTREP::ContinentCode_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::ContinentCode_T:



Public Member Functions

- ContinentCode_T (const std::string &iValue)

23.25.1 Detailed Description

Continent code (e.g., SA for South America).

Definition at line 242 of file OPENTREP_Types.hpp.

23.25.2 Constructor & Destructor Documentation

23.25.2.1 OPENTREP::ContinentCode_T::ContinentCode_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 244 of file OPENTREP_Types.hpp.

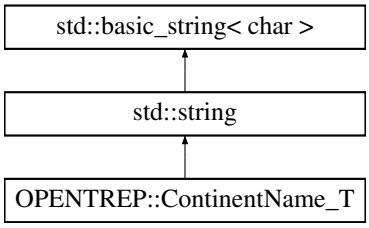
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.26 OPENTREP::ContinentName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::ContinentName_T:



Public Member Functions

- ContinentName_T (const std::string &iValue)

23.26.1 Detailed Description

Continent name (e.g., North America, Europe, Pacific, Asia, Australia).

Definition at line 251 of file OPENTREP_Types.hpp.

23.26.2 Constructor & Destructor Documentation

23.26.2.1 OPENTREP::ContinentName_T::ContinentName_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 253 of file OPENTREP_Types.hpp.

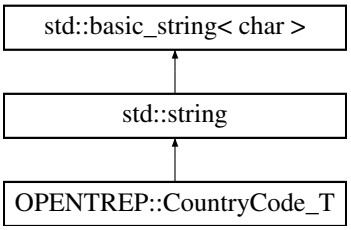
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.27 OPENTREP::CountryCode_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::CountryCode_T:



Public Member Functions

- CountryCode_T (const std::string &iValue)

23.27.1 Detailed Description

Country code (e.g., US for USA).

Definition at line 216 of file OPENTREP_Types.hpp.

23.27.2 Constructor & Destructor Documentation

23.27.2.1 OPENTREP::CountryCode_T::CountryCode_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 218 of file OPENTREP_Types.hpp.

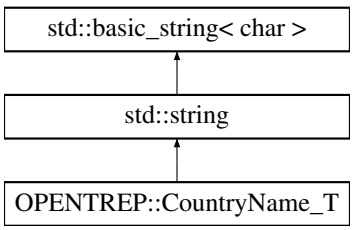
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.28 OPENTREP::CountryName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::CountryName_T:



Public Member Functions

- CountryName_T (const std::string &iValue)

23.28.1 Detailed Description

Country name (e.g., Germany, Switzerland, France).

Definition at line 233 of file OPENTREP_Types.hpp.

23.28.2 Constructor & Destructor Documentation

23.28.2.1 OPENTREP::CountryName_T::CountryName_T (const std::string & *iValue*) [inline],[explicit]

Definition at line 235 of file OPENTREP_Types.hpp.

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.29 OPENTREP::date_time_element< MIN, MAX > Struct Template Reference

`#include <opentrep/OPENTREP_ParserHelperTypes.hpp>`

Public Member Functions

- date_time_element ()
- date_time_element (const date_time_element &t)
- date_time_element (int i)
- void check () const

Public Attributes

- unsigned int _value

23.29.1 Detailed Description

template<int MIN = 0, int MAX = 0>struct OPENTREP::date_time_element< MIN, MAX >

Date & time element parser.

Definition at line 22 of file OPENTREP_ParserHelperTypes.hpp.

23.29.2 Constructor & Destructor Documentation

23.29.2.1 template<int MIN = 0, int MAX = 0> OPENTREP::date_time_element< MIN, MAX >::date_time_element ( ) [inline]

Default constructor.

Definition at line 27 of file OPENTREP_ParserHelperTypes.hpp.

23.29.2.2 template<int MIN = 0, int MAX = 0> OPENTREP::date_time_element< MIN, MAX >::date_time_element (const date_time_element< MIN, MAX > & *t*) [inline]

Default copy constructor.

Definition at line 29 of file OPENTREP_ParserHelperTypes.hpp.

23.29.2.3 template<int MIN = 0, int MAX = 0> OPENTREP::date_time_element< MIN, MAX >::date_time_element (int *i*) [inline]

Constructor.

Definition at line 31 of file OPENTREP_ParserHelperTypes.hpp.

23.29.3 Member Function Documentation

23.29.3.1 template<int MIN = 0, int MAX = 0> void OPENTREP::date_time_element< MIN, MAX >::check ( ) const [inline]

Checker.

Definition at line 33 of file OPENTREP_ParserHelperTypes.hpp.

Referenced by OPENTREP::Location::calculateDate().

23.29.4 Member Data Documentation

23.29.4.1 template<int MIN = 0, int MAX = 0> unsigned int OPENTREP::date_time_element< MIN, MAX >::_value

Definition at line 23 of file OPENTREP_ParserHelperTypes.hpp.

Referenced by OPENTREP::Location::calculateDate(), OPENTREP::date_time_element< 1, 31 >::check(), OPENTREP::operator*(), and OPENTREP::operator+().

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_ParserHelperTypes.hpp

23.30 OPENTREP::DbaAbstract Class Reference

`#include <opentrep/dbadaptor/DbaAbstract.hpp>`

Public Member Functions

- virtual ~DbaAbstract ()
- virtual void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Protected Member Functions

- DbaAbstract ()

23.30.1 Detailed Description

Base class for the Database Adaptor (DBA) layer.

Definition at line 14 of file DbaAbstract.hpp.

23.30.2 Constructor & Destructor Documentation

23.30.2.1 virtual OPENTREP::DbaAbstract::~DbaAbstract ( ) [inline], [virtual]

Destructor.

Definition at line 18 of file DbaAbstract.hpp.

23.30.2.2 OPENTREP::DbaAbstract::DbaAbstract ( ) [inline], [protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 30 of file DbaAbstract.hpp.

23.30.3 Member Function Documentation

23.30.3.1 virtual void OPENTREP::DbaAbstract::toStream (std::ostream & *ioOut*) const [inline], [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Definition at line 22 of file DbaAbstract.hpp.

23.30.3.2 virtual void OPENTREP::DbaAbstract::fromStream (std::istream & *ioIn*) [inline], [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Definition at line 26 of file DbaAbstract.hpp.

Referenced by operator>>().

The documentation for this class was generated from the following file:

- opentrep/dbadaptor/DbaAbstract.hpp

23.31 OPENTREP::DBManager Class Reference

Class building the Business Object Model (BOM) from data retrieved from the database.

`#include <opentrep/command/DBManager.hpp>`

Static Public Member Functions

- static NbOfDBEntries_T buildSQLDB (const PORFilePath_T &, const SQLiteDBFilePath_T &)
- static void updatePlaceInDB (soci::session &, const Place &)
- static bool retrievePlace (soci::session &, const LocationKey &, Place &)
- static bool retrieveClosestPlaces (soci::session &, const double &iLatitude, const double &iLongitude, PlaceOrderedList_T &)
- static void prepareSelectAllStatement (soci::session &, soci::statement &, Place &)
- static void prepareSelectFromCodeStatement (soci::session &, soci::statement &, Place &)
- static void prepareSelectFromCoordStatement (soci::session &, soci::statement &, const double &iLatitude, const double &iLongitude)
- static bool iterateOnStatement (soci::statement &, Place &)

23.31.1 Detailed Description

Class building the Business Object Model (BOM) from data retrieved from the database.

Definition at line 28 of file DBManager.hpp.

23.31.2 Member Function Documentation

23.31.2.1 NbOfDBEntries_T OPENTREP::DBManager::buildSQLDB (const PORFilePath_T & *iPORFilepath,* const SQLiteDBFilePath_T & *iSQLiteDBFilePath*) [static]

Build the SQL database from the file with the ORI-maintained list of POR (points of reference).

Parameters

const	PORFilePath_T& File-path of the file of POR (points of reference).
const	SQLiteDBFilePath_T& File-path of the SQLite3 database.

Returns

NbOfDBEntries_T Number of documents inserted in the SQLite3 DB.

Definition at line 516 of file DBManager.cpp.

References OPENTREP::FacPlace::create(), OPENTREP::Place::getKey(), OPENTREP::FacPlace::instance(), iterateOnStatement(), OPENTREP_LOG_DEBUG, OPENTREP_LOG_ERROR, and prepareSelectAllStatement().

Referenced by OPENTREP::OPENTREP_Service::buildSQLDB().

23.31.2.2 void OPENTREP::DBManager::updatePlaceInDB (soci::session & *ioSociSession,* const Place & *iPlace*) [static]

Update the Xapian document ID field of the database row corresponding to the given Place object.

Parameters

soci::session&	SOCI session handler.
const	Place& The place to be updated.

Definition at line 410 of file DBManager.cpp.

References OPENTREP::Place::getDocID(), OPENTREP::Place::getIataCode(), OPENTREP_LOG_ERROR, and OPENTREP::Place::toString().

23.31.2.3 bool OPENTREP::DBManager::retrievePlace (soci::session & *ioSociSession,* const LocationKey & *iPlaceKey,* Place & *ioPlace*) [static]

Retrieve, from the (SQLite3) database, the row corresponding to the given place code (e.g., 'sfo' for San Francisco Intl airport), and fill the given Place object with that retrieved data.

Parameters

soci::session&	SOCI session handler.
const	LocationKey& The primary key of the place to be retrieved.
Place&	The object corresponding to the place to be retrieved. It has to be given empty, and is filled by the method.

Definition at line 449 of file DBManager.cpp.

References OPENTREP::LocationKey::getGeonamesID(), OPENTREP::LocationKey::getIataCode(), OPENTREP::LocationKey::getIataType(), OPENTREP::IATAType::getTypeAsString(), iterateOnStatement(), OPENTREP_LOG_DEBUG, and OPENTREP_LOG_ERROR.

23.31.2.4 bool OPENTREP::DBManager::retrieveClosestPlaces (soci::session & *ioSociSession,* const double & *iLatitude,* const double & *iLongitude,* PlaceOrderedList_T & *ioPlaceList*) [static]

Retrieve, from the (SQLite3) database, the row corresponding to the given place code (e.g., 'sfo' for San Francisco Intl airport), and fill the given Place object with that retrieved data.

Parameters

soci::session&	SOCI session handler.
const	std::double& The latitude of the place to be retrieved.

const	std::double& The longitude of the place to be retrieved.
PlaceOrdered-List_T&	The list corresponding to the places to be retrieved. It has to be given empty, and is filled by the method.

Definition at line 494 of file DBManager.cpp.

References OPENTREP_LOG_ERROR.

23.31.2.5 void OPENTREP::DBManager::prepareSelectAllStatement (soci::session & *ioSociSession*, soci::statement & *ioSelectStatement*, Place & *ioPlace*) [static]

Prepare (parse and put in cache) the SQL statement.

Parameters

soci::session&	SOCI session handler.
soci::statement&	SOCI SQL statement handler.
Place&	The object corresponding to the place to be retrieved. It has to be given empty, and is filled by the method.

Definition at line 22 of file DBManager.cpp.

References OPENTREP_LOG_ERROR.

Referenced by buildSQLDB().

23.31.2.6 void OPENTREP::DBManager::prepareSelectFromCodeStatement (soci::session & *ioSociSession*, soci::statement & *ioSelectStatement*, Place & *ioPlace*) [static]

Prepare (parse and put in cache) the SQL statement.

Parameters

soci::session&	SOCI session handler.
soci::statement&	SOCI SQL statement handler.
Place&	The object corresponding to the place to be retrieved. It has to be given empty, and is filled by the method.

Definition at line 109 of file DBManager.cpp.

References OPENTREP_LOG_ERROR.

23.31.2.7 void OPENTREP::DBManager::prepareSelectFromCoordStatement (soci::session & *ioSociSession*, soci::statement & *ioSelectStatement*, const double & *iLatitude*, const double & *iLongitude*) [static]

Prepare (parse and put in cache) the SQL statement.

Parameters

soci::session&	SOCI session handler.
soci::statement&	SOCI SQL statement handler.
Place&	The object corresponding to the place to be retrieved. It has to be given empty, and is filled by the method.

Definition at line 172 of file DBManager.cpp.

References OPENTREP::FacPlace::create(), OPENTREP::FacPlace::instance(), and OPENTREP_LOG_ERROR.

23.31.2.8 bool OPENTREP::DBManager::iterateOnStatement (soci::statement & *ioStatement*, Place & *ioPlace*) [static]

Iterate on the SQL statement.

The SQL has to be already prepared.

Parameters

soci::statement&	SOCI SQL statement handler.
Place&	The object corresponding to the place to be retrieved. It has to be given empty, and is filled by the method.

Definition at line 388 of file DBManager.cpp.

References OPENTREP::Place::describeKey(), and OPENTREP_LOG_ERROR.

Referenced by buildSQLDB(), and retrievePlace().

The documentation for this class was generated from the following files:

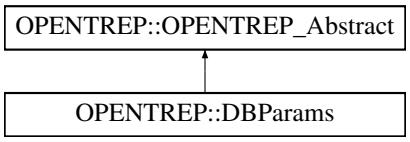
- opentrep/command/DBManager.hpp
- opentrep/command/DBManager.cpp

23.32 OPENTREP::DBParams Struct Reference

Structure modelling a database parameters.

`#include <opentrep/DBParams.hpp>`

Inheritance diagram for OPENTREP::DBParams:



Public Member Functions

- std::string getUser () const
- std::string getPassword () const
- std::string getHost () const
- std::string getPort () const
- std::string getDBName () const
- void setUser (const std::string &iUser)
- void setPassword (const std::string &iPasswd)
- void setHost (const std::string &iHost)
- void setPort (const std::string &iPort)
- void setDBName (const std::string &iDBName)
- bool checkSQLite () const
- bool checkMySQL () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &)
- std::string toShortString () const
- std::string toString () const
- std::string toMySQLConnectionString () const
- std::string toSQLiteConnectionString () const
- DBParams (const std::string &iDBUser, const std::string &iDBPasswd, const std::string &iDBHost, const std::string &iDBPort, const std::string &iDBName)
- DBParams (const std::string &iDBName)
- virtual ~DBParams ()

23.32.1 Detailed Description

Structure modelling a database parameters.

Definition at line 25 of file DBParams.hpp.

23.32.2 Constructor & Destructor Documentation

23.32.2.1 OPENTREP::DBParams::DBParams (const std::string & *iDBUser,* const std::string & *iDBPasswd,* const std::string & *iDBHost,* const std::string & *iDBPort,* const std::string & *iDBName*) [inline]

Main constructors.

Definition at line 187 of file DBParams.hpp.

23.32.2.2 OPENTREP::DBParams::DBParams (const std::string & *iDBName*) [inline]

Definition at line 193 of file DBParams.hpp.

23.32.2.3 virtual OPENTREP::DBParams::~DBParams ( ) [inline], [virtual]

Default Constructor. Default copy constructor. Destructor.

Definition at line 208 of file DBParams.hpp.

23.32.3 Member Function Documentation

23.32.3.1 std::string OPENTREP::DBParams::getUser ( ) const [inline]

Get the database user name.

Definition at line 31 of file DBParams.hpp.

23.32.3.2 std::string OPENTREP::DBParams::getPassword ( ) const [inline]

Get the database user password.

Definition at line 38 of file DBParams.hpp.

23.32.3.3 std::string OPENTREP::DBParams::getHost ( ) const [inline]

Get the database host name.

Definition at line 45 of file DBParams.hpp.

23.32.3.4 std::string OPENTREP::DBParams::getPort ( ) const [inline]

Get the database port number.

Definition at line 52 of file DBParams.hpp.

23.32.3.5 std::string OPENTREP::DBParams::getDBName ( ) const [inline]

Get the database name.

Definition at line 59 of file DBParams.hpp.

23.32.3.6 void OPENTREP::DBParams::setUser (const std::string & *iUser*) [inline]

Set the database user name.

Definition at line 69 of file DBParams.hpp.

23.32.3.7 void OPENTREP::DBParams::setPassword (const std::string & *iPasswd*) [inline]

Set the database password.

Definition at line 76 of file DBParams.hpp.

23.32.3.8 void OPENTREP::DBParams::setHost (const std::string & *iHost*) [inline]

Set the database host name.

Definition at line 83 of file DBParams.hpp.

23.32.3.9 void OPENTREP::DBParams::setPort (const std::string & *iPort*) [inline]

Set the database port number.

Definition at line 90 of file DBParams.hpp.

23.32.3.10 void OPENTREP::DBParams::setDBName (const std::string & *iDBName*) [inline]

Set the database name.

Definition at line 97 of file DBParams.hpp.

23.32.3.11 bool OPENTREP::DBParams::checkSQLite ( ) const [inline]

Check that all the parameters are fine.

Definition at line 107 of file DBParams.hpp.

23.32.3.12 bool OPENTREP::DBParams::checkMySQL ( ) const [inline]

Check that all the parameters are fine.

Definition at line 117 of file DBParams.hpp.

23.32.3.13 void OPENTREP::DBParams::toStream (std::ostream & *ioOut*) const [inline], [virtual]

Dump a structure into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 133 of file DBParams.hpp.

References toString().

23.32.3.14 void OPENTREP::DBParams::fromStream (std::istream &) [inline], [virtual]

Read a structure from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 141 of file DBParams.hpp.

23.32.3.15 std::string OPENTREP::DBParams::toShortString ( ) const [inline]

Get a short display of the DBParams structure.

Definition at line 147 of file DBParams.hpp.

23.32.3.16 std::string OPENTREP::DBParams::toString ( ) const [inline], [virtual]

Get the serialised version of the DBParams structure.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 156 of file DBParams.hpp.

Referenced by toStream().

23.32.3.17 std::string OPENTREP::DBParams::toMySQLConnectionString ( ) const [inline]

Get the serialised version of the DBParams structure.

Definition at line 165 of file DBParams.hpp.

23.32.3.18 std::string OPENTREP::DBParams::toSQLiteConnectionString ( ) const [inline]

Get the serialised version of the DBParams structure.

Definition at line 175 of file DBParams.hpp.

The documentation for this struct was generated from the following file:

- opentrep/DBParams.hpp

23.33 OPENTREP::DBSessionManager Class Reference

Class handling the SOCI session.

`#include <opentrep/command/DBSessionManager.hpp>`

Friends

- class OPENTREP_ServiceContext

23.33.1 Detailed Description

Class handling the SOCI session.

Definition at line 21 of file DBSessionManager.hpp.

23.33.2 Friends And Related Function Documentation

23.33.2.1 friend class OPENTREP_ServiceContext [friend]

Definition at line 22 of file DBSessionManager.hpp.

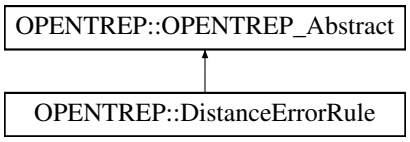
The documentation for this class was generated from the following files:

- opentrep/command/DBSessionManager.hpp
- opentrep/command/DBSessionManager.cpp

23.34 OPENTREP::DistanceErrorRule Struct Reference

`#include <opentrep/DistanceErrorRule.hpp>`

Inheritance diagram for OPENTREP::DistanceErrorRule:



Public Member Functions

- const DistanceErrorScale_T & getDistanceErrorScale () const
- NbOfErrors_T getAllowedDistanceError (const NbOfLetters_T &iNbOfLetters) const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &)
- std::string toShortString () const
- std::string toString () const
- DistanceErrorRule (const DistanceErrorScale_T &iScale)
- DistanceErrorRule (const NbOfErrors_T iSize, const NbOfLetters_T iScaleArray[])
- DistanceErrorRule (const DistanceErrorScaleArray_T &iScaleArray)

23.34.1 Detailed Description

Structure modelling a (geographical) distanceerrorrule.

Definition at line 43 of file DistanceErrorRule.hpp.

23.34.2 Constructor & Destructor Documentation

23.34.2.1 OPENTREP::DistanceErrorRule::DistanceErrorRule (const DistanceErrorScale_T & *iScale*) [inline]

Main Constructor.

Definition at line 129 of file DistanceErrorRule.hpp.

23.34.2.2 OPENTREP::DistanceErrorRule::DistanceErrorRule (const NbOfErrors_T *iSize*, const NbOfLetters_T *iScaleArray[]*) [inline]

Constructor.

Definition at line 136 of file DistanceErrorRule.hpp.

23.34.2.3 OPENTREP::DistanceErrorRule::DistanceErrorRule (const DistanceErrorScaleArray_T & *iScaleArray*) [inline]

Constructor.

Definition at line 147 of file DistanceErrorRule.hpp.

23.34.3 Member Function Documentation

23.34.3.1 const DistanceErrorScale_T& OPENTREP::DistanceErrorRule::getDistanceErrorScale ( ) const [inline]

Get the DistanceErrorRule code.

Definition at line 49 of file DistanceErrorRule.hpp.

23.34.3.2 NbOfErrors_T OPENTREP::DistanceErrorRule::getAllowedDistanceError (const NbOfLetters_T & *iNbOfLetters*) const [inline]

Get, for a given number of letters, the corresponding allowed (distance) error.

Definition at line 63 of file DistanceErrorRule.hpp.

23.34.3.3 void OPENTREP::DistanceErrorRule::toStream (std::ostream & *ioOut*) const [inline], [virtual]

Dump a structure into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 87 of file DistanceErrorRule.hpp.

References toString().

23.34.3.4 void OPENTREP::DistanceErrorRule::fromStream (std::istream &) [inline], [virtual]

Read a structure from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 96 of file DistanceErrorRule.hpp.

23.34.3.5 std::string OPENTREP::DistanceErrorRule::toShortString ( ) const [inline]

Get a short display of the DistanceErrorRule structure.

Definition at line 102 of file DistanceErrorRule.hpp.

Referenced by toString().

23.34.3.6 std::string OPENTREP::DistanceErrorRule::toString ( ) const [inline], [virtual]

Get the serialised version of the DistanceErrorRule structure.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 118 of file DistanceErrorRule.hpp.

References toShortString().

Referenced by toStream().

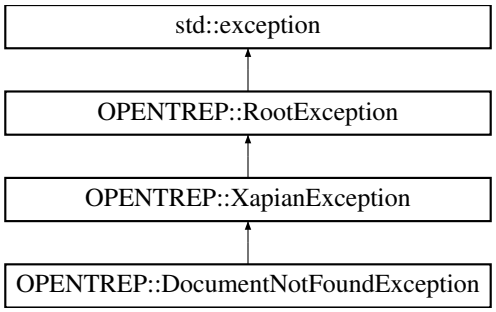
The documentation for this struct was generated from the following file:

- opentrep/DistanceErrorRule.hpp

23.35 OPENTREP::DocumentNotFoundException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::DocumentNotFoundException:



Public Member Functions

- DocumentNotFoundException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.35.1 Detailed Description

The document can not be retrieved.

Definition at line 226 of file OPENTREP_exceptions.hpp.

23.35.2 Constructor & Destructor Documentation

23.35.2.1 OPENTREP::DocumentNotFoundException::DocumentNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 231 of file OPENTREP_exceptions.hpp.

23.35.3 Member Function Documentation

23.35.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.35.4 Member Data Documentation

23.35.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

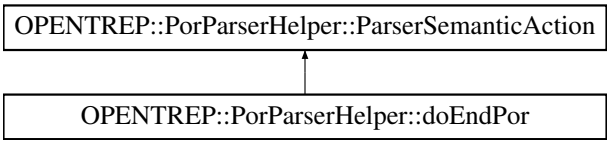
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.36 OPENTREP::PorParserHelper::doEndPor Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::doEndPor:



Public Member Functions

- doEndPor (Location &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.36.1 Detailed Description

Mark the end of the por-rule parsing.

Definition at line 548 of file PORParserHelper.hpp.

23.36.2 Constructor & Destructor Documentation

23.36.2.1 OPENTREP::PorParserHelper::doEndPor::doEndPor (Location & *ioLocation*)

Actor Constructor.

Definition at line 839 of file PORParserHelper.cpp.

23.36.3 Member Function Documentation

23.36.3.1 void OPENTREP::PorParserHelper::doEndPor::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 844 of file PORParserHelper.cpp.

23.36.4 Member Data Documentation

23.36.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

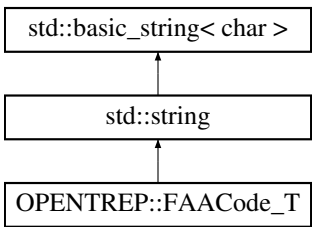
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.37 OPENTREP::FAACode_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::FAACode_T:



Public Member Functions

- FAACode_T (const std::string &iValue)

23.37.1 Detailed Description

FAA three-letter code (e.g., ORD).

Definition at line 101 of file OPENTREP_Types.hpp.

23.37.2 Constructor & Destructor Documentation

23.37.2.1 OPENTREP::FAACode_T::FAACode_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 103 of file OPENTREP_Types.hpp.

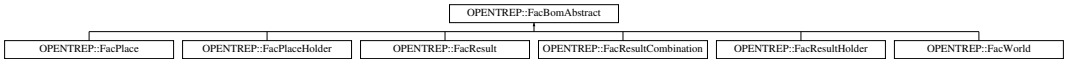
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.38 OPENTREP::FacBomAbstract Class Reference

`#include <opentrep/factory/FacBomAbstract.hpp>`

Inheritance diagram for OPENTREP::FacBomAbstract:



Public Types

- typedef std::vector < BomAbstract * > BomPool_T

Static Public Member Functions

- static std::size_t getID (const BomAbstract *)
- static std::size_t getID (const BomAbstract &)
- static std::string getIDString (const BomAbstract *)
- static std::string getIDString (const BomAbstract &)

Protected Member Functions

- FacBomAbstract ()
- FacBomAbstract (const FacBomAbstract &)
- virtual ~FacBomAbstract ()

Protected Attributes

- BomPool_T _pool

Friends

- class FacSupervisor

23.38.1 Detailed Description

Base class for Factory layer.

Definition at line 17 of file FacBomAbstract.hpp.

23.38.2 Member Typedef Documentation

23.38.2.1 typedef std::vector<BomAbstract*> OPENTREP::FacBomAbstract::BomPool_T

Define the list (pool) of Bom objects.

Definition at line 22 of file FacBomAbstract.hpp.

23.38.3 Constructor & Destructor Documentation

23.38.3.1 OPENTREP::FacBomAbstract::FacBomAbstract ( ) [inline], [protected]

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 41 of file FacBomAbstract.hpp.

23.38.3.2 OPENTREP::FacBomAbstract::FacBomAbstract (const FacBomAbstract &) [inline], [protected]

Definition at line 42 of file FacBomAbstract.hpp.

23.38.3.3 OPENTREP::FacBomAbstract::~FacBomAbstract ( ) [protected], [virtual]

Destructor.

Definition at line 17 of file FacBomAbstract.cpp.

23.38.4 Member Function Documentation

23.38.4.1 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract * *iBomAbstract_ptr*) [static]

Return the ID corresponding to the given object pointer.

Definition at line 36 of file FacBomAbstract.cpp.

Referenced by getID(), and getIDString().

23.38.4.2 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract & *iBomAbstract*) [static]

Return the ID corresponding to the given object reference.

Definition at line 44 of file FacBomAbstract.cpp.

References getID().

23.38.4.3 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract * *iBomAbstract_ptr*) [static]

Return the ID, as a string, corresponding to the given object pointer.

Definition at line 49 of file FacBomAbstract.cpp.

References getID().

Referenced by getIDString().

23.38.4.4 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract & *iBomAbstract*) [static]

Return the ID, as a string, corresponding to the given object reference.

Definition at line 57 of file FacBomAbstract.cpp.

References getIDString().

23.38.5 Friends And Related Function Documentation

23.38.5.1 friend class FacSupervisor [friend]

Definition at line 18 of file FacBomAbstract.hpp.

23.38.6 Member Data Documentation

23.38.6.1 BomPool_T OPENTREP::FacBomAbstract::_pool [protected]

List of instantiated Business Objects

Definition at line 53 of file FacBomAbstract.hpp.

Referenced by OPENTREP::FacWorld::clone(), OPENTREP::FacPlace::clone(), OPENTREP::FacWorld::create(), OPENTREP::FacPlaceHolder::create(), OPENTREP::FacResultHolder::create(), OPENTREP::FacResult::create(), OPENTREP::FacPlace::create(), and OPENTREP::FacResultCombination::create().

The documentation for this class was generated from the following files:

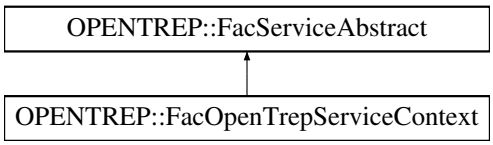
- opentrep/factory/FacBomAbstract.hpp
- opentrep/factory/FacBomAbstract.cpp

23.39 OPENTREP::FacOpenTrepServiceContext Class Reference

Factory for Bucket.

`#include <opentrep/factory/FacOpenTrepServiceContext.hpp>`

Inheritance diagram for OPENTREP::FacOpenTrepServiceContext:



Public Types

- typedef std::vector < ServiceAbstract * > ServicePool_T

Public Member Functions

- ~FacOpenTrepServiceContext ()
- OPENTREP_ServiceContext & create (const TravelDBFilePath_T &, const SQLiteDBFilePath_T &)
- OPENTREP_ServiceContext & create (const PORFilePath_T &, const TravelDBFilePath_T &, const SQLiteDBFilePath_T &)
- void clean ()

Static Public Member Functions

- static FacOpenTrepServiceContext & instance ()

Protected Member Functions

- FacOpenTrepServiceContext ()

Protected Attributes

- ServicePool_T _pool

23.39.1 Detailed Description

Factory for Bucket.

Definition at line 19 of file FacOpenTrepServiceContext.hpp.

23.39.2 Member Typedef Documentation

23.39.2.1 typedef std::vector<ServiceAbstract*> OPENTREP::FacServiceAbstract::ServicePool_T [inherited]

Define the list (pool) of Service objects.

Definition at line 20 of file FacServiceAbstract.hpp.

23.39.3 Constructor & Destructor Documentation

23.39.3.1 OPENTREP::FacOpenTrepServiceContext::~FacOpenTrepServiceContext ( )

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacOpentrepServiceContext::instance()

Definition at line 17 of file FacOpenTrepServiceContext.cpp.

23.39.3.2 OPENTREP::FacOpenTrepServiceContext::FacOpenTrepServiceContext ( ) [inline], [protected]

Default Constructor.

This constructor is protected in order to ensure the singleton pattern.

Definition at line 71 of file FacOpenTrepServiceContext.hpp.

Referenced by instance().

23.39.4 Member Function Documentation

23.39.4.1 FacOpenTrepServiceContext & OPENTREP::FacOpenTrepServiceContext::instance ( ) [static]

Provide the unique instance.

The singleton is instantiated when first used

Returns

FacOpentrepServiceContext&

Definition at line 22 of file FacOpenTrepServiceContext.cpp.

References FacOpenTrepServiceContext(), OPENTREP::FacSupervisor::instance(), and OPENTREP::FacSupervisor::registerServiceFactory().

23.39.4.2 OPENTREP_ServiceContext & OPENTREP::FacOpenTrepServiceContext::create (const TravelDBFilePath_T & *iTravelDBFilePath*, const SQLiteDBFilePath_T & *iSQLiteDBFilePath*)

Create a new OPENTREP_ServiceContext object.

This new object is added to the list of instantiated objects.

Parameters

const	TravelDBFilePath_T& File-path of the Xapian database.
const	SQLiteDBFilePath_T& File-path of the SQLite3 database.

Returns

OPENTREP_ServiceContext& The newly created object.

Definition at line 35 of file FacOpenTrepServiceContext.cpp.

References OPENTREP::FacServiceAbstract::_pool.

23.39.4.3 OPENTREP_ServiceContext & OPENTREP::FacOpenTrepServiceContext::create (const PORFilePath_T & *iPORFilePath,* const TravelDBFilePath_T & *iTravelDBFilePath,* const SQLiteDBFilePath_T & *iSQLiteDBFilePath*)

Create a new OPENTREP_ServiceContext object.

This new object is added to the list of instantiated objects.

Parameters

const	PORFilePath_T& File-path of the POR file.
const	TravelDBFilePath_T& File-path of the Xapian database.
const	SQLiteDBFilePath_T& File-path of the SQLite3 database.

Returns

OPENTREP_ServiceContext& The newly created object.

Definition at line 51 of file FacOpenTrepServiceContext.cpp.

References OPENTREP::FacServiceAbstract::_pool.

23.39.4.4 void OPENTREP::FacServiceAbstract::clean ( ) [inherited]

Destroyed all the object instantiated by this factory.

Definition at line 18 of file FacServiceAbstract.cpp.

References OPENTREP::FacServiceAbstract::_pool.

Referenced by OPENTREP::FacServiceAbstract::~FacServiceAbstract().

23.39.5 Member Data Documentation

23.39.5.1 ServicePool_T OPENTREP::FacServiceAbstract::_pool [protected], [inherited]

List of instantiated Business Objects

Definition at line 34 of file FacServiceAbstract.hpp.

Referenced by OPENTREP::FacServiceAbstract::clean(), and create().

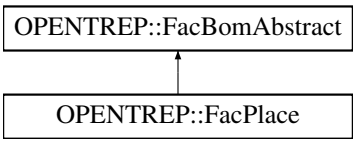
The documentation for this class was generated from the following files:

- opentrep/factory/FacOpenTrepServiceContext.hpp
- opentrep/factory/FacOpenTrepServiceContext.cpp

23.40 OPENTREP::FacPlace Class Reference

`#include <opentrep/factory/FacPlace.hpp>`

Inheritance diagram for OPENTREP::FacPlace:



Public Types

- typedef std::vector < BomAbstract * > BomPool_T

Public Member Functions

- virtual ~FacPlace ()
- Place & create ()
- Place & create (const LocationKey &)
- Place & create (const Location &)
- Place & clone (const Place &)

Static Public Member Functions

- static FacPlace & instance ()
- static void initLinkWithExtraPlace (Place &, Place &)
- static void initLinkWithAlternatePlace (Place &, Place &)
- static std::size_t getID (const BomAbstract *)
- static std::size_t getID (const BomAbstract &)
- static std::string getIDString (const BomAbstract *)
- static std::string getIDString (const BomAbstract &)

Protected Attributes

- BomPool_T _pool

23.40.1 Detailed Description

Factory for Place.

Definition at line 20 of file FacPlace.hpp.

23.40.2 Member Typedef Documentation

23.40.2.1 typedef std::vector<BomAbstract*> OPENTREP::FacBomAbstract::BomPool_T [inherited]

Define the list (pool) of Bom objects.

Definition at line 22 of file FacBomAbstract.hpp.

23.40.3 Constructor & Destructor Documentation

23.40.3.1 OPENTREP::FacPlace::~FacPlace ( ) [virtual]

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacPlace::instance()

Definition at line 24 of file FacPlace.cpp.

23.40.4 Member Function Documentation

23.40.4.1 FacPlace & OPENTREP::FacPlace::instance ( ) [static]

Provide the unique instance.

The singleton is instantiated when first used

Returns

FacPlace&

Definition at line 29 of file FacPlace.cpp.

References OPENTREP::FacSupervisor::instance(), and OPENTREP::FacSupervisor::registerBomFactory().

Referenced by OPENTREP::DBManager::buildSQLDB(), OPENTREP::createPlaces(), and OPENTREP::DBManager::prepareSelectFromCoordStatement().

23.40.4.2 Place & OPENTREP::FacPlace::create ( )

Create a new Place object.

This new object is added to the list of instantiated objects.

Returns

Place& The newly created object.

Definition at line 41 of file FacPlace.cpp.

References OPENTREP::FacBomAbstract::_pool.

Referenced by OPENTREP::DBManager::buildSQLDB(), OPENTREP::createPlaces(), and OPENTREP::DBManager::prepareSelectFromCoordStatement().

23.40.4.3 Place & OPENTREP::FacPlace::create (const LocationKey & *iLocationKey*)

Create a new Place object.

This new object is added to the list of instantiated objects.

Parameters

const	LocationKey& The primary key (IATA and ICAO codes, Geonames ID) of the place

Returns

Place& The newly created object.

Definition at line 54 of file FacPlace.cpp.

References OPENTREP::FacBomAbstract::_pool.

23.40.4.4 Place & OPENTREP::FacPlace::create (const Location & *iLocation*)

Create a new Place object.

This new object is added to the list of instantiated objects.

Parameters

const	Location& The primary key (IATA and ICAO codes, Geonames ID) of the place

Returns

Place& The newly created object.

Definition at line 67 of file FacPlace.cpp.

References OPENTREP::FacBomAbstract::_pool.

23.40.4.5 Place & OPENTREP::FacPlace::clone (const Place & *iPlace*)

Create a copy of a Place object.

This new object is added to the list of instantiated objects.

Returns

Place& The newly created object.

Definition at line 80 of file FacPlace.cpp.

References OPENTREP::FacBomAbstract::_pool.

23.40.4.6 void OPENTREP::FacPlace::initLinkWithExtraPlace (Place & *ioMainPlace,* Place & *ioExtraPlace*) [static]

Initialise the link between a Place and an extra Place.

Parameters

Place&	Main Place object.
Place&	Extra Place object.

Exceptions

FacExceptionNullPointer	
FacException.	

Definition at line 93 of file FacPlace.cpp.

23.40.4.7 void OPENTREP::FacPlace::initLinkWithAlternatePlace (Place & *ioMainPlace,* Place & *ioAlternatePlace*) [static]

Initialise the link between a Place and an alternate Place.

Parameters

Place&	Main Place object.
Place&	Alternate Place object.

Exceptions

FacExceptionNullPointer	
FacException.	

Definition at line 104 of file FacPlace.cpp.

23.40.4.8 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID corresponding to the given object pointer.

Definition at line 36 of file FacBomAbstract.cpp.

Referenced by OPENTREP::FacBomAbstract::getID(), and OPENTREP::FacBomAbstract::getIDString().

23.40.4.9 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID corresponding to the given object reference.

Definition at line 44 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

23.40.4.10 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID, as a string, corresponding to the given object pointer.

Definition at line 49 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

Referenced by OPENTREP::FacBomAbstract::getIDString().

23.40.4.11 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID, as a string, corresponding to the given object reference.

Definition at line 57 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getIDString().

23.40.5 Member Data Documentation

23.40.5.1 BomPool_T OPENTREP::FacBomAbstract::_pool [protected], [inherited]

List of instantiated Business Objects

Definition at line 53 of file FacBomAbstract.hpp.

Referenced by OPENTREP::FacWorld::clone(), clone(), OPENTREP::FacPlaceHolder::create(), OPENTREP::FacWorld::create(), OPENTREP::FacResultHolder::create(), OPENTREP::FacResult::create(), create(), and OPENTREP::FacResultCombination::create().

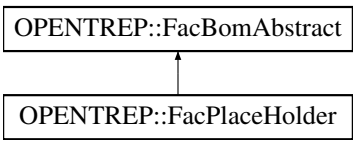
The documentation for this class was generated from the following files:

- opentrep/factory/FacPlace.hpp
- opentrep/factory/FacPlace.cpp

23.41 OPENTREP::FacPlaceHolder Class Reference

`#include <opentrep/factory/FacPlaceHolder.hpp>`

Inheritance diagram for OPENTREP::FacPlaceHolder:



Public Types

- typedef std::vector < BomAbstract * > BomPool_T

Public Member Functions

- virtual ~FacPlaceHolder ()
- PlaceHolder & create ()

Static Public Member Functions

- static FacPlaceHolder & instance ()
- static void initLinkWithPlace (PlaceHolder &, Place &)
- static std::size_t getID (const BomAbstract *)
- static std::size_t getID (const BomAbstract &)
- static std::string getIDString (const BomAbstract *)
- static std::string getIDString (const BomAbstract &)

Protected Attributes

- BomPool_T _pool

23.41.1 Detailed Description

Factory for Place.

Definition at line 17 of file FacPlaceHolder.hpp.

23.41.2 Member Typedef Documentation

23.41.2.1 typedef std::vector<BomAbstract*> OPENTREP::FacBomAbstract::BomPool_T [inherited]

Define the list (pool) of Bom objects.

Definition at line 22 of file FacBomAbstract.hpp.

23.41.3 Constructor & Destructor Documentation

23.41.3.1 OPENTREP::FacPlaceHolder::~FacPlaceHolder ( ) [virtual]

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacPlaceHolder::instance()

Definition at line 26 of file FacPlaceHolder.cpp.

23.41.4 Member Function Documentation

23.41.4.1 FacPlaceHolder & OPENTREP::FacPlaceHolder::instance ( ) [static]

Provide the unique instance.

The singleton is instantiated when first used

Returns

FacPlaceHolder&

Definition at line 31 of file FacPlaceHolder.cpp.

References OPENTREP::FacSupervisor::instance(), and OPENTREP::FacSupervisor::registerBomFactory().

23.41.4.2 PlaceHolder & OPENTREP::FacPlaceHolder::create ( )

Create a new PlaceHolder object.

This new object is added to the list of instantiated objects.

Returns

PlaceHolder& The newly created object.

Definition at line 43 of file FacPlaceHolder.cpp.

References OPENTREP::FacBomAbstract::_pool.

23.41.4.3 void OPENTREP::FacPlaceHolder::initLinkWithPlace (PlaceHolder & *ioPlaceHolder,* Place & *ioPlace*) [static]

Initialise the link between a PlaceHolder and a Place.

Parameters

PlaceHolder&	
Place&	

Exceptions

FacExceptionNullPointer	
FacException.	

Definition at line 56 of file FacPlaceHolder.cpp.

References OPENTREP::Place::describeShortKey().

Referenced by OPENTREP::createPlaces().

23.41.4.4 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID corresponding to the given object pointer.

Definition at line 36 of file FacBomAbstract.cpp.

Referenced by OPENTREP::FacBomAbstract::getID(), and OPENTREP::FacBomAbstract::getIDString().

23.41.4.5 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID corresponding to the given object reference.

Definition at line 44 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

23.41.4.6 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID, as a string, corresponding to the given object pointer.

Definition at line 49 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

Referenced by OPENTREP::FacBomAbstract::getIDString().

23.41.4.7 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID, as a string, corresponding to the given object reference.

Definition at line 57 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getIDString().

23.41.5 Member Data Documentation

23.41.5.1 BomPool_T OPENTREP::FacBomAbstract::_pool [protected],[inherited]

List of instantiated Business Objects

Definition at line 53 of file FacBomAbstract.hpp.

Referenced by OPENTREP::FacWorld::clone(), OPENTREP::FacPlace::clone(), OPENTREP::FacWorld::create(), create(), OPENTREP::FacResultHolder::create(), OPENTREP::FacResult::create(), OPENTREP::FacPlace-::create(), and OPENTREP::FacResultCombination::create().

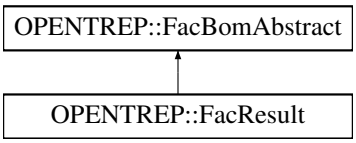
The documentation for this class was generated from the following files:

- opentrep/factory/FacPlaceHolder.hpp
- opentrep/factory/FacPlaceHolder.cpp

23.42 OPENTREP::FacResult Class Reference

`#include <opentrep/factory/FacResult.hpp>`

Inheritance diagram for OPENTREP::FacResult:



Public Types

- typedef std::vector
  < BomAbstract ∗ > BomPool_T

Public Member Functions

- virtual ∼FacResult ()
- Result & create (const TravelQuery_T &, const Xapian::Database &)

Static Public Member Functions

- static FacResult & instance ()
- static std::size_t getID (const BomAbstract ∗)
- static std::size_t getID (const BomAbstract &)
- static std::string getIDString (const BomAbstract ∗)
- static std::string getIDString (const BomAbstract &)

Protected Attributes

- BomPool_T _pool

23.42.1 Detailed Description

Factory for Result.

Definition at line 18 of file FacResult.hpp.

23.42.2 Member Typedef Documentation

23.42.2.1 typedef std::vector<BomAbstract*> OPENTREP::FacBomAbstract::BomPool_T [inherited]

Define the list (pool) of Bom objects.

Definition at line 22 of file FacBomAbstract.hpp.

23.42.3 Constructor & Destructor Documentation

23.42.3.1 OPENTREP::FacResult::~FacResult ( ) [virtual]

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacResult::instance()

Definition at line 24 of file FacResult.cpp.

23.42.4 Member Function Documentation

23.42.4.1 FacResult & OPENTREP::FacResult::instance ( ) [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns

FacResult&

Definition at line 29 of file FacResult.cpp.

References OPENTREP::FacSupervisor::instance(), and OPENTREP::FacSupervisor::registerBomFactory().

Referenced by OPENTREP::searchString().

23.42.4.2 Result & OPENTREP::FacResult::create (const TravelQuery_T & *iQueryString*, const Xapian::Database & *iXapianDatabase*)

Create a new Result object.

This new object is added to the list of instantiated objects.

Returns

Result& The newly created object.

Definition at line 41 of file FacResult.cpp.

References OPENTREP::FacBomAbstract::_pool.

Referenced by OPENTREP::searchString().

23.42.4.3 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID corresponding to the given object pointer.

Definition at line 36 of file FacBomAbstract.cpp.

Referenced by OPENTREP::FacBomAbstract::getID(), and OPENTREP::FacBomAbstract::getIDString().

23.42.4.4 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID corresponding to the given object reference.

Definition at line 44 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

23.42.4.5 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID, as a string, corresponding to the given object pointer.

Definition at line 49 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

Referenced by OPENTREP::FacBomAbstract::getIDString().

23.42.4.6 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID, as a string, corresponding to the given object reference.

Definition at line 57 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getIDString().

23.42.5 Member Data Documentation

23.42.5.1 BomPool_T OPENTREP::FacBomAbstract::_pool [protected], [inherited]

List of instantiated Business Objects

Definition at line 53 of file FacBomAbstract.hpp.

Referenced by OPENTREP::FacWorld::clone(), OPENTREP::FacPlace::clone(), OPENTREP::FacPlaceHolder::create(), OPENTREP::FacWorld::create(), OPENTREP::FacResultHolder::create(), create(), OPENTREP::FacPlace::create(), and OPENTREP::FacResultCombination::create().

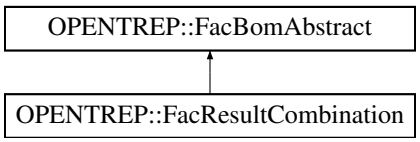
The documentation for this class was generated from the following files:

- opentrep/factory/FacResult.hpp
- opentrep/factory/FacResult.cpp

23.43 OPENTREP::FacResultCombination Class Reference

`#include <opentrep/factory/FacResultCombination.hpp>`

Inheritance diagram for OPENTREP::FacResultCombination:



Public Types

- typedef std::vector < BomAbstract * > BomPool_T

Public Member Functions

- virtual ~FacResultCombination ()
- ResultCombination & create (const TravelQuery_T &iQueryString)

Static Public Member Functions

- static FacResultCombination & instance ()
- static void initLinkWithResultHolder (ResultCombination &, ResultHolder &)
- static std::size_t getID (const BomAbstract *)
- static std::size_t getID (const BomAbstract &)
- static std::string getIDString (const BomAbstract *)
- static std::string getIDString (const BomAbstract &)

Protected Attributes

- BomPool_T _pool

23.43.1 Detailed Description

Factory for ResultCombination.

Definition at line 25 of file FacResultCombination.hpp.

23.43.2 Member Typedef Documentation

23.43.2.1 typedef std::vector<BomAbstract*> OPENTREP::FacBomAbstract::BomPool_T [inherited]

Define the list (pool) of Bom objects.

Definition at line 22 of file FacBomAbstract.hpp.

23.43.3 Constructor & Destructor Documentation

23.43.3.1 OPENTREP::FacResultCombination::~FacResultCombination ( ) [virtual]

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacResultCombination::instance()

Definition at line 26 of file FacResultCombination.cpp.

23.43.4 Member Function Documentation

23.43.4.1 FacResultCombination & OPENTREP::FacResultCombination::instance ( ) [static]

Provide the unique instance.

The singleton is instantiated when first used

Returns

FacResultCombination&

Definition at line 31 of file FacResultCombination.cpp.

References OPENTREP::FacSupervisor::instance(), and OPENTREP::FacSupervisor::registerBomFactory().

23.43.4.2 ResultCombination & OPENTREP::FacResultCombination::create (const TravelQuery_T & *iQueryString*)

Create a new ResultCombination object.

This new object is added to the list of instantiated objects.

Returns

ResultCombination& The newly created object.

Definition at line 44 of file FacResultCombination.cpp.

References OPENTREP::FacBomAbstract::_pool.

23.43.4.3 void OPENTREP::FacResultCombination::initLinkWithResultHolder (ResultCombination & *ioResultCombination*, ResultHolder & *ioResultHolder*) [static]

Initialise the link between a ResultCombination and a ResultHolder.

Parameters

Result-Combination&	
ResultHolder&	

Exceptions

FacExceptionNullPointer	
FacException.	

Definition at line 58 of file FacResultCombination.cpp.

Referenced by OPENTREP::searchString().

23.43.4.4 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID corresponding to the given object pointer.

Definition at line 36 of file FacBomAbstract.cpp.

Referenced by OPENTREP::FacBomAbstract::getID(), and OPENTREP::FacBomAbstract::getIDString().

23.43.4.5 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID corresponding to the given object reference.

Definition at line 44 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

23.43.4.6 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID, as a string, corresponding to the given object pointer.

Definition at line 49 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

Referenced by OPENTREP::FacBomAbstract::getIDString().

23.43.4.7 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID, as a string, corresponding to the given object reference.

Definition at line 57 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getIDString().

23.43.5 Member Data Documentation

23.43.5.1 BomPool_T OPENTREP::FacBomAbstract::_pool [protected], [inherited]

List of instantiated Business Objects

Definition at line 53 of file FacBomAbstract.hpp.

Referenced by OPENTREP::FacWorld::clone(), OPENTREP::FacPlace::clone(), OPENTREP::FacWorld::create(), OPENTREP::FacPlaceHolder::create(), OPENTREP::FacResultHolder::create(), OPENTREP::FacResult::create(), OPENTREP::FacPlace::create(), and create().

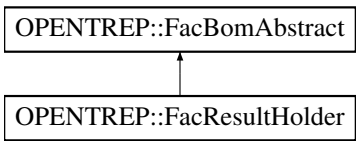
The documentation for this class was generated from the following files:

- opentrep/factory/FacResultCombination.hpp
- opentrep/factory/FacResultCombination.cpp

23.44 OPENTREP::FacResultHolder Class Reference

`#include <opentrep/factory/FacResultHolder.hpp>`

Inheritance diagram for OPENTREP::FacResultHolder:



Public Types

- typedef std::vector < BomAbstract * > BomPool_T

Public Member Functions

- virtual ~FacResultHolder ()
- ResultHolder & create (const TravelQuery_T &iQueryString, const Xapian::Database &iDatabase)

Static Public Member Functions

- static FacResultHolder & instance ()
- static void initLinkWithResult (ResultHolder &, Result &)
- static std::size_t getID (const BomAbstract *)
- static std::size_t getID (const BomAbstract &)
- static std::string getIDString (const BomAbstract *)
- static std::string getIDString (const BomAbstract &)

Protected Attributes

- BomPool_T _pool

23.44.1 Detailed Description

Factory for Result.

Definition at line 23 of file FacResultHolder.hpp.

23.44.2 Member Typedef Documentation

23.44.2.1 typedef std::vector<BomAbstract*> OPENTREP::FacBomAbstract::BomPool_T [inherited]

Define the list (pool) of Bom objects.

Definition at line 22 of file FacBomAbstract.hpp.

23.44.3 Constructor & Destructor Documentation

23.44.3.1 OPENTREP::FacResultHolder::~FacResultHolder ( ) [virtual]

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacResultHolder::instance()

Definition at line 26 of file FacResultHolder.cpp.

23.44.4 Member Function Documentation

23.44.4.1 FacResultHolder & OPENTREP::FacResultHolder::instance ( ) [static]

Provide the unique instance.

The singleton is instantiated when first used

Returns

FacResultHolder&

Definition at line 31 of file FacResultHolder.cpp.

References OPENTREP::FacSupervisor::instance(), and OPENTREP::FacSupervisor::registerBomFactory().

Referenced by OPENTREP::searchString().

23.44.4.2 ResultHolder & OPENTREP::FacResultHolder::create (const TravelQuery_T & *iQueryString*, const Xapian::Database & *iDatabase*)

Create a new ResultHolder object.

This new object is added to the list of instantiated objects.

Returns

ResultHolder& The newly created object.

Definition at line 43 of file FacResultHolder.cpp.

References OPENTREP::FacBomAbstract::_pool.

Referenced by OPENTREP::searchString().

23.44.4.3 void OPENTREP::FacResultHolder::initLinkWithResult (ResultHolder & *ioResultHolder,* Result & *ioResult*) [static]

Initialise the link between a ResultHolder and a Result.

Parameters

ResultHolder&	
Result&	

Exceptions

FacExceptionNullPointer	
FacException.	

Definition at line 57 of file FacResultHolder.cpp.

Referenced by OPENTREP::searchString().

23.44.4.4 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID corresponding to the given object pointer.

Definition at line 36 of file FacBomAbstract.cpp.

Referenced by OPENTREP::FacBomAbstract::getID(), and OPENTREP::FacBomAbstract::getIDString().

23.44.4.5 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID corresponding to the given object reference.

Definition at line 44 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

23.44.4.6 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID, as a string, corresponding to the given object pointer.

Definition at line 49 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

Referenced by OPENTREP::FacBomAbstract::getIDString().

23.44.4.7 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID, as a string, corresponding to the given object reference.

Definition at line 57 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getIDString().

23.44.5 Member Data Documentation

23.44.5.1 BomPool_T OPENTREP::FacBomAbstract::_pool [protected], [inherited]

List of instantiated Business Objects

Definition at line 53 of file FacBomAbstract.hpp.

Referenced by OPENTREP::FacWorld::clone(), OPENTREP::FacPlace::clone(), OPENTREP::FacPlaceHolder::create(), OPENTREP::FacWorld::create(), create(), OPENTREP::FacResult::create(), OPENTREP::FacPlace::create(), and OPENTREP::FacResultCombination::create().

The documentation for this class was generated from the following files:

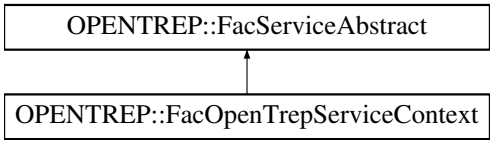
- opentrep/factory/FacResultHolder.hpp

- opentrep/factory/FacResultHolder.cpp

23.45 OPENTREP::FacServiceAbstract Class Reference

`#include <opentrep/factory/FacServiceAbstract.hpp>`

Inheritance diagram for OPENTREP::FacServiceAbstract:



Public Types

- typedef std::vector < ServiceAbstract * > ServicePool_T

Public Member Functions

- virtual ~FacServiceAbstract ()
- void clean ()

Protected Member Functions

- FacServiceAbstract ()

Protected Attributes

- ServicePool_T _pool

23.45.1 Detailed Description

Base class for the (Service) Factory layer.

Definition at line 16 of file FacServiceAbstract.hpp.

23.45.2 Member Typedef Documentation

23.45.2.1 typedef std::vector<ServiceAbstract*> OPENTREP::FacServiceAbstract::ServicePool_T

Define the list (pool) of Service objects.

Definition at line 20 of file FacServiceAbstract.hpp.

23.45.3 Constructor & Destructor Documentation

23.45.3.1 OPENTREP::FacServiceAbstract::~FacServiceAbstract ( ) [virtual]

Destructor.

Definition at line 13 of file FacServiceAbstract.cpp.

References clean().

23.45.3.2 OPENTREP::FacServiceAbstract::FacServiceAbstract ( ) [inline],[protected]

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 31 of file FacServiceAbstract.hpp.

23.45.4 Member Function Documentation

23.45.4.1 void OPENTREP::FacServiceAbstract::clean ( )

Destroyed all the object instantiated by this factory.

Definition at line 18 of file FacServiceAbstract.cpp.

References _pool.

Referenced by ~FacServiceAbstract().

23.45.5 Member Data Documentation

23.45.5.1 ServicePool_T OPENTREP::FacServiceAbstract::_pool [protected]

List of instantiated Business Objects

Definition at line 34 of file FacServiceAbstract.hpp.

Referenced by clean(), and OPENTREP::FacOpenTrepServiceContext::create().

The documentation for this class was generated from the following files:

- opentrep/factory/FacServiceAbstract.hpp
- opentrep/factory/FacServiceAbstract.cpp

23.46 OPENTREP::FacSupervisor Class Reference

`#include <opentrep/factory/FacSupervisor.hpp>`

Public Types

- typedef std::vector < FacBomAbstract * > BomFactoryPool_T
- typedef std::vector < FacServiceAbstract * > ServiceFactoryPool_T

Public Member Functions

- void registerBomFactory (FacBomAbstract *)
- void registerServiceFactory (FacServiceAbstract *)
- void registerLoggerService (Logger *)
- void cleanBomLayer ()
- void cleanServiceLayer ()
- void cleanLoggerService ()
- ~FacSupervisor ()

Static Public Member Functions

- static FacSupervisor & instance ()
- static void cleanFactory ()

Protected Member Functions

- FacSupervisor ()
- FacSupervisor (const FacSupervisor &)

23.46.1 Detailed Description

Singleton class to register and clean all Factories.

Definition at line 18 of file FacSupervisor.hpp.

23.46.2 Member Typedef Documentation

23.46.2.1 typedef std::vector<FacBomAbstract*> OPENTREP::FacSupervisor::BomFactoryPool_T

Define the pool (list) of factories.

Definition at line 22 of file FacSupervisor.hpp.

23.46.2.2 typedef std::vector<FacServiceAbstract*> OPENTREP::FacSupervisor::ServiceFactoryPool_T

Definition at line 23 of file FacSupervisor.hpp.

23.46.3 Constructor & Destructor Documentation

23.46.3.1 OPENTREP::FacSupervisor::~FacSupervisor ( )

Destructor

The static instance is deleted (and reset to NULL) by the static cleanFactory() method.

Definition at line 48 of file FacSupervisor.cpp.

References cleanBomLayer(), cleanLoggerService(), and cleanServiceLayer().

23.46.3.2 OPENTREP::FacSupervisor::FacSupervisor ( ) [protected]

Default Constructor.

This constructor is protected to ensure the singleton pattern.

Definition at line 17 of file FacSupervisor.cpp.

Referenced by instance().

23.46.3.3 OPENTREP::FacSupervisor::FacSupervisor (const FacSupervisor &) [inline], [protected]

Definition at line 80 of file FacSupervisor.hpp.

23.46.4 Member Function Documentation

23.46.4.1 FacSupervisor & OPENTREP::FacSupervisor::instance ( ) [static]

Provides the unique instance.

The singleton is instantiated when first used.

Returns

FacSupervisor&

Definition at line 22 of file FacSupervisor.cpp.

References FacSupervisor().

Referenced by OPENTREP::FacPlaceHolder::instance(), OPENTREP::FacWorld::instance(), OPENTREP::FacResult::instance(), OPENTREP::FacResultHolder::instance(), OPENTREP::FacOpenTrepServiceContext::instance(), OPENTREP::FacPlace::instance(), and OPENTREP::FacResultCombination::instance().

23.46.4.2 void OPENTREP::FacSupervisor::registerBomFactory (FacBomAbstract * *ioFacBomAbstract_ptr*)

Register a newly instantiated concrete factory for the Bom layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the FacSupervisor

Parameters

FacAbstract&	the concrete Factory to register.

Definition at line 32 of file FacSupervisor.cpp.

Referenced by OPENTREP::FacPlaceHolder::instance(), OPENTREP::FacWorld::instance(), OPENTREP::FacResult::instance(), OPENTREP::FacResultHolder::instance(), OPENTREP::FacPlace::instance(), and OPENTREP::FacResultCombination::instance().

23.46.4.3 void OPENTREP::FacSupervisor::registerServiceFactory (FacServiceAbstract * *ioFacServiceAbstract_ptr*)

Register a newly instantiated concrete factory for the Service layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the FacSupervisor.

Parameters

FacService-Abstract&	the concrete Factory to register.

Definition at line 38 of file FacSupervisor.cpp.

Referenced by OPENTREP::FacOpenTrepServiceContext::instance().

23.46.4.4 void OPENTREP::FacSupervisor::registerLoggerService (Logger * *ioLogger_ptr*)

Register a newly instantiated concrete factory for the Logger object. In fact, as the Logger object follows the singleton pattern, the concrete factory is the Logger object itself.

When a concrete Factory is firstly instantiated this factory have to register itself to the FacSupervisor.

Parameters

FacService-Abstract&	the concrete Factory to register.

Definition at line 43 of file FacSupervisor.cpp.

23.46.4.5 void OPENTREP::FacSupervisor::cleanBomLayer ( )

Clean all created object.

Call the clean method of all the instantiated factories for the Bom layer.

Definition at line 55 of file FacSupervisor.cpp.

Referenced by cleanFactory(), and ~FacSupervisor().

23.46.4.6 void OPENTREP::FacSupervisor::cleanServiceLayer ( )

Clean all Service created object.

Call the clean method of all the instantiated factories for the Service layer.

Definition at line 69 of file FacSupervisor.cpp.

Referenced by cleanFactory(), and ~FacSupervisor().

23.46.4.7 void OPENTREP::FacSupervisor::cleanLoggerService ( )

Delete the Logger object.

Definition at line 83 of file FacSupervisor.cpp.

Referenced by cleanFactory(), and ~FacSupervisor().

23.46.4.8 void OPENTREP::FacSupervisor::cleanFactory ( ) [static]

Clean the static instance.

The singleton is deleted.

Definition at line 88 of file FacSupervisor.cpp.

References cleanBomLayer(), cleanLoggerService(), and cleanServiceLayer().

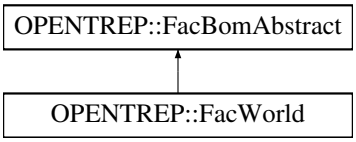
The documentation for this class was generated from the following files:

- opentrep/factory/FacSupervisor.hpp
- opentrep/factory/FacSupervisor.cpp

23.47 OPENTREP::FacWorld Class Reference

`#include <opentrep/factory/FacWorld.hpp>`

Inheritance diagram for OPENTREP::FacWorld:



Public Types

- typedef std::vector < BomAbstract * > BomPool_T

Public Member Functions

- virtual ~FacWorld ()
- World & create ()
- World & clone (const World &)

Static Public Member Functions

- static FacWorld & instance ()
- static void linkPlaceToWorld (const bool iNeedsGeneric, World &, Place &)

- static std::size_t getID (const BomAbstract *)
- static std::size_t getID (const BomAbstract &)
- static std::string getIDString (const BomAbstract *)
- static std::string getIDString (const BomAbstract &)

Protected Attributes

- BomPool_T _pool

23.47.1 Detailed Description

Factory for World.

Definition at line 17 of file FacWorld.hpp.

23.47.2 Member Typedef Documentation

23.47.2.1 typedef std::vector<BomAbstract*> OPENTREP::FacBomAbstract::BomPool_T [inherited]

Define the list (pool) of Bom objects.

Definition at line 22 of file FacBomAbstract.hpp.

23.47.3 Constructor & Destructor Documentation

23.47.3.1 OPENTREP::FacWorld::~FacWorld ( ) [virtual]

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacWorld::instance()

Definition at line 21 of file FacWorld.cpp.

23.47.4 Member Function Documentation

23.47.4.1 FacWorld & OPENTREP::FacWorld::instance ( ) [static]

Provide the unique instance.

The singleton is instantiated when first used

Returns

FacWorld&

Definition at line 26 of file FacWorld.cpp.

References OPENTREP::FacSupervisor::instance(), and OPENTREP::FacSupervisor::registerBomFactory().

23.47.4.2 World & OPENTREP::FacWorld::create ( )

Create a new World object.

This new object is added to the list of instantiated objects.

Returns

World& The newly created object.

Definition at line 38 of file FacWorld.cpp.

References OPENTREP::FacBomAbstract::_pool.

23.47.4.3 void OPENTREP::FacWorld::linkPlaceToWorld (const bool *iNeedsGeneric*, World & *ioWorld*, Place & *ioPlace*) [static]

Link a Place object to a World one.

Returns

void.

Definition at line 51 of file FacWorld.cpp.

References OPENTREP::World::describeKey(), OPENTREP::Place::describeShortKey(), OPENTREP::Place::getDocID(), OPENTREP_LOG_ERROR, and OPENTREP::BomType::PLACE.

23.47.4.4 World & OPENTREP::FacWorld::clone (const World & *iWorld*)

Create a copy of a World object.

This new object is added to the list of instantiated objects.

Returns

World& The newly created object.

Definition at line 90 of file FacWorld.cpp.

References OPENTREP::FacBomAbstract::_pool.

23.47.4.5 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID corresponding to the given object pointer.

Definition at line 36 of file FacBomAbstract.cpp.

Referenced by OPENTREP::FacBomAbstract::getID(), and OPENTREP::FacBomAbstract::getIDString().

23.47.4.6 std::size_t OPENTREP::FacBomAbstract::getID (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID corresponding to the given object reference.

Definition at line 44 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

23.47.4.7 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract * *iBomAbstract_ptr*) [static], [inherited]

Return the ID, as a string, corresponding to the given object pointer.

Definition at line 49 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getID().

Referenced by OPENTREP::FacBomAbstract::getIDString().

23.47.4.8 std::string OPENTREP::FacBomAbstract::getIDString (const BomAbstract & *iBomAbstract*) [static], [inherited]

Return the ID, as a string, corresponding to the given object reference.

Definition at line 57 of file FacBomAbstract.cpp.

References OPENTREP::FacBomAbstract::getIDString().

23.47.5 Member Data Documentation

23.47.5.1 BomPool_T OPENTREP::FacBomAbstract::_pool [protected], [inherited]

List of instantiated Business Objects

Definition at line 53 of file FacBomAbstract.hpp.

Referenced by clone(), OPENTREP::FacPlace::clone(), create(), OPENTREP::FacPlaceHolder::create(), OPENTREP::FacResultHolder::create(), OPENTREP::FacResult::create(), OPENTREP::FacPlace::create(), and OPENTREP::FacResultCombination::create().

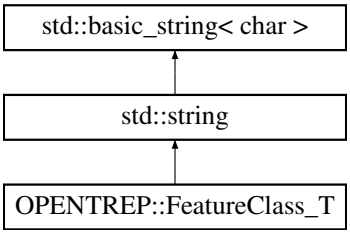
The documentation for this class was generated from the following files:

- opentrep/factory/FacWorld.hpp
- opentrep/factory/FacWorld.cpp

23.48 OPENTREP::FeatureClass_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::FeatureClass_T:



Public Member Functions

- FeatureClass_T (const std::string &iValue)

23.48.1 Detailed Description

Feature class (e.g., S for spot).

Definition at line 328 of file OPENTREP_Types.hpp.

23.48.2 Constructor & Destructor Documentation

23.48.2.1 OPENTREP::FeatureClass_T::FeatureClass_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 330 of file OPENTREP_Types.hpp.

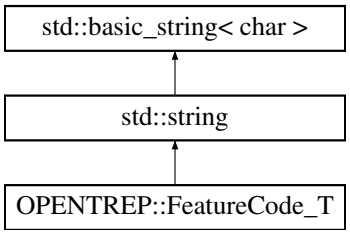
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.49 OPENTREP::FeatureCode_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::FeatureCode_T:



Public Member Functions

- FeatureCode_T (const std::string &iValue)

23.49.1 Detailed Description

Feature code (e.g., AIRP for airport).

Definition at line 337 of file OPENTREP_Types.hpp.

23.49.2 Constructor & Destructor Documentation

23.49.2.1 OPENTREP::FeatureCode_T::FeatureCode_T (const std::string & *iValue*) [inline],[explicit]

Definition at line 339 of file OPENTREP_Types.hpp.

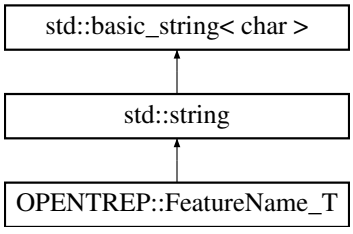
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.50 OPENTREP::FeatureName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::FeatureName_T:



Public Member Functions

- FeatureName_T (const std::string &iValue)

23.50.1 Detailed Description

Feature name (e.g., airport, airfield, airstrip, airbase, heliport).

Definition at line 345 of file OPENTREP_Types.hpp.

23.50.2 Constructor & Destructor Documentation

23.50.2.1 OPENTREP::FeatureName_T::FeatureName_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 347 of file OPENTREP_Types.hpp.

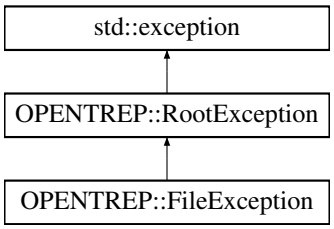
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.51 OPENTREP::FileException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::FileException:



Public Member Functions

- FileException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.51.1 Detailed Description

File.

Definition at line 134 of file OPENTREP_exceptions.hpp.

23.51.2 Constructor & Destructor Documentation

23.51.2.1 OPENTREP::FileException::FileException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 139 of file OPENTREP_exceptions.hpp.

23.51.3 Member Function Documentation

23.51.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.51.4 Member Data Documentation

23.51.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

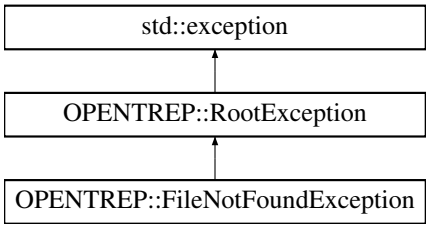
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.52 OPENTREP::FileNotFoundException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::FileNotFoundException:



Public Member Functions

- FileNotFoundException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.52.1 Detailed Description

File not found.

Definition at line 145 of file OPENTREP_exceptions.hpp.

23.52.2 Constructor & Destructor Documentation

23.52.2.1 OPENTREP::FileNotFoundException::FileNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 150 of file OPENTREP_exceptions.hpp.

23.52.3 Member Function Documentation

23.52.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.52.4 Member Data Documentation

23.52.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

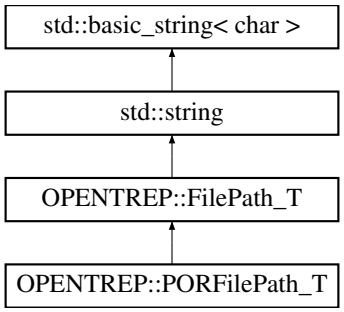
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.53 OPENTREP::FilePath_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::FilePath_T:



Public Member Functions

- FilePath_T (const std::string &iValue)

23.53.1 Detailed Description

File path.

Definition at line 28 of file OPENTREP_Types.hpp.

23.53.2 Constructor & Destructor Documentation

23.53.2.1 OPENTREP::FilePath_T::FilePath_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 30 of file OPENTREP_Types.hpp.

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.54 OPENTREP::Filter Struct Reference

Class filtering out the words not suitable for indexing and/or searching, when part of greater strings. Hence, most of the methods take as parameter the "initial"/greater string.

`#include <opentrep/bom/Filter.hpp>`

Static Public Member Functions

- static void trim (std::string &ioPhrase, const NbOfLetters_T &iMinWordLength=4)
- static bool shouldKeep (const std::string &iPhrase, const std::string &iWord)

23.54.1 Detailed Description

Class filtering out the words not suitable for indexing and/or searching, when part of greater strings. Hence, most of the methods take as parameter the "initial"/greater string.

For instance, words of length less than 3 (e.g., "de", "a", "san"), when part of greater strings (e.g., respectively, "rio de janeiro", "san francisco"), should not be indexed and searched for.

Definition at line 21 of file Filter.hpp.

23.54.2 Member Function Documentation

23.54.2.1 void OPENTREP::Filter::trim (std::string & *ioPhrase*, const NbOfLetters_T & *iMinWordLength* = 4) [static]

Trim all the non-relevant words from the given phrase.

The following rules are applied to the right and left outer words, iteratively until no more outer word can be stripped out:

- If the left or right outer word has no more than <iMinWordLength> letters (e.g., 'de', 'san'), it should be stripped out
- If the left or right outer word is part of the "black-list" (e.g., 'airport', 'intl', 'international'), it should be filtered out

Parameters

std::string&	The phrase to be amended (e.g., 'de san francisco', part of the 'aeroport de san francisco' global phrase).
const	NbOfLetters_T& The minimum length of the words (default is 4 letters).

Definition at line 131 of file Filter.cpp.

References OPENTREP::createStringFromWordList(), OPENTREP::tokeniseStringIntoWordList(), and OPENTREP::trim().

Referenced by OPENTREP::Result::calculateCodeMatches().

23.54.2.2 bool OPENTREP::Filter::shouldKeep (const std::string & *iPhrase*, const std::string & *iWord*) [static]

State whether or not to keep the given word, as opposed to filter out a non-indexable/searchable word.

The following rules are applied in sequence (if a rule applies, then the method returns, and the other rules are not processed/checked):

- When the word is equal to the phrase (e.g., 'san'), it should be kept (not filtered out), as it is obviously here intentionally
- If the word has no more than 3 letters (e.g., 'de', 'san'), it should be filtered out
- If the word is part of the "black-list" (e.g., 'airport', 'intl', 'international'), it should be filtered out

Parameters

const	std::string& The initial phrase (e.g., 'san francisco airport').
const	std::string& The word on which a decision has to be made

Returns

bool Whether or not the word should be kept / filtered out

Definition at line 144 of file Filter.cpp.

References OPENTREP::hasGoodSize(), and OPENTREP::isBlackListed().

Referenced by OPENTREP::addUnmatchedWord(), OPENTREP::Result::calculateCombinedWeights(), and OPENTREP::Result::fullTextMatch().

The documentation for this struct was generated from the following files:

- opentrep/bom/Filter.hpp
- opentrep/bom/Filter.cpp

23.55 FloatingPoint< RawType > Class Template Reference

`#include <opentrep/basic/float_utils_google.hpp>`

Public Types

- typedef TypeWithSize< sizeof(RawType)> ::UInt Bits

Public Member Functions

- FloatingPoint (const RawType &x)
- const Bits & bits () const
- Bits exponent_bits () const
- Bits fraction_bits () const
- Bits sign_bit () const
- bool is_nan () const
- bool AlmostEquals (const FloatingPoint &rhs) const

Static Public Member Functions

- static RawType ReinterpretBits (const Bits bits)
- static RawType Infinity ()

Static Public Attributes

- static const size_t kBitCount = 8*sizeof(RawType)
- static const size_t kFractionBitCount
- static const size_t kExponentBitCount = kBitCount - 1 - kFractionBitCount
- static const Bits kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1)
- static const Bits kFractionBitMask
- static const Bits kExponentBitMask = ~(kSignBitMask | kFractionBitMask)
- static const size_t kMaxUlps = 4

23.55.1 Detailed Description

template<typename RawType>class FloatingPoint< RawType >

Definition at line 123 of file float_utils_google.hpp.

23.55.2 Member Typedef Documentation

23.55.2.1 template<typename RawType> typedef TypeWithSize<sizeof(RawType)>::UInt FloatingPoint< RawType >::Bits

Definition at line 127 of file float_utils_google.hpp.

23.55.3 Constructor & Destructor Documentation

23.55.3.1 template<typename RawType> FloatingPoint< RawType >::FloatingPoint (const RawType & *x*) [inline], [explicit]

Definition at line 171 of file float_utils_google.hpp.

23.55.4 Member Function Documentation

23.55.4.1 template<typename RawType> static RawType FloatingPoint< RawType >::ReinterpretBits (const Bits *bits*) [inline], [static]

Definition at line 178 of file float_utils_google.hpp.

References FloatingPoint< RawType >::bits().

Referenced by FloatingPoint< RawType >::Infinity().

23.55.4.2 template<typename RawType> static RawType FloatingPoint< RawType >::Infinity ( ) [inline], [static]

Definition at line 185 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kExponentBitMask, and FloatingPoint< RawType >::ReinterpretBits().

23.55.4.3 template<typename RawType> const Bits& FloatingPoint< RawType >::bits ( ) const [inline]

Definition at line 192 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::ReinterpretBits().

23.55.4.4 template<typename RawType> Bits FloatingPoint< RawType >::exponent_bits ( ) const [inline]

Definition at line 195 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kExponentBitMask.

Referenced by FloatingPoint< RawType >::is_nan().

23.55.4.5 template<typename RawType> Bits FloatingPoint< RawType >::fraction_bits ( ) const [inline]

Definition at line 198 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kFractionBitMask.

Referenced by FloatingPoint< RawType >::is_nan().

23.55.4.6 template<typename RawType> Bits FloatingPoint< RawType >::sign_bit ( ) const [inline]

Definition at line 201 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kSignBitMask.

23.55.4.7 template<typename RawType> bool FloatingPoint< RawType >::is_nan ( ) const [inline]

Definition at line 204 of file float_utils_google.hpp.

References FloatingPoint< RawType >::exponent_bits(), FloatingPoint< RawType >::fraction_bits(), and FloatingPoint< RawType >::kExponentBitMask.

Referenced by FloatingPoint< RawType >::AlmostEquals().

23.55.4.8 template<typename RawType> bool FloatingPoint< RawType >::AlmostEquals (const FloatingPoint< RawType > & *rhs*) const [inline]

Definition at line 216 of file float_utils_google.hpp.

References FloatingPoint< RawType >::is_nan(), and FloatingPoint< RawType >::kMaxUlps.

Referenced by OPENTREP::ScoreBoard::setScore().

23.55.5 Member Data Documentation

23.55.5.1 template<typename RawType> const size_t FloatingPoint< RawType >::kBitCount = 8*sizeof(RawType) [static]

Definition at line 132 of file float_utils_google.hpp.

23.55.5.2 template<typename RawType> const size_t FloatingPoint< RawType >::kFractionBitCount [static]

Initial value:

```
=
    std::numeric_limits<RawType>::digits - 1
```

Definition at line 135 of file float_utils_google.hpp.

23.55.5.3 template<typename RawType> const size_t FloatingPoint< RawType >::kExponentBitCount = kBitCount - 1 - kFractionBitCount [static]

Definition at line 139 of file float_utils_google.hpp.

23.55.5.4 template<typename RawType> const Bits FloatingPoint< RawType >::kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1) [static]

Definition at line 142 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::sign_bit().

23.55.5.5 template<typename RawType> const Bits FloatingPoint< RawType >::kFractionBitMask [static]

Initial value:

```
=
    ~static_cast<Bits>(0) >> (kExponentBitCount + 1)
```

Definition at line 145 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::fraction_bits().

23.55.5.6 template<typename RawType> const Bits FloatingPoint< RawType >::kExponentBitMask = ~(kSignBitMask | kFractionBitMask) [static]

Definition at line 149 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::exponent_bits(), FloatingPoint< RawType >::Infinity(), and FloatingPoint< RawType >::is_nan().

23.55.5.7 template<typename RawType> const size_t FloatingPoint< RawType >::kMaxUlps = 4 [static]

Definition at line 163 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::AlmostEquals().

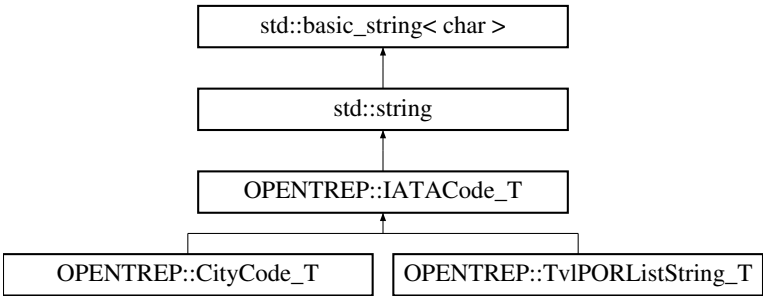
The documentation for this class was generated from the following file:

- opentrep/basic/float_utils_google.hpp

23.56 OPENTREP::IATACode_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::IATACode_T:



Public Member Functions

- IATACode_T (const std::string &iValue)

23.56.1 Detailed Description

IATA three-letter code (e.g., ORD for Chicago O'Hare, IL, USA).

Definition at line 75 of file OPENTREP_Types.hpp.

23.56.2 Constructor & Destructor Documentation

23.56.2.1 OPENTREP::IATACode_T::IATACode_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 77 of file OPENTREP_Types.hpp.

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.57 OPENTREP::IATAType Struct Reference

Enumeration of output types.

`#include <opentrep/IATAType.hpp>`

Public Types

- enum EN_IATAType {
  CTY_AIRP = 0, CTY_HPT, CTY_RSTN, CTY_BSTN,
  CTY_FERRY, CITY, AIRP, HPT,
  RSTN, BSTN, FERRY, OFF,
  LAST_VALUE }

Public Member Functions

- EN_IATAType getType () const
- char getTypeAsChar () const
- std::string getTypeAsString () const
- treppb::PlaceType getTypeAsPB () const
- const std::string describe () const
- bool operator== (const EN_IATAType &) const
- bool operator== (const IATAType &) const
- IATAType (const EN_IATAType &)
- IATAType (const char iType)
- IATAType (const std::string &iType)
- IATAType (const IATAType &)

Static Public Member Functions

- static const std::string & getLabel (const EN_IATAType &)
- static EN_IATAType getType (const char)
- static char getTypeLabel (const EN_IATAType &)
- static std::string getTypeLabelAsString (const EN_IATAType &)
- static treppb::PlaceType getTypeLabelAsPB (const EN_IATAType &)
- static std::string describeLabels ()

23.57.1 Detailed Description

Enumeration of output types.

Definition at line 22 of file IATAType.hpp.

23.57.2 Member Enumeration Documentation

23.57.2.1 enum OPENTREP::IATAType::EN_IATAType

Enumerator

CTY_AIRP

CTY_HPT

CTY_RSTN

CTY_BSTN

CTY_FERRY

CITY

AIRP

HPT

RSTN

BSTN

FERRY

OFF

LAST_VALUE

Definition at line 24 of file IATAType.hpp.

23.57.3 Constructor & Destructor Documentation

23.57.3.1 OPENTREP::IATAType::IATAType (const EN_IATAType & *ilATAType*)

Main constructor.

Definition at line 34 of file IATAType.cpp.

23.57.3.2 OPENTREP::IATAType::IATAType (const char *iType*)

Alternative constructor.

Definition at line 69 of file IATAType.cpp.

23.57.3.3 OPENTREP::IATAType::IATAType (const std::string & *iType*)

Alternative constructor.

Definition at line 74 of file IATAType.cpp.

References AIRP, BSTN, CITY, CTY_AIRP, CTY_BSTN, CTY_FERRY, CTY_HPT, CTY_RSTN, describeLabels(), FERRY, HPT, LAST_VALUE, OFF, and RSTN.

23.57.3.4 OPENTREP::IATAType::IATAType (const IATAType & *ilATAType*)

Default copy constructor.

Definition at line 29 of file IATAType.cpp.

23.57.4 Member Function Documentation

23.57.4.1 const std::string & OPENTREP::IATAType::getLabel (const EN_IATAType & *iType*) [static]

Get the label as a string (e.g., "CTY_AIRP", "CITY", "OFF").

Definition at line 113 of file IATAType.cpp.

23.57.4.2 IATAType::EN_IATAType OPENTREP::IATAType::getType (const char *iTypeChar*) [static]

Get the type value from parsing a single char (e.g., 'a', 'C', 'O')

Definition at line 39 of file IATAType.cpp.

References AIRP, BSTN, CITY, CTY_AIRP, CTY_BSTN, CTY_FERRY, CTY_HPT, CTY_RSTN, describeLabels(), FERRY, HPT, LAST_VALUE, OFF, and RSTN.

23.57.4.3 char OPENTREP::IATAType::getTypeLabel (const EN_IATAType & *iType*) [static]

Get the label as a single char (e.g., 'a', 'C', 'O').

Definition at line 118 of file IATAType.cpp.

23.57.4.4 std::string OPENTREP::IATAType::getTypeLabelAsString (const EN_IATAType & *iType*) [static]

Get the label as a string of a single char (e.g., "a", "C", "O").

Definition at line 123 of file IATAType.cpp.

23.57.4.5 treppb::PlaceType OPENTREP::IATAType::getTypeLabelAsPB (const EN_IATAType & *iType*) [static]

Get the label as a Protobuf enum (e.g., treppb::CA, treppb::C).

Definition at line 130 of file IATAType.cpp.

References AIRP, BSTN, CITY, CTY_AIRP, CTY_BSTN, CTY_FERRY, CTY_HPT, CTY_RSTN, FERRY, HPT, OFF, and RSTN.

Referenced by getTypeAsPB().

23.57.4.6 std::string OPENTREP::IATAType::describeLabels ( ) [static]

List the labels.

Definition at line 151 of file IATAType.cpp.

References LAST_VALUE.

Referenced by getType(), and IATAType().

23.57.4.7 IATAType::EN_IATAType OPENTREP::IATAType::getType ( ) const

Get the enumerated value.

Definition at line 163 of file IATAType.cpp.

23.57.4.8 char OPENTREP::IATAType::getTypeAsChar ( ) const

Get the enumerated value as a short string (e.g., 'a', 'C', 'O').

Definition at line 168 of file IATAType.cpp.

23.57.4.9 std::string OPENTREP::IATAType::getTypeAsString ( ) const

Get the enumerated value as a short string (e.g., "a", "C", "O").

Definition at line 174 of file IATAType.cpp.

Referenced by OPENTREP::LocationKey::describe(), and OPENTREP::DBManager::retrievePlace().

23.57.4.10 treppb::PlaceType OPENTREP::IATAType::getTypeAsPB ( ) const

Get the enumerated value as a Protobuf enum (e.g., treppb::CA, treppb::C).

Definition at line 181 of file IATAType.cpp.

References getTypeLabelAsPB().

Referenced by OPENTREP::LocationExchange::exportLocation().

23.57.4.11 const std::string OPENTREP::IATAType::describe ( ) const

Give a description of the structure (e.g., "CTY_AIRP", "CITY", "OFF").

Definition at line 186 of file IATAType.cpp.

23.57.4.12 bool OPENTREP::IATAType::operator== (const EN_IATAType & *iType*) const

Comparison operators.

Definition at line 193 of file IATAType.cpp.

23.57.4.13 bool OPENTREP::IATAType::operator== (const IATAType & *ilIATAType*) const

Definition at line 198 of file IATAType.cpp.

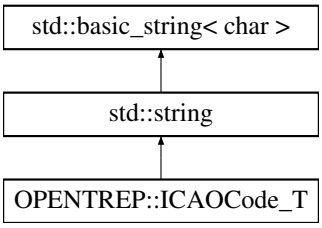
The documentation for this struct was generated from the following files:

- opentrep/IATAType.hpp
- opentrep/bom/IATAType.cpp

23.58 OPENTREP::ICAOCode_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::ICAOCode_T:



Public Member Functions

- ICAOCode_T (const std::string &iValue)

23.58.1 Detailed Description

ICAO four-letter code (e.g., KORD).

Definition at line 83 of file OPENTREP_Types.hpp.

23.58.2 Constructor & Destructor Documentation

23.58.2.1 OPENTREP::ICAOCode_T::ICAOCode_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 85 of file OPENTREP_Types.hpp.

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.59 OPENTREP::IndexBuilder Class Reference

Command wrapping the travel request process.

`#include <opentrep/command/IndexBuilder.hpp>`

Friends

- class OPENTREP_Service

23.59.1 Detailed Description

Command wrapping the travel request process.

Definition at line 24 of file IndexBuilder.hpp.

23.59.2 Friends And Related Function Documentation

23.59.2.1 friend class OPENTREP_Service [friend]

Definition at line 25 of file IndexBuilder.hpp.

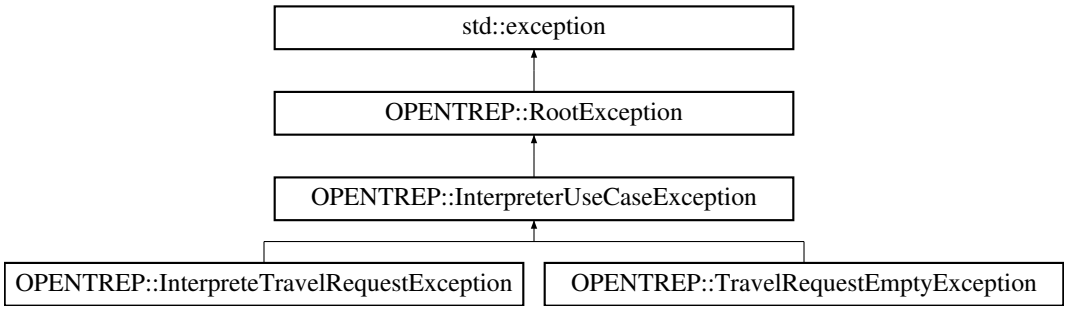
The documentation for this class was generated from the following files:

- opentrep/command/IndexBuilder.hpp
- opentrep/command/IndexBuilder.cpp

23.60 OPENTREP::InterpreterUseCaseException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::InterpreterUseCaseException:



Public Member Functions

- InterpreterUseCaseException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.60.1 Detailed Description

Interpreter issue.

Definition at line 320 of file OPENTREP_exceptions.hpp.

23.60.2 Constructor & Destructor Documentation

23.60.2.1 OPENTREP::InterpreterUseCaseException::InterpreterUseCaseException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 325 of file OPENTREP_exceptions.hpp.

23.60.3 Member Function Documentation

23.60.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.60.4 Member Data Documentation

23.60.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

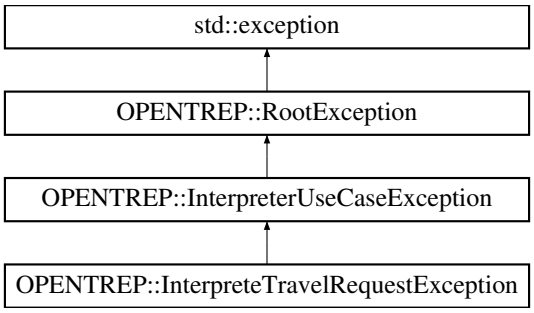
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.61 OPENTREP::InterpreteTravelRequestException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::InterpreteTravelRequestException:



Public Member Functions

- InterpreteTravelRequestException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.61.1 Detailed Description

Travel request interpreter issue.

Definition at line 332 of file OPENTREP_exceptions.hpp.

23.61.2 Constructor & Destructor Documentation

23.61.2.1 OPENTREP::InterpreteTravelRequestException::InterpreteTravelRequestException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 337 of file OPENTREP_exceptions.hpp.

23.61.3 Member Function Documentation

23.61.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.61.4 Member Data Documentation

23.61.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.62 OPENTREP::KeyAbstract Class Reference

Base class for the Business Object Model (KEY) layer.

`#include <opentrep/bom/KeyAbstract.hpp>`

Public Member Functions

- virtual void toStream (std::ostream &) const =0
- virtual void fromStream (std::istream &)=0
- virtual std::string toString () const =0
- virtual std::string describe () const =0

Protected Member Functions

- KeyAbstract ()
- KeyAbstract (const KeyAbstract &)
- virtual ~KeyAbstract ()

23.62.1 Detailed Description

Base class for the Business Object Model (KEY) layer.

Definition at line 17 of file KeyAbstract.hpp.

23.62.2 Constructor & Destructor Documentation

23.62.2.1 OPENTREP::KeyAbstract::KeyAbstract ( ) [inline], [protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 50 of file KeyAbstract.hpp.

23.62.2.2 OPENTREP::KeyAbstract::KeyAbstract (const KeyAbstract &) [inline], [protected]

Definition at line 51 of file KeyAbstract.hpp.

23.62.2.3 virtual OPENTREP::KeyAbstract::~KeyAbstract ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 56 of file KeyAbstract.hpp.

23.62.3 Member Function Documentation

23.62.3.1 virtual void OPENTREP::KeyAbstract::toStream (std::ostream &) const [pure virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

23.62.3.2 virtual void OPENTREP::KeyAbstract::fromStream (std::istream &) [pure virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Referenced by operator>>().

23.62.3.3 virtual std::string OPENTREP::KeyAbstract::toString ( ) const [pure virtual]

Get the serialised version of the Business Object.

23.62.3.4 virtual std::string OPENTREP::KeyAbstract::describe ( ) const [pure virtual]

Get a string describing the key (differentiating two objects at any level).

The documentation for this class was generated from the following file:

- opentrep/bom/KeyAbstract.hpp

23.63 OPENTREP::Language Struct Reference

`#include <opentrep/Language.hpp>`

Public Types

- enum EN_Language {
  lg_com = 0, ar, ar_DZ, ar_MA,
  ar_SA, ar_TN, de, de_DE,
  en, en_GB, en_US, es,
  es_AR, es_ES, es_MX, fa,
  fa_IR, fr, fr_BE, fr_CA,
  fr_CH, fr_FR, he, he_IL,
  ko, ko_KR, pl, pl_PL,
  pt, pt_BR, pt_PT, sv,
  sv_SE, ru, ru_RU, uk,
  uk_UA, zh, zh_CN, zh_HK,
  zh_TW, LAST_VALUE }

Static Public Member Functions

- static const std::string & getLongLabel (const EN_Language &)
- static const std::string & getShortLabel (const EN_Language &)
- static EN_Language getCode (const std::string &)

23.63.1 Detailed Description

Enumeration of languages.

On Unix, see also the /usr/share/locale/ directory. The iso-codes (RPM, DEB) package provides all the ISO-related code information.

Definition at line 18 of file Language.hpp.

23.63.2 Member Enumeration Documentation

23.63.2.1 enum OPENTREP::Language::EN_Language

The standard language form, i.e., the one without the country code, must be placed at the beginning of the list of the corresponding language variations. For instance, "en" must be placed before "en_GB" or "en_US".

Enumerator

lg_com

ar

ar_DZ

ar_MA

ar_SA

ar_TN

de

de_DE

en

en_GB

en_US

es

es_AR

es_ES

es_MX

fa

fa_IR

fr

fr_BE

fr_CA

fr_CH

fr_FR

he

he_IL

ko

ko_KR

pl

pl_PL

pt

pt_BR

pt_PT

sv

sv_SE

ru

ru_RU

uk

uk_UA

zh

zh_CN

zh_HK

zh_TW

LAST_VALUE

Definition at line 25 of file Language.hpp.

23.63.3 Member Function Documentation

23.63.3.1 const std::string & OPENTREP::Language::getLongLabel (const EN_Language & *iLanguage*) [static]

Get the long label (language and localisation) as a string.

Definition at line 37 of file Language.cpp.

23.63.3.2 const std::string & OPENTREP::Language::getShortLabel (const EN_Language & *iLanguage*) [static]

Get the short label (just the language code) as a string.

Definition at line 43 of file Language.cpp.

Referenced by getCode().

23.63.3.3 Language::EN_Language OPENTREP::Language::getCode (const std::string & *iLanguageString*) [static]

Get the language code corresponding to the label (language).

Definition at line 48 of file Language.cpp.

References getShortLabel(), LAST_VALUE, and lg_com.

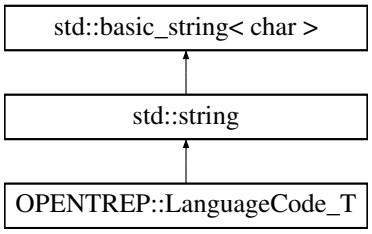
The documentation for this struct was generated from the following files:

- opentrep/Language.hpp
- opentrep/bom/Language.cpp

23.64 OPENTREP::LanguageCode_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::LanguageCode_T:



Public Member Functions

- LanguageCode_T (const std::string &iValue)

23.64.1 Detailed Description

Language code (e.g., en). Note that some language codes may contain numerics; for instance, fr_1793.

Definition at line 163 of file OPENTREP_Types.hpp.

23.64.2 Constructor & Destructor Documentation

23.64.2.1 OPENTREP::LanguageCode_T::LanguageCode_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 165 of file OPENTREP_Types.hpp.

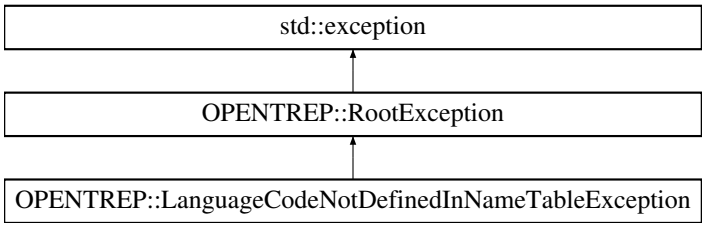
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.65 OPENTREP::LanguageCodeNotDefinedInNameTableException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::LanguageCodeNotDefinedInNameTableException:



Public Member Functions

- LanguageCodeNotDefinedInNameTableException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.65.1 Detailed Description

Language code is not recognised.

Definition at line 100 of file OPENTREP_exceptions.hpp.

23.65.2 Constructor & Destructor Documentation

23.65.2.1 OPENTREP::LanguageCodeNotDefinedInNameTableException::LanguageCodeNotDefinedInNameTableException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 105 of file OPENTREP_exceptions.hpp.

23.65.3 Member Function Documentation

23.65.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.65.4 Member Data Documentation

23.65.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

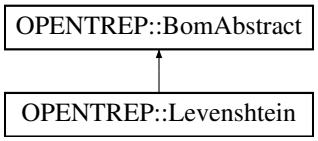
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.66 OPENTREP::Levenshtein Class Reference

`#include <opentrep/bom/Levenshtein.hpp>`

Inheritance diagram for OPENTREP::Levenshtein:



Public Member Functions

- virtual void toStream (std::ostream &) const =0
- virtual void fromStream (std::istream &)=0
- virtual std::string toString () const =0
- virtual std::string describeKey () const =0
- virtual std::string describeShortKey () const =0

Static Public Member Functions

- static int getDistance (const std::string &iSource, const std::string &iTarget)

23.66.1 Detailed Description

Class aggregating utilities around the Levenshtein edit distance/error.

Definition at line 20 of file Levenshtein.hpp.

23.66.2 Member Function Documentation

23.66.2.1 int OPENTREP::Levenshtein::getDistance (const std::string & *iSource,* const std::string & *iTarget*) [static]

Calculate the edit distance between two strings.

Definition at line 13 of file Levenshtein.cpp.

Referenced by OPENTREP::doesMatch().

23.66.2.2 virtual void OPENTREP::BomAbstract::toStream (std::ostream &) const [pure virtual], [inherited]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::World, and OPENTREP::PlaceHolder.

23.66.2.3 virtual void OPENTREP::BomAbstract::fromStream (std::istream &) [pure virtual],[inherited]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::World, and OPENTREP::PlaceHolder.

Referenced by operator>>().

23.66.2.4 virtual std::string OPENTREP::BomAbstract::toString ( ) const [pure virtual], [inherited]

Get the serialised version of the Business Object.

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::World, and OPENTREP::PlaceHolder.

23.66.2.5 virtual std::string OPENTREP::BomAbstract::describeKey ( ) const [pure virtual], [inherited]

Get a string describing the whole key (differentiating two objects at any level).

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::PlaceHolder, and OPENTREP::World.

23.66.2.6 virtual std::string OPENTREP::BomAbstract::describeShortKey ( ) const [pure virtual], [inherited]

Get a string describing the short key (differentiating two objects at the same level).

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::PlaceHolder, and OPENTREP::World.

The documentation for this class was generated from the following files:

- opentrep/bom/Levenshtein.hpp
- opentrep/bom/Levenshtein.cpp

23.67 OPENTREP::ListGenerator Struct Reference

`#include <opentrep/basic/BasConst_General.hpp>`

Static Public Member Functions

- static BlackList_T createBlackList ()

23.67.1 Detailed Description

Default "black list".

Definition at line 66 of file BasConst_General.hpp.

23.67.2 Member Function Documentation

23.67.2.1 BlackList_T OPENTREP::ListGenerator::createBlackList ( ) [static]

Definition at line 133 of file BasConst.cpp.

The documentation for this struct was generated from the following files:

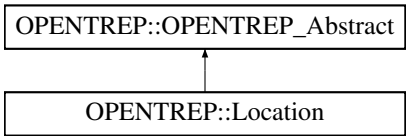
- opentrep/basic/BasConst_General.hpp
- opentrep/basic/BasConst.cpp

23.68 OPENTREP::Location Struct Reference

Structure modelling a (geographical) location.

`#include <opentrep/Location.hpp>`

Inheritance diagram for OPENTREP::Location:



Public Member Functions

- const LocationKey & getKey () const
- const IATACode_T & getIataCode () const
- const IATAType & getIataType () const
- const GeonamesID_T & getGeonamesID () const
- const IsGeonames_T & isGeonames () const
- const ICAOCode_T & getIcaoCode () const
- const FAACode_T & getFaaCode () const
- const CommonName_T & getCommonName () const
- const ASCIIName_T & getAsciiName () const
- const AltNameShortListString_T & getAltNameShortListString () const
- const TvlPORListString_T & getTvlPORListString () const
- const EnvelopeID_T & getEnvelopeID () const
- const Date_T & getDateFrom () const
- const Date_T & getDateEnd () const
- const Comment_T & getComment () const
- const CityCode_T & getCityCode () const
- const CityUTFName_T & getCityUtfName () const
- const CityASCIIName_T & getCityAsciiName () const
- const GeonamesID_T & getCityGeonamesID () const
- const StateCode_T & getStateCode () const
- const CountryCode_T & getCountryCode () const
- const AltCountryCode_T & getAltCountryCode () const
- const CountryName_T & getCountryName () const
- const ContinentCode_T & getContinentCode () const
- const ContinentName_T & getContinentName () const
- const TimeZone_T & getTimeZone () const
- const GMTOffset_T & getGMTOffset () const
- const DSTOffset_T & getDSTOffset () const
- const RawOffset_T & getRawOffset () const
- const Latitude_T & getLatitude () const
- const Longitude_T & getLongitude () const
- const FeatureClass_T & getFeatureClass () const
- const FeatureCode_T & getFeatureCode () const
- const Admin1Code_T & getAdmin1Code () const
- const Admin1UTFName_T & getAdmin1UtfName () const
- const Admin1ASCIIName_T & getAdmin1AsciiName () const
- const Admin2Code_T & getAdmin2Code () const
- const Admin2UTFName_T & getAdmin2UtfName () const
- const Admin2ASCIIName_T & getAdmin2AsciiName () const

- const Admin3Code_T & getAdmin3Code () const
- const Admin4Code_T & getAdmin4Code () const
- const Population_T & getPopulation () const
- const Elevation_T & getElevation () const
- const GTopo30_T & getGTopo30 () const
- const PageRank_T & getPageRank () const
- const Date_T & getModificationDate () const
- const WikiLink_T & getWikiLink () const
- const NameMatrix & getNameMatrix () const
- bool getNameList (const LanguageCode_T &iLanguageCode, NameList_T &ioNameList) const
- const std::string & getOriginalKeywords () const
- const std::string & getCorrectedKeywords () const
- const MatchingPercentage_T & getPercentage () const
- const NbOfErrors_T & getEditDistance () const
- const NbOfErrors_T & getAllowableEditDistance () const
- const LocationList_T & getExtraLocationList () const
- const LocationList_T & getAlternateLocationList () const
- const RawDataString_T & getRawDataString () const
- void setKey (const LocationKey &iKey)
- void setIataCode (const std::string &iIataCode)
- void setIataType (const IATAType &iIATAType)
- void setGeonamesID (const GeonamesID_T &iGeonamesID)
- void setIcaoCode (const std::string &iIcaoCode)
- void setFaaCode (const std::string &iFaaCode)
- void setCommonName (const std::string &iName)
- void setAsciiName (const std::string &iName)
- void setAltNameShortListString (const std::string &iNameListString)
- void setTvlPORListString (const std::string &iPORListString)
- void setEnvelopeID (const EnvelopeID_T &iEnvelopeID)
- void setDateFrom (const Date_T &iDate)
- void setDateEnd (const Date_T &iDate)
- void setComment (const std::string &iComment)
- void setCityCode (const std::string &iCityCode)
- void setCityUtfName (const std::string &iCityUtfName)
- void setCityAsciiName (const std::string &iCityAsciiName)
- void setCityGeonamesID (const GeonamesID_T &iGeonamesID)
- void setCountryCode (const std::string &iCountryCode)
- void setAltCountryCode (const std::string &iCountryCode)
- void setCountryName (const std::string &iCountryName)
- void setStateCode (const std::string &iStateCode)
- void setContinentCode (const std::string &iContinentCode)
- void setContinentName (const std::string &iContinentName)
- void setTimeZone (const std::string &iTimeZone)
- void setGMTOffset (const GMTOffset_T &iOffset)
- void setDSTOffset (const DSTOffset_T &iOffset)
- void setRawOffset (const RawOffset_T &iOffset)
- void setLatitude (const Latitude_T &iLatitude)
- void setLongitude (const Longitude_T &iLongitude)
- void setFeatureClass (const std::string &iFeatClass)
- void setFeatureCode (const std::string &iFeatCode)
- void setAdmin1Code (const std::string &iAdminCode)
- void setAdmin1UtfName (const std::string &iAdminName)
- void setAdmin1AsciiName (const std::string &iAdminName)
- void setAdmin2Code (const std::string &iAdminCode)
- void setAdmin2UtfName (const std::string &iAdminName)

- void setAdmin2AsciiName (const std::string &iAdminName)
- void setAdmin3Code (const std::string &iAdminCode)
- void setAdmin4Code (const std::string &iAdminCode)
- void setPopulation (const Population_T &iPopulation)
- void setElevation (const Elevation_T &iElevation)
- void setGTopo30 (const GTopo30_T &iGTopo30)
- void setPageRank (const PageRank_T &iPageRank)
- void setModificationDate (const Date_T &iModDate)
- void setWikiLink (const std::string &iWikiLink)
- void addName (const LanguageCode_T &iLanguageCode, const std::string &iName)
- void resetMatrix ()
- void setOriginalKeywords (const std::string &iOriginalKeywords)
- void setCorrectedKeywords (const std::string &iCorrectedKeywords)
- void setPercentage (const MatchingPercentage_T &iPercentage)
- void setEditDistance (const NbOfErrors_T &iEditDistance)
- void setAllowableEditDistance (const NbOfErrors_T &iAllowableEditDistance)
- void addExtraLocation (const Location &iExtraLocation)
- void addAlternateLocation (const Location &iAlternateLocation)
- void setRawDataString (const std::string &iRawDataString)
- Date_T calculateDate () const
- void consolidateAltNameShortListString ()
- void consolidateTvlPORListString ()
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- std::string toString () const
- std::string describeKey () const
- std::string describeShortKey () const
- std::string toBasicString () const
- std::string toShortString () const
- std::string toSingleLocationString () const
- Location (const IATACode_T &, const IATAType &, const GeonamesID_T &, const ICAOCode_T &, const FAACode_T &, const CommonName_T &, const ASCIIName_T &, const EnvelopeID_T &, const Date_T &iDateFrom, const Date_T &iDateEnd, const Comment_T &, const CityCode_T &, const CityUTFName_T &, const CityASCIIName_T &, const GeonamesID_T &, const StateCode_T &, const CountryCode_T &, const AltCountryCode_T &, const CountryName_T &, const ContinentName_T &, const Latitude_T &, const Longitude_T &, const FeatureClass_T &, const FeatureCode_T &, const Admin1Code_T &, const Admin1UTFName_T &, const Admin1ASCIIName_T &, const Admin2Code_T &, const Admin2UTFName_T &, const Admin2ASCIIName_T &, const Admin3Code_T &, const Admin4Code_T &, const Population_T &, const Elevation_T &, const GTopo30_T &, const TimeZone_T &, const GMTOffset_T &, const DSTOffset_T &, const RawOffset_T &, const Date_T &iModDate, const TvlPORListString_T &, const WikiLink_T &, const PageRank_T &, const std::string &iOriginalKeywords, const std::string &iCorrectedKeywords, const MatchingPercentage_T &iPercentage, const NbOfErrors_T &iEditDistance, const NbOfErrors_T &iAllowableEditDistance, const RawDataString_T &)
- Location ()
- Location (const Location &)
- ~Location ()

Static Public Member Functions

- static FeatureNameList_T getFeatureList (const FeatureCode_T &)

Public Attributes

- year_t _itYear
- month_t _itMonth
- day_t _itDay
- Language::EN_Language _itLanguageCodeNum
- LanguageCode_T _itLanguageCode
- std::string _itLangCode2Char
- std::string _itLangCodeExt
- std::string _itLangCodeHist
- AltNameShortList_T _itAltNameShortList
- TvlPORList_T _itTvlPORList

23.68.1 Detailed Description

Structure modelling a (geographical) location.

Definition at line 24 of file Location.hpp.

23.68.2 Constructor & Destructor Documentation

23.68.2.1 OPENTREP::Location::Location (const IATACode_T & *ilataCode,* const IATAType & *ilataType,* const GeonamesID_T & *iGeonamesID,* const ICAOCode_T & *ilcaoCode,* const FAACode_T & *iFaaCode,* const CommonName_T & *iCommonName,* const ASCIIName_T & *iASCIIName,* const EnvelopeID_T & *iEnvelopeID,* const Date_T & *iDateFrom,* const Date_T & *iDateEnd,* const Comment_T & *iComment,* const CityCode_T & *iCityCode,* const CityUTFName_T & *iCityUtfName,* const CityASCIIName_T & *iCityAsciiName,* const GeonamesID_T & *iCityGeonamesID,* const StateCode_T & *iStateCode,* const CountryCode_T & *iCountryCode,* const AltCountryCode_T & *iAltCountryCode,* const CountryName_T & *iCountryName,* const ContinentName_T & *iContinentName,* const Latitude_T & *iLatitude,* const Longitude_T & *iLongitude,* const FeatureClass_T & *iFeatureClass,* const FeatureCode_T & *iFeatureCode,* const Admin1Code_T & *iAdmin1Code,* const Admin1UTFName_T & *iAdmin1UtfName,* const Admin1ASCIIName_T & *iAdmin1AsciiName,* const Admin2Code_T & *iAdmin2Code,* const Admin2UTFName_T & *iAdmin2UtfName,* const Admin2ASCIIName_T & *iAdmin2AsciiName,* const Admin3Code_T & *iAdmin3Code,* const Admin4Code_T & *iAdmin4Code,* const Population_T & *iPopulation,* const Elevation_T & *iElevation,* const GTopo30_T & *iGTopo30,* const TimeZone_T & *iTimeZone,* const GMTOffset_T & *iGMTOffset,* const DSTOffset_T & *iDSTOffset,* const RawOffset_T & *iRawOffset,* const Date_T & *iModDate,* const TvlPORListString_T & *iTvlPORListString,* const WikiLink_T & *iWikiLink,* const PageRank_T & *iPageRank,* const std::string & *iOriginalKeywords,* const std::string & *iCorrectedKeywords,* const MatchingPercentage_T & *iPercentage,* const NbOfErrors_T & *iEditDistance,* const NbOfErrors_T & *iAllowableEditDistance,* const RawDataString_T & *iRawDataString*)

Main Constructor.

Definition at line 102 of file Location.cpp.

23.68.2.2 OPENTREP::Location::Location ( )

Default Constructor.

Definition at line 16 of file Location.cpp.

23.68.2.3 OPENTREP::Location::Location (const Location & *iLocation*)

Default copy constructor.

Definition at line 55 of file Location.cpp.

23.68.2.4 OPENTREP::Location::~Location ( )

Destructor.

Definition at line 184 of file Location.cpp.

23.68.3 Member Function Documentation

23.68.3.1 const LocationKey& OPENTREP::Location::getKey ( ) const [inline]

Get the primary key (IATA and ICAO codes, Geonames ID) of that location.

Definition at line 30 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getKey(), and OPENTREP::Result::getPrimaryKey().

23.68.3.2 const IATACode_T& OPENTREP::Location::getIataCode ( ) const [inline]

Get the IATA code.

Definition at line 37 of file Location.hpp.

References OPENTREP::LocationKey::getIataCode().

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::Place::getIataCode(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.3 const IATAType& OPENTREP::Location::getIataType ( ) const [inline]

Get the IATA location type (e.g., A for airport).

Definition at line 44 of file Location.hpp.

References OPENTREP::LocationKey::getIataType().

Referenced by OPENTREP::Place::getIataType().

23.68.3.4 const GeonamesID_T& OPENTREP::Location::getGeonamesID ( ) const [inline]

Get the Geonames ID.

Definition at line 51 of file Location.hpp.

References OPENTREP::LocationKey::getGeonamesID().

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::Place::getGeonamesID(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.5 const IsGeonames_T& OPENTREP::Location::isGeonames ( ) const [inline]

State whether that POR is referenced by Geonames.

Definition at line 58 of file Location.hpp.

References OPENTREP::LocationKey::isGeonames().

Referenced by OPENTREP::Place::isGeonames().

23.68.3.6 const ICAOCode_T& OPENTREP::Location::getIcaoCode ( ) const [inline]

Get the ICAO code.

Definition at line 65 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getIcaoCode(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.7 const FAACode_T& OPENTREP::Location::getFaaCode ( ) const [inline]

Get the FAA code.

Definition at line 72 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getFaaCode(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.8 const CommonName_T& OPENTREP::Location::getCommonName ( ) const [inline]

Get the common name (usually in American English, but not necessarily in ASCII).

Definition at line 80 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getCommonName().

23.68.3.9 const ASCIIName_T& OPENTREP::Location::getAsciiName ( ) const [inline]

Get the ASCII name (not necessarily in English).

Definition at line 87 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getAsciiName().

23.68.3.10 const AltNameShortListString_T& OPENTREP::Location::getAltNameShortListString ( ) const [inline]

Get the short list of alternate names (without language codes).

Definition at line 94 of file Location.hpp.

Referenced by OPENTREP::Place::getAltNameShortListString().

23.68.3.11 const TvlPORListString_T& OPENTREP::Location::getTvlPORListString ( ) const [inline]

Get the list of IATA codes of the travel-related POR.

Definition at line 101 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.12 const EnvelopeID_T& OPENTREP::Location::getEnvelopeID ( ) const [inline]

Get the envelope ID.

Definition at line 108 of file Location.hpp.

Referenced by OPENTREP::Place::getEnvelopeID(), and OPENTREP::Result::getEnvelopeID().

23.68.3.13 const Date_T& OPENTREP::Location::getDateFrom ( ) const [inline]

Get the beginning date of the validity period

Definition at line 115 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getDateFrom().

23.68.3.14 const Date_T& OPENTREP::Location::getDateEnd ( ) const [inline]

Get the end date of the validity period

Definition at line 122 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getDateEnd().

23.68.3.15 const Comment_T& OPENTREP::Location::getComment ( ) const [inline]

Get the commentaries.

Definition at line 129 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getComment().

23.68.3.16 const CityCode_T& OPENTREP::Location::getCityCode ( ) const [inline]

Get the city code.

Definition at line 136 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getCityCode(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.17 const CityUTFName_T& OPENTREP::Location::getCityUtfName ( ) const [inline]

Get the city name in UTF8.

Definition at line 143 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getCityUtfName(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.18 const CityASCIIName_T& OPENTREP::Location::getCityAsciiName ( ) const [inline]

Get the city name in ASCII (not necessarily in English).

Definition at line 150 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getCityAsciiName(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.19 const GeonamesID_T& OPENTREP::Location::getCityGeonamesID ( ) const [inline]

Get the city Geonames ID (may be 0 if unknown).

Definition at line 157 of file Location.hpp.

Referenced by OPENTREP::Place::getCityGeonamesID().

23.68.3.20 const StateCode_T& OPENTREP::Location::getStateCode ( ) const [inline]

Get the state code.

Definition at line 164 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getStateCode(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.21 const CountryCode_T& OPENTREP::Location::getCountryCode ( ) const [inline]

Get the country code.

Definition at line 171 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getCountryCode(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.22 const AltCountryCode_T& OPENTREP::Location::getAltCountryCode ( ) const [inline]

Get the alternative country code.

Definition at line 178 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getAltCountryCode().

23.68.3.23 const CountryName_T& OPENTREP::Location::getCountryName ( ) const [inline]

Get the country name.

Definition at line 185 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getCountryName(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.24 const ContinentCode_T& OPENTREP::Location::getContinentCode ( ) const [inline]

Get the continent code.

Definition at line 192 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getContinentCode().

23.68.3.25 const ContinentName_T& OPENTREP::Location::getContinentName ( ) const [inline]

Get the continent name.

Definition at line 199 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getContinentName(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.26 const TimeZone_T& OPENTREP::Location::getTimeZone ( ) const [inline]

Get the time-zone.

Definition at line 206 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getTimeZone(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.27 const GMTOffset_T& OPENTREP::Location::getGMTOffset ( ) const [inline]

Get the GMT offset (e.g., 1)

Definition at line 213 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getGMTOffset().

23.68.3.28 const DSTOffset_T& OPENTREP::Location::getDSTOffset ( ) const [inline]

Get the DST offset (e.g., 2)

Definition at line 220 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getDSTOffset().

23.68.3.29 const RawOffset_T& OPENTREP::Location::getRawOffset ( ) const [inline]

Get the raw offset (e.g., 1)

Definition at line 227 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getRawOffset().

23.68.3.30 const Latitude_T& OPENTREP::Location::getLatitude ( ) const [inline]

Get the latitude.

Definition at line 234 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getLatitude(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.31 const Longitude_T& OPENTREP::Location::getLongitude ( ) const [inline]

Get the longitude.

Definition at line 241 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getLongitude(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.32 const FeatureClass_T& OPENTREP::Location::getFeatureClass ( ) const [inline]

Get the feature class (e.g., S for spot).

Definition at line 248 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getFeatureClass().

23.68.3.33 const FeatureCode_T& OPENTREP::Location::getFeatureCode ( ) const [inline]

Get the feature code (e.g., AIRP for airport).

Definition at line 255 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getFeatureCode().

23.68.3.34 const Admin1Code_T& OPENTREP::Location::getAdmin1Code ( ) const [inline]

Get the administrative code of rank 1 (e.g., IL for Illinois).

Definition at line 262 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getAdmin1Code(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.35 const Admin1UTFName_T& OPENTREP::Location::getAdmin1UtfName ( ) const [inline]

Get the administrative UTF8 name of rank 1 (e.g., Illinois).

Definition at line 269 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getAdmin1UtfName(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.36 const Admin1ASCIIName_T& OPENTREP::Location::getAdmin1AsciiName ( ) const [inline]

Get the administrative ASCII name of rank 1 (e.g., Illinois).

Definition at line 276 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getAdmin1AsciiName(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.37 const Admin2Code_T& OPENTREP::Location::getAdmin2Code ( ) const [inline]

Get the administrative code of rank 2.

Definition at line 283 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getAdmin2Code(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.38 const Admin2UTFName_T& OPENTREP::Location::getAdmin2UtfName ( ) const [inline]

Get the administrative UTF8 name of rank 2 (e.g., Alpes-Maritimes).

Definition at line 290 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTR-

EP::Place::getAdmin2UtfName(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.39 const Admin2ASCIIName_T& OPENTREP::Location::getAdmin2AsciiName ( ) const [inline]

Get the administrative ASCII name of rank 2 (e.g., Alpes-Maritimes).

Definition at line 297 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getAdmin2AsciiName(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.40 const Admin3Code_T& OPENTREP::Location::getAdmin3Code ( ) const [inline]

Get the administrative code of rank 3.

Definition at line 304 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getAdmin3Code(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.41 const Admin4Code_T& OPENTREP::Location::getAdmin4Code ( ) const [inline]

Get the administrative code of rank 4.

Definition at line 311 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getAdmin4Code(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.42 const Population_T& OPENTREP::Location::getPopulation ( ) const [inline]

Get the population (number of inhabitants).

Definition at line 318 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getPopulation().

23.68.3.43 const Elevation_T& OPENTREP::Location::getElevation ( ) const [inline]

Get the elevation.

Definition at line 325 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getElevation().

23.68.3.44 const GTopo30_T& OPENTREP::Location::getGTopo30 ( ) const [inline]

Get the GTopo30.

Definition at line 332 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getGTopo30().

23.68.3.45 const PageRank_T& OPENTREP::Location::getPageRank ( ) const [inline]

Get the PageRank/importance.

Definition at line 339 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Result::getPageRank(), OPENTREP::Place::getPageRank(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.46 const Date_T& OPENTREP::Location::getModificationDate ( ) const [inline]

Get the modification date.

Definition at line 346 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::Place::getModificationDate().

23.68.3.47 const WikiLink_T& OPENTREP::Location::getWikiLink ( ) const [inline]

Get the Wikipedia link.

Definition at line 353 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getWikiLink(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.48 const NameMatrix& OPENTREP::Location::getNameMatrix ( ) const [inline]

Get the map of name lists.

Definition at line 360 of file Location.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getNameMatrix(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.49 bool OPENTREP::Location::getNameList (const LanguageCode_T & *iLanguageCode,* NameList_T & *ioNameList*) const [inline]

Get, for a given language (code), the corresponding list of names.

Parameters

const	LanguageCode_T& Language code.
NameList_T&	Empty list of names, which will be filled by the method if a list exists for that language code.

Returns

bool Whether or not such a list exists for the given language.

Definition at line 373 of file Location.hpp.

References OPENTREP::NameMatrix::getNameList().

Referenced by OPENTREP::Place::getNameList().

23.68.3.50 const std::string& OPENTREP::Location::getOriginalKeywords ( ) const [inline]

Get the original keywords.

Definition at line 381 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getOriginalKeywords(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.51 const std::string& OPENTREP::Location::getCorrectedKeywords ( ) const [inline]

Get the corrected keywords.

Definition at line 388 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getCorrectedKeywords(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.52 const MatchingPercentage_T& OPENTREP::Location::getPercentage ( ) const [inline]

Get the matching percentage.

Definition at line 395 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getPercentage(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.53 const NbOfErrors_T& OPENTREP::Location::getEditDistance ( ) const [inline]

Get the allowed edit distance/error.

Definition at line 402 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getEditDistance(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.54 const NbOfErrors_T& OPENTREP::Location::getAllowableEditDistance ( ) const [inline]

Get the maximal allowable edit distance/error, with which the matching has been made.

Definition at line 410 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Place::getAllowableEditDistance(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.68.3.55 const LocationList_T& OPENTREP::Location::getExtraLocationList ( ) const [inline]

Get the list of extra matching (similar) locations.

Definition at line 417 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::BomJSONExport::jsonExportLocationList().

23.68.3.56 const LocationList_T& OPENTREP::Location::getAlternateLocationList ( ) const [inline]

Get the list of alternate matching (less similar) locations.

Definition at line 424 of file Location.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), and OPENTREP::BomJSONExport::jsonExportLocationList().

23.68.3.57 const RawDataString_T& OPENTREP::Location::getRawDataString ( ) const [inline]

Get the raw data string, as stored and retrieved by Xapian.

Definition at line 431 of file Location.hpp.

Referenced by OPENTREP::Place::getRawDataString().

23.68.3.58 void OPENTREP::Location::setKey (const LocationKey & *iKey*) [inline]

Set the primary key (IATA and ICAO codes, Geonames ID) of that location.

Definition at line 441 of file Location.hpp.

Referenced by OPENTREP::Place::setKey().

23.68.3.59 void OPENTREP::Location::setIataCode (const std::string & *ilataCode*) [inline]

Set the IATA code.

Definition at line 448 of file Location.hpp.

References OPENTREP::LocationKey::setIataCode().

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), and OPENTREP::Place::setIataCode().

23.68.3.60 void OPENTREP::Location::setIataType (const IATAType & *ilATAType*) [inline]

Set the IATA location type (e.g., A for airport).

Definition at line 455 of file Location.hpp.

References OPENTREP::LocationKey::setIataType().

Referenced by OPENTREP::PorParserHelper::storePORType::operator()(), and OPENTREP::Place::setIataType().

23.68.3.61 void OPENTREP::Location::setGeonamesID (const GeonamesID_T & *iGeonamesID*) [inline]

Set the Geonames ID.

Definition at line 462 of file Location.hpp.

References OPENTREP::LocationKey::setGeonamesID().

Referenced by OPENTREP::PorParserHelper::storeGeonamesID::operator()(), and OPENTREP::Place::setGeonamesID().

23.68.3.62 void OPENTREP::Location::setIcaoCode (const std::string & *iIcaoCode*) [inline]

Set the ICAO code.

Definition at line 469 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeIcaoCode::operator()(), and OPENTREP::Place::setIcaoCode().

23.68.3.63 void OPENTREP::Location::setFaaCode (const std::string & *iFaaCode*) [inline]

Set the FAA code.

Definition at line 476 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeFaaCode::operator()(), and OPENTREP::Place::setFaaCode().

23.68.3.64 void OPENTREP::Location::setCommonName (const std::string & *iName*) [inline]

Set the common name (usually in American English, but not necessarily in ASCII).

Definition at line 484 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeCommonName::operator()(), and OPENTREP::Place::setCommonName().

23.68.3.65 void OPENTREP::Location::setAsciiName (const std::string & *iName*) [inline]

Set the ASCII name (not necessarily in English).

Definition at line 491 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAsciiName::operator()(), and OPENTREP::Place::setAsciiName().

23.68.3.66 void OPENTREP::Location::setAltNameShortListString (const std::string & *iNameListString*) [inline]

Set the short list of alternate names (without language codes).

Definition at line 498 of file Location.hpp.

Referenced by consolidateAltNameShortListString(), and OPENTREP::Place::setAltNameShortListString().

23.68.3.67 void OPENTREP::Location::setTvlPORListString (const std::string & *iPORListString*) [inline]

Set the list of IATA codes of the travel-related POR.

Definition at line 505 of file Location.hpp.

Referenced by consolidateTvlPORListString().

23.68.3.68 void OPENTREP::Location::setEnvelopeID (const EnvelopeID_T & *iEnvelopeID*) [inline]

Set the envelope ID.

Definition at line 512 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), and OPENTREP::Place::setEnvelopeID().

23.68.3.69 void OPENTREP::Location::setDateFrom (const Date_T & *iDate*) [inline]

Set the beginning date of the validity period

Definition at line 519 of file Location.hpp.

Referenced by OPENTREP::Place::setDateFrom().

23.68.3.70 void OPENTREP::Location::setDateEnd (const Date_T & *iDate*) [inline]

Set the end date of the validity period

Definition at line 526 of file Location.hpp.

Referenced by OPENTREP::Place::setDateEnd().

23.68.3.71 void OPENTREP::Location::setComment (const std::string & *iComment*) [inline]

Set the commentaries.

Definition at line 533 of file Location.hpp.

Referenced by OPENTREP::Place::setComment().

23.68.3.72 void OPENTREP::Location::setCityCode (const std::string & *iCityCode*) [inline]

Set the related/served IATA city code.

Definition at line 540 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeCityCode::operator()(), and OPENTREP::Place::setCityCode().

23.68.3.73 void OPENTREP::Location::setCityUtfName (const std::string & *iCityUtfName*) [inline]

Set the city name in UTF8.

Definition at line 547 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeCityUtfName::operator()(), and OPENTREP::Place::setCityUtfName().

23.68.3.74 void OPENTREP::Location::setCityAsciiName (const std::string & *iCityAsciiName*) [inline]

Set the city name in ASCII (not necessarily in English).

Definition at line 554 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), and OPENTREP::Place::setCityAsciiName().

23.68.3.75 void OPENTREP::Location::setCityGeonamesID (const GeonamesID_T & *iGeonamesID*) [inline]

Set the city Geonames ID (may be 0 if unknown).

Definition at line 561 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), and OPENTREP::Place::setCityGeonamesID().

23.68.3.76 void OPENTREP::Location::setCountryCode (const std::string & *iCountryCode*) [inline]

Set the country code.

Definition at line 568 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeCountryCode::operator()(), and OPENTREP::Place::set-

CountryCode().

23.68.3.77 void OPENTREP::Location::setAltCountryCode (const std::string & *iCountryCode*) [inline]

Set the alternative country code.

Definition at line 575 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), and OPENTREP::Place::setAltCountryCode().

23.68.3.78 void OPENTREP::Location::setCountryName (const std::string & *iCountryName*) [inline]

Set the country name.

Definition at line 582 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeCountryName::operator()(), and OPENTREP::Place::setCountryName().

23.68.3.79 void OPENTREP::Location::setStateCode (const std::string & *iStateCode*) [inline]

Set the state code.

Definition at line 589 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeStateCode::operator()(), and OPENTREP::Place::setStateCode().

23.68.3.80 void OPENTREP::Location::setContinentCode (const std::string & *iContinentCode*) [inline]

Set the continent code.

Definition at line 596 of file Location.hpp.

Referenced by OPENTREP::Place::setContinentCode().

23.68.3.81 void OPENTREP::Location::setContinentName (const std::string & *iContinentName*) [inline]

Set the continent name.

Definition at line 603 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeContinentName::operator()(), and OPENTREP::Place::setContinentName().

23.68.3.82 void OPENTREP::Location::setTimeZone (const std::string & *iTimeZone*) [inline]

Set the time-zone.

Definition at line 610 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeTimeZone::operator()(), and OPENTREP::Place::setTimeZone().

23.68.3.83 void OPENTREP::Location::setGMTOffset (const GMTOffset_T & *iOffset*) [inline]

Set the GMT offset (e.g., 1)

Definition at line 617 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeGMTOffset::operator()(), and OPENTREP::Place::setGMTOffset().

23.68.3.84 void OPENTREP::Location::setDSTOffset (const DSTOffset_T & *iOffset*) [inline]

Set the DST offset (e.g., 2)

Definition at line 624 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeDSTOffset::operator()(), and OPENTREP::Place::setDSTOffset().

23.68.3.85 void OPENTREP::Location::setRawOffset (const RawOffset_T & *iOffset*) [inline]

Set the raw offset (e.g., 1)

Definition at line 631 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeRawOffset::operator()(), and OPENTREP::Place::setRawOffset().

23.68.3.86 void OPENTREP::Location::setLatitude (const Latitude_T & *iLatitude*) [inline]

Set the geographical latitude.

Definition at line 638 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeLatitude::operator()(), and OPENTREP::Place::setLatitude().

23.68.3.87 void OPENTREP::Location::setLongitude (const Longitude_T & *iLongitude*) [inline]

Set the geographical longitude.

Definition at line 645 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeLongitude::operator()(), and OPENTREP::Place::setLongitude().

23.68.3.88 void OPENTREP::Location::setFeatureClass (const std::string & *iFeatClass*) [inline]

Set the feature class (e.g., S for spot).

Definition at line 652 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeFeatureClass::operator()(), and OPENTREP::Place::setFeatureClass().

23.68.3.89 void OPENTREP::Location::setFeatureCode (const std::string & *iFeatCode*) [inline]

Set the feature code (e.g., AIRP for airport).

Definition at line 659 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeFeatureCode::operator()(), and OPENTREP::Place::setFeatureCode().

23.68.3.90 void OPENTREP::Location::setAdmin1Code (const std::string & *iAdminCode*) [inline]

Set the administrative code of rank 1 (e.g., IL for Illinois).

Definition at line 666 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAdm1Code::operator()(), and OPENTREP::Place::setAdmin1Code().

23.68.3.91 void OPENTREP::Location::setAdmin1UtfName (const std::string & *iAdminName*) [inline]

Set the administrative UTF8 name of rank 1 (e.g., Illinois).

Definition at line 673 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), and OPENTREP::Place::setAdmin1UtfName().

23.68.3.92 void OPENTREP::Location::setAdmin1AsciiName (const std::string & *iAdminName*) [inline]

Set the administrative ASCII name of rank 1 (e.g., Illinois).

Definition at line 680 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), and OPENTREP::Place::setAdmin1AsciiName().

23.68.3.93 void OPENTREP::Location::setAdmin2Code (const std::string & *iAdminCode*) [inline]

Set the administrative code of rank 2.

Definition at line 687 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAdm2Code::operator()(), and OPENTREP::Place::setAdmin2Code().

23.68.3.94 void OPENTREP::Location::setAdmin2UtfName (const std::string & *iAdminName*) [inline]

Set the administrative UTF8 name of rank 2 (e.g., Alpes-Maritimes).

Definition at line 694 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), and OPENTREP::Place::setAdmin2UtfName().

23.68.3.95 void OPENTREP::Location::setAdmin2AsciiName (const std::string & *iAdminName*) [inline]

Set the administrative ASCII name of rank 2 (e.g., Alpes-Maritimes).

Definition at line 701 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), and OPENTREP::Place::setAdmin2AsciiName().

23.68.3.96 void OPENTREP::Location::setAdmin3Code (const std::string & *iAdminCode*) [inline]

Set the administrative code of rank 3.

Definition at line 708 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAdm3Code::operator()(), and OPENTREP::Place::setAdmin3Code().

23.68.3.97 void OPENTREP::Location::setAdmin4Code (const std::string & *iAdminCode*) [inline]

Set the administrative code of rank 4.

Definition at line 715 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAdm4Code::operator()(), and OPENTREP::Place::setAdmin4Code().

23.68.3.98 void OPENTREP::Location::setPopulation (const Population_T & *iPopulation*) [inline]

Set the population (number of inhabitants).

Definition at line 722 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storePopulation::operator()(), and OPENTREP::Place::setPopulation().

23.68.3.99 void OPENTREP::Location::setElevation (const Elevation_T & *iElevation*) [inline]

Set the elevation.

Definition at line 729 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeElevation::operator()(), and OPENTREP::Place::setElevation().

23.68.3.100 void OPENTREP::Location::setGTopo30 (const GTopo30_T & *iGTopo30*) [inline]

Set the GTopo30.

Definition at line 736 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeGTopo30::operator()(), and OPENTREP::Place::setGTopo30().

23.68.3.101 void OPENTREP::Location::setPageRank (const PageRank_T & *iPageRank*) [inline]

Set the PageRank.

Definition at line 743 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storePageRank::operator()(), and OPENTREP::Place::setPageRank().

23.68.3.102 void OPENTREP::Location::setModificationDate (const Date_T & *iModDate*) [inline]

Set the modification date.

Definition at line 750 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeModDate::operator()(), and OPENTREP::Place::setModificationDate().

23.68.3.103 void OPENTREP::Location::setWikiLink (const std::string & *iWikiLink*) [inline]

Set the Wikipedia link.

Definition at line 757 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeWikiLink::operator()(), and OPENTREP::Place::setWikiLink().

23.68.3.104 void OPENTREP::Location::addName (const LanguageCode_T & *iLanguageCode,* const std::string & *iName*) [inline]

Add a name for that location.

Parameters

const	LanguageCode_T& Language in which to add the name.
const	std::string& Name to be added.

Definition at line 767 of file Location.hpp.

References OPENTREP::NameMatrix::addName().

Referenced by OPENTREP::Place::addName(), and OPENTREP::PorParserHelper::storeAltName::operator()().

23.68.3.105 void OPENTREP::Location::resetMatrix ( ) [inline]

Reset the map of name lists.

Definition at line 775 of file Location.hpp.

References OPENTREP::NameMatrix::reset().

Referenced by OPENTREP::Place::resetMatrix().

23.68.3.106 void OPENTREP::Location::setOriginalKeywords (const std::string & *iOriginalKeywords*) [inline]

Set the original keywords.

Definition at line 782 of file Location.hpp.

Referenced by OPENTREP::Place::setOriginalKeywords().

23.68.3.107 void OPENTREP::Location::setCorrectedKeywords (const std::string & *iCorrectedKeywords*) [inline]

Set the corrected keywords.

Definition at line 789 of file Location.hpp.

Referenced by OPENTREP::Place::setCorrectedKeywords().

23.68.3.108 void OPENTREP::Location::setPercentage (const MatchingPercentage_T & *iPercentage*) [inline]

Set the Xapian matching percentage.

Definition at line 796 of file Location.hpp.

Referenced by OPENTREP::Place::setPercentage().

23.68.3.109 void OPENTREP::Location::setEditDistance (const NbOfErrors_T & *iEditDistance*) [inline]

Set the allowed edit distance/error.

Definition at line 803 of file Location.hpp.

Referenced by OPENTREP::Place::setEditDistance().

23.68.3.110 void OPENTREP::Location::setAllowableEditDistance (const NbOfErrors_T & *iAllowableEditDistance*) [inline]

Set the maxiaml allowable edit distance/error, with which the matching has been made.

Definition at line 811 of file Location.hpp.

Referenced by OPENTREP::Place::setAllowableEditDistance().

23.68.3.111 void OPENTREP::Location::addExtraLocation (const Location & *iExtraLocation*) [inline]

Add an extra matching location.

Definition at line 818 of file Location.hpp.

Referenced by OPENTREP::Place::completeLocation().

23.68.3.112 void OPENTREP::Location::addAlternateLocation (const Location & *iAlternateLocation*) [inline]

Add an alternate matching location.

Definition at line 825 of file Location.hpp.

Referenced by OPENTREP::Place::completeLocation().

23.68.3.113 void OPENTREP::Location::setRawDataString (const std::string & *iRawDataString*) [inline]

Set the raw data string, as stored and retrieved by Xapian.

Definition at line 832 of file Location.hpp.

Referenced by OPENTREP::Place::setRawDataString().

23.68.3.114 Date_T OPENTREP::Location::calculateDate ( ) const

Calculate the date from the staging details.

Definition at line 307 of file Location.cpp.

References _itDay, _itMonth, _itYear, OPENTREP::date_time_element< MIN, MAX >::_value, and OPENTREP::date_time_element< MIN, MAX >::check().

Referenced by OPENTREP::PorParserHelper::storeModDate::operator()().

23.68.3.115 void OPENTREP::Location::consolidateAltNameShortListString ( )

Aggregate the temporary alternate names into the short list of alternate names.

Definition at line 313 of file Location.cpp.

References _itAltNameShortList, and setAltNameShortListString().

Referenced by OPENTREP::PorParserHelper::storeAltNameShortListString::operator()().

23.68.3.116 void OPENTREP::Location::consolidateTvlPORListString ( )

Aggregate the temporary IATA codes into the list of travel-related POR.

Definition at line 331 of file Location.cpp.

References _itTvlPORList, and setTvlPORListString().

Referenced by OPENTREP::PorParserHelper::storeTvlPORListString::operator()().

23.68.3.117 FeatureNameList_T OPENTREP::Location::getFeatureList (const FeatureCode_T & *iFeatureCode*) [static]

Derive a list from a (Geonames) feature code. For instance, the 'AIRP' feature code provides:

- airport
- airdrome
- aerodrome
- airfield
- airstrip
- airbase

Parameters

const	FeatureCode_T& The feature code.

Returns

const FeatureNameList_T The list of feature names.

Definition at line 349 of file Location.cpp.

Referenced by OPENTREP::Place::addNameToXapianSets().

23.68.3.118 void OPENTREP::Location::toStream (std::ostream & *ioOut*) const [virtual]

Dump a structure into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 188 of file Location.cpp.

References toString().

23.68.3.119 void OPENTREP::Location::fromStream (std::istream &) [virtual]

Read a structure from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 193 of file Location.cpp.

23.68.3.120 std::string OPENTREP::Location::toString ( ) const [virtual]

Display of the Location structure with its associated names.

The alternate and extra matches are also fully displayed.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 269 of file Location.cpp.

References OPENTREP::NameMatrix::describe(), toShortString(), and toSingleLocationString().

Referenced by toStream(), and OPENTREP::Place::toString().

23.68.3.121 std::string OPENTREP::Location::describeKey ( ) const

Get a string describing the whole key (IATA and ICAO codes, Geonames ID).

Definition at line 204 of file Location.cpp.

References describeShortKey().

Referenced by OPENTREP::Place::describeKey().

23.68.3.122 std::string OPENTREP::Location::describeShortKey ( ) const

Get a string describing the whole key (IATA and ICAO codes, Geonames ID).

Definition at line 197 of file Location.cpp.

References OPENTREP::LocationKey::describe().

Referenced by describeKey(), OPENTREP::Place::describeShortKey(), and toBasicString().

23.68.3.123 std::string OPENTREP::Location::toBasicString ( ) const

Basic display of the Location structure.

The names are not displayed. No information is displayed about alternate and extra matches.

Definition at line 209 of file Location.cpp.

References describeShortKey().

Referenced by OPENTREP::Place::shortDisplay(), toShortString(), and toSingleLocationString().

23.68.3.124 std::string OPENTREP::Location::toShortString ( ) const

Short display of the Location structure.

The names are not displayed. Basic information is displayed about alternate and extra matches.

Definition at line 245 of file Location.cpp.

References toBasicString().

Referenced by OPENTREP::Place::toShortString(), and toString().

23.68.3.125 std::string OPENTREP::Location::toSingleLocationString ( ) const

Display of the Location structure with its associated names.

Contrary to the toString() method, the alternate and extra matches are not displayed.

Definition at line 262 of file Location.cpp.

References toBasicString().

Referenced by toString().

23.68.4 Member Data Documentation

23.68.4.1 year_t OPENTREP::Location::_itYear

Staging Date.

Definition at line 1275 of file Location.hpp.

Referenced by calculateDate(), and OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.68.4.2 month_t OPENTREP::Location::_itMonth

Definition at line 1276 of file Location.hpp.

Referenced by calculateDate(), and OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.68.4.3 day_t OPENTREP::Location::_itDay

Definition at line 1277 of file Location.hpp.

Referenced by calculateDate(), and OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.68.4.4 Language::EN_Language OPENTREP::Location::_itLanguageCodeNum

Staging alternate names.

Definition at line 1282 of file Location.hpp.

23.68.4.5 LanguageCode_T OPENTREP::Location::_itLanguageCode

Definition at line 1283 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

23.68.4.6 std::string OPENTREP::Location::_itLangCode2Char

Definition at line 1284 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), and OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()().

23.68.4.7 std::string OPENTREP::Location::_itLangCodeExt

Definition at line 1285 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), and OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()().

23.68.4.8 std::string OPENTREP::Location::_itLangCodeHist

Definition at line 1286 of file Location.hpp.

Referenced by OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), and OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()().

23.68.4.9 AltNameShortList_T OPENTREP::Location::_itAltNameShortList

Definition at line 1287 of file Location.hpp.

Referenced by consolidateAltNameShortListString(), and OPENTREP::PorParserHelper::storeAltNameShort-::operator()().

23.68.4.10 TvlPORList_T OPENTREP::Location::_itTvlPORList

Definition at line 1288 of file Location.hpp.

Referenced by consolidateTvlPORListString(), and OPENTREP::PorParserHelper::storeTvlPORCode::operator()().

The documentation for this struct was generated from the following files:

- opentrep/Location.hpp
- opentrep/basic/Location.cpp

23.69 OPENTREP::LocationExchange Class Reference

Utility class to export Opentrep structures in a Protobuf format.

`#include <opentrep/bom/LocationExchange.hpp>`

Static Public Member Functions

- static void exportLocationList (std::ostream &, const LocationList_T &, const WordList_T &iNonMatchedWord-List)
- static void exportLocation (treppb::Place &, const Location &)

23.69.1 Detailed Description

Utility class to export Opentrep structures in a Protobuf format.

Definition at line 26 of file LocationExchange.hpp.

23.69.2 Member Function Documentation

23.69.2.1 void OPENTREP::LocationExchange::exportLocationList (std::ostream & *oStream*, const LocationList_T & *iLocationList*, const WordList_T & *iNonMatchedWordList*) [static]

Export (dump in the underlying output log stream and in Protobuf format) a list of Location objects.

Parameters

std::ostream&	Output stream in which the Location objects should be logged/dumped.
const	LocationList_T& List of Location objects to be exported.
const	WordList_T& The list of non-matching keywords.

Definition at line 19 of file LocationExchange.cpp.

References exportLocation().

23.69.2.2 void OPENTREP::LocationExchange::exportLocation (treppb::Place & *ioPlace*, const Location & *iLocation*) [static]

Export (dump in the underlying output log stream and in Protobuf format) a Location object.

Parameters

treppb::Place&	Protobuf holder in which the Location structure should be logged/dumped.
const	Location& Location object to be exported.

Definition at line 73 of file LocationExchange.cpp.

References OPENTREP::Location::getAdmin1AsciiName(), OPENTREP::Location::getAdmin1Code(), OPENTREP::Location::getAdmin1UtfName(), OPENTREP::Location::getAdmin2AsciiName(), OPENTREP::Location::getAdmin2Code(), OPENTREP::Location::getAdmin2UtfName(), OPENTREP::Location::getAdmin3Code(), OPENTREP::Location::getAdmin4Code(), OPENTREP::Location::getAllowableEditDistance(), OPENTREP::Location::getAltCountryCode(), OPENTREP::Location::getAlternateLocationList(), OPENTREP::Location::getAsciiName(), OPENTREP::Location::getCityAsciiName(), OPENTREP::Location::getCityCode(), OPENTREP::Location::getCityUtfName(), OPENTREP::Location::getComment(), OPENTREP::Location::getCommonName(), OPENTREP::Location::getContinentCode(), OPENTREP::Location::getContinentName(), OPENTREP::Location::getCorrectedKeywords(), OPENTREP::Location::getCountryCode(), OPENTREP::Location::getCountryName(), OPENTREP::Location::getDateEnd(), OPENTREP::Location::getDateFrom(), OPENTREP::Location::getDSTOffset(), OPENTREP::Location::getEditDistance(), OPENTREP::Location::getElevation(), OPENTREP::Location::getExtraLocationList(), OPENTREP::Location::getFaaCode(), OPENTREP::Location::getFeatureClass(), OPENTREP::Location::getFeatureCode(), OPENTREP::LocationKey::getGeonamesID(), OPENTREP::Location::getGMTOffset(), OPENTREP::Location::getGTopo30(), OPENTREP::LocationKey::getIataCode(), OPENTREP::LocationKey::getIataType(), OPENTREP::Location::getIcaoCode(), OPENTREP::Location::getKey(), OPENTREP::Names::getLanguageCode(), OPENTREP::Location::getLatitude(), OPENTREP::Location::getLongitude(), OPENTREP::Location::getModificationDate(), OPENTREP::Names::getNameList(), OPENTREP::NameMatrix::getNameMatrix(), OPENTREP::Location::getNameMatrix(), OPENTREP::Location::getOriginalKeywords(), OPENTREP::Location::getPageRank(), OPENTREP::Location::getPercentage(), OPENTREP::Location::getPopulation(), OPENTREP::Location::getRawOffset(), OPENTREP::Location::getStateCode(), OPENTREP::Location::getTimeZone(), OPENTREP::Location::getTvlPORListString(), OPENTREP::IATAType::getTypeAsPB(), and OPENTREP::Location::getWikiLink().

Referenced by exportLocationList().

The documentation for this class was generated from the following files:

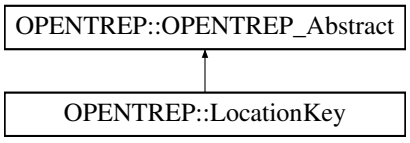
- opentrep/bom/LocationExchange.hpp
- opentrep/bom/LocationExchange.cpp

23.70 OPENTREP::LocationKey Struct Reference

Class modelling the primary key of a location/POR (point of reference).

`#include <opentrep/LocationKey.hpp>`

Inheritance diagram for OPENTREP::LocationKey:



Public Member Functions

- const IATACode_T & getIataCode () const
- const IATAType & getIataType () const
- const GeonamesID_T & getGeonamesID () const
- const IsGeonames_T & isGeonames () const
- void setIataCode (const std::string &iIataCode)
- void setIataType (const IATAType &iIATAType)
- void setGeonamesID (const GeonamesID_T &iGeonamesID)

- void setIsGeonames (const IsGeonames_T &isGeonames)
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- std::string toString () const
- std::string describe () const
- LocationKey (const IATACode_T &, const IATAType &, const GeonamesID_T &)
- LocationKey (const LocationKey &)
- ~LocationKey ()
- bool operator== (const LocationKey &) const
- bool operator!= (const LocationKey &iLocationKey) const

23.70.1 Detailed Description

Class modelling the primary key of a location/POR (point of reference).

Definition at line 21 of file LocationKey.hpp.

23.70.2 Constructor & Destructor Documentation

23.70.2.1 OPENTREP::LocationKey::LocationKey (const IATACode_T & *ilataCode,* const IATAType & *ilataType,* const GeonamesID_T & *iGeonamesID*)

Main constructor.

Definition at line 13 of file LocationKey.cpp.

23.70.2.2 OPENTREP::LocationKey::LocationKey (const LocationKey & *iLocationKey*)

Default copy constructor.

Definition at line 32 of file LocationKey.cpp.

23.70.2.3 OPENTREP::LocationKey::~LocationKey ( )

Destructor.

Definition at line 39 of file LocationKey.cpp.

23.70.3 Member Function Documentation

23.70.3.1 const IATACode_T& OPENTREP::LocationKey::getIataCode ( ) const [inline]

Get the IATA code.

Definition at line 27 of file LocationKey.hpp.

Referenced by OPENTREP::Result::calculateCodeMatches(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::Location::getIataCode(), and OPENTREP::DBManager::retrievePlace().

23.70.3.2 const IATAType& OPENTREP::LocationKey::getIataType ( ) const [inline]

Get the IATA location type (e.g., A for airport).

Definition at line 34 of file LocationKey.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Location::getIataType(), and OPENTREP::DBManager::retrievePlace().

23.70.3.3 const GeonamesID_T& OPENTREP::LocationKey::getGeonamesID ( ) const [inline]

Get the Geonames ID.

Definition at line 41 of file LocationKey.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation(), OPENTREP::Location::getGeonamesID(), and OPENTREP::DBManager::retrievePlace().

23.70.3.4 const IsGeonames_T& OPENTREP::LocationKey::isGeonames ( ) const [inline]

State whether that POR is referenced by Geonames.

Definition at line 48 of file LocationKey.hpp.

Referenced by OPENTREP::Location::isGeonames(), and setIsGeonames().

23.70.3.5 void OPENTREP::LocationKey::setIataCode (const std::string & *ilataCode*) [inline]

Set the IATA code.

Definition at line 58 of file LocationKey.hpp.

Referenced by OPENTREP::Location::setIataCode().

23.70.3.6 void OPENTREP::LocationKey::setIataType (const IATAType & *ilATAType*) [inline]

Set the IATA location type (e.g., A for airport).

Definition at line 65 of file LocationKey.hpp.

Referenced by OPENTREP::Location::setIataType().

23.70.3.7 void OPENTREP::LocationKey::setGeonamesID (const GeonamesID_T & *iGeonamesID*) [inline]

Set the Geonames ID.

Definition at line 72 of file LocationKey.hpp.

Referenced by OPENTREP::Location::setGeonamesID().

23.70.3.8 void OPENTREP::LocationKey::setIsGeonames (const IsGeonames_T & *isGeonames*) [inline]

Set the Geonames flag.

Definition at line 79 of file LocationKey.hpp.

References isGeonames().

23.70.3.9 void OPENTREP::LocationKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump the structure into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 67 of file LocationKey.cpp.

References toString().

23.70.3.10 void OPENTREP::LocationKey::fromStream (std::istream & *ioIn*) [virtual]

Read a structure from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 72 of file LocationKey.cpp.

23.70.3.11 std::string OPENTREP::LocationKey::toString ( ) const [virtual]

Get the serialised version of the structure.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 60 of file LocationKey.cpp.

References describe().

Referenced by toStream().

23.70.3.12 std::string OPENTREP::LocationKey::describe ( ) const

Get a string describing the whole key (differentiating two objects at any level).

Definition at line 51 of file LocationKey.cpp.

References OPENTREP::IATAType::getTypeAsString().

Referenced by OPENTREP::Location::describeShortKey(), and toString().

23.70.3.13 bool OPENTREP::LocationKey::operator== (const LocationKey & *iLocationKey*) const

Comparison operator.

Definition at line 43 of file LocationKey.cpp.

23.70.3.14 bool OPENTREP::LocationKey::operator!= (const LocationKey & *iLocationKey*) const [inline]

Comparison operator.

Definition at line 137 of file LocationKey.hpp.

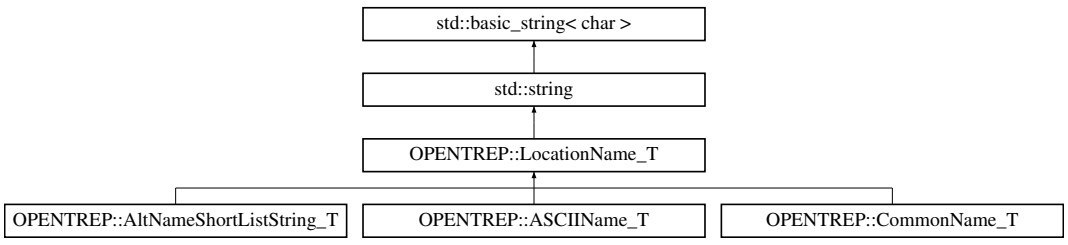
The documentation for this struct was generated from the following files:

- opentrep/LocationKey.hpp
- opentrep/basic/LocationKey.cpp

23.71 OPENTREP::LocationName_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::LocationName_T:



Public Member Functions

- LocationName_T (const std::string &iValue)

23.71.1 Detailed Description

Location name (e.g., Nice Côte d'Azur).

Definition at line 109 of file OPENTREP_Types.hpp.

23.71.2 Constructor & Destructor Documentation

23.71.2.1 OPENTREP::LocationName_T::LocationName_T (const std::string & *iValue*) [inline], [explicit]

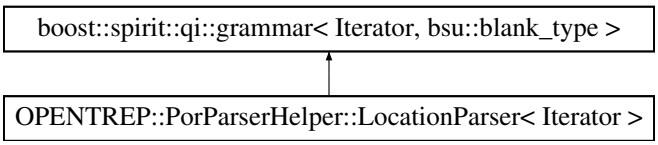
Definition at line 111 of file OPENTREP_Types.hpp.

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.72 OPENTREP::PorParserHelper::LocationParser< Iterator > Struct Template Reference

Inheritance diagram for OPENTREP::PorParserHelper::LocationParser< Iterator >:



Public Member Functions

- LocationParser (Location &ioPORRule)

Public Attributes

- bsq::rule< Iterator, bsu::blank_type > start
- bsq::rule< Iterator, bsu::blank_type > header
- bsq::rule< Iterator, bsu::blank_type > por_rule
- bsq::rule< Iterator, bsu::blank_type > por_rule_end
- bsq::rule< Iterator, bsu::blank_type > por_key
- bsq::rule< Iterator, bsu::blank_type > por_details
- bsq::rule< Iterator, bsu::blank_type > iata_code
- bsq::rule< Iterator, bsu::blank_type > icao_code
- bsq::rule< Iterator, bsu::blank_type > faa_code
- bsq::rule< Iterator, bsu::blank_type > geoname_id
- bsq::rule< Iterator, bsu::blank_type > envelope_id

- bsq::rule< Iterator, bsu::blank_type > is_geonames
- bsq::rule< Iterator, bsu::blank_type > common_name
- bsq::rule< Iterator, bsu::blank_type > ascii_name
- bsq::rule< Iterator, bsu::blank_type > alt_name_short_list
- bsq::rule< Iterator, bsu::blank_type > alt_name_short
- bsq::rule< Iterator, bsu::blank_type > alt_name_sep
- bsq::rule< Iterator, bsu::blank_type > tvl_por_code_list
- bsq::rule< Iterator, bsu::blank_type > tvl_por_code
- bsq::rule< Iterator, bsu::blank_type > tvl_por_sep
- bsq::rule< Iterator, bsu::blank_type > latitude
- bsq::rule< Iterator, bsu::blank_type > longitude
- bsq::rule< Iterator, bsu::blank_type > feat_class
- bsq::rule< Iterator, bsu::blank_type > feat_code
- bsq::rule< Iterator, bsu::blank_type > page_rank
- bsq::rule< Iterator, bsu::blank_type > date_from
- bsq::rule< Iterator, bsu::blank_type > date_until
- bsq::rule< Iterator, bsu::blank_type > comments
- bsq::rule< Iterator, bsu::blank_type > country_code
- bsq::rule< Iterator, bsu::blank_type > country_code2
- bsq::rule< Iterator, bsu::blank_type > country_name
- bsq::rule< Iterator, bsu::blank_type > continent_name
- bsq::rule< Iterator, bsu::blank_type > adm1_code
- bsq::rule< Iterator, bsu::blank_type > adm1_name_utf
- bsq::rule< Iterator, bsu::blank_type > adm1_name_ascii
- bsq::rule< Iterator, bsu::blank_type > adm2_code
- bsq::rule< Iterator, bsu::blank_type > adm2_name_utf
- bsq::rule< Iterator, bsu::blank_type > adm2_name_ascii
- bsq::rule< Iterator, bsu::blank_type > adm3_code

- bsq::rule< Iterator, bsu::blank_type > adm4_code
- bsq::rule< Iterator, bsu::blank_type > population
- bsq::rule< Iterator, bsu::blank_type > elevation
- bsq::rule< Iterator, bsu::blank_type > gtopo30
- bsq::rule< Iterator, bsu::blank_type > time_zone
- bsq::rule< Iterator, bsu::blank_type > gmt_offset
- bsq::rule< Iterator, bsu::blank_type > dst_offset
- bsq::rule< Iterator, bsu::blank_type > raw_offset
- bsq::rule< Iterator, bsu::blank_type > mod_date
- bsq::rule< Iterator, bsu::blank_type > date
- bsq::rule< Iterator, bsu::blank_type > city_code_list
- bsq::rule< Iterator, bsu::blank_type > city_code
- bsq::rule< Iterator, bsu::blank_type > city_name_list
- bsq::rule< Iterator, bsu::blank_type > city_name_utf
- bsq::rule< Iterator, bsu::blank_type > city_name_ascii
- bsq::rule< Iterator, bsu::blank_type > city_detail_list
- bsq::rule< Iterator, bsu::blank_type > city_details
- bsq::rule< Iterator, bsu::blank_type > city_geoname_id
- bsq::rule< Iterator, bsu::blank_type > state_code
- bsq::rule< Iterator, bsu::blank_type > por_type
- bsq::rule< Iterator, bsu::blank_type > wiki_link
- bsq::rule< Iterator, bsu::blank_type > alt_name_section
- bsq::rule< Iterator, bsu::blank_type > alt_name_details
- bsq::rule< Iterator, bsu::blank_type > alt_lang_code
- bsq::rule< Iterator, bsu::blank_type > alt_lang_code_ftd
- bsq::rule< Iterator, bsu::blank_type > alt_name
- bsq::rule< Iterator, bsu::blank_type > alt_name_qualifiers
- bsq::rule< Iterator, bsu::blank_type > lang_code_opt

- bsq::rule< Iterator, bsu::blank_type > lang_code_2char
- bsq::rule< Iterator, bsu::blank_type > lang_code_ext
- bsq::rule< Iterator, bsu::blank_type > lang_code_hist
- bsq::rule< Iterator, bsu::blank_type > destination
- Location & _location

23.72.1 Detailed Description

template<typename Iterator>struct OPENTREP::PorParserHelper::LocationParser< Iterator >

– Geonames-related part:

– iata_code : IATA code; varchar(3). See also: – http://www.iata.org/ps/publications/-Pages/code-search.aspx – icao_code : ICAO code; varchar(4) – faa_code : FAA code; varchar(4) – geoname_id : Integer ID of record in geonames database – name : Name of geographical point; – (UTF8) varchar(200) – asciiname : Name of geographical point in plain ASCII – characters; – (ASCII) varchar(200) – alternatenames : Alternate names, comma separated – varchar(5000) – latitude : Latitude in decimal degrees (WGS84) – longitude : Longitude in decimal degrees (WGS84) – feature class : See http://www.-geonames.org/export/codes.html – char(1) – feature code : See http://www.geonames.-org/export/codes.html – varchar(10) – country code : ISO-3166 2-letter country code, 2 characters – cc2 : Alternate country codes, comma separated, – ISO-3166 2-letter country code, 60 characters – country name : Name of the country, as found in the Geonames – countryInfo.txt data file – varchar(80) – continent name : The name of the related continent, as found in – the Geonames continentCodes.txt data file; – see also below the 'Continents' section – varchar(80) – admin1 code : FIPS code (subject to change to ISO code), – see exceptions below. See file admin1Codes.txt – for display names of this code; – varchar(20) – admin1 UTF8 name : UTF8 name for the first administrative division – admin1 ASCII name : ASCII name for the first administrative division – admin2 code : Code for the second administrative division, – a county in the US. See file admin2Codes.txt; – varchar(80) – admin2 UTF8 name : UTF8 name for the second administrative division – admin2 ASCII name : ASCII name for the second administrative division – admin3 code : Code for third level administrative division; – varchar(20) – admin4 code : Code for fourth level administrative division; – varchar(20) – population : bigint (8 byte int) – elevation : In meters, integer – dem : Digital elevation model, srtm3 or gtopo30, average – elevation of 3"x3" (ca 90mx90m) or 30"x30" – (ca 900mx900m) area in meters, integer. – srtm processed by cgiar/ciat. – timezone : The time-zone ID (see file timeZone.txt) – gmt offset : GMT offset on 1st of January – dst offset : DST offset to GMT on 1st of July (of the – current year) – raw offset : Raw Offset without DST

– modification date : Date of last modification in yyyy-MM-dd format

– – ORI-related part:

– ----------

– is_geonames : Whether that POR is known by Geonames; varchar(1) – varchar(1) – envelope_id : ID of the envelope. Empty when valid, i.e., for the – current envelope/state. Incrementing ID when no – longer valid; the validity date range then allows – to derive when that POR was valid. – city_code_list : The (list of) IATA code(s) of the related (list of) – cities. – varchar(100) – city_name_list : UTF-8 names of the served (list of) cities – varchar(500) – city_detail_list : Details (Geonames ID, names) of the served – (list of) cities – varchar(500) – tvl_por_list : list of the IATA codes of the travel-related POR – serving the city. – varchar(100) – state code : The ISO code of the related state; varchar(3) – location type : A/APT airport; B/BUS bus/coach station; – C/CITY City; – G/GRD ground transport (this code is used for SK in – Sweden only); H/HELI Heliport; – O/OFF-PT off-line point, i.e. a city without – an airport; – R/RAIL railway Station; – S/ASSOC a location without its own IATA code,

– but attached to an IATA location.

– Continents:

– AF Africa 6255146 – AS Asia 6255147 – EU Europe 6255148 – NA North America 6255149 – OC Oceania 6255151 – SA South America 6255150

– AN Antarctica 6255152

– Samples: – CDG^LFPG^^Y^6269554^^Paris - Charles-de-Gaulle^Paris - Charles-de-Gaulle^49.012779^2.55^-S^AIRP^0.650357283214^^^^FR^^France^Europe^A8^Île-de-France^Ile-de-France^95^Département du Val-d'Oise^Departement du Val-d'Oise^^^0^119^106^Europe/Paris^1.0^2.0^1.0^2008-07-09^PAR^Paris^-Paris^^^A^http://en.wikipedia.org/wiki/Paris-Charles_de_Gaulle_Airport^es|París - Charles de Gaulle|p=|-Roissy Charles de Gaulle| – PAR^ZZZZ^^Y^2988507^^Paris^Paris^48.85341^2.3488^P^PPLC^1.0^^^^-FR^^France^Europe^A8^Île-de-France^Ile-de-France^75^Paris^Paris^751^75056^2138551^^42^Europe/-Paris^1.0^2.0^1.0^2012-08-19^PAR^Paris^Paris^BVA,CDG,JDP,JPU,LBG,ORY,POX,TNF,VIY,XCR,XEX,XG-B,XHP,XJY,XPG,XTT^^C^http://en.wikipedia.org/wiki/Paris^la|Lutetia Parisorum|=fr|Lutece|h=fr|Ville-Lumière|c=eo|-Parizo|=es|París|ps=de|Paris|=en|Paris|p=af|Parys|=als|Paris|=fr|Paris|p

– HDQ^ZZZZ^^N^0^^Headquarters ZZ^Headquarters ZZ^^^X^XXXX^^^^^ZZ^^Not relevant/available^Not relevant/available^^^^^^^^^^^^^^^1^HDQ^Headquarters ZZ^Headquarters ZZ^^^O^^

iata_code varchar(3) icao_code varchar(4) faa_code varchar(4) is_geonames varchar(1) geoname_id int(11) envelope_id int(11) name varchar(200) asciiname varchar(200) latitude decimal(10,7) longitude decimal(10,7) fclass varchar(1) fcode varchar(10) page_rank decimal(15,12) date_from date date_until date comments varchar(4000) country_code varchar(2) cc2 varchar(60) country_name varchar(200) continent_name varchar(80) adm1_code varchar(20) adm1_name_utf varchar(400) adm1_name_ascii varchar(400) adm2_code varchar(80) adm2_name_utf varchar(400) adm2_name_ascii varchar(400) adm3_code varchar(20) adm4_code varchar(20) population int(11) elevation int(11) gtopo30 int(11) timezone varchar(40) gmt_offset decimal(3,1) dst_offset decimal(3,1) raw_offset decimal(3,1) moddate date city_code_list varchar(100) city_name_list varchar(500) city_detail_list varchar(500) tvl-_por_list varchar(100) state_code varchar(3) location_type varchar(4) wiki_link varchar(200) alt_name_section text

iata_code^icao_code^faa_code^is_geonames^geoname_id^envelope_id^ name^asciiname^ latitude^longitude^fclass^fcode^ page_rank^date_from^date_until^comment^ country_code^cc2^country_name^continent_name^ adm1_-code^adm1_name_utf^adm1_name_ascii^ adm2_code^adm2_name_utf^adm2_name_ascii^ adm3_code^adm4-_code^ population^elevation^gtopo30^ timezone^gmt_offset^dst_offset^raw_offset^moddate^ city_code_-list^city_name_list^city_detail_list^tvl_por_list^ state_code^location_type^wiki_link^ alt_name_section Grammar for the Por-Rule parser.

Definition at line 1069 of file PORParserHelper.cpp.

23.72.2 Constructor & Destructor Documentation

23.72.2.1 template<typename Iterator> OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser (Location & *ioPORRule*) [inline]

Definition at line 1071 of file PORParserHelper.cpp.

References OPENTREP::Location::_itDay, OPENTREP::Location::_itMonth, OPENTREP::Location::_itYear, O-PENTREP::PorParserHelper::LocationParser< Iterator >::_location, OPENTREP::PorParserHelper::Location-Parser< Iterator >::adm1_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::adm1_name_ascii, OPENTREP::PorParserHelper::LocationParser< Iterator >::adm1_name_utf, OPENTREP::PorParserHelper::-LocationParser< Iterator >::adm2_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::adm2-_name_ascii, OPENTREP::PorParserHelper::LocationParser< Iterator >::adm2_name_utf, OPENTREP::Por-ParserHelper::LocationParser< Iterator >::adm3_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::adm4_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::alt_lang_code, OPENTREP::Por-ParserHelper::LocationParser< Iterator >::alt_lang_code_ftd, OPENTREP::PorParserHelper::LocationParser< Iterator >::alt_name, OPENTREP::PorParserHelper::LocationParser< Iterator >::alt_name_details, OPENTRE-P::PorParserHelper::LocationParser< Iterator >::alt_name_qualifiers, OPENTREP::PorParserHelper::Location-Parser< Iterator >::alt_name_section, OPENTREP::PorParserHelper::LocationParser< Iterator >::alt_name_sep, OPENTREP::PorParserHelper::LocationParser< Iterator >::alt_name_short, OPENTREP::PorParserHelper::-LocationParser< Iterator >::alt_name_short_list, OPENTREP::PorParserHelper::LocationParser< Iterator >-::ascii_name, OPENTREP::PorParserHelper::boolean_p, OPENTREP::PorParserHelper::LocationParser< Iterator >::city_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::city_code_list, OPENTREP::PorParser-

Helper::LocationParser< Iterator >::city_detail_list, OPENTREP::PorParserHelper::LocationParser< Iterator >::city_details, OPENTREP::PorParserHelper::LocationParser< Iterator >::city_geoname_id, OPENTREP::PorParserHelper::LocationParser< Iterator >::city_name_ascii, OPENTREP::PorParserHelper::LocationParser< Iterator >::city_name_list, OPENTREP::PorParserHelper::LocationParser< Iterator >::city_name_utf, OPENTREP::PorParserHelper::LocationParser< Iterator >::comments, OPENTREP::PorParserHelper::LocationParser< Iterator >::common_name, OPENTREP::PorParserHelper::LocationParser< Iterator >::continent_name, OPENTREP::PorParserHelper::LocationParser< Iterator >::country_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::country_code2, OPENTREP::PorParserHelper::LocationParser< Iterator >::country_name, OPENTREP::PorParserHelper::LocationParser< Iterator >::date, OPENTREP::PorParserHelper::LocationParser< Iterator >::date_from, OPENTREP::PorParserHelper::LocationParser< Iterator >::date_until, OPENTREP::PorParserHelper::day_p, OPENTREP::PorParserHelper::LocationParser< Iterator >::dst_offset, OPENTREP::PorParserHelper::LocationParser< Iterator >::elevation, OPENTREP::PorParserHelper::LocationParser< Iterator >::envelope_id, OPENTREP::PorParserHelper::LocationParser< Iterator >::faa_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::feat_class, OPENTREP::PorParserHelper::LocationParser< Iterator >::feat_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::geoname_id, OPENTREP::PorParserHelper::LocationParser< Iterator >::gmt_offset, OPENTREP::PorParserHelper::LocationParser< Iterator >::gtopo30, OPENTREP::PorParserHelper::LocationParser< Iterator >::header, OPENTREP::PorParserHelper::LocationParser< Iterator >::iata_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::icao_code, OPENTREP::PorParserHelper::int1_5_p, OPENTREP::PorParserHelper::LocationParser< Iterator >::is_geonames, OPENTREP::PorParserHelper::LocationParser< Iterator >::lang_code_2char, OPENTREP::PorParserHelper::LocationParser< Iterator >::lang_code_ext, OPENTREP::PorParserHelper::LocationParser< Iterator >::lang_code_hist, OPENTREP::PorParserHelper::LocationParser< Iterator >::lang_code_opt, OPENTREP::PorParserHelper::LocationParser< Iterator >::latitude, OPENTREP::PorParserHelper::LocationParser< Iterator >::longitude, OPENTREP::PorParserHelper::LocationParser< Iterator >::mod_date, OPENTREP::PorParserHelper::month_p, OPENTREP::PorParserHelper::LocationParser< Iterator >::page_rank, OPENTREP::PorParserHelper::LocationParser< Iterator >::population, OPENTREP::PorParserHelper::LocationParser< Iterator >::por_details, OPENTREP::PorParserHelper::LocationParser< Iterator >::por_key, OPENTREP::PorParserHelper::LocationParser< Iterator >::por_rule, OPENTREP::PorParserHelper::LocationParser< Iterator >::por_rule_end, OPENTREP::PorParserHelper::LocationParser< Iterator >::por_type, OPENTREP::PorParserHelper::LocationParser< Iterator >::raw_offset, OPENTREP::PorParserHelper::LocationParser< Iterator >::start, OPENTREP::PorParserHelper::LocationParser< Iterator >::state_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::time_zone, OPENTREP::PorParserHelper::LocationParser< Iterator >::tvl_por_code, OPENTREP::PorParserHelper::LocationParser< Iterator >::tvl_por_code_list, OPENTREP::PorParserHelper::LocationParser< Iterator >::tvl_por_sep, OPENTREP::PorParserHelper::uint1_4_p, OPENTREP::PorParserHelper::uint1_9_p, OPENTREP::PorParserHelper::LocationParser< Iterator >::wiki_link, and OPENTREP::PorParserHelper::year_p.

23.72.3 Member Data Documentation

23.72.3.1 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::start

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.2 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::header

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.3 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::por_rule

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.4 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::por_rule_end

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.5 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::por_key

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.6 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::por_details

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.7 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::iata_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.8 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::icao_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.9 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::faa_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.10 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::geoname_id

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.11 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::envelope_id

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.12 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::is_geonames

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.13 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::common_name

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.14 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::ascii_name

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.15 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::alt_name_short_list

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.16 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::alt_name_short

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.17 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::alt_name_sep

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.18 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::tvl_por_code_list

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.19 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::tvl_por_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.20 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::tvl_por_sep

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.21 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::latitude

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.22 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::longitude

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.23 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::feat_class

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.24 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::feat_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.25 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::page_rank

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.26 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::date_from

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.27 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::date_until

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.28 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::comments

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.29 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::country_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.30 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::country_code2

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.31 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::country_name

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.32 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::continent_name

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.33 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::adm1_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.34 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::adm1_name_utf

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.35 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::adm1_name_ascii

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.36 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::adm2_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.37 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::adm2_name_utf

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.38 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::adm2_name_ascii

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.39 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::adm3_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.40 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::adm4_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.41 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::population

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.42 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::elevation

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.43 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::gtopo30

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.44 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::time_zone

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.45 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::gmt_offset

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.46 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::dst_offset

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.47 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::raw_offset

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.48 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::mod_date

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.49 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::date

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.50 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::city_code_list

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.51 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::city_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.52 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::city_name_list

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.53 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::city_name_utf

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.54 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::city_name_ascii

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.55 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::city_detail_list

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.56 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::city_details

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.57 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::city_geoname_id

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.58 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::state_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.59 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::por_type

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.60 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::wiki_link

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.61 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::alt_name_section

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.62 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::alt_name_details

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.63 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::alt_lang_code

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.64 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::alt_lang_code_ftd

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.65 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::alt_name

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.66 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::-LocationParser< Iterator >::alt_name_qualifiers

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.67 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::lang_code_opt

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.68 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::lang_code_2char

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.69 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::lang_code_ext

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.70 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::lang_code_hist

Definition at line 1437 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

23.72.3.71 template<typename Iterator> bsq::rule<Iterator, bsu::blank_type> OPENTREP::PorParserHelper::LocationParser< Iterator >::destination

Definition at line 1437 of file PORParserHelper.cpp.

23.72.3.72 template<typename Iterator> Location& OPENTREP::PorParserHelper::LocationParser< Iterator >::_location

Definition at line 1461 of file PORParserHelper.cpp.

Referenced by OPENTREP::PorParserHelper::LocationParser< Iterator >::LocationParser().

The documentation for this struct was generated from the following file:

- opentrep/bom/PORParserHelper.cpp

23.73 OPENTREP::Logger Class Reference

`#include <opentrep/service/Logger.hpp>`

Public Member Functions

- template<typename T >
  void log (const LOG::EN_LogLevel iLevel, const int iLineNumber, const std::string &iFileName, const T &iToBeLogged)
- LOG::EN_LogLevel getLogLevel ()
- std::ostream & getLogStream ()
- void setLogParameters (const LOG::EN_LogLevel iLogLevel, std::ostream &ioLogStream)

Static Public Member Functions

- static Logger & instance ()

Friends

- class FacSupervisor

23.73.1 Detailed Description

Class holding the stream for logs.

Note that the error logs are seen as standard output logs, but with a higher level of visibility.

Definition at line 45 of file Logger.hpp.

23.73.2 Member Function Documentation

23.73.2.1 template<typename T > void OPENTREP::Logger::log (const LOG::EN_LogLevel *iLevel,* const int *iLineNumber,* const std::string & *iFileName,* const T & *iToBeLogged*) [inline]

Main log entry.

Definition at line 52 of file Logger.hpp.

23.73.2.2 LOG::EN_LogLevel OPENTREP::Logger::getLogLevel ( )

Get the log level.

Definition at line 37 of file Logger.cpp.

23.73.2.3 std::ostream & OPENTREP::Logger::getLogStream ( )

get the log stream.

Definition at line 42 of file Logger.cpp.

23.73.2.4 void OPENTREP::Logger::setLogParameters (const LOG::EN_LogLevel *iLogLevel,* std::ostream & *ioLogStream*)

Set the logger parameters (level and stream).

Definition at line 48 of file Logger.cpp.

Referenced by OPENTREP::logInit().

23.73.2.5 Logger & OPENTREP::Logger::instance ( ) [static]

Returns a current Logger instance.

Definition at line 55 of file Logger.cpp.

References OPENTREP::LOG::DEBUG.

Referenced by OPENTREP::logInit().

23.73.3 Friends And Related Function Documentation

23.73.3.1 friend class FacSupervisor [friend]

Definition at line 47 of file Logger.hpp.

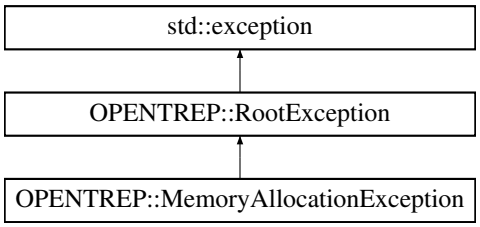
The documentation for this class was generated from the following files:

- opentrep/service/Logger.hpp
- opentrep/service/Logger.cpp

23.74 OPENTREP::MemoryAllocationException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::MemoryAllocationException:



Public Member Functions

- MemoryAllocationException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.74.1 Detailed Description

Memory allocation issue.

Definition at line 112 of file OPENTREP_exceptions.hpp.

23.74.2 Constructor & Destructor Documentation

23.74.2.1 OPENTREP::MemoryAllocationException::MemoryAllocationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 117 of file OPENTREP_exceptions.hpp.

23.74.3 Member Function Documentation

23.74.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.74.4 Member Data Documentation

23.74.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

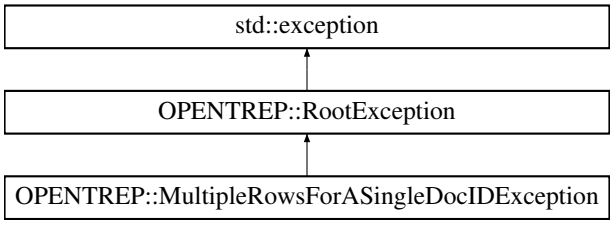
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.75 OPENTREP::MultipleRowsForASingleDocIDException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::MultipleRowsForASingleDocIDException:



Public Member Functions

- MultipleRowsForASingleDocIDException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.75.1 Detailed Description

Multiple rows for a single Xapian document ID.

Definition at line 88 of file OPENTREP_exceptions.hpp.

23.75.2 Constructor & Destructor Documentation

23.75.2.1 OPENTREP::MultipleRowsForASingleDocIDException::MultipleRowsForASingleDocIDException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 93 of file OPENTREP_exceptions.hpp.

23.75.3 Member Function Documentation

23.75.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.75.4 Member Data Documentation

23.75.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

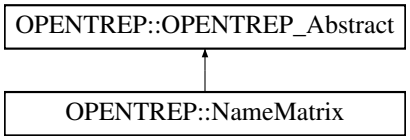
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.76 OPENTREP::NameMatrix Struct Reference

`#include <opentrep/NameMatrix.hpp>`

Inheritance diagram for OPENTREP::NameMatrix:



Public Member Functions

- NameMatrix ()
- NameMatrix (const NameMatrix &)
- ~NameMatrix ()
- const NameMatrix_T & getNameMatrix () const
- bool getNameList (const LanguageCode_T &, NameList_T &) const
- void addName (const LanguageCode_T &, const std::string &iName)
- void reset ()
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- std::string toString () const
- std::string describeKey () const
- std::string describe () const

23.76.1 Detailed Description

Structure holding names for a BOM type (place, day, month, etc.), for a list of given languages.

Definition at line 22 of file NameMatrix.hpp.

23.76.2 Constructor & Destructor Documentation

23.76.2.1 OPENTREP::NameMatrix::NameMatrix ( )

Default constructor.

Definition at line 14 of file NameMatrix.cpp.

23.76.2.2 OPENTREP::NameMatrix::NameMatrix (const NameMatrix & *iNameMatrix*)

Copy constructor.

Definition at line 18 of file NameMatrix.cpp.

23.76.2.3 OPENTREP::NameMatrix::~NameMatrix ( )

Destructor.

Definition at line 23 of file NameMatrix.cpp.

23.76.3 Member Function Documentation

23.76.3.1 const NameMatrix_T& OPENTREP::NameMatrix::getNameMatrix ( ) const [inline]

Get the map of name lists.

Definition at line 44 of file NameMatrix.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.76.3.2 bool OPENTREP::NameMatrix::getNameList (const LanguageCode_T & *iLanguageCode,* NameList_T & *ioNameList*) const

Get, for a given language (code), the corresponding list of names.

Parameters

const	LanguageCode_T& Language code.
NameList_T&	Empty list of names, which will be filled by the method if a list exists for that language code.

Returns

bool Whether or not such a list exists for the given language.

Definition at line 67 of file NameMatrix.cpp.

References OPENTREP::Names::getNameList().

Referenced by OPENTREP::Location::getNameList().

23.76.3.3 void OPENTREP::NameMatrix::addName (const LanguageCode_T & *iLanguageCode,* const std::string & *iName*)

Add a name for the place.

Parameters

const	LanguageCode_T& Language in which to add the name.
const	std::string& Name to be added.

Definition at line 82 of file NameMatrix.cpp.

References OPENTREP::Names::addName(), and OPENTREP_LOG_ERROR.

Referenced by OPENTREP::Location::addName().

23.76.3.4 void OPENTREP::NameMatrix::reset ( )

Reset the map of name lists.

Definition at line 112 of file NameMatrix.cpp.

Referenced by OPENTREP::Location::resetMatrix().

23.76.3.5 void OPENTREP::NameMatrix::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 51 of file NameMatrix.cpp.

References describe().

23.76.3.6 void OPENTREP::NameMatrix::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 56 of file NameMatrix.cpp.

23.76.3.7 std::string OPENTREP::NameMatrix::toString ( ) const [virtual]

Get the serialised version of the structure.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 60 of file NameMatrix.cpp.

References describe().

23.76.3.8 std::string OPENTREP::NameMatrix::describeKey ( ) const

Display the key of the structure.

Returns

std::string Dump of the structure.

Definition at line 27 of file NameMatrix.cpp.

Referenced by describe().

23.76.3.9 std::string OPENTREP::NameMatrix::describe ( ) const

Display of the structure.

Returns

std::string Dump of the structure.

Definition at line 33 of file NameMatrix.cpp.

References OPENTREP::Names::describe(), and describeKey().

Referenced by toStream(), toString(), and OPENTREP::Location::toString().

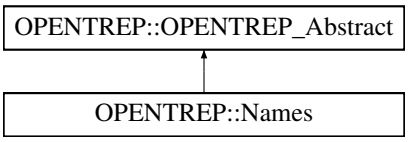
The documentation for this struct was generated from the following files:

- opentrep/NameMatrix.hpp
- opentrep/basic/NameMatrix.cpp

23.77 OPENTREP::Names Struct Reference

`#include <opentrep/Names.hpp>`

Inheritance diagram for OPENTREP::Names:



Public Member Functions

- Names (const LanguageCode_T &)
- Names (const Names &)
- ~Names ()
- LanguageCode_T getLanguageCode () const
- const NameList_T & getNameList () const
- std::string getFirstName () const
- void setLanguageCode (const LanguageCode_T &iLanguageCode)
- void addName (const std::string &iName)
- void resetList ()
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- std::string describeKey () const
- std::string describe () const

23.77.1 Detailed Description

Structure holding names for a BOM type (place, day, month, etc.), in a given language.

Definition at line 26 of file Names.hpp.

23.77.2 Constructor & Destructor Documentation

23.77.2.1 OPENTREP::Names::Names (const LanguageCode_T & *iLanguageCode*)

Main constructor.

Definition at line 18 of file Names.cpp.

23.77.2.2 OPENTREP::Names::Names (const Names & *iName*)

Copy constructor.

Definition at line 23 of file Names.cpp.

23.77.2.3 OPENTREP::Names::~Names ( )

Destructor.

Definition at line 29 of file Names.cpp.

23.77.3 Member Function Documentation

23.77.3.1 LanguageCode_T OPENTREP::Names::getLanguageCode ( ) const [inline]

Get the language code.

Definition at line 53 of file Names.hpp.

Referenced by OPENTREP::LocationExchange::exportLocation().

23.77.3.2 const NameList_T& OPENTREP::Names::getNameList ( ) const [inline]

Get the name list.

Definition at line 60 of file Names.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), OPENTREP::LocationExchange::exportLocation(), OPENTREP::NameMatrix::getNameList(), and OPENTREP::BomJSONExport::jsonExportLocation().

23.77.3.3 std::string OPENTREP::Names::getFirstName ( ) const

Get the first name of the list.

Note that it can be empty (when the list is itself empty).

Definition at line 33 of file Names.cpp.

23.77.3.4 void OPENTREP::Names::setLanguageCode (const LanguageCode_T & *iLanguageCode*) [inline]

Set the language code.

Definition at line 77 of file Names.hpp.

23.77.3.5 void OPENTREP::Names::addName (const std::string & *iName*)

Add a name to the dedicated list.

Definition at line 89 of file Names.cpp.

Referenced by OPENTREP::NameMatrix::addName().

23.77.3.6 void OPENTREP::Names::resetList ( )

Reset the list of names to null (i.e., to an empty list).

Definition at line 94 of file Names.cpp.

23.77.3.7 void OPENTREP::Names::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 73 of file Names.cpp.

References describe().

23.77.3.8 void OPENTREP::Names::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 78 of file Names.cpp.

23.77.3.9 std::string OPENTREP::Names::toString ( ) const [virtual]

Get the serialised version of the structure.

Implements OPENTREP::OPENTREP_Abstract.

Definition at line 82 of file Names.cpp.

References describe().

23.77.3.10 std::string OPENTREP::Names::describeKey ( ) const

Display the key of the structure.

Returns

std::string Dump of the structure.

Definition at line 44 of file Names.cpp.

Referenced by describe().

23.77.3.11 std::string OPENTREP::Names::describe ( ) const

Display of the structure.

Returns

std::string Dump of the structure.

Definition at line 51 of file Names.cpp.

References describeKey().

Referenced by OPENTREP::NameMatrix::describe(), toStream(), and toString().

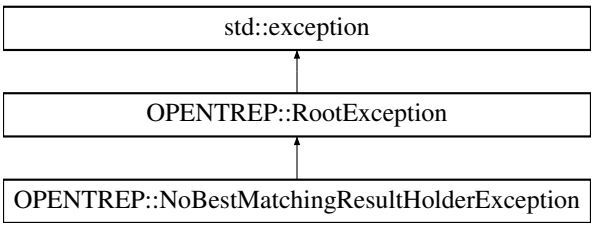
The documentation for this struct was generated from the following files:

- opentrep/Names.hpp
- opentrep/basic/Names.cpp

23.78 OPENTREP::NoBestMatchingResultHolderException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::NoBestMatchingResultHolderException:



Public Member Functions

- NoBestMatchingResultHolderException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.78.1 Detailed Description

There is best matching ResultHolder object.

Definition at line 274 of file OPENTREP_exceptions.hpp.

23.78.2 Constructor & Destructor Documentation

23.78.2.1 OPENTREP::NoBestMatchingResultHolderException::NoBestMatchingResultHolderException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 279 of file OPENTREP_exceptions.hpp.

23.78.3 Member Function Documentation

23.78.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.78.4 Member Data Documentation

23.78.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

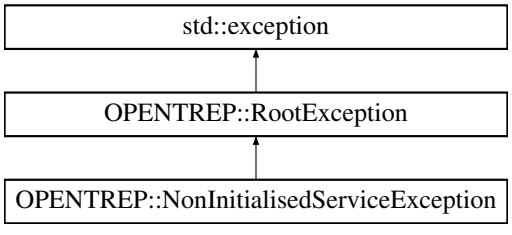
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.79 OPENTREP::NonInitialisedServiceException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::NonInitialisedServiceException:



Public Member Functions

- NonInitialisedServiceException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.79.1 Detailed Description

OpenTREP service not initialised.

Definition at line 53 of file OPENTREP_exceptions.hpp.

23.79.2 Constructor & Destructor Documentation

23.79.2.1 OPENTREP::NonInitialisedServiceException::NonInitialisedServiceException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 58 of file OPENTREP_exceptions.hpp.

23.79.3 Member Function Documentation

23.79.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.79.4 Member Data Documentation

23.79.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

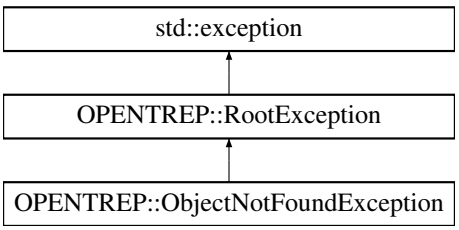
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.80 OPENTREP::ObjectNotFoundException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::ObjectNotFoundException:



Public Member Functions

- ObjectNotFoundException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.80.1 Detailed Description

The object cannot be retrieved.

Definition at line 123 of file OPENTREP_exceptions.hpp.

23.80.2 Constructor & Destructor Documentation

23.80.2.1 OPENTREP::ObjectNotFoundException::ObjectNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 128 of file OPENTREP_exceptions.hpp.

23.80.3 Member Function Documentation

23.80.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.80.4 Member Data Documentation

23.80.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

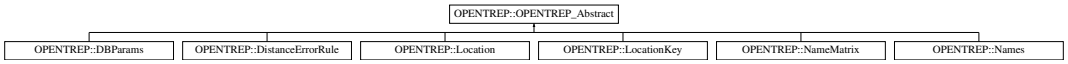
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.81 OPENTREP::OPENTREP_Abstract Struct Reference

`#include <opentrep/OPENTREP_Abstract.hpp>`

Inheritance diagram for OPENTREP::OPENTREP_Abstract:



Public Member Functions

- virtual void toStream (std::ostream &ioOut) const =0
- virtual void fromStream (std::istream &ioIn)=0
- virtual std::string toString () const =0

Protected Member Functions

- OPENTREP_Abstract ()
- OPENTREP_Abstract (const OPENTREP_Abstract &)
- virtual ~OPENTREP_Abstract ()

23.81.1 Detailed Description

Base class for the OPENTREP interface structures.

Definition at line 16 of file OPENTREP_Abstract.hpp.

23.81.2 Constructor & Destructor Documentation

23.81.2.1 OPENTREP::OPENTREP_Abstract::OPENTREP_Abstract ( ) [inline], [protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 43 of file OPENTREP_Abstract.hpp.

23.81.2.2 OPENTREP::OPENTREP_Abstract::OPENTREP_Abstract (const OPENTREP_Abstract &) [inline], [protected]

Definition at line 44 of file OPENTREP_Abstract.hpp.

23.81.2.3 virtual OPENTREP::OPENTREP_Abstract::~OPENTREP_Abstract ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 49 of file OPENTREP_Abstract.hpp.

23.81.3 Member Function Documentation

23.81.3.1 virtual void OPENTREP::OPENTREP_Abstract::toStream (std::ostream & *ioOut*) const [pure virtual]

Dump a structure into an output stream.

Parameters

ostream&	the output stream.

Implemented in OPENTREP::Location, OPENTREP::DBParams, OPENTREP::Names, OPENTREP::LocationKey, OPENTREP::DistanceErrorRule, and OPENTREP::NameMatrix.

23.81.3.2 virtual void OPENTREP::OPENTREP_Abstract::fromStream (std::istream & *ioIn*) [pure virtual]

Read a structure from an input stream.

Parameters

istream&	the input stream.

Implemented in OPENTREP::Location, OPENTREP::DBParams, OPENTREP::Names, OPENTREP::LocationKey, OPENTREP::DistanceErrorRule, and OPENTREP::NameMatrix.

Referenced by operator>>().

23.81.3.3 virtual std::string OPENTREP::OPENTREP_Abstract::toString ( ) const [pure virtual]

Get the serialised version of the structure.

Implemented in OPENTREP::Location, OPENTREP::DBParams, OPENTREP::DistanceErrorRule, OPENTREP::Names, OPENTREP::LocationKey, and OPENTREP::NameMatrix.

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Abstract.hpp

23.82 OPENTREP::OPENTREP_Service Class Reference

Interface for the OPENTREP Services.

`#include <opentrep/OPENTREP_Service.hpp>`

Public Types

- typedef std::pair< const TravelDBFilePath_T, const SQLiteDBFilePath_T > DBFilePathPair_T
- typedef std::pair< const PORFilePath_T, const DBFilePathPair_T > FilePathSet_T

Public Member Functions

- NbOfDBEntries_T getIndexSize ()
- NbOfMatches_T drawRandomLocations (const NbOfMatches_T &iNbOfDraws, LocationList_T &)
- NbOfDBEntries_T buildSQLDB ()
- NbOfDBEntries_T buildSearchIndex ()
- NbOfMatches_T interpretTravelRequest (const std::string &iTravelQuery, LocationList_T &, WordList_T &)
- FilePathSet_T getFilePaths () const
- OPENTREP_Service (std::ostream &ioLogStream, const TravelDBFilePath_T &, const SQLiteDBFilePath_T &)
- OPENTREP_Service (std::ostream &ioLogStream, const PORFilePath_T &, const TravelDBFilePath_T &, const SQLiteDBFilePath_T &)
- ~OPENTREP_Service ()

23.82.1 Detailed Description

Interface for the OPENTREP Services.

Definition at line 24 of file OPENTREP_Service.hpp.

23.82.2 Member Typedef Documentation

23.82.2.1 typedef std::pair<const TravelDBFilePath_T, const SQLiteDBFilePath_T> OPENTREP::OPENTREP_Service::DBFilePathPair_T

Get the file-paths of the Xapian database/index and of the ORI-maintained POR (points of reference).

Returns

std::pair<const PORFilePath_T, File-path of the file of POR (points of reference). std::pair<const TravelDBFilePath_T, File-path of the Xapian database. const SQLiteDBFilePath_T> > File-path of the SQLite3 database.

Definition at line 90 of file OPENTREP_Service.hpp.

23.82.2.2 typedef std::pair<const PORFilePath_T, const DBFilePathPair_T> OPENTREP::OPENTREP_Service::FilePathSet_T

Definition at line 91 of file OPENTREP_Service.hpp.

23.82.3 Constructor & Destructor Documentation

23.82.3.1 OPENTREP::OPENTREP_Service::OPENTREP_Service (std::ostream & *ioLogStream,* const TravelDBFilePath_T & *iTravelDBFilePath,* const SQLiteDBFilePath_T & *iSQLiteDBFilePath*)

Constructor.

Parameters

std::ostream&	Output log stream (for instance, std::cout).
const	TravelDBFilePath_T& File-path of the Xapian index/database.
const	SQLiteDBFilePath_T& File-path of the SQLite3 database.

Definition at line 37 of file OPENTREP_Service.cpp.

23.82.3.2 OPENTREP::OPENTREP_Service::OPENTREP_Service (std::ostream & *ioLogStream,* const PORFilePath_T & *iPORFilepath,* const TravelDBFilePath_T & *iTravelDBFilePath,* const SQLiteDBFilePath_T & *iSQLiteDBFilePath*)

Constructor.

Parameters

std::ostream&	Output log stream (for instance, std::cout).
const	PORFilePath_T& File-path of the file of POR (points of reference).
const	TravelDBFilePath_T& File-path of the Xapian index/database.
const	SQLiteDBFilePath_T& File-path of the SQLite3 database.

Definition at line 28 of file OPENTREP_Service.cpp.

23.82.3.3 OPENTREP::OPENTREP_Service::~OPENTREP_Service ( )

Destructor.

Definition at line 55 of file OPENTREP_Service.cpp.

23.82.4 Member Function Documentation

23.82.4.1 NbOfDBEntries_T OPENTREP::OPENTREP_Service::getIndexSize ( )

Get the number of documents within the Xapian database (index).

Returns

NbOfDBEntries_T Number of documents within the Xapian database.

Definition at line 132 of file OPENTREP_Service.cpp.

References OPENTREP::OPENTREP_ServiceContext::display(), OPENTREP::BasChronometer::elapsed(), OPENTREP::OPENTREP_ServiceContext::getTravelDBFilePath(), OPENTREP_LOG_DEBUG, and OPENTREP::BasChronometer::start().

23.82.4.2 NbOfMatches_T OPENTREP::OPENTREP_Service::drawRandomLocations (const NbOfMatches_T & *iNbOfDraws,* LocationList_T & *ioLocationList*)

Return a given number of locations, all randomly draw from the Xapian database (index).

Parameters

LocationList_T&	List of (geographical) locations randomly picked up from the Xapian database.

Returns

NbOfMatches_T Number of documents actually retrieved from the Xapian database/index.

Definition at line 161 of file OPENTREP_Service.cpp.

References OPENTREP::OPENTREP_ServiceContext::display(), OPENTREP::BasChronometer::elapsed(), OPENTREP::OPENTREP_ServiceContext::getTravelDBFilePath(), OPENTREP_LOG_DEBUG, and OPENTREP::BasChronometer::start().

23.82.4.3 NbOfDBEntries_T OPENTREP::OPENTREP_Service::buildSQLDB ( )

Build the SQL database from the file with the ORI-maintained list of POR (points of reference).

Returns

NbOfDBEntries_T Number of documents inserted in the SQLite3 DB.

Definition at line 192 of file OPENTREP_Service.cpp.

References OPENTREP::DBManager::buildSQLDB(), OPENTREP::OPENTREP_ServiceContext::display(), OPENTREP::BasChronometer::elapsed(), OPENTREP::OPENTREP_ServiceContext::getPORFilePath(), OPENTREP::OPENTREP_ServiceContext::getSQLiteDBFilePath(), OPENTREP_LOG_DEBUG, and OPENTREP::BasChronometer::start().

Referenced by main().

23.82.4.4 NbOfDBEntries_T OPENTREP::OPENTREP_Service::buildSearchIndex ( )

Build the Xapian database (index) from the file with the ORI-maintained list of POR (points of reference).

Returns

NbOfDBEntries_T Number of documents indexed by the Xapian database/index.

Definition at line 223 of file OPENTREP_Service.cpp.

References OPENTREP::OPENTREP_ServiceContext::display(), OPENTREP::BasChronometer::elapsed(), OPENTREP::OPENTREP_ServiceContext::getPORFilePath(), OPENTREP::OPENTREP_ServiceContext::getTransliterator(), OPENTREP::OPENTREP_ServiceContext::getTravelDBFilePath(), OPENTREP_LOG_DEBUG, and OPENTREP::BasChronometer::start().

Referenced by main().

23.82.4.5 NbOfMatches_T OPENTREP::OPENTREP_Service::interpretTravelRequest (const std::string & *iTravelQuery*, LocationList_T & *ioLocationList*, WordList_T & *ioWordList*)

Match the given string, thanks to a full-text search on the underlying Xapian index (named "database").

Parameters

const	std::string& (Travel-related) query string (e.g., "sna francicso rio de janero lso angles reykyavki nce iev mow").
LocationList_T&	List of (geographical) locations, if any, matching the given query string.
WordList_T&	List of non-matched words of the query string.

Returns

NbOfMatches_T Number of matches.

Definition at line 263 of file OPENTREP_Service.cpp.

References OPENTREP::OPENTREP_ServiceContext::display(), OPENTREP::BasChronometer::elapsed(), OPENTREP::OPENTREP_ServiceContext::getTravelDBFilePath(), OPENTREP_LOG_DEBUG, OPENTREP_LOG_ERROR, and OPENTREP::BasChronometer::start().

Referenced by parseQuery().

23.82.4.6 OPENTREP_Service::FilePathSet_T OPENTREP::OPENTREP_Service::getFilePaths ( ) const

Definition at line 107 of file OPENTREP_Service.cpp.

References OPENTREP::OPENTREP_ServiceContext::getPORFilePath(), OPENTREP::OPENTREP_ServiceContext::getSQLiteDBFilePath(), and OPENTREP::OPENTREP_ServiceContext::getTravelDBFilePath().

The documentation for this class was generated from the following files:

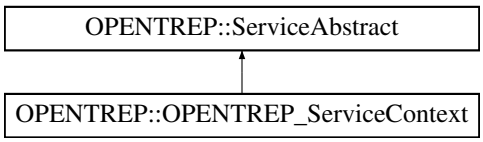
- opentrep/OPENTREP_Service.hpp
- opentrep/service/OPENTREP_Service.cpp

23.83 OPENTREP::OPENTREP_ServiceContext Class Reference

Class holding the context of the OpenTrep services.

`#include <opentrep/service/OPENTREP_ServiceContext.hpp>`

Inheritance diagram for OPENTREP::OPENTREP_ServiceContext:



Public Member Functions

- World * getWorld () const
- World & getWorldHandler () const

- const PORFilePath_T & getPORFilePath () const
- const TravelDBFilePath_T & getTravelDBFilePath () const
- const SQLiteDBFilePath_T & getSQLiteDBFilePath () const
- const OTransliterator & getTransliterator () const
- void setWorld (World &ioWorld)
- void setPORFilePath (const std::string &iPORFilePath)
- void setTravelDBFilePath (const std::string &iTravelDBFilePath)
- void setSQLiteDBFilePath (const std::string &iSQLiteDBFilePath)
- void setTransliterator (const OTransliterator &iTransliterator)
- const std::string shortDisplay () const
- const std::string display () const
- virtual void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Friends

- class FacOpenTrepServiceContext

23.83.1 Detailed Description

Class holding the context of the OpenTrep services.

Definition at line 27 of file OPENTREP_ServiceContext.hpp.

23.83.2 Member Function Documentation

23.83.2.1 World* OPENTREP::OPENTREP_ServiceContext::getWorld ( ) const [inline]

Get the World object (BOM root).

Definition at line 34 of file OPENTREP_ServiceContext.hpp.

23.83.2.2 World & OPENTREP::OPENTREP_ServiceContext::getWorldHandler ( ) const

Get the World object (BOM root).

Definition at line 49 of file OPENTREP_ServiceContext.cpp.

23.83.2.3 const PORFilePath_T& OPENTREP::OPENTREP_ServiceContext::getPORFilePath ( ) const [inline]

Get the file-path of the file of POR (points of reference).

Definition at line 46 of file OPENTREP_ServiceContext.hpp.

Referenced by OPENTREP::OPENTREP_Service::buildSearchIndex(), OPENTREP::OPENTREP_Service::buildSQLDB(), and OPENTREP::OPENTREP_Service::getFilePaths().

23.83.2.4 const TravelDBFilePath_T& OPENTREP::OPENTREP_ServiceContext::getTravelDBFilePath ( ) const [inline]

Get the Xapian database name.

Definition at line 53 of file OPENTREP_ServiceContext.hpp.

Referenced by OPENTREP::OPENTREP_Service::buildSearchIndex(), OPENTREP::OPENTREP_Service::drawRandomLocations(), OPENTREP::OPENTREP_Service::getFilePaths(), OPENTREP::OPENTREP_Service::getIndexSize(), and OPENTREP::OPENTREP_Service::interpretTravelRequest().

23.83.2.5 const SQLiteDBFilePath_T& OPENTREP::OPENTREP_ServiceContext::getSQLiteDBFilePath ( ) const [inline]

Get the SQLite3 database file-path.

Definition at line 60 of file OPENTREP_ServiceContext.hpp.

Referenced by OPENTREP::OPENTREP_Service::buildSQLDB(), and OPENTREP::OPENTREP_Service::getFilePaths().

23.83.2.6 const OTransliterator& OPENTREP::OPENTREP_ServiceContext::getTransliterator ( ) const [inline]

Get the Unicode transliterator.

Definition at line 67 of file OPENTREP_ServiceContext.hpp.

Referenced by OPENTREP::OPENTREP_Service::buildSearchIndex().

23.83.2.7 void OPENTREP::OPENTREP_ServiceContext::setWorld (World & *ioWorld*) [inline]

Set the World object (BOM root).

Definition at line 76 of file OPENTREP_ServiceContext.hpp.

23.83.2.8 void OPENTREP::OPENTREP_ServiceContext::setPORFilePath (const std::string & *iPORFilePath*) [inline]

Set the file-path of the file of POR (points of reference).

Definition at line 83 of file OPENTREP_ServiceContext.hpp.

23.83.2.9 void OPENTREP::OPENTREP_ServiceContext::setTravelDBFilePath (const std::string & *iTravelDBFilePath*) [inline]

Set the Xapian database name.

Definition at line 90 of file OPENTREP_ServiceContext.hpp.

23.83.2.10 void OPENTREP::OPENTREP_ServiceContext::setSQLiteDBFilePath (const std::string & *iSQLiteDBFilePath*) [inline]

Set the SQLite3 database file-path.

Definition at line 97 of file OPENTREP_ServiceContext.hpp.

23.83.2.11 void OPENTREP::OPENTREP_ServiceContext::setTransliterator (const OTransliterator & *iTransliterator*) [inline]

Set the Unicode transliterator.

Definition at line 104 of file OPENTREP_ServiceContext.hpp.

23.83.2.12 const std::string OPENTREP::OPENTREP_ServiceContext::shortDisplay ( ) const

Display the short OPENTREP_ServiceContext content.

Definition at line 55 of file OPENTREP_ServiceContext.cpp.

Referenced by display().

23.83.2.13 const std::string OPENTREP::OPENTREP_ServiceContext::display ( ) const

Display the full OPENTREP_ServiceContext content.

Definition at line 66 of file OPENTREP_ServiceContext.cpp.

References OPENTREP::World::display(), and shortDisplay().

Referenced by OPENTREP::OPENTREP_Service::buildSearchIndex(), OPENTREP::OPENTREP_Service::build-

SQLDB(), OPENTREP::OPENTREP_Service::drawRandomLocations(), OPENTREP::OPENTREP_Service::getIndexSize(), and OPENTREP::OPENTREP_Service::interpretTravelRequest().

23.83.2.14 virtual void OPENTREP::ServiceAbstract::toStream (std::ostream & *ioOut*) const [inline], [virtual], [inherited]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Definition at line 22 of file ServiceAbstract.hpp.

23.83.2.15 virtual void OPENTREP::ServiceAbstract::fromStream (std::istream & *ioIn*) [inline], [virtual], [inherited]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Definition at line 26 of file ServiceAbstract.hpp.

Referenced by operator>>().

23.83.3 Friends And Related Function Documentation

23.83.3.1 friend class FacOpenTrepServiceContext [friend]

Definition at line 28 of file OPENTREP_ServiceContext.hpp.

The documentation for this class was generated from the following files:

- opentrep/service/OPENTREP_ServiceContext.hpp
- opentrep/service/OPENTREP_ServiceContext.cpp

23.84 OPENTREP::OpenTrepSearcher Struct Reference

API wrapper around the OpenTREP C++ API, so that Python scripts can use it seamlessly.

Public Member Functions

- std::string getPaths ()
- std::string index ()
- std::string generate (const std::string &iOutputFormatString, const NbOfMatches_T &iNbOfDraws)
- std::string search (const std::string &iOutputFormatString, const std::string &iTravelQuery)
- OpenTrepSearcher ()
- OpenTrepSearcher (const OpenTrepSearcher &iOpenTrepSearcher)
- ~OpenTrepSearcher ()
- bool init (const std::string &iTravelDBFilePath, const std::string &iSQLiteDBFilePath, const std::string &iLogFilePath)
- bool finalize ()

23.84.1 Detailed Description

API wrapper around the OpenTREP C++ API, so that Python scripts can use it seamlessly.

Definition at line 26 of file pyopentrep.cpp.

23.84.2 Constructor & Destructor Documentation

23.84.2.1 OPENTREP::OpenTrepSearcher::OpenTrepSearcher ( ) [inline]

Default constructor.

Definition at line 594 of file pyopentrep.cpp.

23.84.2.2 OPENTREP::OpenTrepSearcher::OpenTrepSearcher (const OpenTrepSearcher & *iOpenTrepSearcher*) [inline]

Default copy constructor.

Definition at line 600 of file pyopentrep.cpp.

23.84.2.3 OPENTREP::OpenTrepSearcher::~OpenTrepSearcher ( ) [inline]

Default constructor.

Definition at line 608 of file pyopentrep.cpp.

23.84.3 Member Function Documentation

23.84.3.1 std::string OPENTREP::OpenTrepSearcher::getPaths ( ) [inline]

Get the file-paths for the:

- ORI-maintained list of POR (points of reference)
- Xapian-based travel database/index

Definition at line 36 of file pyopentrep.cpp.

Referenced by BOOST_PYTHON_MODULE().

23.84.3.2 std::string OPENTREP::OpenTrepSearcher::index ( ) [inline]

Public wrapper around the indexation use case.

Definition at line 43 of file pyopentrep.cpp.

Referenced by BOOST_PYTHON_MODULE().

23.84.3.3 std::string OPENTREP::OpenTrepSearcher::generate (const std::string & *iOutputFormatString,* const NbOfMatches_T & *iNbOfDraws*) [inline]

Public wrapper around the random generation use case.

Definition at line 50 of file pyopentrep.cpp.

References OPENTREP::OutputFormat::getFormat().

Referenced by BOOST_PYTHON_MODULE().

23.84.3.4 std::string OPENTREP::OpenTrepSearcher::search (const std::string & *iOutputFormatString,* const std::string & *iTravelQuery*) [inline]

Public wrapper around the search use case.

Definition at line 61 of file pyopentrep.cpp.

References OPENTREP::OutputFormat::getFormat().

Referenced by BOOST_PYTHON_MODULE().

23.84.3.5 bool OPENTREP::OpenTrepSearcher::init (const std::string & *iTravelDBFilePath,* const std::string & *iSQLiteDBFilePath,* const std::string & *iLogFilePath*) [inline]

Wrapper around the search use case.

Definition at line 616 of file pyopentrep.cpp.

References OPENTREP::RootException::what().

Referenced by BOOST_PYTHON_MODULE().

23.84.3.6 bool OPENTREP::OpenTrepSearcher::finalize ( ) [inline]

Wrapper around the search use case.

Definition at line 671 of file pyopentrep.cpp.

Referenced by BOOST_PYTHON_MODULE().

The documentation for this struct was generated from the following file:

- opentrep/python/pyopentrep.cpp

23.85 OPENTREP::OTransliterator Class Reference

`#include <opentrep/basic/OTransliterator.hpp>`

Public Member Functions

- std::string normalise (const std::string &iString) const
- OTransliterator ()
- OTransliterator (const OTransliterator &)
- ~OTransliterator ()

23.85.1 Detailed Description

Wrapper around a Unicode transliterator.

Definition at line 20 of file OTransliterator.hpp.

23.85.2 Constructor & Destructor Documentation

23.85.2.1 OPENTREP::OTransliterator::OTransliterator ( )

Default Constructor.

Definition at line 17 of file OTransliterator.cpp.

23.85.2.2 OPENTREP::OTransliterator::OTransliterator (const OTransliterator & *iTransliterator*)

Copy Constructor.

Definition at line 24 of file OTransliterator.cpp.

23.85.2.3 OPENTREP::OTransliterator::~OTransliterator ( )

Destructor.

Definition at line 42 of file OTransliterator.cpp.

23.85.3 Member Function Documentation

23.85.3.1 std::string OPENTREP::OTransliterator::normalise (const std::string & *iString*) const

Remove the accents of the given string.

Parameters

const	std::string& The string for which the strings must be removed

Returns

std::string The unaccentuated string.

Definition at line 134 of file OTransliterator.cpp.

References OPENTREP::getUTF8().

Referenced by OPENTREP::Place::addNameToXapianSets(), BOOST_AUTO_TEST_CASE(), and OPENTREP::Place::buildIndexSets().

The documentation for this class was generated from the following files:

- opentrep/basic/OTransliterator.hpp
- opentrep/basic/OTransliterator.cpp

23.86 OPENTREP::OutputFormat Struct Reference

Enumeration of output formats.

`#include <opentrep/OutputFormat.hpp>`

Public Types

- enum EN_OutputFormat {
  SHORT = 0, FULL, JSON, PROTOBUF,
  LAST_VALUE }

Public Member Functions

- EN_OutputFormat getFormat () const
- char getFormatAsChar () const
- std::string getFormatAsString () const
- const std::string describe () const
- bool operator== (const EN_OutputFormat &) const
- OutputFormat (const EN_OutputFormat &)
- OutputFormat (const char iFormat)
- OutputFormat (const std::string &iFormat)
- OutputFormat (const OutputFormat &)

Static Public Member Functions

- static const std::string & getLabel (const EN_OutputFormat &)
- static EN_OutputFormat getFormat (const char)
- static char getFormatLabel (const EN_OutputFormat &)
- static std::string getFormatLabelAsString (const EN_OutputFormat &)
- static std::string describeLabels ()

23.86.1 Detailed Description

Enumeration of output formats.

Definition at line 17 of file OutputFormat.hpp.

23.86.2 Member Enumeration Documentation

23.86.2.1 enum OPENTREP::OutputFormat::EN_OutputFormat

Enumerator

SHORT

FULL

JSON

PROTOBUF

LAST_VALUE

Definition at line 19 of file OutputFormat.hpp.

23.86.3 Constructor & Destructor Documentation

23.86.3.1 OPENTREP::OutputFormat::OutputFormat (const EN_OutputFormat & *iOutputFormat*)

Main constructor.

Definition at line 33 of file OutputFormat.cpp.

23.86.3.2 OPENTREP::OutputFormat::OutputFormat (const char *iFormat*)

Alternative constructor.

Definition at line 61 of file OutputFormat.cpp.

23.86.3.3 OPENTREP::OutputFormat::OutputFormat (const std::string & *iFormat*)

Alternative constructor.

Definition at line 67 of file OutputFormat.cpp.

References getFormat().

23.86.3.4 OPENTREP::OutputFormat::OutputFormat (const OutputFormat & *iOutputFormat*)

Default copy constructor.

Definition at line 27 of file OutputFormat.cpp.

23.86.4 Member Function Documentation

23.86.4.1 const std::string & OPENTREP::OutputFormat::getLabel (const EN_OutputFormat & *iFormat*) [static]

Get the label as a string (e.g., "Short", "Full", "JSON" or "PROTOBUF").

Definition at line 77 of file OutputFormat.cpp.

23.86.4.2 OutputFormat::EN_OutputFormat OPENTREP::OutputFormat::getFormat (const char *iFormatChar*) [static]

Get the format value from parsing a single char (e.g., 'S', 'F', 'J' or 'P').

Definition at line 39 of file OutputFormat.cpp.

References describeLabels(), FULL, JSON, LAST_VALUE, PROTOBUF, and SHORT.

Referenced by OPENTREP::OpenTrepSearcher::generate(), and OPENTREP::OpenTrepSearcher::search().

23.86.4.3 char OPENTREP::OutputFormat::getFormatLabel (const EN_OutputFormat & *iFormat*) [static]

Get the label as a single char (e.g., 'S', 'F', 'J' or 'P').

Definition at line 83 of file OutputFormat.cpp.

23.86.4.4 std::string OPENTREP::OutputFormat::getFormatLabelAsString (const EN_OutputFormat & *iFormat*) [static]

Get the label as a string of a single char (e.g., "S", "F", "J" or "P").

Definition at line 89 of file OutputFormat.cpp.

23.86.4.5 std::string OPENTREP::OutputFormat::describeLabels ( ) [static]

List the labels.

Definition at line 96 of file OutputFormat.cpp.

References LAST_VALUE.

Referenced by getFormat().

23.86.4.6 OutputFormat::EN_OutputFormat OPENTREP::OutputFormat::getFormat ( ) const

Get the enumerated value.

Definition at line 109 of file OutputFormat.cpp.

Referenced by OutputFormat().

23.86.4.7 char OPENTREP::OutputFormat::getFormatAsChar ( ) const

Get the enumerated value as a short string (e.g., 'S', 'F', 'J' or 'P').

Definition at line 114 of file OutputFormat.cpp.

23.86.4.8 std::string OPENTREP::OutputFormat::getFormatAsString ( ) const

Get the enumerated value as a short string (e.g., "S", "F", "J" or "P").

Definition at line 120 of file OutputFormat.cpp.

23.86.4.9 const std::string OPENTREP::OutputFormat::describe ( ) const

Give a description of the structure (e.g., "Short", "Full", "JSON" or "PROTOBUF").

Definition at line 127 of file OutputFormat.cpp.

23.86.4.10 bool OPENTREP::OutputFormat::operator== (const EN_OutputFormat & *iFormat*) const

Comparison operator.

Definition at line 135 of file OutputFormat.cpp.

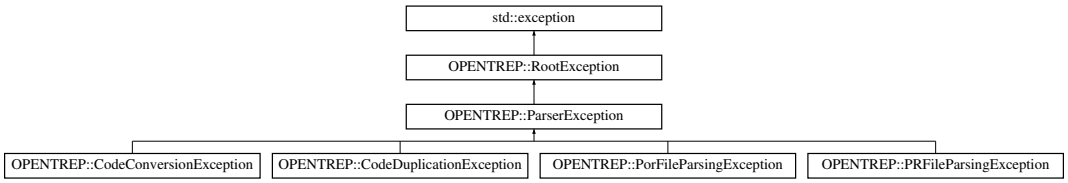
The documentation for this struct was generated from the following files:

- opentrep/OutputFormat.hpp
- opentrep/basic/OutputFormat.cpp

23.87 OPENTREP::ParserException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::ParserException:



Public Member Functions

- ParserException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.87.1 Detailed Description

Parser.

Definition at line 156 of file OPENTREP_exceptions.hpp.

23.87.2 Constructor & Destructor Documentation

23.87.2.1 OPENTREP::ParserException::ParserException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 161 of file OPENTREP_exceptions.hpp.

23.87.3 Member Function Documentation

23.87.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.87.4 Member Data Documentation

23.87.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

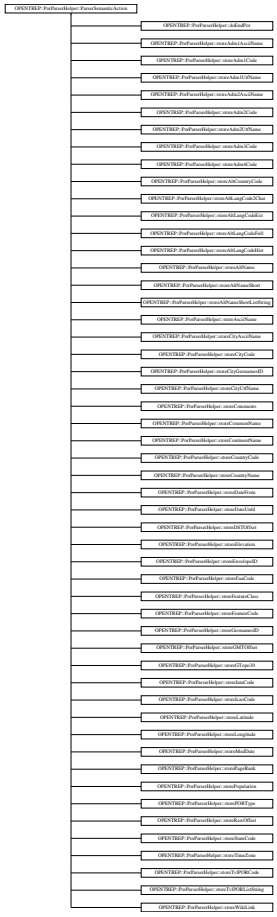
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.88 OPENTREP::PorParserHelper::ParserSemanticAction Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::ParserSemanticAction:



Public Member Functions

- ParserSemanticAction (Location &)

Public Attributes

- Location & _location

23.88.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the Por Parser.

Definition at line 29 of file PORParserHelper.hpp.

23.88.2 Constructor & Destructor Documentation

23.88.2.1 OPENTREP::PorParserHelper::ParserSemanticAction::ParserSemanticAction (Location & *ioLocation*)

Actor Constructor.

Definition at line 31 of file PORParserHelper.cpp.

23.88.3 Member Data Documentation

23.88.3.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

The documentation for this struct was generated from the following files:

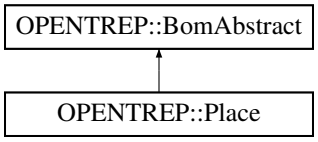
- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.89 OPENTREP::Place Class Reference

Class modelling a place/POR (point of reference).

`#include <opentrep/bom/Place.hpp>`

Inheritance diagram for OPENTREP::Place:



Public Types

- typedef std::set< std::string > StringSet_T

- typedef std::map< const Weight_T, StringSet_T > TermSetMap_T

Public Member Functions

- const Location & getLocation () const
- const LocationKey & getKey () const
- const IATACode_T & getIataCode () const
- const IATAType & getIataType () const
- const GeonamesID_T & getGeonamesID () const
- const IsGeonames_T & isGeonames () const
- const ICAOCode_T & getIcaoCode () const
- const FAACode_T & getFaaCode () const
- const CommonName_T & getCommonName () const
- const ASCIIName_T & getAsciiName () const
- const AltNameShortListString_T & getAltNameShortListString () const
- const EnvelopeID_T & getEnvelopeID () const
- const Date_T & getDateFrom () const
- const Date_T & getDateEnd () const
- const Comment_T & getComment () const
- const CityCode_T & getCityCode () const
- const CityUTFName_T & getCityUtfName () const
- const CityASCIIName_T & getCityAsciiName () const
- const GeonamesID_T & getCityGeonamesID () const
- const StateCode_T & getStateCode () const
- const CountryCode_T & getCountryCode () const
- const AltCountryCode_T & getAltCountryCode () const
- const CountryName_T & getCountryName () const
- const ContinentCode_T & getContinentCode () const
- const ContinentName_T & getContinentName () const
- const TimeZone_T & getTimeZone () const
- const GMTOffset_T & getGMTOffset () const
- const DSTOffset_T & getDSTOffset () const
- const RawOffset_T & getRawOffset () const
- const Latitude_T & getLatitude () const
- const Longitude_T & getLongitude () const
- const FeatureClass_T & getFeatureClass () const
- const FeatureCode_T & getFeatureCode () const
- const Admin1Code_T & getAdmin1Code () const
- const Admin1UTFName_T & getAdmin1UtfName () const
- const Admin1ASCIIName_T & getAdmin1AsciiName () const
- const Admin2Code_T & getAdmin2Code () const
- const Admin2UTFName_T & getAdmin2UtfName () const
- const Admin2ASCIIName_T & getAdmin2AsciiName () const
- const Admin3Code_T & getAdmin3Code () const
- const Admin4Code_T & getAdmin4Code () const
- const Population_T & getPopulation () const
- const Elevation_T & getElevation () const
- const GTopo30_T & getGTopo30 () const
- const PageRank_T & getPageRank () const
- const Date_T & getModificationDate () const
- const WikiLink_T & getWikiLink () const
- const NameMatrix & getNameMatrix () const
- bool getNameList (const LanguageCode_T &iLanguageCode, NameList_T &ioNameList) const

- const std::string & getOriginalKeywords () const
- const std::string & getCorrectedKeywords () const
- const MatchingPercentage_T & getPercentage () const
- const NbOfErrors_T & getEditDistance () const
- const NbOfErrors_T & getAllowableEditDistance () const
- const RawDataString_T & getRawDataString () const
- const XapianDocID_T & getDocID () const
- const PlaceOrderedList_T & getExtraPlaceList () const
- const PlaceOrderedList_T & getAlternatePlaceList () const
- const TermSetMap_T & getTermSetMap () const
- StringSet_T getTermSet (const Weight_T &) const
- const StringSet_T & getSpellingSet () const
- const StringSet_T & getStemmingSet () const
- const StringSet_T & getSynonymSet () const
- void setLocation (const Location &iLocation)
- void setKey (const LocationKey &iKey)
- void setIataCode (const std::string &iIataCode)
- void setIataType (const std::string &iIATAType)
- void setGeonamesID (const GeonamesID_T &iGeonamesID)
- void setIcaoCode (const std::string &iIcaoCode)
- void setFaaCode (const std::string &iFaaCode)
- void setCommonName (const std::string &iName)
- void setAsciiName (const std::string &iName)
- void setAltNameShortListString (const std::string &iNameListString)
- void setEnvelopeID (const EnvelopeID_T &iEnvelopeID)
- void setDateFrom (const Date_T &iDate)
- void setDateEnd (const Date_T &iDate)
- void setComment (const std::string &iComment)
- void setCityCode (const std::string &iCityCode)
- void setCityUtfName (const std::string &iCityUtfName)
- void setCityAsciiName (const std::string &iCityAsciiName)
- void setCityGeonamesID (const GeonamesID_T &iGeonamesID)
- void setStateCode (const std::string &iStateCode)
- void setCountryCode (const std::string &iCountryCode)
- void setAltCountryCode (const std::string &iCountryCode)
- void setCountryName (const std::string &iCountryName)
- void setContinentCode (const std::string &iContinentCode)
- void setContinentName (const std::string &iContinentName)
- void setTimeZone (const std::string &iTimeZone)
- void setGMTOffset (const GMTOffset_T &iOffset)
- void setDSTOffset (const DSTOffset_T &iOffset)
- void setRawOffset (const RawOffset_T &iOffset)
- void setLatitude (const Latitude_T &iLatitude)
- void setLongitude (const Longitude_T &iLongitude)
- void setFeatureClass (const std::string &iFeatClass)
- void setFeatureCode (const std::string &iFeatCode)
- void setAdmin1Code (const std::string &iAdminCode)
- void setAdmin1UtfName (const std::string &iAdminName)
- void setAdmin1AsciiName (const std::string &iAdminName)
- void setAdmin2Code (const std::string &iAdminCode)
- void setAdmin2UtfName (const std::string &iAdminName)
- void setAdmin2AsciiName (const std::string &iAdminName)
- void setAdmin3Code (const std::string &iAdminCode)
- void setAdmin4Code (const std::string &iAdminCode)
- void setPopulation (const Population_T &iPopulation)

- void setElevation (const Elevation_T &iElevation)
- void setGTopo30 (const GTopo30_T &iGTopo30)
- void setPageRank (const PageRank_T &iPageRank)
- void setModificationDate (const Date_T &iModDate)
- void setWikiLink (const std::string &iWikiLink)
- void setOriginalKeywords (const std::string &iOriginalKeywords)
- void setCorrectedKeywords (const std::string &iCorrectedKeywords)
- void setPercentage (const MatchingPercentage_T &iPercentage)
- void setEditDistance (const NbOfErrors_T &iEditDistance)
- void setAllowableEditDistance (const NbOfErrors_T &iAllowableEditDistance)
- void setRawDataString (const std::string &iRawDataString)
- void setDocID (const XapianDocID_T &iDocID)
- bool addTermSet (const Weight_T &, const StringSet_T &)
- void addName (const LanguageCode_T &iLanguageCode, const std::string &iName)
- void resetMatrix ()
- void resetIndexSets ()
- const Location & completeLocation ()
- void addNameToXapianSets (const Weight_T &, const LocationName_T &, const FeatureCode_T &, const CityUTFName_T &, const CityASCIIName_T &, const Admin1UTFName_T &, const Admin1ASCIIName_T &, const Admin2UTFName_T &, const Admin2ASCIIName_T &, const StateCode_T &, const CountryCode_T &, const CountryName_T &, const ContinentName_T &, const OTransliterator &)
- void buildIndexSets (const OTransliterator &)
- void addNameToXapianSets (const Weight_T &, const std::string &iBaseName, const FeatureCode_T &)
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- std::string describeKey () const
- std::string describeShortKey () const
- std::string toString () const
- std::string toShortString () const
- std::string describeSets () const
- std::string display () const
- std::string shortDisplay () const

Friends

- class FacWorld
- class FacPlace
- class FacPlaceHolder
- class DbaPlace

23.89.1 Detailed Description

Class modelling a place/POR (point of reference).

Definition at line 28 of file Place.hpp.

23.89.2 Member Typedef Documentation

23.89.2.1 typedef std::set<std::string> OPENTREP::Place::StringSet_T

(STL) Set of strings, to be added in the Xapian database (for indexing, spelling, stemming, synonyms, etc).

Definition at line 39 of file Place.hpp.

23.89.2.2 typedef std::map<const Weight_T, StringSet_T> OPENTREP::Place::TermSetMap_T

Definition at line 40 of file Place.hpp.

23.89.3 Member Function Documentation

23.89.3.1 const Location& OPENTREP::Place::getLocation ( ) const [inline]

Get the underlying Location structure, which is an extract of the Place object.

Returns

const Location& The underlying Location structure.

Definition at line 51 of file Place.hpp.

Referenced by completeLocation().

23.89.3.2 const LocationKey& OPENTREP::Place::getKey ( ) const [inline]

Get the primary key (IATA and ICAO codes, Geonames ID) of the place.

Definition at line 58 of file Place.hpp.

References OPENTREP::Location::getKey().

Referenced by OPENTREP::DBManager::buildSQLDB(), OPENTREP::Result::fillPlace(), and soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.3 const IATACode_T& OPENTREP::Place::getIataCode ( ) const [inline]

Get the IATA code.

Definition at line 65 of file Place.hpp.

References OPENTREP::Location::getIataCode().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base(), and OPENTREP::DBManager::updatePlaceInDB().

23.89.3.4 const IATAType& OPENTREP::Place::getIataType ( ) const [inline]

Get the IATA location type (e.g., A for airport).

Definition at line 72 of file Place.hpp.

References OPENTREP::Location::getIataType().

23.89.3.5 const GeonamesID_T& OPENTREP::Place::getGeonamesID ( ) const [inline]

Get the Geonames ID.

Definition at line 79 of file Place.hpp.

References OPENTREP::Location::getGeonamesID().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.6 const IsGeonames_T& OPENTREP::Place::isGeonames ( ) const [inline]

State whether that POR is referenced by Geonames.

Definition at line 86 of file Place.hpp.

References OPENTREP::Location::isGeonames().

23.89.3.7 const ICAOCode_T& OPENTREP::Place::getIcaoCode ( ) const [inline]

Get the ICAO code.

Definition at line 93 of file Place.hpp.

References OPENTREP::Location::getIcaoCode().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.8 const FAACode_T& OPENTREP::Place::getFaaCode ( ) const [inline]

Get the FAA code.

Definition at line 100 of file Place.hpp.

References OPENTREP::Location::getFaaCode().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.9 const CommonName_T& OPENTREP::Place::getCommonName ( ) const [inline]

Get the common name (usually in American English, but not necessarily in ASCII).

Definition at line 108 of file Place.hpp.

References OPENTREP::Location::getCommonName().

23.89.3.10 const ASCIIName_T& OPENTREP::Place::getAsciiName ( ) const [inline]

Get the ASCII name (not necessarily in English).

Definition at line 115 of file Place.hpp.

References OPENTREP::Location::getAsciiName().

23.89.3.11 const AltNameShortListString_T& OPENTREP::Place::getAltNameShortListString ( ) const [inline]

Get the short list of alternate names (without language codes).

Definition at line 122 of file Place.hpp.

References OPENTREP::Location::getAltNameShortListString().

23.89.3.12 const EnvelopeID_T& OPENTREP::Place::getEnvelopeID ( ) const [inline]

Get the envelope ID.

Definition at line 129 of file Place.hpp.

References OPENTREP::Location::getEnvelopeID().

23.89.3.13 const Date_T& OPENTREP::Place::getDateFrom ( ) const [inline]

Get the beginning date of the validity period

Definition at line 136 of file Place.hpp.

References OPENTREP::Location::getDateFrom().

23.89.3.14 const Date_T& OPENTREP::Place::getDateEnd ( ) const [inline]

Get the end date of the validity period

Definition at line 143 of file Place.hpp.

References OPENTREP::Location::getDateEnd().

23.89.3.15 const Comment_T& OPENTREP::Place::getComment ( ) const [inline]

Get the commentaries.

Definition at line 150 of file Place.hpp.

References OPENTREP::Location::getComment().

23.89.3.16 const CityCode_T& OPENTREP::Place::getCityCode ( ) const [inline]

Get the related/served IATA city code.

Definition at line 157 of file Place.hpp.

References OPENTREP::Location::getCityCode().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.17 const CityUTFName_T& OPENTREP::Place::getCityUtfName ( ) const [inline]

Get the city name in UTF8.

Definition at line 164 of file Place.hpp.

References OPENTREP::Location::getCityUtfName().

23.89.3.18 const CityASCIIName_T& OPENTREP::Place::getCityAsciiName ( ) const [inline]

Get the city name in ASCII (not necessarily in English).

Definition at line 171 of file Place.hpp.

References OPENTREP::Location::getCityAsciiName().

23.89.3.19 const GeonamesID_T& OPENTREP::Place::getCityGeonamesID ( ) const [inline]

Get the city Geonames ID (may be 0 if unknown).

Definition at line 178 of file Place.hpp.

References OPENTREP::Location::getCityGeonamesID().

23.89.3.20 const StateCode_T& OPENTREP::Place::getStateCode ( ) const [inline]

Get the state code.

Definition at line 185 of file Place.hpp.

References OPENTREP::Location::getStateCode().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.21 const CountryCode_T& OPENTREP::Place::getCountryCode ( ) const [inline]

Get the country code.

Definition at line 192 of file Place.hpp.

References OPENTREP::Location::getCountryCode().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.22 const AltCountryCode_T& OPENTREP::Place::getAltCountryCode ( ) const [inline]

Get the alternative country code.

Definition at line 199 of file Place.hpp.

References OPENTREP::Location::getAltCountryCode().

23.89.3.23 const CountryName_T& OPENTREP::Place::getCountryName ( ) const [inline]

Get the country name.

Definition at line 206 of file Place.hpp.

References OPENTREP::Location::getCountryName().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.24 const ContinentCode_T& OPENTREP::Place::getContinentCode ( ) const [inline]

Get the continent code.

Definition at line 213 of file Place.hpp.

References OPENTREP::Location::getContinentCode().

23.89.3.25 const ContinentName_T& OPENTREP::Place::getContinentName ( ) const [inline]

Get the continent name.

Definition at line 220 of file Place.hpp.

References OPENTREP::Location::getContinentName().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.26 const TimeZone_T& OPENTREP::Place::getTimeZone ( ) const [inline]

Get the time-zone.

Definition at line 227 of file Place.hpp.

References OPENTREP::Location::getTimeZone().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.27 const GMTOffset_T& OPENTREP::Place::getGMTOffset ( ) const [inline]

Get the GMT offset (e.g., 1)

Definition at line 234 of file Place.hpp.

References OPENTREP::Location::getGMTOffset().

23.89.3.28 const DSTOffset_T& OPENTREP::Place::getDSTOffset ( ) const [inline]

Get the DST offset (e.g., 2)

Definition at line 241 of file Place.hpp.

References OPENTREP::Location::getDSTOffset().

23.89.3.29 const RawOffset_T& OPENTREP::Place::getRawOffset ( ) const [inline]

Get the raw offset (e.g., -1)

Definition at line 248 of file Place.hpp.

References OPENTREP::Location::getRawOffset().

23.89.3.30 const Latitude_T& OPENTREP::Place::getLatitude ( ) const [inline]

Get the geographical latitude.

Definition at line 255 of file Place.hpp.

References OPENTREP::Location::getLatitude().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.31 const Longitude_T& OPENTREP::Place::getLongitude ( ) const [inline]

Get the geographical longitude.

Definition at line 262 of file Place.hpp.

References OPENTREP::Location::getLongitude().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.32 const FeatureClass_T& OPENTREP::Place::getFeatureClass ( ) const [inline]

Get the feature class (e.g., S for spot).

Definition at line 269 of file Place.hpp.

References OPENTREP::Location::getFeatureClass().

23.89.3.33 const FeatureCode_T& OPENTREP::Place::getFeatureCode ( ) const [inline]

Get the feature code (e.g., AIRP for airport).

Definition at line 276 of file Place.hpp.

References OPENTREP::Location::getFeatureCode().

23.89.3.34 const Admin1Code_T& OPENTREP::Place::getAdmin1Code ( ) const [inline]

Get the administrative code of rank 1 (e.g., IL for Illinois).

Definition at line 283 of file Place.hpp.

References OPENTREP::Location::getAdmin1Code().

23.89.3.35 const Admin1UTFName_T& OPENTREP::Place::getAdmin1UtfName ( ) const [inline]

Get the administrative UTF8 name of rank 1 (e.g., Illinois).

Definition at line 290 of file Place.hpp.

References OPENTREP::Location::getAdmin1UtfName().

23.89.3.36 const Admin1ASCIIName_T& OPENTREP::Place::getAdmin1AsciiName ( ) const [inline]

Get the administrative ASCII name of rank 1 (e.g., Illinois).

Definition at line 297 of file Place.hpp.

References OPENTREP::Location::getAdmin1AsciiName().

23.89.3.37 const Admin2Code_T& OPENTREP::Place::getAdmin2Code ( ) const [inline]

Get the administrative code of rank 2.

Definition at line 304 of file Place.hpp.

References OPENTREP::Location::getAdmin2Code().

23.89.3.38 const Admin2UTFName_T& OPENTREP::Place::getAdmin2UtfName ( ) const [inline]

Get the administrative UTF8 name of rank 2 (e.g., Alpes-Maritimes).

Definition at line 311 of file Place.hpp.

References OPENTREP::Location::getAdmin2UtfName().

23.89.3.39 const Admin2ASCIIName_T& OPENTREP::Place::getAdmin2AsciiName ( ) const [inline]

Get the administrative ASCII name of rank 2 (e.g., Alpes-Maritimes).

Definition at line 318 of file Place.hpp.

References OPENTREP::Location::getAdmin2AsciiName().

23.89.3.40 const Admin3Code_T& OPENTREP::Place::getAdmin3Code ( ) const [inline]

Get the administrative code of rank 3.

Definition at line 325 of file Place.hpp.

References OPENTREP::Location::getAdmin3Code().

23.89.3.41 const Admin4Code_T& OPENTREP::Place::getAdmin4Code ( ) const [inline]

Get the administrative code of rank 4.

Definition at line 332 of file Place.hpp.

References OPENTREP::Location::getAdmin4Code().

23.89.3.42 const Population_T& OPENTREP::Place::getPopulation ( ) const [inline]

Get the population (number of inhabitants).

Definition at line 339 of file Place.hpp.

References OPENTREP::Location::getPopulation().

23.89.3.43 const Elevation_T& OPENTREP::Place::getElevation ( ) const [inline]

Get the elevation.

Definition at line 346 of file Place.hpp.

References OPENTREP::Location::getElevation().

23.89.3.44 const GTopo30_T& OPENTREP::Place::getGTopo30 ( ) const [inline]

Get the GTopo30.

Definition at line 353 of file Place.hpp.

References OPENTREP::Location::getGTopo30().

23.89.3.45 const PageRank_T& OPENTREP::Place::getPageRank ( ) const [inline]

Get the PageRank/importance.

Definition at line 360 of file Place.hpp.

References OPENTREP::Location::getPageRank().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.46 const Date_T& OPENTREP::Place::getModificationDate ( ) const [inline]

Get the modification date.

Definition at line 367 of file Place.hpp.

References OPENTREP::Location::getModificationDate().

23.89.3.47 const WikiLink_T& OPENTREP::Place::getWikiLink ( ) const [inline]

Get the Wikipedia link.

Definition at line 374 of file Place.hpp.

References OPENTREP::Location::getWikiLink().

Referenced by soci::type_conversion< OPENTREP::Place >::to_base().

23.89.3.48 const NameMatrix& OPENTREP::Place::getNameMatrix ( ) const [inline]

Get the map of name lists.

Definition at line 381 of file Place.hpp.

References OPENTREP::Location::getNameMatrix().

23.89.3.49 bool OPENTREP::Place::getNameList (const LanguageCode_T & *iLanguageCode,* NameList_T & *ioNameList*) const [inline]

Get, for a given language (code), the corresponding list of names.

Parameters

const	LanguageCode_T& Language code.
NameList_T&	Empty list of names, which will be filled by the method if a list exists for that language code.

Returns

bool Whether or not such a list exists for the given language.

Definition at line 394 of file Place.hpp.

References OPENTREP::Location::getNameList().

Referenced by completeLocation().

23.89.3.50 const std::string& OPENTREP::Place::getOriginalKeywords ( ) const [inline]

Get the original keywords.

Definition at line 402 of file Place.hpp.

References OPENTREP::Location::getOriginalKeywords().

23.89.3.51 const std::string& OPENTREP::Place::getCorrectedKeywords ( ) const [inline]

Get the corrected keywords.

Definition at line 409 of file Place.hpp.

References OPENTREP::Location::getCorrectedKeywords().

23.89.3.52 const MatchingPercentage_T& OPENTREP::Place::getPercentage ( ) const [inline]

Get the matching percentage.

Definition at line 416 of file Place.hpp.

References OPENTREP::Location::getPercentage().

23.89.3.53 const NbOfErrors_T& OPENTREP::Place::getEditDistance ( ) const [inline]

Get the allowed edit distance/error.

Definition at line 423 of file Place.hpp.

References OPENTREP::Location::getEditDistance().

23.89.3.54 const NbOfErrors_T& OPENTREP::Place::getAllowableEditDistance ( ) const [inline]

Get the maximal allowable edit distance/error, with which the matching has been made.

Definition at line 431 of file Place.hpp.

References OPENTREP::Location::getAllowableEditDistance().

23.89.3.55 const RawDataString_T& OPENTREP::Place::getRawDataString ( ) const [inline]

Get the raw data string, as stored and retrieved by Xapian.

Definition at line 438 of file Place.hpp.

References OPENTREP::Location::getRawDataString().

23.89.3.56 const XapianDocID_T& OPENTREP::Place::getDocID ( ) const [inline]

Get the Xapian document ID.

Definition at line 445 of file Place.hpp.

Referenced by OPENTREP::FacWorld::linkPlaceToWorld(), and OPENTREP::DBManager::updatePlaceInDB().

23.89.3.57 const PlaceOrderedList_T& OPENTREP::Place::getExtraPlaceList ( ) const [inline]

Get the list of extra matching (similar) places.

Definition at line 452 of file Place.hpp.

23.89.3.58 const PlaceOrderedList_T& OPENTREP::Place::getAlternatePlaceList ( ) const [inline]

Get the list of alternate matching (less similar) places.

Definition at line 459 of file Place.hpp.

23.89.3.59 const TermSetMap_T& OPENTREP::Place::getTermSetMap ( ) const [inline]

Get the (STL) map of term set (for the Xapian index).

Definition at line 466 of file Place.hpp.

Referenced by OPENTREP::addToXapian().

23.89.3.60 Place::StringSet_T OPENTREP::Place::getTermSet (const Weight_T & *iWeight*) const

Get the (STL) set of terms (for the Xapian index for the given weight).

Parameters

const	Weight_T& Weight with which the terms should be indexed.

Definition at line 226 of file Place.cpp.

Referenced by addNameToXapianSets(), and buildIndexSets().

23.89.3.61 const StringSet_T& OPENTREP::Place::getSpellingSet ( ) const [inline]

Get the (STL) set of spelling terms (for the Xapian spelling dictionary).

Definition at line 480 of file Place.hpp.

Referenced by OPENTREP::addToXapian().

23.89.3.62 const StringSet_T& OPENTREP::Place::getStemmingSet ( ) const [inline]

Get the (STL) set of stemming terms (for the Xapian stemming dictionary).

Definition at line 487 of file Place.hpp.

23.89.3.63 const StringSet_T& OPENTREP::Place::getSynonymSet ( ) const [inline]

Get the (STL) set of synonym terms (for the Xapian synonym dictionary).

Definition at line 494 of file Place.hpp.

23.89.3.64 void OPENTREP::Place::setLocation (const Location & *iLocation*) [inline]

Set the full Location structure.

Definition at line 504 of file Place.hpp.

23.89.3.65 void OPENTREP::Place::setKey (const LocationKey & *iKey*) [inline]

Set the primary key (IATA and ICAO codes, Geonames ID) of the place.

Definition at line 511 of file Place.hpp.

References OPENTREP::Location::setKey().

23.89.3.66 void OPENTREP::Place::setIataCode (const std::string & *ilataCode*) [inline]

Set the IATA code.

Definition at line 518 of file Place.hpp.

References OPENTREP::Location::setIataCode().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.67 void OPENTREP::Place::setIataType (const std::string & *iIATAType*) [inline]

Set the IATA location type (e.g., A for airport).

Definition at line 525 of file Place.hpp.

References OPENTREP::Location::setIataType().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.68 void OPENTREP::Place::setGeonamesID (const GeonamesID_T & *iGeonamesID*) [inline]

Set the Geonames ID.

Definition at line 532 of file Place.hpp.

References OPENTREP::Location::setGeonamesID().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.69 void OPENTREP::Place::setIcaoCode (const std::string & *iIcaoCode*) [inline]

Set the ICAO code.

Definition at line 539 of file Place.hpp.

References OPENTREP::Location::setIcaoCode().

23.89.3.70 void OPENTREP::Place::setFaaCode (const std::string & *iFaaCode*) [inline]

Set the FAA code.

Definition at line 546 of file Place.hpp.

References OPENTREP::Location::setFaaCode().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.71 void OPENTREP::Place::setCommonName (const std::string & *iName*) [inline]

Set the common name (usually in American English, but not necessarily in ASCII).

Definition at line 554 of file Place.hpp.

References OPENTREP::Location::setCommonName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.72 void OPENTREP::Place::setAsciiName (const std::string & *iName*) [inline]

Set the ASCII name (not necessarily in English).

Definition at line 561 of file Place.hpp.

References OPENTREP::Location::setAsciiName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.73 void OPENTREP::Place::setAltNameShortListString (const std::string & *iNameListString*) [inline]

Set the short list of alternate names (without language codes).

Definition at line 568 of file Place.hpp.

References OPENTREP::Location::setAltNameShortListString().

23.89.3.74 void OPENTREP::Place::setEnvelopeID (const EnvelopeID_T & *iEnvelopeID*) [inline]

Set the envelope ID.

Definition at line 575 of file Place.hpp.

References OPENTREP::Location::setEnvelopeID().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.75 void OPENTREP::Place::setDateFrom (const Date_T & *iDate*) [inline]

Set the beginning date of the validity period

Definition at line 582 of file Place.hpp.

References OPENTREP::Location::setDateFrom().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.76 void OPENTREP::Place::setDateEnd (const Date_T & *iDate*) [inline]

Set the end date of the validity period

Definition at line 589 of file Place.hpp.

References OPENTREP::Location::setDateEnd().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.77 void OPENTREP::Place::setComment (const std::string & *iComment*) [inline]

Set the commentaries.

Definition at line 596 of file Place.hpp.

References OPENTREP::Location::setComment().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.78 void OPENTREP::Place::setCityCode (const std::string & *iCityCode*) [inline]

Set the related/served IATA city code.

Definition at line 603 of file Place.hpp.

References OPENTREP::Location::setCityCode().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.79 void OPENTREP::Place::setCityUtfName (const std::string & *iCityUtfName*) [inline]

Set the city name in UTF8.

Definition at line 610 of file Place.hpp.

References OPENTREP::Location::setCityUtfName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.80 void OPENTREP::Place::setCityAsciiName (const std::string & *iCityAsciiName*) [inline]

Set the city name in ASCII (not necessarily in English).

Definition at line 617 of file Place.hpp.

References OPENTREP::Location::setCityAsciiName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.81 void OPENTREP::Place::setCityGeonamesID (const GeonamesID_T & *iGeonamesID*) [inline]

Set the city Geonames ID (may be 0 if unknown).

Definition at line 624 of file Place.hpp.

References OPENTREP::Location::setCityGeonamesID().

23.89.3.82 void OPENTREP::Place::setStateCode (const std::string & *iStateCode*) [inline]

Set the state code.

Definition at line 631 of file Place.hpp.

References OPENTREP::Location::setStateCode().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.83 void OPENTREP::Place::setCountryCode (const std::string & *iCountryCode*) [inline]

Set the country code.

Definition at line 638 of file Place.hpp.

References OPENTREP::Location::setCountryCode().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.84 void OPENTREP::Place::setAltCountryCode (const std::string & *iCountryCode*) [inline]

Set the alternative country code.

Definition at line 645 of file Place.hpp.

References OPENTREP::Location::setAltCountryCode().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.85 void OPENTREP::Place::setCountryName (const std::string & *iCountryName*) [inline]

Set the country name.

Definition at line 652 of file Place.hpp.

References OPENTREP::Location::setCountryName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.86 void OPENTREP::Place::setContinentCode (const std::string & *iContinentCode*) [inline]

Set the continent code.

Definition at line 659 of file Place.hpp.

References OPENTREP::Location::setContinentCode().

23.89.3.87 void OPENTREP::Place::setContinentName (const std::string & *iContinentName*) [inline]

Set the continent name.

Definition at line 666 of file Place.hpp.

References OPENTREP::Location::setContinentName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.88 void OPENTREP::Place::setTimeZone (const std::string & *iTimeZone*) [inline]

Set the time-zone.

Definition at line 673 of file Place.hpp.

References OPENTREP::Location::setTimeZone().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.89 void OPENTREP::Place::setGMTOffset (const GMTOffset_T & *iOffset*) [inline]

Set the GMT offset (e.g., 1)

Definition at line 680 of file Place.hpp.

References OPENTREP::Location::setGMTOffset().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.90 void OPENTREP::Place::setDSTOffset (const DSTOffset_T & *iOffset*) [inline]

Set the DST offset (e.g., 2)

Definition at line 687 of file Place.hpp.

References OPENTREP::Location::setDSTOffset().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.91 void OPENTREP::Place::setRawOffset (const RawOffset_T & *iOffset*) [inline]

Set the raw offset (e.g., 1)

Definition at line 694 of file Place.hpp.

References OPENTREP::Location::setRawOffset().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.92 void OPENTREP::Place::setLatitude (const Latitude_T & *iLatitude*) [inline]

Set the geographical latitude.

Definition at line 701 of file Place.hpp.

References OPENTREP::Location::setLatitude().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.93 void OPENTREP::Place::setLongitude (const Longitude_T & *iLongitude*) [inline]

Set the geographical longitude.

Definition at line 708 of file Place.hpp.

References OPENTREP::Location::setLongitude().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.94 void OPENTREP::Place::setFeatureClass (const std::string & *iFeatClass*) [inline]

Set the feature class (e.g., S for spot).

Definition at line 715 of file Place.hpp.

References OPENTREP::Location::setFeatureClass().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.95 void OPENTREP::Place::setFeatureCode (const std::string & *iFeatCode*) [inline]

Set the feature code (e.g., AIRP for airport).

Definition at line 722 of file Place.hpp.

References OPENTREP::Location::setFeatureCode().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.96 void OPENTREP::Place::setAdmin1Code (const std::string & *iAdminCode*) [inline]

Set the administrative code of rank 1 (e.g., IL for Illinois).

Definition at line 729 of file Place.hpp.

References OPENTREP::Location::setAdmin1Code().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.97 void OPENTREP::Place::setAdmin1UtfName (const std::string & *iAdminName*) [inline]

Set the administrative UTF8 name of rank 1 (e.g., Illinois).

Definition at line 736 of file Place.hpp.

References OPENTREP::Location::setAdmin1UtfName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.98 void OPENTREP::Place::setAdmin1AsciiName (const std::string & *iAdminName*) [inline]

Set the administrative ASCII name of rank 1 (e.g., Illinois).

Definition at line 743 of file Place.hpp.

References OPENTREP::Location::setAdmin1AsciiName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.99 void OPENTREP::Place::setAdmin2Code (const std::string & *iAdminCode*) [inline]

Set the administrative code of rank 2.

Definition at line 750 of file Place.hpp.

References OPENTREP::Location::setAdmin2Code().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.100 void OPENTREP::Place::setAdmin2UtfName (const std::string & *iAdminName*) [inline]

Set the administrative UTF8 name of rank 2 (e.g., Alpes-Maritimes).

Definition at line 757 of file Place.hpp.

References OPENTREP::Location::setAdmin2UtfName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.101 void OPENTREP::Place::setAdmin2AsciiName (const std::string & *iAdminName*) [inline]

Set the administrative ASCII name of rank 2 (e.g., Alpes-Maritimes).

Definition at line 764 of file Place.hpp.

References OPENTREP::Location::setAdmin2AsciiName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.102 void OPENTREP::Place::setAdmin3Code (const std::string & *iAdminCode*) [inline]

Set the administrative code of rank 3.

Definition at line 771 of file Place.hpp.

References OPENTREP::Location::setAdmin3Code().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.103 void OPENTREP::Place::setAdmin4Code (const std::string & *iAdminCode*) [inline]

Set the administrative code of rank 4.

Definition at line 778 of file Place.hpp.

References OPENTREP::Location::setAdmin4Code().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.104 void OPENTREP::Place::setPopulation (const Population_T & *iPopulation*) [inline]

Set the population (number of inhabitants).

Definition at line 785 of file Place.hpp.

References OPENTREP::Location::setPopulation().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.105 void OPENTREP::Place::setElevation (const Elevation_T & *iElevation*) [inline]

Set the elevation.

Definition at line 792 of file Place.hpp.

References OPENTREP::Location::setElevation().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.106 void OPENTREP::Place::setGTopo30 (const GTopo30_T & *iGTopo30*) [inline]

Set the GTopo30.

Definition at line 799 of file Place.hpp.

References OPENTREP::Location::setGTopo30().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.107 void OPENTREP::Place::setPageRank (const PageRank_T & *iPageRank*) [inline]

Set the PageRank.

Definition at line 806 of file Place.hpp.

References OPENTREP::Location::setPageRank().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.108 void OPENTREP::Place::setModificationDate (const Date_T & *iModDate*) [inline]

Set the modification date.

Definition at line 813 of file Place.hpp.

References OPENTREP::Location::setModificationDate().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.109 void OPENTREP::Place::setWikiLink (const std::string & *iWikiLink*) [inline]

Set the Wikipedia link.

Definition at line 820 of file Place.hpp.

References OPENTREP::Location::setWikiLink().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.110 void OPENTREP::Place::setOriginalKeywords (const std::string & *iOriginalKeywords*) [inline]

Set the original keywords.

Definition at line 827 of file Place.hpp.

References OPENTREP::Location::setOriginalKeywords().

Referenced by OPENTREP::Result::fillPlace().

23.89.3.111 void OPENTREP::Place::setCorrectedKeywords (const std::string & *iCorrectedKeywords*) [inline]

Set the corrected keywords.

Definition at line 834 of file Place.hpp.

References OPENTREP::Location::setCorrectedKeywords().

Referenced by OPENTREP::Result::fillPlace().

23.89.3.112 void OPENTREP::Place::setPercentage (const MatchingPercentage_T & *iPercentage*) [inline]

Set the Xapian matching percentage.

Definition at line 841 of file Place.hpp.

References OPENTREP::Location::setPercentage().

Referenced by OPENTREP::Result::fillPlace().

23.89.3.113 void OPENTREP::Place::setEditDistance (const NbOfErrors_T & *iEditDistance*) [inline]

Set the allowed edit distance/error.

Definition at line 848 of file Place.hpp.

References OPENTREP::Location::setEditDistance().

Referenced by OPENTREP::Result::fillPlace().

23.89.3.114 void OPENTREP::Place::setAllowableEditDistance (const NbOfErrors_T & *iAllowableEditDistance*) [inline]

Set the maxiaml allowable edit distance/error, with which the matching has been made.

Definition at line 856 of file Place.hpp.

References OPENTREP::Location::setAllowableEditDistance().

Referenced by OPENTREP::Result::fillPlace().

23.89.3.115 void OPENTREP::Place::setRawDataString (const std::string & *iRawDataString*) [inline]

Set the raw data string, as stored and retrieved by Xapian.

Definition at line 863 of file Place.hpp.

References OPENTREP::Location::setRawDataString().

23.89.3.116 void OPENTREP::Place::setDocID (const XapianDocID_T & *iDocID*) [inline]

Set the Xapian document ID.

Definition at line 870 of file Place.hpp.

Referenced by OPENTREP::Result::fillPlace().

23.89.3.117 bool OPENTREP::Place::addTermSet (const Weight_T & *iWeight,* const StringSet_T & *iTermSet*)

Set the (STL) set of terms (for the Xapian index for the given weight).

If no set of terms existed for the given weight, it is inserted in the (STL) map. Otherwise, the existing set of terms is just added to the existing set of terms.

Parameters

const	Weight_T& Weight with which the terms should be indexed.
const	StringSet_T& Set of terms (to be indexed by Xapian).

Definition at line 240 of file Place.cpp.

Referenced by addNameToXapianSets(), and buildIndexSets().

23.89.3.118 void OPENTREP::Place::addName (const LanguageCode_T & *iLanguageCode,* const std::string & *iName*) [inline]

Add a name for the place.

Parameters

const	LanguageCode_T& Language in which to add the name.
const	std::string& Name to be added.

Definition at line 896 of file Place.hpp.

References OPENTREP::Location::addName().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.119 void OPENTREP::Place::resetMatrix ( ) [inline]

Reset the map of name lists.

Definition at line 904 of file Place.hpp.

References OPENTREP::Location::resetMatrix().

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.120 void OPENTREP::Place::resetIndexSets ( )

Reset the index/spelling (STL) sets.

Definition at line 218 of file Place.cpp.

Referenced by soci::type_conversion< OPENTREP::Place >::from_base().

23.89.3.121 const Location & OPENTREP::Place::completeLocation ( )

Get the underlying Location structure, which is an extract of the Place object. That (Location) structure is passed back to the caller of the service.

Returns

Location The Location structure just created and filled with the parameters of the Place object.

Definition at line 699 of file Place.cpp.

References OPENTREP::Location::addAlternateLocation(), OPENTREP::Location::addExtraLocation(), getLocation(), getNameList(), OPENTREP_LOG_ERROR, and toShortString().

Referenced by OPENTREP::PlaceHolder::createLocations().

23.89.3.122 void OPENTREP::Place::addNameToXapianSets (const Weight_T & *iWeight,* const LocationName_T & *iLocationName,* const FeatureCode_T & *iFeatureCode,* const CityUTFName_T & *iCityUtfName,* const CityASCIIName_T & *iCityAsciiName,* const Admin1UTFName_T & *iAdm1UtfName,* const Admin1ASCIIName_T & *iAdm1AsciiName,* const Admin2UTFName_T & *iAdm2UtfName,* const Admin2ASCIIName_T & *iAdm2AsciiName,* const StateCode_T & *iStateCode,* const CountryCode_T & *iCountryCode,* const CountryName_T & *iCountryName,* const ContinentName_T & *iContinentName,* const OTransliterator & *iTransliterator*)

Add the given name to the Xapian index with the given weight, and add that name to the spelling dictionary. Tokenise and re-assemble the given name, so as to replace all the punctuations and other separators by mere spaces. For instance, "Paris/FR/Gare" is transformed into "Paris FR Gare".

Parameters

const	Weight_T& The weight with which the terms should be indexed
const	LocationName_T& Name of the POR (point of reference)
const	FeatureCode_T& Geonames feature code
const	CityUTFName_T& UTF8 name of the served city
const	CityASCIIName_T& ASCII name of the served city
const	Admin1UTFName_T& UTF8 Name of the administrative level 1 of the POR
const	Admin1ASCIIName_T& ASCII Name of the administrative level 1 of the POR
const	Admin2UTFName_T& UTF8 Name of the administrative level 2 of the POR
const	Admin2ASCIIName_T& ASCII Name of the administrative level 2 of the POR
const	StateCode_T& Code of the state of the POR
const	CountryCode_T& ISO code of the country of the POR
const	CountryName_T& Name of the country of the POR
const	ContinentName_T& Name of the continent of the POR
const	OTransliterator& Unicode transliterator

Definition at line 296 of file Place.cpp.

References addTermSet(), OPENTREP::createStringFromWordList(), getTermSet(), OPENTREP::OTransliterator::normalise(), and OPENTREP::tokeniseStringIntoWordList().

Referenced by buildIndexSets().

23.89.3.123 void OPENTREP::Place::buildIndexSets (const OTransliterator & *iTransliterator*)

Build the (STL) sets of (Xapian-related) terms, spelling, synonyms, etc.

Parameters

const	OTransliterator& Unicode transliterator

Definition at line 445 of file Place.cpp.

References OPENTREP::WordCombinationHolder::_list, addNameToXapianSets(), addTermSet(), OPENTREP::Location::getAdmin1AsciiName(), OPENTREP::Location::getAdmin1Code(), OPENTREP::Location::getAdmin1-

UtfName(), OPENTREP::Location::getAdmin2AsciiName(), OPENTREP::Location::getAdmin2UtfName(), OPENTREP::Location::getAsciiName(), OPENTREP::Location::getCityAsciiName(), OPENTREP::Location::getCityCode(), OPENTREP::Location::getCityUtfName(), OPENTREP::Location::getCommonName(), OPENTREP::Location::getContinentName(), OPENTREP::Location::getCountryCode(), OPENTREP::Location::getCountryName(), OPENTREP::Location::getFaaCode(), OPENTREP::Location::getFeatureCode(), OPENTREP::Location::getGeonamesID(), OPENTREP::Location::getIataCode(), OPENTREP::Location::getIcaoCode(), OPENTREP::Names::getNameList(), OPENTREP::NameMatrix::getNameMatrix(), OPENTREP::Location::getNameMatrix(), OPENTREP::Location::getPageRank(), OPENTREP::Location::getStateCode(), getTermSet(), OPENTREP::K_DEFAULT_INDEXING_EXTRA_WEIGHT, OPENTREP::K_DEFAULT_INDEXING_STD_WEIGHT, and OPENTREP::OTransliterator::normalise().

23.89.3.124 void OPENTREP::Place::addNameToXapianSets (const Weight_T & *iWeight*, const std::string & *iBaseName*, const FeatureCode_T & *iFeatureCode*)

Add the given name to the Xapian index with the given weight. Derive a list of feature names from the feature code. See the Location::getFeatureList() method for more information.

Parameters

const	Weight_T& The weight with which the terms should be indexed
const	std::string& Name of the POR (point of reference)
const	FeatureCode_T& Geonames feature code

Definition at line 269 of file Place.cpp.

References addTermSet(), OPENTREP::Location::getFeatureList(), and getTermSet().

23.89.3.125 void OPENTREP::Place::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::BomAbstract.

Definition at line 146 of file Place.cpp.

References toString().

23.89.3.126 void OPENTREP::Place::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::BomAbstract.

Definition at line 151 of file Place.cpp.

23.89.3.127 std::string OPENTREP::Place::describeKey ( ) const [inline], [virtual]

Get a string describing the whole key (IATA and ICAO codes, Geonames ID).

Implements OPENTREP::BomAbstract.

Definition at line 1002 of file Place.hpp.

References OPENTREP::Location::describeKey().

Referenced by OPENTREP::addToXapian(), and OPENTREP::DBManager::iterateOnStatement().

23.89.3.128 std::string OPENTREP::Place::describeShortKey ( ) const [inline], [virtual]

Get a string describing the whole key (IATA and ICAO codes, Geonames ID).

Implements OPENTREP::BomAbstract.

Definition at line 1009 of file Place.hpp.

References OPENTREP::Location::describeShortKey().

Referenced by OPENTREP::FacPlaceHolder::initLinkWithPlace(), and OPENTREP::FacWorld::linkPlaceToWorld().

23.89.3.129 std::string OPENTREP::Place::toString ( ) const [virtual]

Get the serialised version of the Place object.

Implements OPENTREP::BomAbstract.

Definition at line 83 of file Place.cpp.

References toShortString(), and OPENTREP::Location::toString().

Referenced by OPENTREP::createPlaces(), toStream(), OPENTREP::PlaceHolder::toString(), and OPENTREP::DBManager::updatePlaceInDB().

23.89.3.130 std::string OPENTREP::Place::toShortString ( ) const

Get a short display of the Business Object.

Definition at line 124 of file Place.cpp.

References OPENTREP::Location::toShortString().

Referenced by completeLocation(), OPENTREP::PlaceHolder::toShortString(), and toString().

23.89.3.131 std::string OPENTREP::Place::describeSets ( ) const

Get a string describing the (STL) sets of terms for the Xapian database.

Definition at line 155 of file Place.cpp.

Referenced by OPENTREP::addToXapian().

23.89.3.132 std::string OPENTREP::Place::display ( ) const

Display the full Place context.

Definition at line 211 of file Place.cpp.

References shortDisplay().

Referenced by OPENTREP::World::display().

23.89.3.133 std::string OPENTREP::Place::shortDisplay ( ) const

Display a short Place context.

Definition at line 200 of file Place.cpp.

References OPENTREP::Location::toBasicString().

Referenced by display().

23.89.4 Friends And Related Function Documentation

23.89.4.1 friend class FacWorld [friend]

Definition at line 29 of file Place.hpp.

23.89.4.2 friend class FacPlace [friend]

Definition at line 30 of file Place.hpp.

23.89.4.3 friend class FacPlaceHolder [friend]

Definition at line 31 of file Place.hpp.

23.89.4.4 friend class DbaPlace [friend]

Definition at line 32 of file Place.hpp.

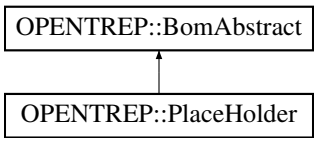
The documentation for this class was generated from the following files:

- opentrep/bom/Place.hpp
- opentrep/bom/Place.cpp

23.90 OPENTREP::PlaceHolder Class Reference

`#include <opentrep/bom/PlaceHolder.hpp>`

Inheritance diagram for OPENTREP::PlaceHolder:



Public Member Functions

- const PlaceList_T & getPlaceList () const
- void createLocations (LocationList_T &) const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- std::string toShortString () const
- std::string describeKey () const
- std::string describeShortKey () const

Friends

- class FacPlaceHolder

23.90.1 Detailed Description

Class wrapping functions on a list of Place objects.

Definition at line 16 of file PlaceHolder.hpp.

23.90.2 Member Function Documentation

23.90.2.1 const PlaceList_T& OPENTREP::PlaceHolder::getPlaceList ( ) const [inline]

Retrieve the list of place objects.

Definition at line 21 of file PlaceHolder.hpp.

23.90.2.2 void OPENTREP::PlaceHolder::createLocations (LocationList_T & *ioLocationList*) const

Create the list of Location structures, which are light copies of the Place objects. Those (Location) structures are passed back to the caller of the service.

Definition at line 82 of file PlaceHolder.cpp.

References OPENTREP::Place::completeLocation().

23.90.2.3 void OPENTREP::PlaceHolder::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::BomAbstract.

Definition at line 73 of file PlaceHolder.cpp.

References toString().

23.90.2.4 void OPENTREP::PlaceHolder::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::BomAbstract.

Definition at line 78 of file PlaceHolder.cpp.

23.90.2.5 std::string OPENTREP::PlaceHolder::toString ( ) const [virtual]

Get the serialised version of the Business Object.

Implements OPENTREP::BomAbstract.

Definition at line 41 of file PlaceHolder.cpp.

References describeShortKey(), and OPENTREP::Place::toString().

Referenced by toStream().

23.90.2.6 std::string OPENTREP::PlaceHolder::toShortString ( ) const

Get a short display of the Business Object.

Definition at line 57 of file PlaceHolder.cpp.

References describeShortKey(), and OPENTREP::Place::toShortString().

23.90.2.7 std::string OPENTREP::PlaceHolder::describeKey ( ) const [virtual]

Get a string describing the whole key (differentiating two objects at any level).

Implements OPENTREP::BomAbstract.

Definition at line 36 of file PlaceHolder.cpp.

References describeShortKey().

23.90.2.8 std::string OPENTREP::PlaceHolder::describeShortKey ( ) const [virtual]

Get a string describing the short key (differentiating two objects at the same level).

Implements OPENTREP::BomAbstract.

Definition at line 30 of file PlaceHolder.cpp.

Referenced by describeKey(), toShortString(), and toString().

23.90.3 Friends And Related Function Documentation

23.90.3.1 friend class FacPlaceHolder [friend]

Definition at line 17 of file PlaceHolder.hpp.

The documentation for this class was generated from the following files:

- opentrep/bom/PlaceHolder.hpp
- opentrep/bom/PlaceHolder.cpp

23.91 OPENTREP::PORFileParser Class Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Public Member Functions

- PORFileParser (const PORFilePath_T &iFilename)
- void generateLocations ()

23.91.1 Detailed Description

Class wrapping the initialisation and entry point of the parser.

The seemingly redundancy is used to force the instantiation of the actual parser, which is a templatised Boost Spirit grammar. Hence, the actual parser is instantiated within that class object code.

Definition at line 621 of file PORParserHelper.hpp.

23.91.2 Constructor & Destructor Documentation

23.91.2.1 OPENTREP::PORFileParser::PORFileParser (const PORFilePath_T & *iFilename*)

Constructor.

Definition at line 1563 of file PORParserHelper.cpp.

23.91.3 Member Function Documentation

23.91.3.1 void OPENTREP::PORFileParser::generateLocations ( )

Parse the input file and generate the Location structures.

Definition at line 1573 of file PORParserHelper.cpp.

References OPENTREP_LOG_DEBUG, and OPENTREP_LOG_ERROR.

Referenced by OPENTREP::PORParser::PORGeneration().

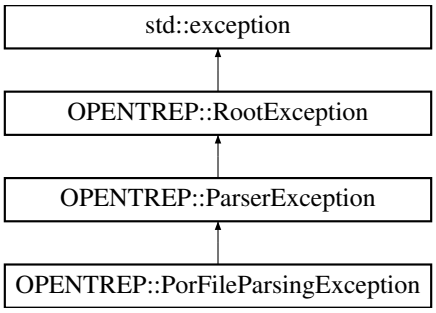
The documentation for this class was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.92 OPENTREP::PorFileParsingException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::PorFileParsingException:



Public Member Functions

- PorFileParsingException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.92.1 Detailed Description

POR file parsing exception.

Definition at line 167 of file OPENTREP_exceptions.hpp.

23.92.2 Constructor & Destructor Documentation

23.92.2.1 OPENTREP::PorFileParsingException::PorFileParsingException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 172 of file OPENTREP_exceptions.hpp.

23.92.3 Member Function Documentation

23.92.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.92.4 Member Data Documentation

23.92.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

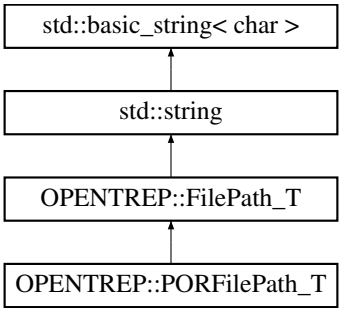
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.93 OPENTREP::PORFilePath_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::PORFilePath_T:



Public Member Functions

- PORFilePath_T (const std::string &iValue)

23.93.1 Detailed Description

File-path for the ORI-maintained list of POR (points of reference).

Definition at line 37 of file OPENTREP_Types.hpp.

23.93.2 Constructor & Destructor Documentation

23.93.2.1 OPENTREP::PORFilePath_T::PORFilePath_T (const std::string & *iValue*) [inline],[explicit]

Definition at line 39 of file OPENTREP_Types.hpp.

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.94 OPENTREP::PORParser Class Reference

`#include <opentrep/command/PORParser.hpp>`

Static Public Member Functions

- static void PORGeneration (const PORFilePath_T &)

23.94.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 15 of file PORParser.hpp.

23.94.2 Member Function Documentation

23.94.2.1 void OPENTREP::PORParser::PORGeneration (const PORFilePath_T & *iPORFilename*) [static]

Parses the CSV file describing the POR (points of reference), and generates the POR BOM tree accordingly.

Parameters

const	PORFilePath_T& The file-name of the CSV-formatted POR input file.

Definition at line 16 of file PORParser.cpp.

References OPENTREP::BasFileMgr::doesExistAndIsReadable(), OPENTREP::PORFileParser::generateLocations(), and OPENTREP_LOG_ERROR.

The documentation for this class was generated from the following files:

- opentrep/command/PORParser.hpp
- opentrep/command/PORParser.cpp

23.95 OPENTREP::PORStringParser Class Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Public Member Functions

- PORStringParser (const std::string &iString)
- ~PORStringParser ()
- const Location & generateLocation ()

23.95.1 Detailed Description

Class wrapping the initialisation and entry point of the parser.

The seemingly redundancy is used to force the instantiation of the actual parser, which is a templatised Boost Spirit grammar. Hence, the actual parser is instantiated within that class object code.

Definition at line 572 of file PORParserHelper.hpp.

23.95.2 Constructor & Destructor Documentation

23.95.2.1 OPENTREP::PORStringParser::PORStringParser (const std::string & *iString*)

Constructor.

Definition at line 1473 of file PORParserHelper.cpp.

23.95.2.2 OPENTREP::PORStringParser::~PORStringParser ( )

Destructor.

Definition at line 1485 of file PORParserHelper.cpp.

23.95.3 Member Function Documentation

23.95.3.1 const Location & OPENTREP::PORStringParser::generateLocation ( )

Parse the input string and generate the Location structures.

Definition at line 1489 of file PORParserHelper.cpp.

References OPENTREP_LOG_ERROR.

Referenced by OPENTREP::Result::retrieveLocation().

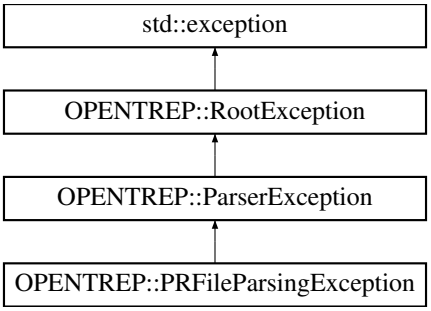
The documentation for this class was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.96 OPENTREP::PRFileParsingException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::PRFileParsingException:



Public Member Functions

- PRFileParsingException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.96.1 Detailed Description

PR file parsing exception.

Definition at line 179 of file OPENTREP_exceptions.hpp.

23.96.2 Constructor & Destructor Documentation

23.96.2.1 OPENTREP::PRFileParsingException::PRFileParsingException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 184 of file OPENTREP_exceptions.hpp.

23.96.3 Member Function Documentation

23.96.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.96.4 Member Data Documentation

23.96.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

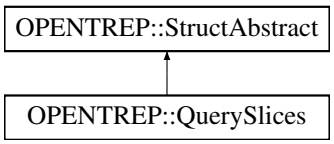
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.97 OPENTREP::QuerySlices Struct Reference

`#include <opentrep/bom/QuerySlices.hpp>`

Inheritance diagram for OPENTREP::QuerySlices:



Public Member Functions

- const StringPartitionList_T & getStringPartitionList () const
- void push_back (const StringPartition &iStringPartition)
- size_t size () const
- bool empty () const
- void clear ()
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string describeKey () const
- std::string describe () const
- QuerySlices (const Xapian::Database &, const TravelQuery_T &)
- ~QuerySlices ()

Public Attributes

- const Xapian::Database & _database
- TravelQuery_T _queryString
- StringPartitionList_T _slices
- std::string _itLeftWords

23.97.1 Detailed Description

Class allowing to slice a query string into multiple slices. Each of those slices will then give birth to the corresponding string partitions.

The initial query string is sliced in the interstices, which split apart any two consecutive words yielding no full text match.

For instance, "san francisco nce rio de janeiro" will give:

- [
- – 0. { {"san francisco"} }
  – 1. { {"nce"} }
  – 2. { {"rio de janeiro"} }
-]

Indeed, "francisco nce" does not yield any match; hence, the query may be split right after the "francisco" word, i.e., separating the "san francisco" sub-query from the "nce rio de janeiro" one. The same way, "nce rio" does not yield any match; hence, the remaining query may be split into two sub-queries: "nce" and "rio de janeiro".

Definition at line 44 of file QuerySlices.hpp.

23.97.2 Constructor & Destructor Documentation

23.97.2.1 OPENTREP::QuerySlices::QuerySlices (const Xapian::Database & *iDatabase,* const TravelQuery_T & *iQueryString*)

Main constructor.

Parameters

const	Xapian::Database& Xapian database (index)
const	TravelQuery_T& The string for which the partitions are sought

Definition at line 19 of file QuerySlices.cpp.

23.97.2.2 OPENTREP::QuerySlices::~QuerySlices ( )

Default destructor.

Definition at line 26 of file QuerySlices.cpp.

23.97.3 Member Function Documentation

23.97.3.1 const StringPartitionList_T& OPENTREP::QuerySlices::getStringPartitionList ( ) const [inline]

Get the underlying list of string partitions.

Definition at line 49 of file QuerySlices.hpp.

References _slices.

23.97.3.2 void OPENTREP::QuerySlices::push_back (const StringPartition & *iStringPartition*)

Add an item (StringPartition) into the list.

Note

When the given string set is empty (zero-length), it is (obviously) not added to the list

Definition at line 30 of file QuerySlices.cpp.

References _slices, and OPENTREP::StringPartition::empty().

23.97.3.3 size_t OPENTREP::QuerySlices::size ( ) const

Return the size of the list.

Definition at line 37 of file QuerySlices.cpp.

References _slices.

Referenced by BOOST_AUTO_TEST_CASE().

23.97.3.4 bool OPENTREP::QuerySlices::empty ( ) const

Return whether or not the list is empty.

Definition at line 42 of file QuerySlices.cpp.

References _slices.

23.97.3.5 void OPENTREP::QuerySlices::clear ( )

Empty the list.

Definition at line 47 of file QuerySlices.cpp.

References _slices.

23.97.3.6 void OPENTREP::QuerySlices::toStream (std::ostream & *ioOut*) const

Dump the structure into an output stream.

Parameters

ostream&	the output stream.

Definition at line 88 of file QuerySlices.cpp.

References describe().

23.97.3.7 void OPENTREP::QuerySlices::fromStream (std::istream & *ioIn*) [virtual]

Read a structure from an input stream.

Parameters

istream&	the input stream.

Reimplemented from OPENTREP::StructAbstract.

Definition at line 93 of file QuerySlices.cpp.

23.97.3.8 std::string OPENTREP::QuerySlices::describeKey ( ) const

Get a string describing the whole key (differentiating two objects at any level).

Definition at line 52 of file QuerySlices.cpp.

Referenced by describe().

23.97.3.9 std::string OPENTREP::QuerySlices::describe ( ) const [virtual]

Get the serialised version of the structure.

Implements OPENTREP::StructAbstract.

Definition at line 59 of file QuerySlices.cpp.

References _slices, and describeKey().

Referenced by toStream().

23.97.4 Member Data Documentation

23.97.4.1 const Xapian::Database& OPENTREP::QuerySlices::_database

Xapian database.

Definition at line 136 of file QuerySlices.hpp.

23.97.4.2 TravelQuery_T OPENTREP::QuerySlices::_queryString

Query string having generated the set of documents.

Definition at line 141 of file QuerySlices.hpp.

23.97.4.3 StringPartitionList_T OPENTREP::QuerySlices::_slices

Partition, i.e., a list containing sub-lists of strings

Definition at line 146 of file QuerySlices.hpp.

Referenced by clear(), describe(), empty(), getStringPartitionList(), push_back(), and size().

23.97.4.4 std::string OPENTREP::QuerySlices::_itLeftWords

Staging string holding the left part of the query

Definition at line 151 of file QuerySlices.hpp.

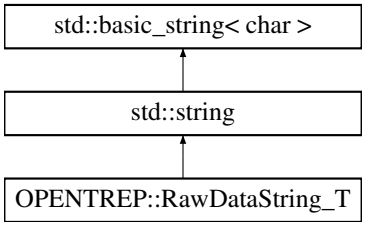
The documentation for this struct was generated from the following files:

- opentrep/bom/QuerySlices.hpp
- opentrep/bom/QuerySlices.cpp

23.98 OPENTREP::RawDataString_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::RawDataString_T:



Public Member Functions

- RawDataString_T (const std::string &iValue)

23.98.1 Detailed Description

Raw data string, before any parsing. It is used to be re-parsed once retrieved by Xapian as document data.

Definition at line 66 of file OPENTREP_Types.hpp.

23.98.2 Constructor & Destructor Documentation

23.98.2.1 OPENTREP::RawDataString_T::RawDataString_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 68 of file OPENTREP_Types.hpp.

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.99 OPENTREP::RequestInterpreter Class Reference

Command wrapping the travel request process.

`#include <opentrep/command/RequestInterpreter.hpp>`

Friends

- class OPENTREP_Service

23.99.1 Detailed Description

Command wrapping the travel request process.

Definition at line 16 of file RequestInterpreter.hpp.

23.99.2 Friends And Related Function Documentation

23.99.2.1 friend class OPENTREP_Service [friend]

Definition at line 17 of file RequestInterpreter.hpp.

The documentation for this class was generated from the following files:

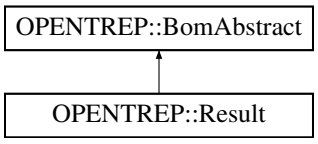
- opentrep/command/RequestInterpreter.hpp
- opentrep/command/RequestInterpreter.cpp

23.100 OPENTREP::Result Class Reference

Class wrapping a set of Xapian documents having matched a given query string.

`#include <opentrep/bom/Result.hpp>`

Inheritance diagram for OPENTREP::Result:



Public Member Functions

- const TravelQuery_T & getQueryString () const
- const TravelQuery_T & getCorrectedTravelQuery () const
- bool hasFullTextMatched () const
- const NbOfErrors_T & getEditDistance () const
- const NbOfErrors_T & getAllowableEditDistance () const
- const DocumentList_T & getDocumentList () const
- const DocumentMap_T & getDocumentMap () const
- const XapianDocumentPair_T & getDocumentPair (const Xapian::docid &) const
- const Xapian::Document & getDocument (const Xapian::docid &) const
- const Xapian::docid & getBestDocID () const
- const Percentage_T & getBestCombinedWeight () const
- const RawDataString_T & getBestDocData () const
- const Xapian::Document & getBestXapianDocument () const
- void setQueryString (const TravelQuery_T &iQueryString)
- void setCorrectedQueryString (const TravelQuery_T &iCorrectedQueryString)
- void setHasFullTextMatched (const bool iHasFullTextMatched)
- void setEditDistance (const NbOfErrors_T &iEditDistance)
- void setAllowableEditDistance (const NbOfErrors_T &iAllowableEditDistance)
- void addDocument (const Xapian::Document &, const Score_T &)
- void setBestDocID (const Xapian::docid &iDocID)
- void setBestCombinedWeight (const Percentage_T &iPercentage)
- void setBestDocData (const std::string &iDocData)
- void fillResult (const Xapian::MSet &iMatchingSet)
- void fillPlace (Place &) const
- std::string fullTextMatch (const Xapian::Database &, const TravelQuery_T &)
- void displayXapianPercentages () const
- void setScoreOnDocMap (const Xapian::docid &, const ScoreType &, const Score_T &)
- void calculateEnvelopeWeights ()
- void calculateCodeMatches ()
- void calculatePageRanks ()
- void calculateHeuristicWeights ()
- void calculateCombinedWeights ()
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- std::string describeKey () const
- std::string describeShortKey () const

Static Public Member Functions

- static Location retrieveLocation (const Xapian::Document &)
- static Location retrieveLocation (const RawDataString_T &)
- static LocationKey getPrimaryKey (const Xapian::Document &)
- static Score_T getEnvelopeID (const Xapian::Document &)
- static PageRank_T getPageRank (const Xapian::Document &)

Friends

- class FacResultHolder
- class FacResult

23.100.1 Detailed Description

Class wrapping a set of Xapian documents having matched a given query string.

Definition at line 48 of file Result.hpp.

23.100.2 Member Function Documentation

23.100.2.1 const TravelQuery_T& OPENTREP::Result::getQueryString ( ) const [inline]

Get the query string.

Definition at line 56 of file Result.hpp.

23.100.2.2 const TravelQuery_T& OPENTREP::Result::getCorrectedTravelQuery ( ) const [inline]

Get the corrected query string. When empty, it means that no correction was necessary.

Definition at line 64 of file Result.hpp.

Referenced by OPENTREP::ResultHolder::getCorrectedStringSet().

23.100.2.3 bool OPENTREP::Result::hasFullTextMatched ( ) const [inline]

State whether there has been a full-text match.

Definition at line 71 of file Result.hpp.

Referenced by OPENTREP::createPlaces().

23.100.2.4 const NbOfErrors_T& OPENTREP::Result::getEditDistance ( ) const [inline]

Get the edit distance/error, with which the matching has been made.

Definition at line 78 of file Result.hpp.

23.100.2.5 const NbOfErrors_T& OPENTREP::Result::getAllowableEditDistance ( ) const [inline]

Get the maximal allowable edit distance/error, with which the matching has been made.

Definition at line 86 of file Result.hpp.

23.100.2.6 const DocumentList_T& OPENTREP::Result::getDocumentList ( ) const [inline]

Get the list of documents.

Definition at line 93 of file Result.hpp.

23.100.2.7 const DocumentMap_T& OPENTREP::Result::getDocumentMap ( ) const [inline]

Get the map of documents.

Definition at line 100 of file Result.hpp.

23.100.2.8 const XapianDocumentPair_T & OPENTREP::Result::getDocumentPair (const Xapian::docid & *iDocID*) const

Get the Xapian document and associated score-board corresponding to the given document ID.

Definition at line 107 of file Result.cpp.

References describeKey(), and OPENTREP_LOG_ERROR.

Referenced by calculateCombinedWeights(), and getDocument().

23.100.2.9 const Xapian::Document & OPENTREP::Result::getDocument (const Xapian::docid & *iDocID*) const

Get the Xapian document corresponding to the given document ID.

Definition at line 128 of file Result.cpp.

References getDocumentPair().

Referenced by getBestXapianDocument().

23.100.2.10 const Xapian::docid& OPENTREP::Result::getBestDocID ( ) const [inline]

Get the Xapian ID of the best matching document.

Definition at line 118 of file Result.hpp.

23.100.2.11 const Percentage_T& OPENTREP::Result::getBestCombinedWeight ( ) const [inline]

Get the combined weight, for all the rules (full-text, PageRank, etc)

Definition at line 125 of file Result.hpp.

Referenced by OPENTREP::ResultHolder::calculateCombinedWeights().

23.100.2.12 const RawDataString_T& OPENTREP::Result::getBestDocData ( ) const [inline]

Get the details of the best matching document.

Definition at line 132 of file Result.hpp.

Referenced by OPENTREP::createPlaces().

23.100.2.13 const Xapian::Document& OPENTREP::Result::getBestXapianDocument ( ) const [inline]

Get the best matching Xapian document.

Definition at line 139 of file Result.hpp.

References getDocument().

23.100.2.14 void OPENTREP::Result::setQueryString (const TravelQuery_T & *iQueryString*) [inline]

Set the query string.

Definition at line 149 of file Result.hpp.

23.100.2.15 void OPENTREP::Result::setCorrectedQueryString (const TravelQuery_T & *iCorrectedQueryString*) [inline]

Set the corrected query string.

Definition at line 156 of file Result.hpp.

23.100.2.16 void OPENTREP::Result::setHasFullTextMatched (const bool *iHasFullTextMatched*) [inline]

Set whether there has been a full-text match.

Definition at line 163 of file Result.hpp.

23.100.2.17 void OPENTREP::Result::setEditDistance (const NbOfErrors_T & *iEditDistance*) [inline]

Set the edit distance/error, with which the matching has been made.

Definition at line 170 of file Result.hpp.

23.100.2.18 void OPENTREP::Result::setAllowableEditDistance (const NbOfErrors_T & *iAllowableEditDistance*) [inline]

Set the maxiaml allowable edit distance/error, with which the matching has been made.

Definition at line 178 of file Result.hpp.

23.100.2.19 void OPENTREP::Result::addDocument (const Xapian::Document & *iDocument,* const Score_T & *iScore*)

Set the latest score for the given type. If no score value has already been stored for that type, create it. void setScore (const ScoreType& iScoreType, const Score_T& iScore) { _scoreBoard.setScore (iScoreType, iScore); } Add a Xapian document to the dedicated (STL) list and (STL) map.

Note

The score type is not specified, as it is corresponding, by construction, to the (Xapian-based) full-text matching. Indeed, when there is no (Xapian-based) full-text matching, by construction, no Xapian document has been found matching a given string.

Parameters

const	Xapian::Document& The Xapian document to be added.
const	Score_T& The matching percentage.

Definition at line 141 of file Result.cpp.

References OPENTREP::ScoreType::XAPIAN_PCT.

Referenced by fillResult().

23.100.2.20 void OPENTREP::Result::setBestDocID (const Xapian::docid & *iDocID*) [inline]

Set the Xapian ID of the best matching document.

Definition at line 207 of file Result.hpp.

Referenced by calculateCombinedWeights().

23.100.2.21 void OPENTREP::Result::setBestCombinedWeight (const Percentage_T & *iPercentage*) [inline]

Set the best combined weight, for all the rules (full-text, PageRank, etc)

Definition at line 215 of file Result.hpp.

Referenced by calculateCombinedWeights().

23.100.2.22 void OPENTREP::Result::setBestDocData (const std::string & *iDocData*) [inline]

Set the details of the best matching document.

Definition at line 222 of file Result.hpp.

Referenced by calculateCombinedWeights().

23.100.2.23 void OPENTREP::Result::fillResult (const Xapian::MSet & *iMatchingSet*)

Extract the best matching Xapian document.

Parameters

Xapian::MSet&	The Xapian matching set. It can be empty.
Result&	The holder for the Xapian documents to be stored.

Definition at line 191 of file Result.cpp.

References addDocument().

Referenced by fullTextMatch().

23.100.2.24 void OPENTREP::Result::fillPlace (Place & *ioPlace*) const

Fill the Place object with the details of the best matching Xapian document.

Parameters

Place&	The Place object.

Definition at line 205 of file Result.cpp.

References OPENTREP::Place::getKey(), OPENTREP_LOG_DEBUG, OPENTREP::Place::setAllowableEditDistance(), OPENTREP::Place::setCorrectedKeywords(), OPENTREP::Place::setDocID(), OPENTREP::Place::setEditDistance(), OPENTREP::Place::setOriginalKeywords(), and OPENTREP::Place::setPercentage().

Referenced by OPENTREP::createPlaces().

23.100.2.25 std::string OPENTREP::Result::fullTextMatch (const Xapian::Database & *iDatabase*, const TravelQuery_T & *iQueryString*)

For all the elements (strings) of the travel query (string set), perform a Xapian-based full-text match. That Xapian-based full-text match gives (potentially) a full set of matches, some with the highest matching percentage and some with a lower percentage.

Parameters

const	Xapian::Database& The Xapian index/database.
const	TravelQuery_T& The query string.

Definition at line 515 of file Result.cpp.

References fillResult(), OPENTREP_LOG_DEBUG, OPENTREP_LOG_ERROR, OPENTREP::Filter::shouldKeep(), and toString().

Referenced by OPENTREP::searchString().

23.100.2.26 Location OPENTREP::Result::retrieveLocation (const Xapian::Document & *iDocument*) [static]

Parse the raw data, as stored by the given Xapian document, and holding all the details of a POR (point of reference).

Parameters

const	Xapian::Document& The Xapian document.

Returns

Location The Location structure holding all the details of the place/POR (point of reference).

Definition at line 266 of file Result.cpp.

Referenced by OPENTREP::createPlaces(), getEnvelopeID(), getPageRank(), and getPrimaryKey().

23.100.2.27 Location OPENTREP::Result::retrieveLocation (const RawDataString_T & *iRawDataString*) [static]

Parse the raw data, as stored by a typical Xapian document, and holding all the details of a POR (point of reference).

Parameters

const	RawDataString_T& The Xapian document data.

Returns

Location The Location structure holding all the details of the place/POR (point of reference).

Definition at line 252 of file Result.cpp.

References OPENTREP::PORStringParser::generateLocation().

23.100.2.28 LocationKey OPENTREP::Result::getPrimaryKey (const Xapian::Document & *iDocument*) [static]

Extract the primary key from the data of the given Xapian document.

The primary key is made of the IATA and ICAO codes, as well as of the Geonames ID. The retrieveLocation() is used to parse the Xapian document raw data.

Parameters

Xapian::Document&	The Xapian document.

Returns

LocationKey& The primary key of the place/POR (point of reference).

Definition at line 278 of file Result.cpp.

References OPENTREP::Location::getKey(), and retrieveLocation().

Referenced by calculateCodeMatches(), calculateCombinedWeights(), calculateEnvelopeWeights(), calculatePageRanks(), and displayXapianPercentages().

23.100.2.29 Score_T OPENTREP::Result::getEnvelopeID (const Xapian::Document & *iDocument*) [static]

Extract the Envelope ID from the data of the given Xapian document.

The retrieveLocation() is used to parse the Xapian document raw data.

Parameters

Xapian::Document&	The Xapian document.

Returns

Score_T& The Envelope ID of the place/POR (point of reference), converted into a floating point (from an integer value).

Definition at line 289 of file Result.cpp.

References OPENTREP::Location::getEnvelopeID(), and retrieveLocation().

Referenced by calculateEnvelopeWeights().

23.100.2.30 PageRank_T OPENTREP::Result::getPageRank (const Xapian::Document & *iDocument*) [static]

Extract the PageRank from the data of the given Xapian document.

The retrieveLocation() is used to parse the Xapian document raw data.

Parameters

Xapian::Document&	The Xapian document.

Returns

PageRank_T& The PageRank of the place/POR (point of reference).

Definition at line 303 of file Result.cpp.

References OPENTREP::Location::getPageRank(), and retrieveLocation().

Referenced by calculatePageRanks().

23.100.2.31 void OPENTREP::Result::displayXapianPercentages ( ) const

Display the Xapian matching percentages for all the matching documents.

That is just a summarising step, for convenience/debug reason. No calculation is done here.

Definition at line 556 of file Result.cpp.

References describeShortKey(), getPrimaryKey(), OPENTREP::ScoreBoard::getScore(), OPENTREP_LOG_NOTIFICATION, and OPENTREP::ScoreType::XAPIAN_PCT.

Referenced by OPENTREP::ResultHolder::displayXapianPercentages().

23.100.2.32 void OPENTREP::Result::setScoreOnDocMap (const Xapian::docid & *iDocID,* const ScoreType & *iType,* const Score_T & *iScore*)

Update the ScoreBoard structure, itself associated to the Xapian document corresponding to the given document ID, with the given score.

Parameters

const	Xapian::docid& Document ID for the Xapian document
const	ScoreType& Score type
const	Score_T& Score/weight value

Definition at line 586 of file Result.cpp.

References describeKey(), OPENTREP_LOG_ERROR, and OPENTREP::ScoreBoard::setScore().

Referenced by calculateCodeMatches(), calculateEnvelopeWeights(), and calculatePageRanks().

23.100.2.33 void OPENTREP::Result::calculateEnvelopeWeights ( )

Calculate/set the envelope weights for all the matching documents.

Some documents have a non-null envelope, meaning that they are no longer valid: they were valid for a given period of time in the past, but no longer are. For instance, the corresponding object (POR, airline, etc) may have been closed, removed, replaced.

Definition at line 608 of file Result.cpp.

References describeShortKey(), OPENTREP::ScoreType::ENV_ID, getEnvelopeID(), getPrimaryKey(), OPENTREP_LOG_NOTIFICATION, OPENTREP::ScoreBoard::setScore(), and setScoreOnDocMap().

Referenced by OPENTREP::ResultHolder::calculateEnvelopeWeights().

23.100.2.34 void OPENTREP::Result::calculateCodeMatches ( )

Calculate/set the IATA/ICAO code matching weights for all the matching documents

Definition at line 647 of file Result.cpp.

References OPENTREP::ScoreType::CODE_FULL_MATCH, describeShortKey(), OPENTREP::LocationKey::getIataCode(), getPrimaryKey(), OPENTREP::K_DEFAULT_FULL_CODE_MATCH_PCT, OPENTREP::K_DEFAULT_MODIFIED_MATCHING_PCT, OPENTREP_LOG_NOTIFICATION, OPENTREP::ScoreBoard::setScore(), setScoreOnDocMap(), OPENTREP::WordHolder::tokeniseStringIntoWordList(), and OPENTREP::Filter::trim().

Referenced by OPENTREP::ResultHolder::calculateCodeMatches().

23.100.2.35 void OPENTREP::Result::calculatePageRanks ( )

Calculate/set the PageRanks for all the matching documents

Definition at line 740 of file Result.cpp.

References describeShortKey(), getPageRank(), getPrimaryKey(), OPENTREP_LOG_NOTIFICATION, OPENTREP::ScoreType::PAGE_RANK, OPENTREP::ScoreBoard::setScore(), and setScoreOnDocMap().

Referenced by OPENTREP::ResultHolder::calculatePageRanks().

23.100.2.36 void OPENTREP::Result::calculateHeuristicWeights ( )

Calculate/set the heuristic weights for all the matching documents

Definition at line 774 of file Result.cpp.

Referenced by OPENTREP::ResultHolder::calculateHeuristicWeights().

23.100.2.37 void OPENTREP::Result::calculateCombinedWeights ( )

Calculate/set the combined weights for all the matching documents. Store the best matching one in the _bestCombinedWeight attribute.

Definition at line 783 of file Result.cpp.

References OPENTREP::ScoreBoard::calculateCombinedWeight(), OPENTREP::ScoreBoard::describe(), describeShortKey(), getDocumentPair(), getPrimaryKey(), OPENTREP_LOG_DEBUG, setBestCombinedWeight(), setBestDocData(), setBestDocID(), OPENTREP::Filter::shouldKeep(), and OPENTREP::WordHolder::tokeniseStringIntoWordList().

Referenced by OPENTREP::ResultHolder::calculateCombinedWeights().

23.100.2.38 void OPENTREP::Result::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::BomAbstract.

Definition at line 97 of file Result.cpp.

References toString().

23.100.2.39 void OPENTREP::Result::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::BomAbstract.

Definition at line 102 of file Result.cpp.

23.100.2.40 std::string OPENTREP::Result::toString ( ) const [virtual]

Get the serialised version of the Business Object.

Implements OPENTREP::BomAbstract.

Definition at line 65 of file Result.cpp.

References OPENTREP::ScoreBoard::describe(), and describeKey().

Referenced by fullTextMatch(), toStream(), and OPENTREP::ResultHolder::toString().

23.100.2.41 std::string OPENTREP::Result::describeKey ( ) const [virtual]

Get a string describing the whole key (differentiating two objects at any level).

Implements OPENTREP::BomAbstract.

Definition at line 49 of file Result.cpp.

References describeShortKey().

Referenced by getDocumentPair(), setScoreOnDocMap(), and toString().

23.100.2.42 std::string OPENTREP::Result::describeShortKey ( ) const [virtual]

Get a string describing the short key (differentiating two objects at the same level).

Implements OPENTREP::BomAbstract.

Definition at line 42 of file Result.cpp.

Referenced by calculateCodeMatches(), calculateCombinedWeights(), calculateEnvelopeWeights(), calculatePageRanks(), describeKey(), and displayXapianPercentages().

23.100.3 Friends And Related Function Documentation

23.100.3.1 friend class FacResultHolder [friend]

Definition at line 49 of file Result.hpp.

23.100.3.2 friend class FacResult [friend]

Definition at line 50 of file Result.hpp.

The documentation for this class was generated from the following files:

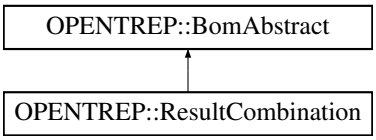
- opentrep/bom/Result.hpp
- opentrep/bom/Result.cpp

23.101 OPENTREP::ResultCombination Class Reference

Class wrapping functions on a list of ResultHolder objects.

`#include <opentrep/bom/ResultCombination.hpp>`

Inheritance diagram for OPENTREP::ResultCombination:



Public Member Functions

- const TravelQuery_T & getQueryString () const
- const ResultHolderList_T & getResultHolderList () const
- bool hasFullTextMatched () const
- const ResultHolder & getBestMatchingResultHolder () const
- const Percentage_T & getBestMatchingWeight () const
- StringSet getCorrectedStringSet () const
- void displayXapianPercentages () const
- void calculateEnvelopeWeights () const
- void calculateCodeMatches () const
- void calculatePageRanks () const
- void calculateHeuristicWeights () const
- void calculateCombinedWeights () const
- bool chooseBestMatchingResultHolder ()
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- std::string toString () const
- std::string describeKey () const
- std::string describeShortKey () const

Friends

- class FacResultCombination

23.101.1 Detailed Description

Class wrapping functions on a list of ResultHolder objects.

Definition at line 25 of file ResultCombination.hpp.

23.101.2 Member Function Documentation

23.101.2.1 const TravelQuery_T& OPENTREP::ResultCombination::getQueryString ( ) const [inline]

Get the query string.

Definition at line 32 of file ResultCombination.hpp.

23.101.2.2 const ResultHolderList_T& OPENTREP::ResultCombination::getResultHolderList ( ) const [inline]

Retrieve the list of result objects.

Definition at line 39 of file ResultCombination.hpp.

23.101.2.3 bool OPENTREP::ResultCombination::hasFullTextMatched ( ) const [inline]

State whether there has been a full-text match.

Definition at line 46 of file ResultCombination.hpp.

23.101.2.4 const ResultHolder & OPENTREP::ResultCombination::getBestMatchingResultHolder ( ) const

Get a reference on the best matching ResultHolder object. If there is no such object, the method throws a NoBestMatchingResultHolderException exception.

Definition at line 46 of file ResultCombination.cpp.

References OPENTREP_LOG_ERROR.

Referenced by OPENTREP::chooseBestMatchingResultHolder(), OPENTREP::createPlaces(), getBestMatchingWeight(), and getCorrectedStringSet().

23.101.2.5 const Percentage_T & OPENTREP::ResultCombination::getBestMatchingWeight ( ) const

Get the weight of the best matching ResultHolder object. If there is no such object, the method throws a NoBestMatchingResultHolderException exception.

Definition at line 59 of file ResultCombination.cpp.

References getBestMatchingResultHolder(), and OPENTREP::ResultHolder::getCombinedWeight().

Referenced by OPENTREP::chooseBestMatchingResultHolder().

23.101.2.6 StringSet OPENTREP::ResultCombination::getCorrectedStringSet ( ) const

Get the list of corrected strings for the best matching ResultHolder object. If there is no such object, the method throws a NoBestMatchingResultHolderException exception.

Definition at line 66 of file ResultCombination.cpp.

References getBestMatchingResultHolder(), and OPENTREP::ResultHolder::getCorrectedStringSet().

Referenced by OPENTREP::chooseBestMatchingResultHolder().

23.101.2.7 void OPENTREP::ResultCombination::displayXapianPercentages ( ) const

Display the Xapian matching percentages for all the matching documents.

Definition at line 111 of file ResultCombination.cpp.

References OPENTREP::ResultHolder::displayXapianPercentages().

23.101.2.8 void OPENTREP::ResultCombination::calculateEnvelopeWeights ( ) const

Calculate/set the envelope weights for all the matching documents

Definition at line 125 of file ResultCombination.cpp.

References OPENTREP::ResultHolder::calculateEnvelopeWeights().

23.101.2.9 void OPENTREP::ResultCombination::calculateCodeMatches ( ) const

Calculate/set the IATA/ICAO code matching weights for all the matching documents

Definition at line 139 of file ResultCombination.cpp.

References OPENTREP::ResultHolder::calculateCodeMatches().

23.101.2.10 void OPENTREP::ResultCombination::calculatePageRanks ( ) const

Calculate/set the PageRanks for all the matching documents

Definition at line 153 of file ResultCombination.cpp.

References OPENTREP::ResultHolder::calculatePageRanks().

23.101.2.11 void OPENTREP::ResultCombination::calculateHeuristicWeights ( ) const

Calculate/set the heuristic weights for all the matching documents

Definition at line 167 of file ResultCombination.cpp.

References OPENTREP::ResultHolder::calculateHeuristicWeights().

23.101.2.12 void OPENTREP::ResultCombination::calculateCombinedWeights ( ) const

Calculate/set the combined weights for all the matching documents

Definition at line 181 of file ResultCombination.cpp.

References OPENTREP::ResultHolder::calculateCombinedWeights().

23.101.2.13 bool OPENTREP::ResultCombination::chooseBestMatchingResultHolder ( )

Choose the best matching ResultHolder object from the underlying list.

Returns

bool Whether or not a best matching ResultHolder exists.

Definition at line 195 of file ResultCombination.cpp.

References OPENTREP::ResultHolder::describeKey(), describeKey(), OPENTREP::ResultHolder::getCombinedWeight(), and OPENTREP_LOG_DEBUG.

Referenced by OPENTREP::chooseBestMatchingResultHolder().

23.101.2.14 void OPENTREP::ResultCombination::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::BomAbstract.

Definition at line 102 of file ResultCombination.cpp.

References toString().

23.101.2.15 void OPENTREP::ResultCombination::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::BomAbstract.

Definition at line 107 of file ResultCombination.cpp.

23.101.2.16 std::string OPENTREP::ResultCombination::toString ( ) const [virtual]

Get the serialised version of the Business Object.

Implements OPENTREP::BomAbstract.

Definition at line 85 of file ResultCombination.cpp.

References describeShortKey(), and OPENTREP::ResultHolder::toString().

Referenced by toStream().

23.101.2.17 std::string OPENTREP::ResultCombination::describeKey ( ) const [virtual]

Get a string describing the whole key (differentiating two objects at any level).

Implements OPENTREP::BomAbstract.

Definition at line 80 of file ResultCombination.cpp.

References describeShortKey().

Referenced by chooseBestMatchingResultHolder().

23.101.2.18 std::string OPENTREP::ResultCombination::describeShortKey ( ) const [virtual]

Get a string describing the short key (differentiating two objects at the same level).

Implements OPENTREP::BomAbstract.

Definition at line 73 of file ResultCombination.cpp.

Referenced by OPENTREP::chooseBestMatchingResultHolder(), describeKey(), and toString().

23.101.3 Friends And Related Function Documentation

23.101.3.1 friend class FacResultCombination [friend]

Definition at line 26 of file ResultCombination.hpp.

The documentation for this class was generated from the following files:

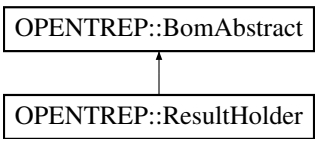
- opentrep/bom/ResultCombination.hpp
- opentrep/bom/ResultCombination.cpp

23.102 OPENTREP::ResultHolder Class Reference

Class wrapping functions on a list of Result objects.

`#include <opentrep/bom/ResultHolder.hpp>`

Inheritance diagram for OPENTREP::ResultHolder:



Public Member Functions

- const TravelQuery_T & getQueryString () const
- const ResultList_T & getResultList () const
- const Percentage_T & getCombinedWeight () const
- StringSet getCorrectedStringSet () const
- void setCombinedWeight (const Percentage_T &iPercentage)
- void displayXapianPercentages () const
- void calculateEnvelopeWeights () const
- void calculateCodeMatches () const
- void calculatePageRanks () const
- void calculateHeuristicWeights () const
- void calculateCombinedWeights ()
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- std::string toString () const
- std::string describeKey () const
- std::string describeShortKey () const

Friends

- class FacResultCombination
- class FacResultHolder

23.102.1 Detailed Description

Class wrapping functions on a list of Result objects.

Definition at line 26 of file ResultHolder.hpp.

23.102.2 Member Function Documentation

23.102.2.1 const TravelQuery_T& OPENTREP::ResultHolder::getQueryString ( ) const [inline]

Get the query string.

Definition at line 34 of file ResultHolder.hpp.

23.102.2.2 const ResultList_T& OPENTREP::ResultHolder::getResultList ( ) const [inline]

Retrieve the list of result objects.

Definition at line 41 of file ResultHolder.hpp.

Referenced by OPENTREP::createPlaces().

23.102.2.3 const Percentage_T& OPENTREP::ResultHolder::getCombinedWeight ( ) const [inline]

Get the combined weight, for all the rules (full-text, PageRank, etc)

Definition at line 48 of file ResultHolder.hpp.

Referenced by OPENTREP::ResultCombination::chooseBestMatchingResultHolder(), and OPENTREP::ResultCombination::getBestMatchingWeight().

23.102.2.4 StringSet OPENTREP::ResultHolder::getCorrectedStringSet ( ) const

Get the list of corrected strings

Definition at line 78 of file ResultHolder.cpp.

References OPENTREP::Result::getCorrectedTravelQuery(), and OPENTREP::StringSet::push_back().

Referenced by OPENTREP::ResultCombination::getCorrectedStringSet().

23.102.2.5 void OPENTREP::ResultHolder::setCombinedWeight (const Percentage_T & *iPercentage*) [inline]

Set the combined weight, for all the rules (full-text, PageRank, etc)

Definition at line 63 of file ResultHolder.hpp.

Referenced by calculateCombinedWeights().

23.102.2.6 void OPENTREP::ResultHolder::displayXapianPercentages ( ) const

Display the Xapian matching percentages for all the matching documents.

Definition at line 96 of file ResultHolder.cpp.

References OPENTREP::Result::displayXapianPercentages().

Referenced by OPENTREP::ResultCombination::displayXapianPercentages().

23.102.2.7 void OPENTREP::ResultHolder::calculateEnvelopeWeights ( ) const

Calculate/set the envelope weights for all the matching documents

Definition at line 109 of file ResultHolder.cpp.

References OPENTREP::Result::calculateEnvelopeWeights().

Referenced by OPENTREP::ResultCombination::calculateEnvelopeWeights().

23.102.2.8 void OPENTREP::ResultHolder::calculateCodeMatches ( ) const

Calculate/set the IATA/ICAO code matching weights for all the matching documents

Definition at line 122 of file ResultHolder.cpp.

References OPENTREP::Result::calculateCodeMatches().

Referenced by OPENTREP::ResultCombination::calculateCodeMatches().

23.102.2.9 void OPENTREP::ResultHolder::calculatePageRanks ( ) const

Calculate/set the PageRanks for all the matching documents

Definition at line 135 of file ResultHolder.cpp.

References OPENTREP::Result::calculatePageRanks().

Referenced by OPENTREP::ResultCombination::calculatePageRanks().

23.102.2.10 void OPENTREP::ResultHolder::calculateHeuristicWeights ( ) const

Calculate/set the heuristic weights for all the matching documents

Definition at line 148 of file ResultHolder.cpp.

References OPENTREP::Result::calculateHeuristicWeights().

Referenced by OPENTREP::ResultCombination::calculateHeuristicWeights().

23.102.2.11 void OPENTREP::ResultHolder::calculateCombinedWeights ( )

Calculate/set the combined weights for all the matching documents

Definition at line 161 of file ResultHolder.cpp.

References OPENTREP::Result::calculateCombinedWeights(), describeKey(), OPENTREP::Result::getBestCombinedWeight(), OPENTREP::K_DEFAULT_ATTENUATION_FCTR, OPENTREP_LOG_DEBUG, and setCombinedWeight().

Referenced by OPENTREP::ResultCombination::calculateCombinedWeights().

23.102.2.12 void OPENTREP::ResultHolder::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::BomAbstract.

Definition at line 69 of file ResultHolder.cpp.

References toString().

23.102.2.13 void OPENTREP::ResultHolder::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::BomAbstract.

Definition at line 74 of file ResultHolder.cpp.

23.102.2.14 std::string OPENTREP::ResultHolder::toString ( ) const [virtual]

Get the serialised version of the Business Object.

Implements OPENTREP::BomAbstract.

Definition at line 49 of file ResultHolder.cpp.

References describeShortKey(), and OPENTREP::Result::toString().

Referenced by OPENTREP::searchString(), toStream(), and OPENTREP::ResultCombination::toString().

23.102.2.15 std::string OPENTREP::ResultHolder::describeKey ( ) const [virtual]

Get a string describing the whole key (differentiating two objects at any level).

Implements OPENTREP::BomAbstract.

Definition at line 44 of file ResultHolder.cpp.

References describeShortKey().

Referenced by calculateCombinedWeights(), and OPENTREP::ResultCombination::chooseBestMatchingResultHolder().

23.102.2.16 std::string OPENTREP::ResultHolder::describeShortKey ( ) const [virtual]

Get a string describing the short key (differentiating two objects at the same level).

Implements OPENTREP::BomAbstract.

Definition at line 37 of file ResultHolder.cpp.

Referenced by OPENTREP::chooseBestMatchingResultHolder(), describeKey(), and toString().

23.102.3 Friends And Related Function Documentation

23.102.3.1 friend class FacResultCombination [friend]

Definition at line 27 of file ResultHolder.hpp.

23.102.3.2 friend class FacResultHolder [friend]

Definition at line 28 of file ResultHolder.hpp.

The documentation for this class was generated from the following files:

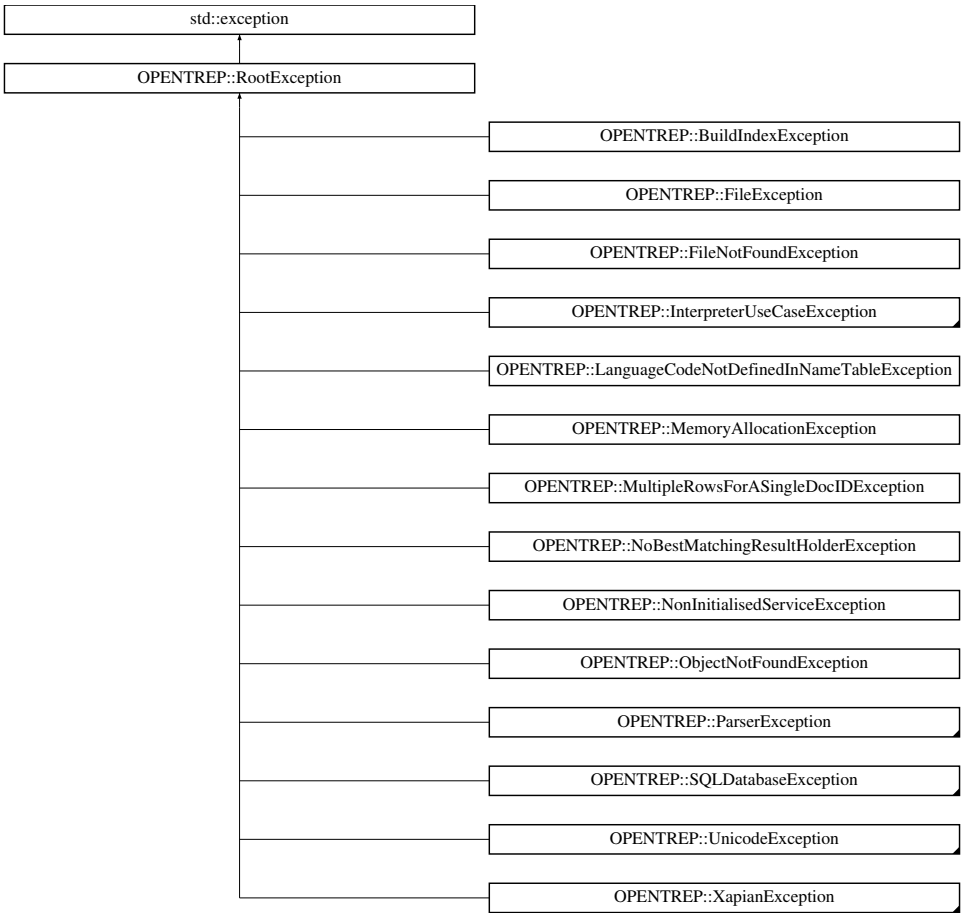
- opentrep/bom/ResultHolder.hpp
- opentrep/bom/ResultHolder.cpp

23.103 OPENTREP::RootException Class Reference

Root of the OpenTREP exceptions.

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::RootException:



Public Member Functions

- RootException (const std::string &iWhat)
- RootException ()
- virtual ~RootException () throw ()
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.103.1 Detailed Description

Root of the OpenTREP exceptions.

All the OpenTREP exceptions inherit from that root, allowing to catch them and to spot them easily when arising in code wrapping the stdair library.

Definition at line 20 of file OPENTREP_exceptions.hpp.

23.103.2 Constructor & Destructor Documentation

23.103.2.1 OPENTREP::RootException::RootException (const std::string & *iWhat*) [inline]

Main Constructor.

Definition at line 25 of file OPENTREP_exceptions.hpp.

23.103.2.2 OPENTREP::RootException::RootException ( ) [inline]

Default constructor.

Definition at line 29 of file OPENTREP_exceptions.hpp.

23.103.2.3 virtual OPENTREP::RootException::~RootException ( ) throw) [inline], [virtual]

Destructor.

Definition at line 34 of file OPENTREP_exceptions.hpp.

23.103.3 Member Function Documentation

23.103.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References _what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.103.4 Member Data Documentation

23.103.4.1 std::string OPENTREP::RootException::_what [protected]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by what().

The documentation for this class was generated from the following file:

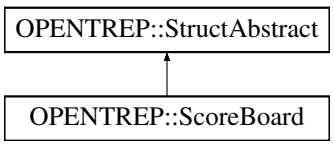
- opentrep/OPENTREP_exceptions.hpp

23.104 OPENTREP::ScoreBoard Struct Reference

Structure holding a board for all the types of score/matching having been performed.

`#include <opentrep/bom/ScoreBoard.hpp>`

Inheritance diagram for OPENTREP::ScoreBoard:



Public Types

- typedef std::map < ScoreType::EN_ScoreType, Score_T > ScoreMap_T

Public Member Functions

- const TravelQuery_T & getQueryString () const
- const ScoreMap_T & getScoreMap () const
- Score_T getScore (const ScoreType &) const
- Score_T getCombinedWeight () const
- void setQueryString (const TravelQuery_T &iQueryString)
- void setScore (const ScoreType &, const Score_T &)
- void setCombinedWeight (const Score_T &iScore)
- Percentage_T calculateCombinedWeight ()
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- std::string describeKey () const
- std::string describe () const
- ScoreBoard (const TravelQuery_T &, const ScoreType &, const Score_T &)
- ScoreBoard (const TravelQuery_T &)
- ScoreBoard (const ScoreBoard &)
- ~ScoreBoard ()

23.104.1 Detailed Description

Structure holding a board for all the types of score/matching having been performed.

Definition at line 22 of file ScoreBoard.hpp.

23.104.2 Member Typedef Documentation

23.104.2.1 typedef std::map<ScoreType::EN_ScoreType, Score_T> OPENTREP::ScoreBoard::ScoreMap_T

(STL) map of scores.

Definition at line 28 of file ScoreBoard.hpp.

23.104.3 Constructor & Destructor Documentation

23.104.3.1 OPENTREP::ScoreBoard::ScoreBoard (const TravelQuery_T & *iQueryString,* const ScoreType & *iType,* const Score_T & *iScore*)

Main constructor.

Definition at line 28 of file ScoreBoard.cpp.

References setScore().

23.104.3.2 OPENTREP::ScoreBoard::ScoreBoard (const TravelQuery_T & *iQueryString*)

Default constructor.

Definition at line 17 of file ScoreBoard.cpp.

23.104.3.3 OPENTREP::ScoreBoard::ScoreBoard (const ScoreBoard & *iScoreBoard*)

Copy constructor.

Definition at line 22 of file ScoreBoard.cpp.

23.104.3.4 OPENTREP::ScoreBoard::~ScoreBoard ( )

Default destructor.

Definition at line 35 of file ScoreBoard.cpp.

23.104.4 Member Function Documentation

23.104.4.1 const TravelQuery_T& OPENTREP::ScoreBoard::getQueryString ( ) const [inline]

Get the query string.

Definition at line 36 of file ScoreBoard.hpp.

23.104.4.2 const ScoreMap_T& OPENTREP::ScoreBoard::getScoreMap ( ) const [inline]

Get the map of scores.

Definition at line 43 of file ScoreBoard.hpp.

23.104.4.3 Score_T OPENTREP::ScoreBoard::getScore (const ScoreType & *iScoreType*) const

Get the score for the given type. If no score value has already been stored for that type, return 0.

Definition at line 40 of file ScoreBoard.cpp.

References OPENTREP::ScoreType::getType().

Referenced by OPENTREP::Result::displayXapianPercentages(), and getCombinedWeight().

23.104.4.4 Score_T OPENTREP::ScoreBoard::getCombinedWeight ( ) const [inline]

Get the combined weight, if existing (0 otherwise).

Definition at line 56 of file ScoreBoard.hpp.

References OPENTREP::ScoreType::COMBINATION, and getScore().

23.104.4.5 void OPENTREP::ScoreBoard::setQueryString (const TravelQuery_T & *iQueryString*) [inline]

Set the query string.

Definition at line 66 of file ScoreBoard.hpp.

23.104.4.6 void OPENTREP::ScoreBoard::setScore (const ScoreType & *iScoreType,* const Score_T & *iScore*)

Set the score for the given type. If no score value has already been stored for that type, create it.

A trick is used to decrease the overall percentage of word combinations, when compared to the whole string. For instance, {"san francisco"} will have a percentage of 99.999%, compared to {"san", "francisco"} which will have a percentage of 99.998%.

A few rules are applied:

- For matching with IATA/ICAO codes:
  - When the query string fully matches with a IATA/ICAO code, the matching percentage is set to 100% (1.00).
  - Otherwise, it is set to 99.999% (0.99999).
- The Xapian matching weight is registered as is.
- For envelope ID:
  - When null, the score is taken as the full matching, i.e., 100% (1.00). There is no need to take 99.999%, as the Xapian matching value above already plays that role.
  - When not null, the score is taken as the minimal default, i.e., 0.10% (0.001).

Definition at line 54 of file ScoreBoard.cpp.

References FloatingPoint< RawType >::AlmostEquals(), OPENTREP::ScoreType::CODE_FULL_MATCH, OPENTREP::ScoreType::describe(), OPENTREP::ScoreType::ENV_ID, OPENTREP::ScoreType::getType(), OPENTRE-

P::K_DEFAULT_ENVELOPE_PCT, OPENTREP::K_DEFAULT_FULL_CODE_MATCH_PCT, OPENTREP::K_DEFAULT_MODIFIED_MATCHING_PCT, and OPENTREP_LOG_ERROR.

Referenced by OPENTREP::Result::calculateCodeMatches(), OPENTREP::Result::calculateEnvelopeWeights(), OPENTREP::Result::calculatePageRanks(), ScoreBoard(), setCombinedWeight(), and OPENTREP::Result::setScoreOnDocMap().

23.104.4.7 void OPENTREP::ScoreBoard::setCombinedWeight (const Score_T & *iScore*) [inline]

Set the combined weight.

Definition at line 103 of file ScoreBoard.hpp.

References OPENTREP::ScoreType::COMBINATION, and setScore().

Referenced by calculateCombinedWeight().

23.104.4.8 Percentage_T OPENTREP::ScoreBoard::calculateCombinedWeight ( )

Calculate the combination of the weights for all the score types, resulting from the full-text matching process, PageRank, user input, etc.

Definition at line 186 of file ScoreBoard.cpp.

References OPENTREP::ScoreType::isIndividualScore(), and setCombinedWeight().

Referenced by OPENTREP::Result::calculateCombinedWeights().

23.104.4.9 void OPENTREP::ScoreBoard::toStream (std::ostream & *ioOut*) const

Dump the structure into an output stream.

Parameters

ostream&	the output stream.

Definition at line 177 of file ScoreBoard.cpp.

References describe().

23.104.4.10 void OPENTREP::ScoreBoard::fromStream (std::istream & *ioIn*) [virtual]

Read a structure from an input stream.

Parameters

istream&	the input stream.

Reimplemented from OPENTREP::StructAbstract.

Definition at line 182 of file ScoreBoard.cpp.

23.104.4.11 std::string OPENTREP::ScoreBoard::describeKey ( ) const

Display the key of the structure.

Returns

std::string Dump of the structure.

Definition at line 150 of file ScoreBoard.cpp.

Referenced by describe().

23.104.4.12 std::string OPENTREP::ScoreBoard::describe ( ) const [virtual]

Display of the structure.

Returns

std::string Dump of the structure.

Implements OPENTREP::StructAbstract.

Definition at line 157 of file ScoreBoard.cpp.

References describeKey(), and OPENTREP::ScoreType::getTypeLabelAsString().

Referenced by OPENTREP::Result::calculateCombinedWeights(), toStream(), and OPENTREP::Result::toString().

The documentation for this struct was generated from the following files:

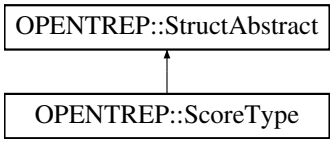
- opentrep/bom/ScoreBoard.hpp
- opentrep/bom/ScoreBoard.cpp

23.105 OPENTREP::ScoreType Struct Reference

Enumeration of score types.

`#include <opentrep/basic/ScoreType.hpp>`

Inheritance diagram for OPENTREP::ScoreType:



Public Types

- enum EN_ScoreType {
  COMBINATION = 0, XAPIAN_PCT, PAGE_RANK, PAX_NB,
  HEURISTIC, ENV_ID, CODE_FULL_MATCH, LAST_VALUE }

Public Member Functions

- const std::string & getLongLabel () const
- EN_ScoreType getType () const
- std::string getTypeAsString () const
- std::string describe () const
- bool isIndividualScore () const
- bool operator== (const EN_ScoreType &) const
- ScoreType (const EN_ScoreType &)
- ScoreType (const char iType)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_ScoreType &)
- static char getTypeLabel (const EN_ScoreType &)
- static std::string getTypeLabelAsString (const EN_ScoreType &)
- static std::string describeLabels ()
- static bool isIndividualScore (const EN_ScoreType &)

23.105.1 Detailed Description

Enumeration of score types.

Each type corresponds to a specific kind of matching:

- Xapian full-text matching
- Page-Rank from the schedule
- Number of passengers registered for that airport/POR (point of reference)

Definition at line 25 of file ScoreType.hpp.

23.105.2 Member Enumeration Documentation

23.105.2.1 enum OPENTREP::ScoreType::EN_ScoreType

Enumerator

COMBINATION

XAPIAN_PCT

PAGE_RANK

PAX_NB

HEURISTIC

ENV_ID

CODE_FULL_MATCH

LAST_VALUE

Definition at line 27 of file ScoreType.hpp.

23.105.3 Constructor & Destructor Documentation

23.105.3.1 OPENTREP::ScoreType::ScoreType (const EN_ScoreType & *iScoreType*)

Constructor.

Definition at line 23 of file ScoreType.cpp.

23.105.3.2 OPENTREP::ScoreType::ScoreType (const char *iType*)

Constructor.

Definition at line 28 of file ScoreType.cpp.

References CODE_FULL_MATCH, COMBINATION, describeLabels(), ENV_ID, HEURISTIC, LAST_VALUE, PAGE_RANK, PAX_NB, and XAPIAN_PCT.

23.105.4 Member Function Documentation

23.105.4.1 const std::string & OPENTREP::ScoreType::getLabel (const EN_ScoreType & *iType*) [static]

Get the label as a string (e.g., "Xapian Percentage" or "Page Rank").

Definition at line 55 of file ScoreType.cpp.

23.105.4.2 const std::string & OPENTREP::ScoreType::getLongLabel ( ) const

Get the label as a string (e.g., "Xapian Percentage" or "Page Rank").

Definition at line 50 of file ScoreType.cpp.

23.105.4.3 char OPENTREP::ScoreType::getTypeLabel (const EN_ScoreType & *iType*) [static]

Get the label as a single char (e.g., 'X' or 'R').

Definition at line 60 of file ScoreType.cpp.

23.105.4.4 std::string OPENTREP::ScoreType::getTypeLabelAsString (const EN_ScoreType & *iType*) [static]

Get the label as a single char (e.g., 'X' or 'R').

Definition at line 66 of file ScoreType.cpp.

Referenced by OPENTREP::ScoreBoard::describe().

23.105.4.5 std::string OPENTREP::ScoreType::describeLabels ( ) [static]

List the labels.

Definition at line 73 of file ScoreType.cpp.

References LAST_VALUE.

Referenced by ScoreType().

23.105.4.6 ScoreType::EN_ScoreType OPENTREP::ScoreType::getType ( ) const

Get the enumerated value.

Definition at line 85 of file ScoreType.cpp.

Referenced by OPENTREP::ScoreBoard::getScore(), and OPENTREP::ScoreBoard::setScore().

23.105.4.7 std::string OPENTREP::ScoreType::getTypeAsString ( ) const

Get the enumerated value as a short string (e.g., 'X' or 'R').

Definition at line 90 of file ScoreType.cpp.

23.105.4.8 std::string OPENTREP::ScoreType::describe ( ) const [virtual]

Give a description of the structure (e.g., "Xapian Percentage" or "Page Rank").

Implements OPENTREP::StructAbstract.

Definition at line 97 of file ScoreType.cpp.

Referenced by OPENTREP::ScoreBoard::setScore().

23.105.4.9 bool OPENTREP::ScoreType::isIndividualScore ( ) const

State whether that score type is valid and is an individual one, i.e., one that is neither COMBINATION nor LAST_-VALUE.

Definition at line 104 of file ScoreType.cpp.

References COMBINATION, and LAST_VALUE.

Referenced by OPENTREP::ScoreBoard::calculateCombinedWeight(), and isIndividualScore().

23.105.4.10 bool OPENTREP::ScoreType::isIndividualScore (const EN_ScoreType & *iTypeEnum*) [static]

State whether that score type is valid and is an individual one, i.e., one that is neither COMBINATION nor LAST_-VALUE.

Definition at line 113 of file ScoreType.cpp.

References isIndividualScore().

23.105.4.11 bool OPENTREP::ScoreType::operator== (const EN_ScoreType & *iType*) const

Comparison operator.

Definition at line 119 of file ScoreType.cpp.

23.105.4.12 void OPENTREP::StructAbstract::toStream (std::ostream & *ioOut*) const [inline], [inherited]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Definition at line 29 of file StructAbstract.hpp.

References OPENTREP::StructAbstract::describe().

23.105.4.13 virtual void OPENTREP::StructAbstract::fromStream (std::istream & *ioIn*) [inline], [virtual], [inherited]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Reimplemented in OPENTREP::ScoreBoard, OPENTREP::StringPartition, OPENTREP::QuerySlices, OPENTREP::WordCombinationHolder, and OPENTREP::StringSet.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

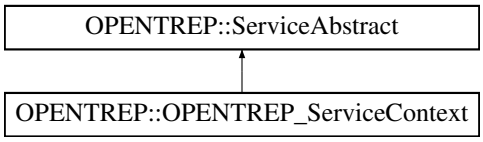
The documentation for this struct was generated from the following files:

- opentrep/basic/ScoreType.hpp
- opentrep/basic/ScoreType.cpp

23.106 OPENTREP::ServiceAbstract Class Reference

`#include <opentrep/service/ServiceAbstract.hpp>`

Inheritance diagram for OPENTREP::ServiceAbstract:



Public Member Functions

- virtual ~ServiceAbstract ()
- virtual void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Protected Member Functions

- ServiceAbstract ()

23.106.1 Detailed Description

Base class for the Service layer.

Definition at line 14 of file ServiceAbstract.hpp.

23.106.2 Constructor & Destructor Documentation

23.106.2.1 virtual OPENTREP::ServiceAbstract::~ServiceAbstract ( ) [inline], [virtual]

Destructor.

Definition at line 18 of file ServiceAbstract.hpp.

23.106.2.2 OPENTREP::ServiceAbstract::ServiceAbstract ( ) [inline], [protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 30 of file ServiceAbstract.hpp.

23.106.3 Member Function Documentation

23.106.3.1 virtual void OPENTREP::ServiceAbstract::toStream (std::ostream & *ioOut*) const [inline], [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Definition at line 22 of file ServiceAbstract.hpp.

23.106.3.2 virtual void OPENTREP::ServiceAbstract::fromStream (std::istream & *ioIn*) [inline], [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Definition at line 26 of file ServiceAbstract.hpp.

Referenced by operator>>().

The documentation for this class was generated from the following file:

- opentrep/service/ServiceAbstract.hpp

23.107 OPENTREP::ServiceUtilities Class Reference

`#include <opentrep/service/ServiceUtilities.hpp>`

Static Public Member Functions

- static bool isDirectory (const std::string &iPath)

23.107.1 Detailed Description

Class wrapping some utility functions.

Definition at line 13 of file ServiceUtilities.hpp.

23.107.2 Member Function Documentation

23.107.2.1 bool OPENTREP::ServiceUtilities::isDirectory (const std::string & *iPath*) [static]

States whether the path given as parameter exists and is a directory.

Parameters

const	std::string& Path to be tested for existence.

Returns

bool Whether or not the path corresponds to a directory.

Definition at line 15 of file ServiceUtilities.cpp.

References OPENTREP_LOG_ERROR.

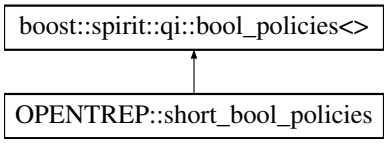
The documentation for this class was generated from the following files:

- opentrep/service/ServiceUtilities.hpp
- opentrep/service/ServiceUtilities.cpp

23.108 OPENTREP::short_bool_policies Struct Reference

`#include <opentrep/basic/BasParserTypes.hpp>`

Inheritance diagram for OPENTREP::short_bool_policies:



Static Public Member Functions

- template<typename Iterator , typename Attribute >
  static bool parse_true (Iterator &first, Iterator const &last, Attribute &attr)
- template<typename Iterator , typename Attribute >
  static bool parse_false (Iterator &first, Iterator const &last, Attribute &attr)

23.108.1 Detailed Description

Boolean parser

Definition at line 71 of file BasParserTypes.hpp.

23.108.2 Member Function Documentation

23.108.2.1 template<typename Iterator , typename Attribute > static bool OPENTREP::short_bool_policies::parse_true (Iterator & *first,* Iterator const & *last,* Attribute & *attr*) [inline], [static]

Definition at line 73 of file BasParserTypes.hpp.

23.108.2.2 template<typename Iterator , typename Attribute > static bool OPENTREP::short_bool_policies::parse_false (Iterator & *first,* Iterator const & *last,* Attribute & *attr*) [inline], [static]

Definition at line 86 of file BasParserTypes.hpp.

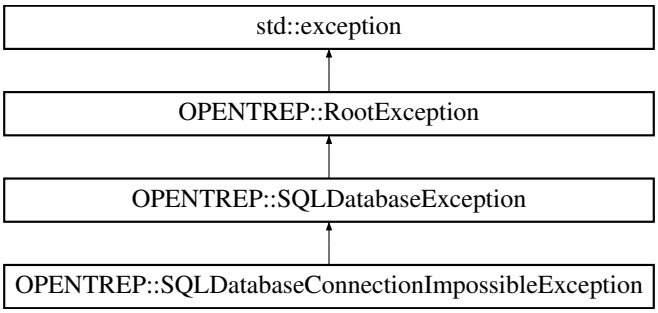
The documentation for this struct was generated from the following file:

- opentrep/basic/BasParserTypes.hpp

23.109 OPENTREP::SQLDatabaseConnectionImpossibleException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::SQLDatabaseConnectionImpossibleException:



Public Member Functions

- SQLDatabaseConnectionImpossibleException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.109.1 Detailed Description

The connection fails with the MySQL database.

Definition at line 297 of file OPENTREP_exceptions.hpp.

23.109.2 Constructor & Destructor Documentation

23.109.2.1 OPENTREP::SQLDatabaseConnectionImpossibleException::SQLDatabaseConnectionImpossibleException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 302 of file OPENTREP_exceptions.hpp.

23.109.3 Member Function Documentation

23.109.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.109.4 Member Data Documentation

23.109.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

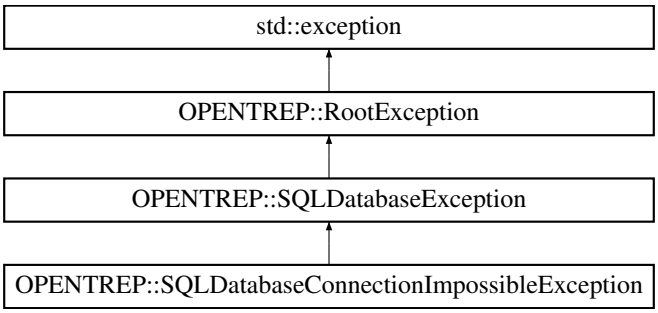
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.110 OPENTREP::SQLDatabaseException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::SQLDatabaseException:



Public Member Functions

- SQLDatabaseException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.110.1 Detailed Description

MySQL Database root exception.

Definition at line 286 of file OPENTREP_exceptions.hpp.

23.110.2 Constructor & Destructor Documentation

23.110.2.1 OPENTREP::SQLDatabaseException::SQLDatabaseException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 291 of file OPENTREP_exceptions.hpp.

23.110.3 Member Function Documentation

23.110.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.110.4 Member Data Documentation

23.110.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

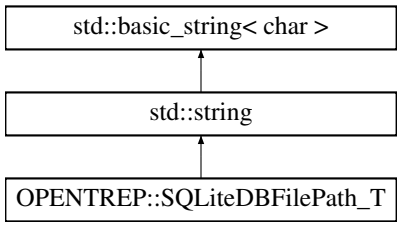
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.111 OPENTREP::SQLiteDBFilePath_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::SQLiteDBFilePath_T:



Public Member Functions

- SQLiteDBFilePath_T (const std::string &iValue)

23.111.1 Detailed Description

SQLite database file-path, corresponding to the (potentially relative) directory name (on the filesystem) where SQLite stores its database.

Definition at line 56 of file OPENTREP_Types.hpp.

23.111.2 Constructor & Destructor Documentation

23.111.2.1 OPENTREP::SQLiteDBFilePath_T::SQLiteDBFilePath_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 58 of file OPENTREP_Types.hpp.

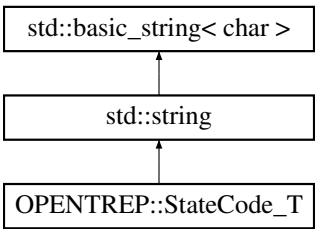
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.112 OPENTREP::StateCode_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::StateCode_T:



Public Member Functions

- StateCode_T (const std::string &iValue)

23.112.1 Detailed Description

State code (e.g., IL for Illinois, USA).

Definition at line 208 of file OPENTREP_Types.hpp.

23.112.2 Constructor & Destructor Documentation

23.112.2.1 OPENTREP::StateCode_T::StateCode_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 210 of file OPENTREP_Types.hpp.

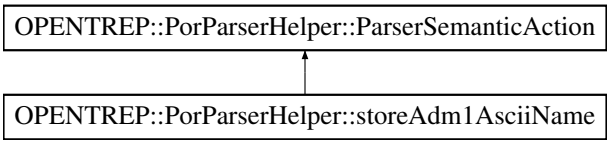
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.113 OPENTREP::PorParserHelper::storeAdm1AsciiName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAdm1AsciiName:



Public Member Functions

- storeAdm1AsciiName (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.113.1 Detailed Description

Store the administrative 1 ASCII name.

Definition at line 287 of file PORParserHelper.hpp.

23.113.2 Constructor & Destructor Documentation

23.113.2.1 OPENTREP::PorParserHelper::storeAdm1AsciiName::storeAdm1AsciiName (Location & *ioLocation*)

Actor Constructor.

Definition at line 435 of file PORParserHelper.cpp.

23.113.3 Member Function Documentation

23.113.3.1 void OPENTREP::PorParserHelper::storeAdm1AsciiName::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 440 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setAdmin1AsciiName().

23.113.4 Member Data Documentation

23.113.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code-

::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

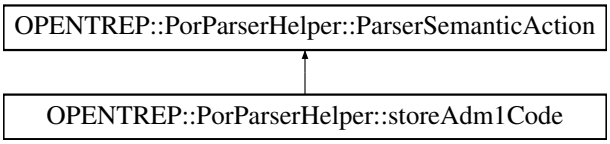
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.114 OPENTREP::PorParserHelper::storeAdm1Code Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAdm1Code:



Public Member Functions

- storeAdm1Code (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.114.1 Detailed Description

Store the administrative 1 code.

Definition at line 267 of file PORParserHelper.hpp.

23.114.2 Constructor & Destructor Documentation

23.114.2.1 OPENTREP::PorParserHelper::storeAdm1Code::storeAdm1Code (Location & *ioLocation*)

Actor Constructor.

Definition at line 403 of file PORParserHelper.cpp.

23.114.3 Member Function Documentation

23.114.3.1 void OPENTREP::PorParserHelper::storeAdm1Code::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 408 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setAdmin1Code().

23.114.4 Member Data Documentation

23.114.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

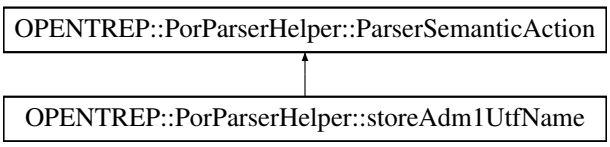
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.115 OPENTREP::PorParserHelper::storeAdm1UtfName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAdm1UtfName:



Public Member Functions

- storeAdm1UtfName (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.115.1 Detailed Description

Store the administrative 1 UTF8 name.

Definition at line 277 of file PORParserHelper.hpp.

23.115.2 Constructor & Destructor Documentation

23.115.2.1 OPENTREP::PorParserHelper::storeAdm1UtfName::storeAdm1UtfName (Location & *ioLocation*)

Actor Constructor.

Definition at line 419 of file PORParserHelper.cpp.

23.115.3 Member Function Documentation

23.115.3.1 void OPENTREP::PorParserHelper::storeAdm1UtfName::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 424 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setAdmin1UtfName().

23.115.4 Member Data Documentation

23.115.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(),

OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

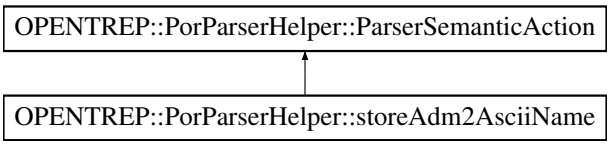
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.116 OPENTREP::PorParserHelper::storeAdm2AsciiName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAdm2AsciiName:



Public Member Functions

- storeAdm2AsciiName (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.116.1 Detailed Description

Store the administrative 1 ASCII name.

Definition at line 317 of file PORParserHelper.hpp.

23.116.2 Constructor & Destructor Documentation

23.116.2.1 OPENTREP::PorParserHelper::storeAdm2AsciiName::storeAdm2AsciiName (Location & *ioLocation*)

Actor Constructor.

Definition at line 483 of file PORParserHelper.cpp.

23.116.3 Member Function Documentation

23.116.3.1 void OPENTREP::PorParserHelper::storeAdm2AsciiName::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 488 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setAdmin2AsciiName().

23.116.4 Member Data Documentation

23.116.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

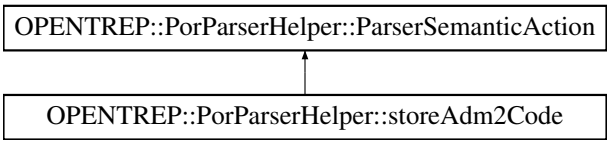
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.117 OPENTREP::PorParserHelper::storeAdm2Code Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAdm2Code:



Public Member Functions

- storeAdm2Code (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.117.1 Detailed Description

Store the administrative 2 code.

Definition at line 297 of file PORParserHelper.hpp.

23.117.2 Constructor & Destructor Documentation

23.117.2.1 OPENTREP::PorParserHelper::storeAdm2Code::storeAdm2Code (Location & *ioLocation*)

Actor Constructor.

Definition at line 451 of file PORParserHelper.cpp.

23.117.3 Member Function Documentation

23.117.3.1 void OPENTREP::PorParserHelper::storeAdm2Code::operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 456 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setAdmin2Code().

23.117.4 Member Data Documentation

23.117.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

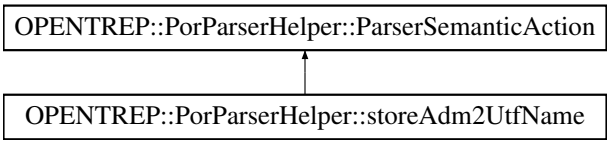
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.118 OPENTREP::PorParserHelper::storeAdm2UtfName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAdm2UtfName:



Public Member Functions

- storeAdm2UtfName (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.118.1 Detailed Description

Store the administrative 1 ASCII name.

Definition at line 307 of file PORParserHelper.hpp.

23.118.2 Constructor & Destructor Documentation

23.118.2.1 OPENTREP::PorParserHelper::storeAdm2UtfName::storeAdm2UtfName (Location & *ioLocation*)

Actor Constructor.

Definition at line 467 of file PORParserHelper.cpp.

23.118.3 Member Function Documentation

23.118.3.1 void OPENTREP::PorParserHelper::storeAdm2UtfName::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 472 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setAdmin2UtfName().

23.118.4 Member Data Documentation

23.118.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2-

Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

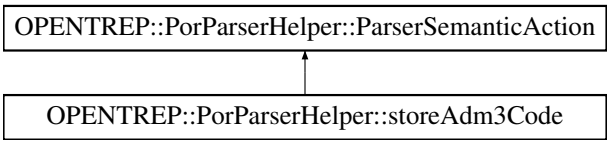
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.119 OPENTREP::PorParserHelper::storeAdm3Code Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAdm3Code:



Public Member Functions

- storeAdm3Code (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.119.1 Detailed Description

Store the administrative 3 code.

Definition at line 327 of file PORParserHelper.hpp.

23.119.2 Constructor & Destructor Documentation

23.119.2.1 OPENTREP::PorParserHelper::storeAdm3Code::storeAdm3Code (Location & *ioLocation*)

Actor Constructor.

Definition at line 499 of file PORParserHelper.cpp.

23.119.3 Member Function Documentation

23.119.3.1 void OPENTREP::PorParserHelper::storeAdm3Code::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 504 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setAdmin3Code().

23.119.4 Member Data Documentation

23.119.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

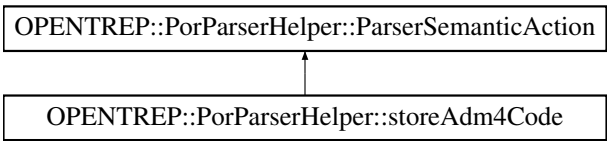
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.120 OPENTREP::PorParserHelper::storeAdm4Code Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAdm4Code:



Public Member Functions

- storeAdm4Code (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.120.1 Detailed Description

Store the administrative 4 code.

Definition at line 337 of file PORParserHelper.hpp.

23.120.2 Constructor & Destructor Documentation

23.120.2.1 OPENTREP::PorParserHelper::storeAdm4Code::storeAdm4Code (Location & *ioLocation*)

Actor Constructor.

Definition at line 515 of file PORParserHelper.cpp.

23.120.3 Member Function Documentation

23.120.3.1 void OPENTREP::PorParserHelper::storeAdm4Code::operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 520 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setAdmin4Code().

23.120.4 Member Data Documentation

23.120.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParser-

Helper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

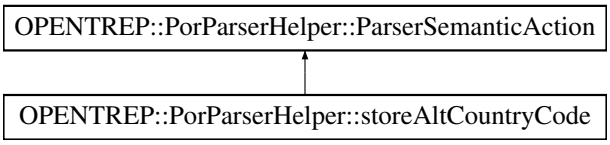
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.121 OPENTREP::PorParserHelper::storeAltCountryCode Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAltCountryCode:



Public Member Functions

- storeAltCountryCode (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.121.1 Detailed Description

Store the alternative country code.

Definition at line 237 of file PORParserHelper.hpp.

23.121.2 Constructor & Destructor Documentation

23.121.2.1 OPENTREP::PorParserHelper::storeAltCountryCode::storeAltCountryCode (Location & *ioLocation*)

Actor Constructor.

Definition at line 355 of file PORParserHelper.cpp.

23.121.3 Member Function Documentation

23.121.3.1 void OPENTREP::PorParserHelper::storeAltCountryCode::operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 360 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setAltCountryCode().

23.121.4 Member Data Documentation

23.121.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

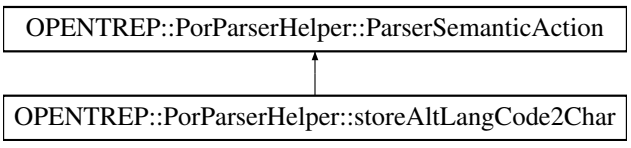
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.122 OPENTREP::PorParserHelper::storeAltLangCode2Char Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAltLangCode2Char:



Public Member Functions

- storeAltLangCode2Char (Location &)
- void operator() (std::vector< boost::uint32_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.122.1 Detailed Description

Store the first 2 characters of the language code of an alternate name.

Definition at line 508 of file PORParserHelper.hpp.

23.122.2 Constructor & Destructor Documentation

23.122.2.1 OPENTREP::PorParserHelper::storeAltLangCode2Char::storeAltLangCode2Char (Location & *ioLocation*)

Actor Constructor.

Definition at line 767 of file PORParserHelper.cpp.

23.122.3 Member Function Documentation

23.122.3.1 void OPENTREP::PorParserHelper::storeAltLangCode2Char::operator() (std::vector< boost::uint32_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 772 of file PORParserHelper.cpp.

References OPENTREP::Location::_itLangCode2Char, OPENTREP::Location::_itLangCodeExt, OPENTREP::Location::_itLangCodeHist, and OPENTREP::PorParserHelper::ParserSemanticAction::_location.

23.122.4 Member Data Documentation

23.122.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OP-

ENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

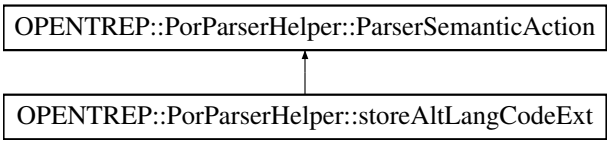
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.123 OPENTREP::PorParserHelper::storeAltLangCodeExt Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAltLangCodeExt:



Public Member Functions

- storeAltLangCodeExt (Location &)
- void operator() (std::vector< boost::uint32_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.123.1 Detailed Description

Store the country-specific part of the language code of an alternate name.

Definition at line 518 of file PORParserHelper.hpp.

23.123.2 Constructor & Destructor Documentation

23.123.2.1 OPENTREP::PorParserHelper::storeAltLangCodeExt::storeAltLangCodeExt (Location & *ioLocation*)

Actor Constructor.

Definition at line 784 of file PORParserHelper.cpp.

23.123.3 Member Function Documentation

23.123.3.1 void OPENTREP::PorParserHelper::storeAltLangCodeExt::operator() (std::vector< boost::uint32_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 789 of file PORParserHelper.cpp.

References OPENTREP::Location::_itLangCode2Char, OPENTREP::Location::_itLangCodeExt, and OPENTREP::PorParserHelper::ParserSemanticAction::_location.

23.123.4 Member Data Documentation

23.123.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

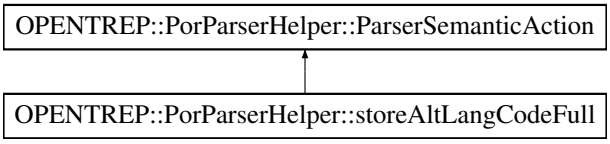
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.124 OPENTREP::PorParserHelper::storeAltLangCodeFull Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAltLangCodeFull:



Public Member Functions

- storeAltLangCodeFull (Location &)
- void operator() (std::vector< boost::uint32_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.124.1 Detailed Description

Store the full language code of an alternate name.

Definition at line 498 of file PORParserHelper.hpp.

23.124.2 Constructor & Destructor Documentation

23.124.2.1 OPENTREP::PorParserHelper::storeAltLangCodeFull::storeAltLangCodeFull (Location & *ioLocation*)

Actor Constructor.

Definition at line 751 of file PORParserHelper.cpp.

23.124.3 Member Function Documentation

23.124.3.1 void OPENTREP::PorParserHelper::storeAltLangCodeFull::operator() (std::vector< boost::uint32_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 756 of file PORParserHelper.cpp.

References OPENTREP::Location::_itLanguageCode, and OPENTREP::PorParserHelper::ParserSemanticAction::_location.

23.124.4 Member Data Documentation

23.124.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(),

OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

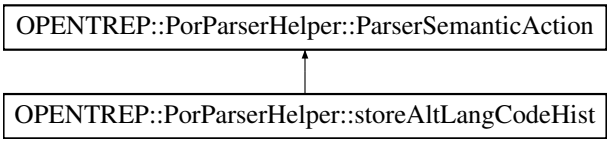
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.125 OPENTREP::PorParserHelper::storeAltLangCodeHist Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAltLangCodeHist:



Public Member Functions

- storeAltLangCodeHist (Location &)
- void operator() (std::vector< boost::uint32_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.125.1 Detailed Description

Store the historical part of the language code of an alternate name.

Definition at line 528 of file PORParserHelper.hpp.

23.125.2 Constructor & Destructor Documentation

23.125.2.1 OPENTREP::PorParserHelper::storeAltLangCodeHist::storeAltLangCodeHist (Location & *ioLocation*)

Actor Constructor.

Definition at line 801 of file PORParserHelper.cpp.

23.125.3 Member Function Documentation

23.125.3.1 void OPENTREP::PorParserHelper::storeAltLangCodeHist::operator() (std::vector< boost::uint32_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 806 of file PORParserHelper.cpp.

References OPENTREP::Location::_itLangCode2Char, OPENTREP::Location::_itLangCodeHist, and OPENTREP::PorParserHelper::ParserSemanticAction::_location.

23.125.4 Member Data Documentation

23.125.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

The documentation for this struct was generated from the following files:

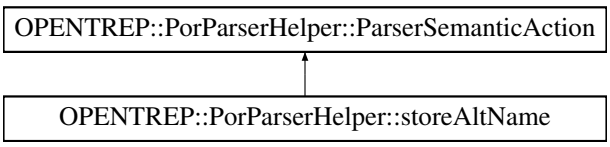
- opentrep/bom/PORParserHelper.hpp

- opentrep/bom/PORParserHelper.cpp

23.126 OPENTREP::PorParserHelper::storeAltName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAltName:



Public Member Functions

- storeAltName (Location &)
- void operator() (std::vector< boost::uint32_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.126.1 Detailed Description

Store an alternate name.

Definition at line 538 of file PORParserHelper.hpp.

23.126.2 Constructor & Destructor Documentation

23.126.2.1 OPENTREP::PorParserHelper::storeAltName::storeAltName (Location & *ioLocation*)

Actor Constructor.

Definition at line 818 of file PORParserHelper.cpp.

23.126.3 Member Function Documentation

23.126.3.1 void OPENTREP::PorParserHelper::storeAltName::operator() (std::vector< boost::uint32_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 823 of file PORParserHelper.cpp.

References OPENTREP::Location::_itLanguageCode, OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::addName().

23.126.4 Member Data Documentation

23.126.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and operator()().

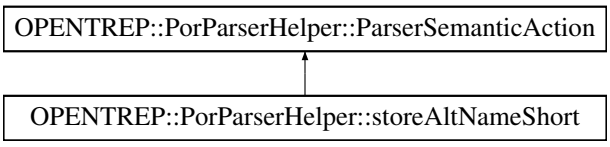
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.127 OPENTREP::PorParserHelper::storeAltNameShort Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAltNameShort:



Public Member Functions

- storeAltNameShort (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.127.1 Detailed Description

Store the short list of alternate names.

Definition at line 107 of file PORParserHelper.hpp.

23.127.2 Constructor & Destructor Documentation

23.127.2.1 OPENTREP::PorParserHelper::storeAltNameShort::storeAltNameShort (Location & *ioLocation*)

Actor Constructor.

Definition at line 151 of file PORParserHelper.cpp.

23.127.3 Member Function Documentation

23.127.3.1 void OPENTREP::PorParserHelper::storeAltNameShort::operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 156 of file PORParserHelper.cpp.

References OPENTREP::Location::_itAltNameShortList, and OPENTREP::PorParserHelper::ParserSemanticAction::_location.

23.127.4 Member Data Documentation

23.127.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2-

Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

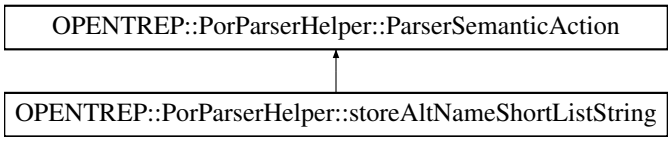
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.128 OPENTREP::PorParserHelper::storeAltNameShortListString Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAltNameShortListString:



Public Member Functions

- storeAltNameShortListString (Location &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.128.1 Detailed Description

Store the short list of alternate names.

Definition at line 117 of file PORParserHelper.hpp.

23.128.2 Constructor & Destructor Documentation

23.128.2.1 OPENTREP::PorParserHelper::storeAltNameShortListString::storeAltNameShortListString (Location & *ioLocation*)

Actor Constructor.

Definition at line 169 of file PORParserHelper.cpp.

23.128.3 Member Function Documentation

23.128.3.1 void OPENTREP::PorParserHelper::storeAltNameShortListString::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 175 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::consolidateAltNameShortListString().

23.128.4 Member Data Documentation

23.128.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

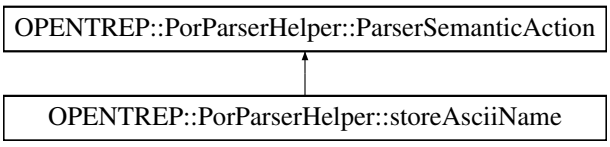
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.129 OPENTREP::PorParserHelper::storeAsciiName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeAsciiName:



Public Member Functions

- storeAsciiName (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.129.1 Detailed Description

Store the ASCII name.

Definition at line 97 of file PORParserHelper.hpp.

23.129.2 Constructor & Destructor Documentation

23.129.2.1 OPENTREP::PorParserHelper::storeAsciiName::storeAsciiName (Location & *ioLocation*)

Actor Constructor.

Definition at line 135 of file PORParserHelper.cpp.

23.129.3 Member Function Documentation

23.129.3.1 void OPENTREP::PorParserHelper::storeAsciiName::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 140 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setAsciiName().

23.129.4 Member Data Documentation

23.129.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(),

OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

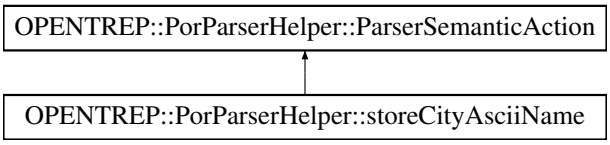
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.130 OPENTREP::PorParserHelper::storeCityAsciiName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeCityAsciiName:



Public Member Functions

- storeCityAsciiName (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.130.1 Detailed Description

Store the parsed city ASCII name.

Definition at line 447 of file PORParserHelper.hpp.

23.130.2 Constructor & Destructor Documentation

23.130.2.1 OPENTREP::PorParserHelper::storeCityAsciiName::storeCityAsciiName (Location & *ioLocation*)

Actor Constructor.

Definition at line 672 of file PORParserHelper.cpp.

23.130.3 Member Function Documentation

23.130.3.1 void OPENTREP::PorParserHelper::storeCityAsciiName::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 677 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setCityAsciiName().

23.130.4 Member Data Documentation

23.130.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

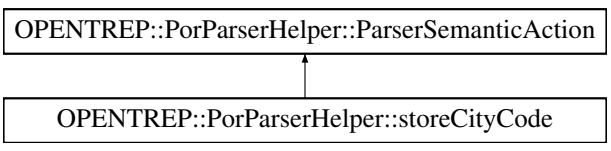
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.131 OPENTREP::PorParserHelper::storeCityCode Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeCityCode:



Public Member Functions

- storeCityCode (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.131.1 Detailed Description

Store the parsed city IATA code.

Definition at line 427 of file PORParserHelper.hpp.

23.131.2 Constructor & Destructor Documentation

23.131.2.1 OPENTREP::PorParserHelper::storeCityCode::storeCityCode (Location & *ioLocation*)

Actor Constructor.

Definition at line 639 of file PORParserHelper.cpp.

23.131.3 Member Function Documentation

23.131.3.1 void OPENTREP::PorParserHelper::storeCityCode::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 644 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setCityCode().

23.131.4 Member Data Documentation

23.131.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName-

::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

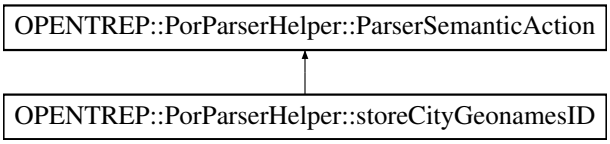
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.132 OPENTREP::PorParserHelper::storeCityGeonamesID Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeCityGeonamesID:



Public Member Functions

- storeCityGeonamesID (Location &)
- void operator() (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.132.1 Detailed Description

Store the parsed city Geonames ID.

Definition at line 457 of file PORParserHelper.hpp.

23.132.2 Constructor & Destructor Documentation

23.132.2.1 OPENTREP::PorParserHelper::storeCityGeonamesID::storeCityGeonamesID (Location & *ioLocation*)

Actor Constructor.

Definition at line 689 of file PORParserHelper.cpp.

23.132.3 Member Function Documentation

23.132.3.1 void OPENTREP::PorParserHelper::storeCityGeonamesID::operator() (unsigned int , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 694 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setCityGeonamesID().

23.132.4 Member Data Documentation

23.132.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

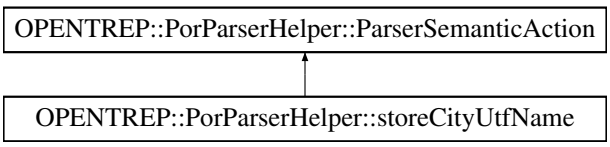
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.133 OPENTREP::PorParserHelper::storeCityUtfName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeCityUtfName:



Public Member Functions

- storeCityUtfName (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.133.1 Detailed Description

Store the parsed city UTF8 name.

Definition at line 437 of file PORParserHelper.hpp.

23.133.2 Constructor & Destructor Documentation

23.133.2.1 OPENTREP::PorParserHelper::storeCityUtfName::storeCityUtfName (Location & *ioLocation*)

Actor Constructor.

Definition at line 655 of file PORParserHelper.cpp.

23.133.3 Member Function Documentation

23.133.3.1 void OPENTREP::PorParserHelper::storeCityUtfName::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 660 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setCityUtfName().

23.133.4 Member Data Documentation

23.133.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass-

::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

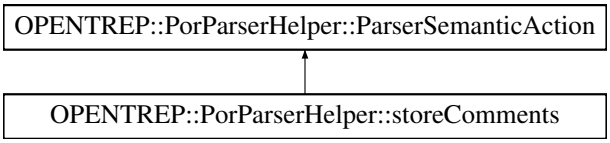
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.134 OPENTREP::PorParserHelper::storeComments Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeComments:



Public Member Functions

- storeComments (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.134.1 Detailed Description

Store the comments.

Definition at line 217 of file PORParserHelper.hpp.

23.134.2 Constructor & Destructor Documentation

23.134.2.1 OPENTREP::PorParserHelper::storeComments::storeComments (Location & *ioLocation*)

Actor Constructor.

Definition at line 321 of file PORParserHelper.cpp.

23.134.3 Member Function Documentation

23.134.3.1 void OPENTREP::PorParserHelper::storeComments::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 326 of file PORParserHelper.cpp.

23.134.4 Member Data Documentation

23.134.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

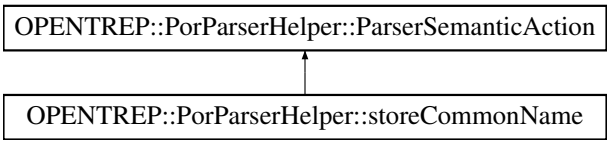
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.135 OPENTREP::PorParserHelper::storeCommonName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeCommonName:



Public Member Functions

- storeCommonName (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.135.1 Detailed Description

Store the common name.

Definition at line 87 of file PORParserHelper.hpp.

23.135.2 Constructor & Destructor Documentation

23.135.2.1 OPENTREP::PorParserHelper::storeCommonName::storeCommonName (Location & *ioLocation*)

Actor Constructor.

Definition at line 119 of file PORParserHelper.cpp.

23.135.3 Member Function Documentation

23.135.3.1 void OPENTREP::PorParserHelper::storeCommonName::operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 124 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setCommonName().

23.135.4 Member Data Documentation

23.135.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

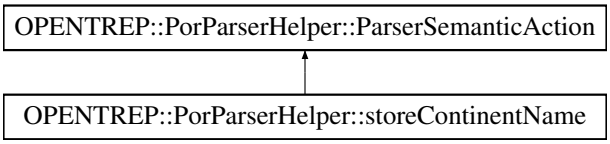
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.136 OPENTREP::PorParserHelper::storeContinentName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeContinentName:



Public Member Functions

- storeContinentName (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.136.1 Detailed Description

Store the parsed continent name.

Definition at line 257 of file PORParserHelper.hpp.

23.136.2 Constructor & Destructor Documentation

23.136.2.1 OPENTREP::PorParserHelper::storeContinentName::storeContinentName (Location & *ioLocation*)

Actor Constructor.

Definition at line 387 of file PORParserHelper.cpp.

23.136.3 Member Function Documentation

23.136.3.1 void OPENTREP::PorParserHelper::storeContinentName::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 392 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setContinentName().

23.136.4 Member Data Documentation

23.136.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2-

Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

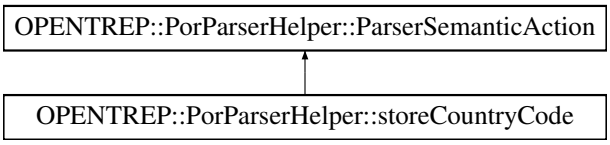
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.137 OPENTREP::PorParserHelper::storeCountryCode Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeCountryCode:



Public Member Functions

- storeCountryCode (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.137.1 Detailed Description

Store the country code.

Definition at line 227 of file PORParserHelper.hpp.

23.137.2 Constructor & Destructor Documentation

23.137.2.1 OPENTREP::PorParserHelper::storeCountryCode::storeCountryCode (Location & *ioLocation*)

Actor Constructor.

Definition at line 339 of file PORParserHelper.cpp.

23.137.3 Member Function Documentation

23.137.3.1 void OPENTREP::PorParserHelper::storeCountryCode::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 344 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setCountryCode().

23.137.4 Member Data Documentation

23.137.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

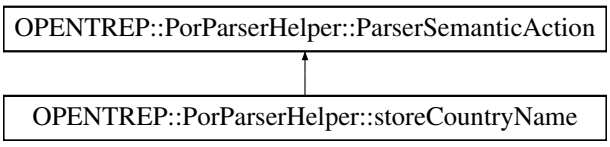
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.138 OPENTREP::PorParserHelper::storeCountryName Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeCountryName:



Public Member Functions

- storeCountryName (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.138.1 Detailed Description

Store the country name.

Definition at line 247 of file PORParserHelper.hpp.

23.138.2 Constructor & Destructor Documentation

23.138.2.1 OPENTREP::PorParserHelper::storeCountryName::storeCountryName (Location & *ioLocation*)

Actor Constructor.

Definition at line 371 of file PORParserHelper.cpp.

23.138.3 Member Function Documentation

23.138.3.1 void OPENTREP::PorParserHelper::storeCountryName::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 376 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setCountryName().

23.138.4 Member Data Documentation

23.138.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(),

OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

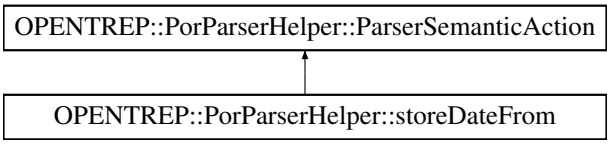
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.139 OPENTREP::PorParserHelper::storeDateFrom Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeDateFrom:



Public Member Functions

- storeDateFrom (Location &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.139.1 Detailed Description

Store the parsed date from.

Definition at line 197 of file PORParserHelper.hpp.

23.139.2 Constructor & Destructor Documentation

23.139.2.1 OPENTREP::PorParserHelper::storeDateFrom::storeDateFrom (Location & *ioLocation*)

Actor Constructor.

Definition at line 287 of file PORParserHelper.cpp.

23.139.3 Member Function Documentation

23.139.3.1 void OPENTREP::PorParserHelper::storeDateFrom::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 292 of file PORParserHelper.cpp.

23.139.4 Member Data Documentation

23.139.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

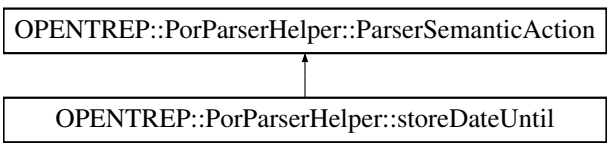
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.140 OPENTREP::PorParserHelper::storeDateUntil Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeDateUntil:



Public Member Functions

- storeDateUntil (Location &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.140.1 Detailed Description

Store the parsed date until.

Definition at line 207 of file PORParserHelper.hpp.

23.140.2 Constructor & Destructor Documentation

23.140.2.1 OPENTREP::PorParserHelper::storeDateUntil::storeDateUntil (Location & *ioLocation*)

Actor Constructor.

Definition at line 304 of file PORParserHelper.cpp.

23.140.3 Member Function Documentation

23.140.3.1 void OPENTREP::PorParserHelper::storeDateUntil::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 309 of file PORParserHelper.cpp.

23.140.4 Member Data Documentation

23.140.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(),

OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

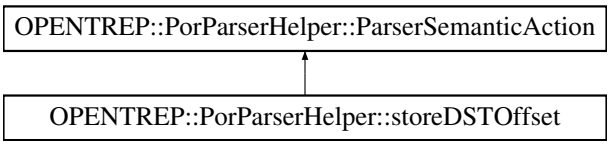
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.141 OPENTREP::PorParserHelper::storeDSTOffset Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeDSTOffset:



Public Member Functions

- storeDSTOffset (Location &)
- void operator() (float, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.141.1 Detailed Description

Store the parsed DST offset.

Definition at line 397 of file PORParserHelper.hpp.

23.141.2 Constructor & Destructor Documentation

23.141.2.1 OPENTREP::PorParserHelper::storeDSTOffset::storeDSTOffset (Location & *ioLocation*)

Actor Constructor.

Definition at line 599 of file PORParserHelper.cpp.

23.141.3 Member Function Documentation

23.141.3.1 void OPENTREP::PorParserHelper::storeDSTOffset::operator() (float , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 604 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setDSTOffset().

23.141.4 Member Data Documentation

23.141.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

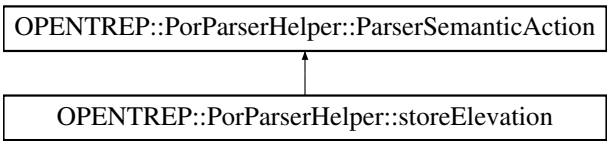
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.142 OPENTREP::PorParserHelper::storeElevation Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeElevation:



Public Member Functions

- storeElevation (Location &)
- void operator() (int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.142.1 Detailed Description

Store the parsed elevation.

Definition at line 357 of file PORParserHelper.hpp.

23.142.2 Constructor & Destructor Documentation

23.142.2.1 OPENTREP::PorParserHelper::storeElevation::storeElevation (Location & *ioLocation*)

Actor Constructor.

Definition at line 544 of file PORParserHelper.cpp.

23.142.3 Member Function Documentation

23.142.3.1 void OPENTREP::PorParserHelper::storeElevation::operator() (int , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 549 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setElevation().

23.142.4 Member Data Documentation

23.142.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParser-

Helper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

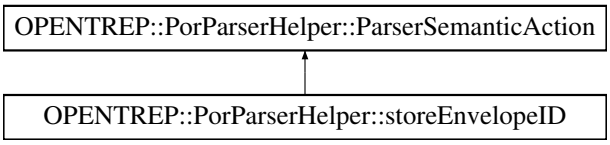
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.143 OPENTREP::PorParserHelper::storeEnvelopeID Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeEnvelopeID:



Public Member Functions

- storeEnvelopeID (Location &)
- void operator() (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.143.1 Detailed Description

Store the parsed envelope ID.

Definition at line 77 of file PORParserHelper.hpp.

23.143.2 Constructor & Destructor Documentation

23.143.2.1 OPENTREP::PorParserHelper::storeEnvelopeID::storeEnvelopeID (Location & *ioLocation*)

Actor Constructor.

Definition at line 105 of file PORParserHelper.cpp.

23.143.3 Member Function Documentation

23.143.3.1 void OPENTREP::PorParserHelper::storeEnvelopeID::operator() (unsigned int , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 110 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setEnvelopeID().

23.143.4 Member Data Documentation

23.143.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

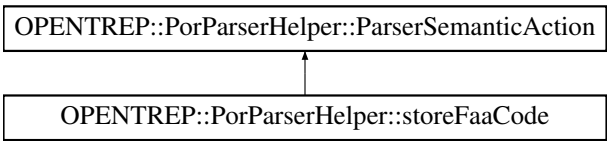
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.144 OPENTREP::PorParserHelper::storeFaaCode Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeFaaCode:



Public Member Functions

- storeFaaCode (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.144.1 Detailed Description

Store the parsed FAA code.

Definition at line 57 of file PORParserHelper.hpp.

23.144.2 Constructor & Destructor Documentation

23.144.2.1 OPENTREP::PorParserHelper::storeFaaCode::storeFaaCode (Location & *ioLocation*)

Actor Constructor.

Definition at line 74 of file PORParserHelper.cpp.

23.144.3 Member Function Documentation

23.144.3.1 void OPENTREP::PorParserHelper::storeFaaCode::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 79 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setFaaCode().

23.144.4 Member Data Documentation

23.144.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code-

::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

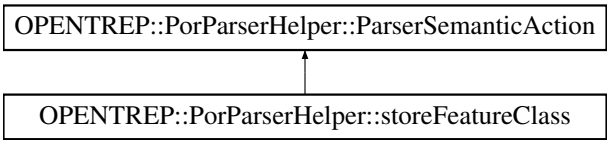
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.145 OPENTREP::PorParserHelper::storeFeatureClass Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeFeatureClass:



Public Member Functions

- storeFeatureClass (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.145.1 Detailed Description

Store the feature class.

Definition at line 167 of file PORParserHelper.hpp.

23.145.2 Constructor & Destructor Documentation

23.145.2.1 OPENTREP::PorParserHelper::storeFeatureClass::storeFeatureClass (Location & *ioLocation*)

Actor Constructor.

Definition at line 239 of file PORParserHelper.cpp.

23.145.3 Member Function Documentation

23.145.3.1 void OPENTREP::PorParserHelper::storeFeatureClass::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 244 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setFeatureClass().

23.145.4 Member Data Documentation

23.145.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

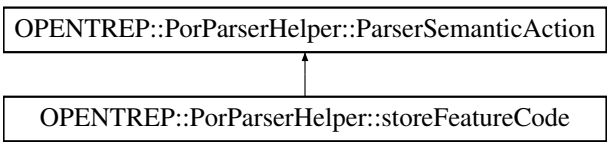
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.146 OPENTREP::PorParserHelper::storeFeatureCode Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeFeatureCode:



Public Member Functions

- storeFeatureCode (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.146.1 Detailed Description

Store the feature code.

Definition at line 177 of file PORParserHelper.hpp.

23.146.2 Constructor & Destructor Documentation

23.146.2.1 OPENTREP::PorParserHelper::storeFeatureCode::storeFeatureCode (Location & *ioLocation*)

Actor Constructor.

Definition at line 256 of file PORParserHelper.cpp.

23.146.3 Member Function Documentation

23.146.3.1 void OPENTREP::PorParserHelper::storeFeatureCode::operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 261 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setFeatureCode().

23.146.4 Member Data Documentation

23.146.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass-

::operator()(), operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

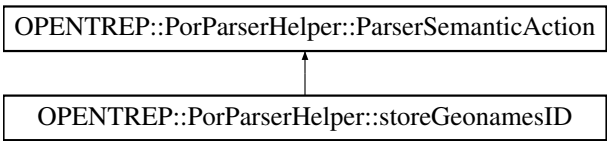
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.147 OPENTREP::PorParserHelper::storeGeonamesID Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeGeonamesID:



Public Member Functions

- storeGeonamesID (Location &)
- void operator() (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.147.1 Detailed Description

Store the parsed Geonames ID.

Definition at line 67 of file PORParserHelper.hpp.

23.147.2 Constructor & Destructor Documentation

23.147.2.1 OPENTREP::PorParserHelper::storeGeonamesID::storeGeonamesID (Location & *ioLocation*)

Actor Constructor.

Definition at line 91 of file PORParserHelper.cpp.

23.147.3 Member Function Documentation

23.147.3.1 void OPENTREP::PorParserHelper::storeGeonamesID::operator() (unsigned int , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 96 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setGeonamesID().

23.147.4 Member Data Documentation

23.147.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

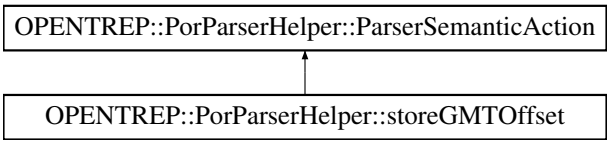
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.148 OPENTREP::PorParserHelper::storeGMTOffset Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeGMTOffset:



Public Member Functions

- storeGMTOffset (Location &)
- void operator() (float, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.148.1 Detailed Description

Store the parsed GMT offset.

Definition at line 387 of file PORParserHelper.hpp.

23.148.2 Constructor & Destructor Documentation

23.148.2.1 OPENTREP::PorParserHelper::storeGMTOffset::storeGMTOffset (Location & *ioLocation*)

Actor Constructor.

Definition at line 586 of file PORParserHelper.cpp.

23.148.3 Member Function Documentation

23.148.3.1 void OPENTREP::PorParserHelper::storeGMTOffset::operator() (float , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 591 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setGMT-Offset().

23.148.4 Member Data Documentation

23.148.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

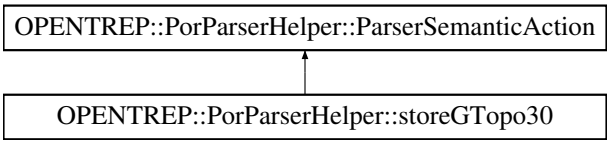
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.149 OPENTREP::PorParserHelper::storeGTopo30 Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeGTopo30:



Public Member Functions

- storeGTopo30 (Location &)
- void operator() (int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.149.1 Detailed Description

Store the parsed GTopo30.

Definition at line 367 of file PORParserHelper.hpp.

23.149.2 Constructor & Destructor Documentation

23.149.2.1 OPENTREP::PorParserHelper::storeGTopo30::storeGTopo30 (Location & *ioLocation*)

Actor Constructor.

Definition at line 557 of file PORParserHelper.cpp.

23.149.3 Member Function Documentation

23.149.3.1 void OPENTREP::PorParserHelper::storeGTopo30::operator() (int , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 562 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setGTopo30().

23.149.4 Member Data Documentation

23.149.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(),

OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

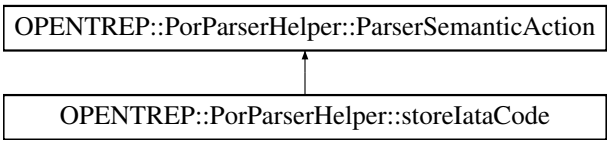
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.150 OPENTREP::PorParserHelper::storelataCode Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storelataCode:



Public Member Functions

- storelataCode (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.150.1 Detailed Description

Store the parsed IATA code.

Definition at line 37 of file PORParserHelper.hpp.

23.150.2 Constructor & Destructor Documentation

23.150.2.1 OPENTREP::PorParserHelper::storelataCode::storelataCode (Location & *ioLocation*)

Actor Constructor.

Definition at line 36 of file PORParserHelper.cpp.

23.150.3 Member Function Documentation

23.150.3.1 void OPENTREP::PorParserHelper::storelataCode::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 41 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setIataCode().

23.150.4 Member Data Documentation

23.150.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

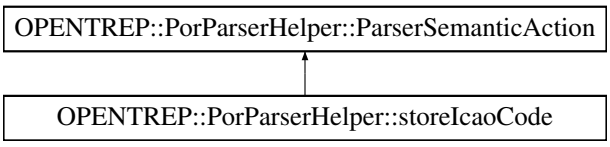
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.151 OPENTREP::PorParserHelper::storeIcaoCode Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeIcaoCode:



Public Member Functions

- storeIcaoCode (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.151.1 Detailed Description

Store the parsed ICAO code.

Definition at line 47 of file PORParserHelper.hpp.

23.151.2 Constructor & Destructor Documentation

23.151.2.1 OPENTREP::PorParserHelper::storeIcaoCode::storeIcaoCode (Location & *ioLocation*)

Actor Constructor.

Definition at line 57 of file PORParserHelper.cpp.

23.151.3 Member Function Documentation

23.151.3.1 void OPENTREP::PorParserHelper::storeIcaoCode::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 62 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setIcaoCode().

23.151.4 Member Data Documentation

23.151.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(),

OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

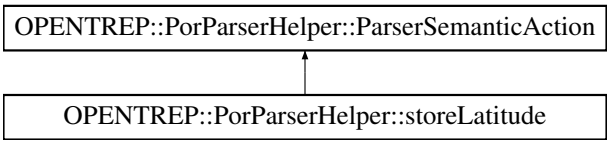
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.152 OPENTREP::PorParserHelper::storeLatitude Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeLatitude:



Public Member Functions

- storeLatitude (Location &)
- void operator() (double, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.152.1 Detailed Description

Store the parsed latitude value.

Definition at line 147 of file PORParserHelper.hpp.

23.152.2 Constructor & Destructor Documentation

23.152.2.1 OPENTREP::PorParserHelper::storeLatitude::storeLatitude (Location & *ioLocation*)

Actor Constructor.

Definition at line 212 of file PORParserHelper.cpp.

23.152.3 Member Function Documentation

23.152.3.1 void OPENTREP::PorParserHelper::storeLatitude::operator() (double , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 217 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setLatitude().

23.152.4 Member Data Documentation

23.152.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

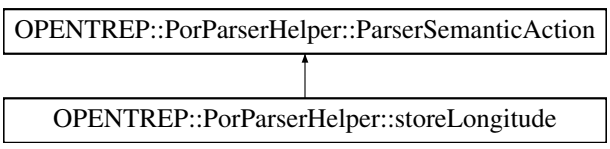
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.153 OPENTREP::PorParserHelper::storeLongitude Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeLongitude:



Public Member Functions

- storeLongitude (Location &)
- void operator() (double, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.153.1 Detailed Description

Store the parsed longitude value.

Definition at line 157 of file PORParserHelper.hpp.

23.153.2 Constructor & Destructor Documentation

23.153.2.1 OPENTREP::PorParserHelper::storeLongitude::storeLongitude (Location & *ioLocation*)

Actor Constructor.

Definition at line 225 of file PORParserHelper.cpp.

23.153.3 Member Function Documentation

23.153.3.1 void OPENTREP::PorParserHelper::storeLongitude::operator() (double , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 230 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setLongitude().

23.153.4 Member Data Documentation

23.153.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code-

::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

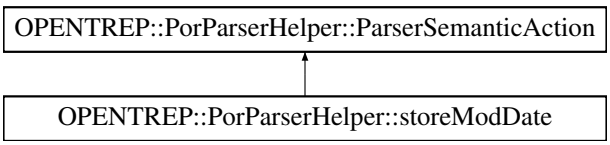
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.154 OPENTREP::PorParserHelper::storeModDate Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeModDate:



Public Member Functions

- storeModDate (Location &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.154.1 Detailed Description

Store the parsed last modification date.

Definition at line 417 of file PORParserHelper.hpp.

23.154.2 Constructor & Destructor Documentation

23.154.2.1 OPENTREP::PorParserHelper::storeModDate::storeModDate (Location & *ioLocation*)

Actor Constructor.

Definition at line 625 of file PORParserHelper.cpp.

23.154.3 Member Function Documentation

23.154.3.1 void OPENTREP::PorParserHelper::storeModDate::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 630 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, OPENTREP::Location::calculateDate(), and OPENTREP::Location::setModificationDate().

23.154.4 Member Data Documentation

23.154.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

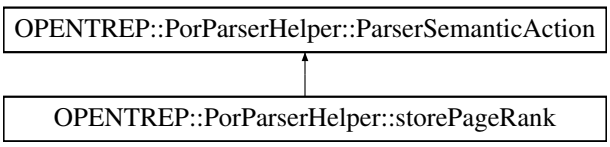
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.155 OPENTREP::PorParserHelper::storePageRank Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storePageRank:



Public Member Functions

- storePageRank (Location &)
- void operator() (double, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.155.1 Detailed Description

Store the parsed PageRank value.

Definition at line 187 of file PORParserHelper.hpp.

23.155.2 Constructor & Destructor Documentation

23.155.2.1 OPENTREP::PorParserHelper::storePageRank::storePageRank (Location & *ioLocation*)

Actor Constructor.

Definition at line 273 of file PORParserHelper.cpp.

23.155.3 Member Function Documentation

23.155.3.1 void OPENTREP::PorParserHelper::storePageRank::operator() (double , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 278 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setPageRank().

23.155.4 Member Data Documentation

23.155.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass-

::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

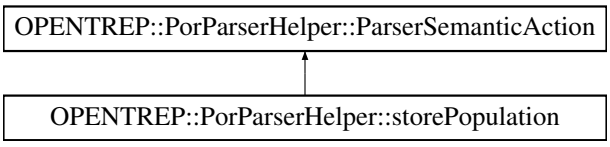
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.156 OPENTREP::PorParserHelper::storePopulation Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storePopulation:



Public Member Functions

- storePopulation (Location &)
- void operator() (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.156.1 Detailed Description

Store the parsed population.

Definition at line 347 of file PORParserHelper.hpp.

23.156.2 Constructor & Destructor Documentation

23.156.2.1 OPENTREP::PorParserHelper::storePopulation::storePopulation (Location & *ioLocation*)

Actor Constructor.

Definition at line 531 of file PORParserHelper.cpp.

23.156.3 Member Function Documentation

23.156.3.1 void OPENTREP::PorParserHelper::storePopulation::operator() (unsigned int , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 536 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setPopulation().

23.156.4 Member Data Documentation

23.156.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

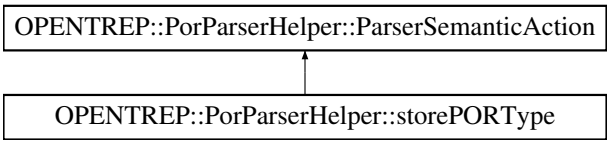
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.157 OPENTREP::PorParserHelper::storePORType Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storePORType:



Public Member Functions

- storePORType (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.157.1 Detailed Description

Store the parsed POR type.

Definition at line 477 of file PORParserHelper.hpp.

23.157.2 Constructor & Destructor Documentation

23.157.2.1 OPENTREP::PorParserHelper::storePORType::storePORType (Location & *ioLocation*)

Actor Constructor.

Definition at line 720 of file PORParserHelper.cpp.

23.157.3 Member Function Documentation

23.157.3.1 void OPENTREP::PorParserHelper::storePORType::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 725 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setIataType().

23.157.4 Member Data Documentation

23.157.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

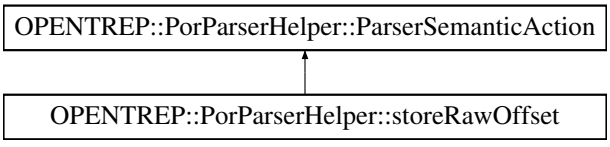
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.158 OPENTREP::PorParserHelper::storeRawOffset Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeRawOffset:



Public Member Functions

- storeRawOffset (Location &)
- void operator() (float, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.158.1 Detailed Description

Store the parsed raw offset.

Definition at line 407 of file PORParserHelper.hpp.

23.158.2 Constructor & Destructor Documentation

23.158.2.1 OPENTREP::PorParserHelper::storeRawOffset::storeRawOffset (Location & *ioLocation*)

Actor Constructor.

Definition at line 612 of file PORParserHelper.cpp.

23.158.3 Member Function Documentation

23.158.3.1 void OPENTREP::PorParserHelper::storeRawOffset::operator() (float , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 617 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setRawOffset().

23.158.4 Member Data Documentation

23.158.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(),

OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

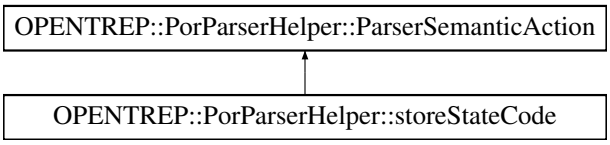
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.159 OPENTREP::PorParserHelper::storeStateCode Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeStateCode:



Public Member Functions

- storeStateCode (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.159.1 Detailed Description

Store the parsed state code.

Definition at line 467 of file PORParserHelper.hpp.

23.159.2 Constructor & Destructor Documentation

23.159.2.1 OPENTREP::PorParserHelper::storeStateCode::storeStateCode (Location & *ioLocation*)

Actor Constructor.

Definition at line 704 of file PORParserHelper.cpp.

23.159.3 Member Function Documentation

23.159.3.1 void OPENTREP::PorParserHelper::storeStateCode::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 709 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setStateCode().

23.159.4 Member Data Documentation

23.159.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

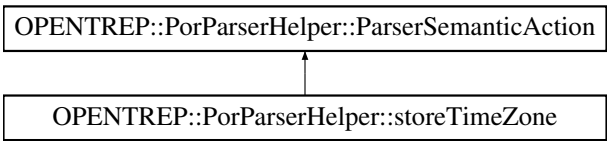
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.160 OPENTREP::PorParserHelper::storeTimeZone Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeTimeZone:



Public Member Functions

- storeTimeZone (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.160.1 Detailed Description

Store the time zone.

Definition at line 377 of file PORParserHelper.hpp.

23.160.2 Constructor & Destructor Documentation

23.160.2.1 OPENTREP::PorParserHelper::storeTimeZone::storeTimeZone (Location & *ioLocation*)

Actor Constructor.

Definition at line 570 of file PORParserHelper.cpp.

23.160.3 Member Function Documentation

23.160.3.1 void OPENTREP::PorParserHelper::storeTimeZone::operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 575 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setTimeZone().

23.160.4 Member Data Documentation

23.160.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParser-

Helper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

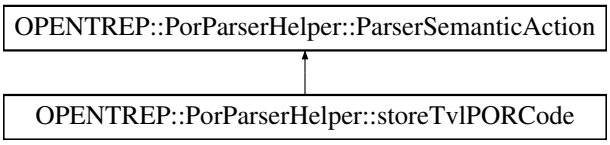
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.161 OPENTREP::PorParserHelper::storeTvlPORCode Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeTvlPORCode:



Public Member Functions

- storeTvlPORCode (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.161.1 Detailed Description

Store the list of IATA codes of the travel-related POR.

Definition at line 127 of file PORParserHelper.hpp.

23.161.2 Constructor & Destructor Documentation

23.161.2.1 OPENTREP::PorParserHelper::storeTvlPORCode::storeTvlPORCode (Location & *ioLocation*)

Actor Constructor.

Definition at line 182 of file PORParserHelper.cpp.

23.161.3 Member Function Documentation

23.161.3.1 void OPENTREP::PorParserHelper::storeTvlPORCode::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 187 of file PORParserHelper.cpp.

References OPENTREP::Location::_itTvlPORList, and OPENTREP::PorParserHelper::ParserSemanticAction::_location.

23.161.4 Member Data Documentation

23.161.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

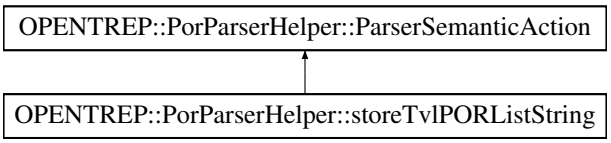
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.162 OPENTREP::PorParserHelper::storeTvlPORListString Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeTvlPORListString:



Public Member Functions

- storeTvlPORListString (Location &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.162.1 Detailed Description

Store the list of IATA codes of the travel-related POR.

Definition at line 137 of file PORParserHelper.hpp.

23.162.2 Constructor & Destructor Documentation

23.162.2.1 OPENTREP::PorParserHelper::storeTvlPORListString::storeTvlPORListString (Location & *ioLocation*)

Actor Constructor.

Definition at line 199 of file PORParserHelper.cpp.

23.162.3 Member Function Documentation

23.162.3.1 void OPENTREP::PorParserHelper::storeTvlPORListString::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 204 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::consolidateTvlPORListString().

23.162.4 Member Data Documentation

23.162.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), O-

PENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), OPENTREP::PorParserHelper::storeWikiLink::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

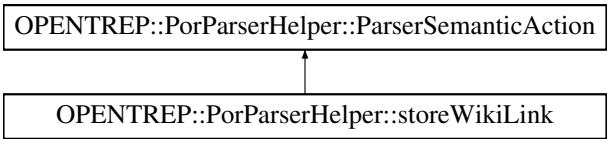
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.163 OPENTREP::PorParserHelper::storeWikiLink Struct Reference

`#include <opentrep/bom/PORParserHelper.hpp>`

Inheritance diagram for OPENTREP::PorParserHelper::storeWikiLink:



Public Member Functions

- storeWikiLink (Location &)
- void operator() (std::vector< uchar_t >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- Location & _location

23.163.1 Detailed Description

Store the parsed Wiki link.

Definition at line 488 of file PORParserHelper.hpp.

23.163.2 Constructor & Destructor Documentation

23.163.2.1 OPENTREP::PorParserHelper::storeWikiLink::storeWikiLink (Location & *ioLocation*)

Actor Constructor.

Definition at line 735 of file PORParserHelper.cpp.

23.163.3 Member Function Documentation

23.163.3.1 void OPENTREP::PorParserHelper::storeWikiLink::operator() (std::vector< uchar_t > , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 740 of file PORParserHelper.cpp.

References OPENTREP::PorParserHelper::ParserSemanticAction::_location, and OPENTREP::Location::setWikiLink().

23.163.4 Member Data Documentation

23.163.4.1 Location& OPENTREP::PorParserHelper::ParserSemanticAction::_location [inherited]

Actor Context.

Definition at line 33 of file PORParserHelper.hpp.

Referenced by OPENTREP::PorParserHelper::storeIataCode::operator()(), OPENTREP::PorParserHelper::storeIcaoCode::operator()(), OPENTREP::PorParserHelper::storeFaaCode::operator()(), OPENTREP::PorParserHelper::storeGeonamesID::operator()(), OPENTREP::PorParserHelper::storeEnvelopeID::operator()(), OPENTREP::PorParserHelper::storeCommonName::operator()(), OPENTREP::PorParserHelper::storeAsciiName::operator()(), OPENTREP::PorParserHelper::storeAltNameShort::operator()(), OPENTREP::PorParserHelper::storeAltNameShortListString::operator()(), OPENTREP::PorParserHelper::storeTvlPORCode::operator()(), OPENTREP::PorParserHelper::storeTvlPORListString::operator()(), OPENTREP::PorParserHelper::storeLatitude::operator()(), OPENTREP::PorParserHelper::storeLongitude::operator()(), OPENTREP::PorParserHelper::storeFeatureClass::operator()(), OPENTREP::PorParserHelper::storeFeatureCode::operator()(), OPENTREP::PorParserHelper::storePageRank::operator()(), OPENTREP::PorParserHelper::storeCountryCode::operator()(), OPENTREP::PorParserHelper::storeAltCountryCode::operator()(), OPENTREP::PorParserHelper::storeCountryName::operator()(), OPENTREP::PorParserHelper::storeContinentName::operator()(), OPENTREP::PorParserHelper::storeAdm1Code::operator()(), OPENTREP::PorParserHelper::storeAdm1UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm1AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm2Code::operator()(), OPENTREP::PorParserHelper::storeAdm2UtfName::operator()(), OPENTREP::PorParserHelper::storeAdm2AsciiName::operator()(), OPENTREP::PorParserHelper::storeAdm3Code::operator()(), OPENTREP::PorParserHelper::storeAdm4Code::operator()(), OPENTREP::PorParserHelper::storePopulation::operator()(), OPENTREP::PorParserHelper::storeElevation::operator()(), OPENTREP::PorParserHelper::storeGTopo30::operator()(), OPENTREP::PorParserHelper::storeTimeZone::operator()(), OPENTREP::PorParserHelper::storeGMTOffset::operator()(), OPENTREP::PorParserHelper::storeDSTOffset::operator()(), OPENTREP::PorParserHelper::storeRawOffset::operator()(), OPENTREP::PorParserHelper::storeModDate::operator()(), OPENTREP::PorParserHelper::storeCityCode::operator()(), OPENTREP::PorParserHelper::storeCityUtfName::operator()(), OPENTREP::PorParserHelper::storeCityAsciiName::operator()(), OPENTREP::PorParserHelper::storeCityGeonamesID::operator()(), OPENTREP::PorParserHelper::storeStateCode::operator()(), OPENTREP::PorParserHelper::storePORType::operator()(), operator()(), OPENTREP::PorParserHelper::storeAltLangCodeFull::operator()(), OPENTREP::PorParserHelper::storeAltLangCode2Char::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeExt::operator()(), OPENTREP::PorParserHelper::storeAltLangCodeHist::operator()(), and OPENTREP::PorParserHelper::storeAltName::operator()().

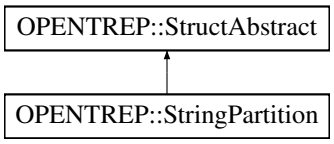
The documentation for this struct was generated from the following files:

- opentrep/bom/PORParserHelper.hpp
- opentrep/bom/PORParserHelper.cpp

23.164 OPENTREP::StringPartition Struct Reference

`#include <opentrep/bom/StringPartition.hpp>`

Inheritance diagram for OPENTREP::StringPartition:



Public Types

- typedef std::list< StringSet > StringPartition_T

Public Member Functions

- void push_back (const StringSet &iStringSet)
- size_t size () const
- bool empty () const
- void clear ()
- const std::string & getInitialString () const
- StringSet calculateUniqueCombinations () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string describeKey () const
- std::string describe () const
- StringPartition (const std::string &iStringToBePartitioned)
- ~StringPartition ()

Public Attributes

- std::string _initialString
- StringPartition_T _partition

23.164.1 Detailed Description

Class holding a string partition.

A string partition contains all the ways to combine, serially, all the words of a given string.

For instance, "rio de janeiro" will give:

- {
- – {"rio", "de", "janeiro"}
  – {"rio", "de janeiro"}
  – {"rio de", "janeiro"}
  – {"rio de janeiro"}
- }

Definition at line 35 of file StringPartition.hpp.

23.164.2 Member Typedef Documentation

23.164.2.1 typedef std::list<StringSet> OPENTREP::StringPartition::StringPartition_T

Type gathering all the partitions of a string.

Definition at line 40 of file StringPartition.hpp.

23.164.3 Constructor & Destructor Documentation

23.164.3.1 OPENTREP::StringPartition::StringPartition (const std::string & *iStringToBePartitioned*)

Constructor.

Parameters

const	std::string& The string for which the partitions are sought

Definition at line 17 of file StringPartition.cpp.

23.164.3.2 OPENTREP::StringPartition::~StringPartition ( )

Default destructor.

Definition at line 23 of file StringPartition.cpp.

23.164.4 Member Function Documentation

23.164.4.1 void OPENTREP::StringPartition::push_back (const StringSet & *iStringSet*)

Add an item (StringSet) into the list.

Note

When the given string set is empty (zero-length), it is (obviously) not added to the list

Definition at line 27 of file StringPartition.cpp.

References _partition, and OPENTREP::StringSet::empty().

23.164.4.2 size_t OPENTREP::StringPartition::size ( ) const

Return the size of the list.

Definition at line 34 of file StringPartition.cpp.

References _partition.

Referenced by BOOST_AUTO_TEST_CASE().

23.164.4.3 bool OPENTREP::StringPartition::empty ( ) const

Return whether or not the list is empty.

Definition at line 39 of file StringPartition.cpp.

References _partition.

Referenced by OPENTREP::QuerySlices::push_back().

23.164.4.4 void OPENTREP::StringPartition::clear ( )

Empty the list.

Definition at line 44 of file StringPartition.cpp.

References _partition.

23.164.4.5 const std::string& OPENTREP::StringPartition::getInitialString ( ) const [inline]

Get the initial string, that is, the string having been given to be partitioned.

Definition at line 70 of file StringPartition.hpp.

References _initialString.

23.164.4.6 StringSet OPENTREP::StringPartition::calculateUniqueCombinations ( ) const

Return the list of all the unique word combinations, which can be made from the partitions.

For instance, with "rio de janeiro", the list of unique strings would be:

- "rio"
- "de"
- "janeiro"
- "rio de"
- "de janeiro"
- "rio de janeiro"

Returns

StringSet The list of unique strings.

Definition at line 171 of file StringPartition.cpp.

References _partition, OPENTREP::StringSet::_set, and OPENTREP::StringSet::push_back().

23.164.4.7 void OPENTREP::StringPartition::toStream (std::ostream & *ioOut*) const

Dump the structure into an output stream.

Parameters

ostream&	the output stream.

Definition at line 85 of file StringPartition.cpp.

References describe().

23.164.4.8 void OPENTREP::StringPartition::fromStream (std::istream & *ioIn*) [virtual]

Read a structure from an input stream.

Parameters

istream&	the input stream.

Reimplemented from OPENTREP::StructAbstract.

Definition at line 90 of file StringPartition.cpp.

23.164.4.9 std::string OPENTREP::StringPartition::describeKey ( ) const

Get a string describing the whole key (differentiating two objects at any level).

Definition at line 49 of file StringPartition.cpp.

Referenced by describe().

23.164.4.10 std::string OPENTREP::StringPartition::describe ( ) const [virtual]

Get the serialised version of the structure.

Implements OPENTREP::StructAbstract.

Definition at line 56 of file StringPartition.cpp.

References _partition, and describeKey().

Referenced by toStream().

23.164.5 Member Data Documentation

23.164.5.1 std::string OPENTREP::StringPartition::_initialString

String to be partitioned.

Definition at line 153 of file StringPartition.hpp.

Referenced by getInitialString().

23.164.5.2 StringPartition_T OPENTREP::StringPartition::_partition

Partition, i.e., a list containing sub-lists of strings

Definition at line 158 of file StringPartition.hpp.

Referenced by calculateUniqueCombinations(), clear(), describe(), empty(), push_back(), OPENTREP::searchString(), and size().

The documentation for this struct was generated from the following files:

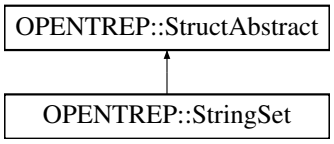
- opentrep/bom/StringPartition.hpp
- opentrep/bom/StringPartition.cpp

23.165 OPENTREP::StringSet Struct Reference

Class holding a set of strings, e.g., {"rio", "de", "janeiro"}.

`#include <opentrep/bom/StringSet.hpp>`

Inheritance diagram for OPENTREP::StringSet:



Public Types

- typedef std::list< std::string > StringSet_T

Public Member Functions

- void push_back (const std::string &)
- void push_back (const StringSet &)
- size_t size () const
- bool empty () const
- void clear ()
- std::string getFirstString () const
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- std::string describeKey () const
- std::string describe () const
- StringSet ()
- StringSet (const StringSet &)
- StringSet (const std::string &)
- ~StringSet ()

Public Attributes

- StringSet_T _set

23.165.1 Detailed Description

Class holding a set of strings, e.g., {"rio", "de", "janeiro"}.

Definition at line 19 of file StringSet.hpp.

23.165.2 Member Typedef Documentation

23.165.2.1 typedef std::list<std::string> OPENTREP::StringSet::StringSet_T

List of strings.

Definition at line 25 of file StringSet.hpp.

23.165.3 Constructor & Destructor Documentation

23.165.3.1 OPENTREP::StringSet::StringSet ( )

Default constructor.

Definition at line 14 of file StringSet.cpp.

23.165.3.2 OPENTREP::StringSet::StringSet (const StringSet & *iStringSet*)

Default copy constructor.

Definition at line 18 of file StringSet.cpp.

References _set.

23.165.3.3 OPENTREP::StringSet::StringSet (const std::string & *iString*)

Main constructor.

Definition at line 23 of file StringSet.cpp.

References _set.

23.165.3.4 OPENTREP::StringSet::~StringSet ( )

Default destructor.

Definition at line 30 of file StringSet.cpp.

23.165.4 Member Function Documentation

23.165.4.1 void OPENTREP::StringSet::push_back (const std::string & *iString*)

Add an item (string) into the list.

Note

When the given string is empty (zero-length), it is (obviously) not added to the list

Definition at line 49 of file StringSet.cpp.

References _set.

Referenced by OPENTREP::StringPartition::calculateUniqueCombinations(), OPENTREP::ResultHolder::getCorrectedStringSet(), and push_back().

23.165.4.2 void OPENTREP::StringSet::push_back (const StringSet & *iStringSet*)

Add all the items (string) into the list.

Definition at line 54 of file StringSet.cpp.

References _set, and push_back().

23.165.4.3 size_t OPENTREP::StringSet::size ( ) const

Return the size of the list.

Definition at line 34 of file StringSet.cpp.

References _set.

23.165.4.4 bool OPENTREP::StringSet::empty ( ) const

Return whether or not the list is empty.

Definition at line 39 of file StringSet.cpp.

References _set.

Referenced by OPENTREP::StringPartition::push_back().

23.165.4.5 void OPENTREP::StringSet::clear ( )

Empty the list.

Definition at line 44 of file StringSet.cpp.

References _set.

23.165.4.6 std::string OPENTREP::StringSet::getFirstString ( ) const

Get the first string of the underlying list/set.

Definition at line 64 of file StringSet.cpp.

References _set.

23.165.4.7 void OPENTREP::StringSet::toStream (std::ostream & *ioOut*) const

Dump the structure into an output stream.

Parameters

ostream&	the output stream.

Definition at line 117 of file StringSet.cpp.

References describe().

23.165.4.8 void OPENTREP::StringSet::fromStream (std::istream & *ioIn*) [virtual]

Read a structure from an input stream.

Parameters

istream&	the input stream.

Reimplemented from OPENTREP::StructAbstract.

Definition at line 122 of file StringSet.cpp.

23.165.4.9 std::string OPENTREP::StringSet::describeKey ( ) const

Get a string describing the whole key (differentiating two objects at any level).

Definition at line 81 of file StringSet.cpp.

Referenced by describe().

23.165.4.10 std::string OPENTREP::StringSet::describe ( ) const [virtual]

Get the serialised version of the structure.

Implements OPENTREP::StructAbstract.

Definition at line 88 of file StringSet.cpp.

References _set, and describeKey().

Referenced by OPENTREP::searchString(), and toStream().

23.165.5 Member Data Documentation

23.165.5.1 StringSet_T OPENTREP::StringSet::_set

String set, i.e., a list of strings

Definition at line 118 of file StringSet.hpp.

Referenced by OPENTREP::StringPartition::calculateUniqueCombinations(), clear(), describe(), empty(), getFirstString(), push_back(), OPENTREP::searchString(), size(), and StringSet().

The documentation for this struct was generated from the following files:

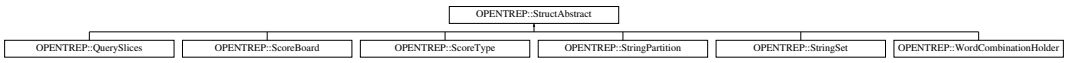
- opentrep/bom/StringSet.hpp
- opentrep/bom/StringSet.cpp

23.166 OPENTREP::StructAbstract Struct Reference

Base class for the light structures.

`#include <opentrep/basic/StructAbstract.hpp>`

Inheritance diagram for OPENTREP::StructAbstract:



Public Member Functions

- virtual ~StructAbstract ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)
- virtual std::string describe () const =0

Protected Member Functions

- StructAbstract ()

23.166.1 Detailed Description

Base class for the light structures.

Definition at line 16 of file StructAbstract.hpp.

23.166.2 Constructor & Destructor Documentation

23.166.2.1 virtual OPENTREP::StructAbstract::~StructAbstract ( ) [inline], [virtual]

Destructor.

Definition at line 22 of file StructAbstract.hpp.

23.166.2.2 OPENTREP::StructAbstract::StructAbstract ( ) [inline], [protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 51 of file StructAbstract.hpp.

23.166.3 Member Function Documentation

23.166.3.1 void OPENTREP::StructAbstract::toStream (std::ostream & *ioOut*) const [inline]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Definition at line 29 of file StructAbstract.hpp.

References describe().

23.166.3.2 virtual void OPENTREP::StructAbstract::fromStream (std::istream & *ioIn*) [inline], [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Reimplemented in OPENTREP::ScoreBoard, OPENTREP::StringPartition, OPENTREP::QuerySlices, OPENTREP::WordCombinationHolder, and OPENTREP::StringSet.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

23.166.3.3 virtual std::string OPENTREP::StructAbstract::describe ( ) const [pure virtual]

Display of the structure.

Returns

std::string Dump of the structure.

Implemented in OPENTREP::ScoreBoard, OPENTREP::StringPartition, OPENTREP::QuerySlices, OPENTREP::WordCombinationHolder, OPENTREP::StringSet, and OPENTREP::ScoreType.

Referenced by toStream().

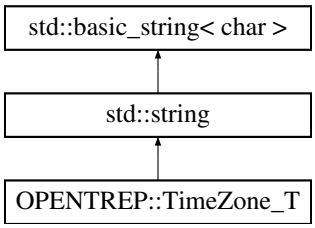
The documentation for this struct was generated from the following file:

- opentrep/basic/StructAbstract.hpp

23.167 OPENTREP::TimeZone_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::TimeZone_T:



Public Member Functions

- TimeZone_T (const std::string &iValue)

23.167.1 Detailed Description

Time-zone (e.g., America/Chicago).

Definition at line 389 of file OPENTREP_Types.hpp.

23.167.2 Constructor & Destructor Documentation

23.167.2.1 OPENTREP::TimeZone_T::TimeZone_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 391 of file OPENTREP_Types.hpp.

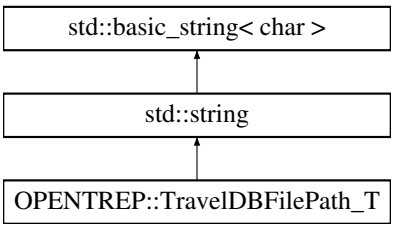
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.168 OPENTREP::TravelDBFilePath_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::TravelDBFilePath_T:



Public Member Functions

- TravelDBFilePath_T (const std::string &iValue)

23.168.1 Detailed Description

Xapian database file-path/name, corresponding to the (potentially relative) directory name (on the filesystem) where Xapian stores its index.

Definition at line 46 of file OPENTREP_Types.hpp.

23.168.2 Constructor & Destructor Documentation

23.168.2.1 OPENTREP::TravelDBFilePath_T::TravelDBFilePath_T (const std::string & *iValue*) [inline],[explicit]

Definition at line 48 of file OPENTREP_Types.hpp.

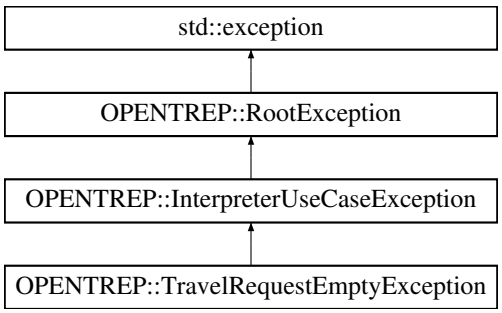
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.169 OPENTREP::TravelRequestEmptyException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::TravelRequestEmptyException:



Public Member Functions

- TravelRequestEmptyException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.169.1 Detailed Description

The travel request is empty.

Definition at line 344 of file OPENTREP_exceptions.hpp.

23.169.2 Constructor & Destructor Documentation

23.169.2.1 OPENTREP::TravelRequestEmptyException::TravelRequestEmptyException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 349 of file OPENTREP_exceptions.hpp.

23.169.3 Member Function Documentation

23.169.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.169.4 Member Data Documentation

23.169.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

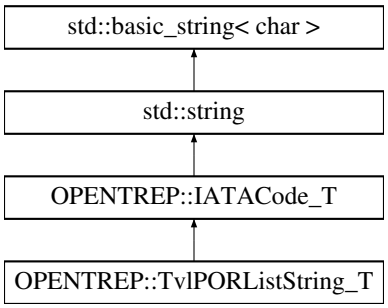
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.170 OPENTREP::TvlPORListString_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::TvlPORListString_T:



Public Member Functions

- TvlPORListString_T (const std::string &iValue)

23.170.1 Detailed Description

List of the IATA codes of the travel-related POR, aggregated within a single string (e.g., "EMB,JCC,SFO" for San Francisco, California, USA).

Definition at line 151 of file OPENTREP_Types.hpp.

23.170.2 Constructor & Destructor Documentation

23.170.2.1 OPENTREP::TvlPORListString_T::TvlPORListString_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 153 of file OPENTREP_Types.hpp.

The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.171 soci::type_conversion< OPENTREP::Place > Struct Template Reference

Specify how the Place class can be converted to (resp. from) values stored into (resp. retrieved from) database, using the SOCI framework.

`#include <opentrep/dbadaptor/DbaPlace.hpp>`

Public Types

- typedef values base_type

Static Public Member Functions

- static void from_base (values const &iPlaceValues, indicator, OPENTREP::Place &)
- static void to_base (const OPENTREP::Place &, values &ioPlaceValues, indicator &)

23.171.1 Detailed Description

template<>struct soci::type_conversion< OPENTREP::Place >

Specify how the Place class can be converted to (resp. from) values stored into (resp. retrieved from) database, using the SOCI framework.

Definition at line 27 of file DbaPlace.hpp.

23.171.2 Member Typedef Documentation

23.171.2.1 typedef values soci::type_conversion< OPENTREP::Place >::base_type

Definition at line 29 of file DbaPlace.hpp.

23.171.3 Member Function Documentation

23.171.3.1 void soci::type_conversion< OPENTREP::Place >::from_base (values const & *iPlaceValues*, indicator , OPENTREP::Place & *ioPlace*) [static]

Fill an Place object from the database values.

Definition at line 19 of file DbaPlace.cpp.

References OPENTREP::Place::addName(), OPENTREP::DEFAULT_DATE_STR, OPENTREP::Place::getKey(), OPENTREP::K_DEFAULT_PAGE_RANK, OPENTREP::Place::resetIndexSets(), OPENTREP::Place::resetMatrix(), OPENTREP::Place::setAdmin1AsciiName(), OPENTREP::Place::setAdmin1Code(), OPENTREP::Place::setAdmin1UtfName(), OPENTREP::Place::setAdmin2AsciiName(), OPENTREP::Place::setAdmin2Code(), OPENTREP::Place::setAdmin2UtfName(), OPENTREP::Place::setAdmin3Code(), OPENTREP::Place::setAdmin4Code(), OPENTREP::Place::setAltCountryCode(), OPENTREP::Place::setAsciiName(), OPENTREP::Place::setCityAsciiName(), OPENTREP::Place::setCityCode(), OPENTREP::Place::setCityUtfName(), OPENTREP::Place-

::setComment(), OPENTREP::Place::setCommonName(), OPENTREP::Place::setContinentName(), OPENTREP::Place::setCountryCode(), OPENTREP::Place::setCountryName(), OPENTREP::Place::setDateEnd(), OPENTREP::Place::setDateFrom(), OPENTREP::Place::setDSTOffset(), OPENTREP::Place::setElevation(), OPENTREP::Place::setEnvelopeID(), OPENTREP::Place::setFaaCode(), OPENTREP::Place::setFeatureClass(), OPENTREP::Place::setFeatureCode(), OPENTREP::Place::setGeonamesID(), OPENTREP::Place::setGMTOffset(), OPENTREP::Place::setGTopo30(), OPENTREP::Place::setIataCode(), OPENTREP::Place::setIataType(), OPENTREP::Place::setLatitude(), OPENTREP::Place::setLongitude(), OPENTREP::Place::setModificationDate(), OPENTREP::Place::setPageRank(), OPENTREP::Place::setPopulation(), OPENTREP::Place::setRawOffset(), OPENTREP::Place::setStateCode(), OPENTREP::Place::setTimeZone(), and OPENTREP::Place::setWikiLink().

23.171.3.2 void soci::type_conversion< OPENTREP::Place >::to_base (const OPENTREP::Place & *iPlace,* values & *ioPlaceValues,* indicator & *ioIndicator*) [static]

Fill the database values from an Place object.

Definition at line 264 of file DbaPlace.cpp.

References OPENTREP::Place::getCityCode(), OPENTREP::Place::getContinentName(), OPENTREP::Place::getCountryCode(), OPENTREP::Place::getCountryName(), OPENTREP::Place::getFaaCode(), OPENTREP::Place::getGeonamesID(), OPENTREP::Place::getIataCode(), OPENTREP::Place::getIcaoCode(), OPENTREP::Place::getLatitude(), OPENTREP::Place::getLongitude(), OPENTREP::Place::getPageRank(), OPENTREP::Place::getStateCode(), OPENTREP::Place::getTimeZone(), and OPENTREP::Place::getWikiLink().

The documentation for this struct was generated from the following files:

- opentrep/dbadaptor/DbaPlace.hpp
- opentrep/dbadaptor/DbaPlace.cpp

23.172 TypeWithSize< size > Class Template Reference

`#include <opentrep/basic/float_utils_google.hpp>`

Public Types

- typedef void UInt

23.172.1 Detailed Description

template<size_t size>class TypeWithSize< size >

Definition at line 60 of file float_utils_google.hpp.

23.172.2 Member Typedef Documentation

23.172.2.1 template<size_t size> typedef void TypeWithSize< size >::UInt

Definition at line 64 of file float_utils_google.hpp.

The documentation for this class was generated from the following file:

- opentrep/basic/float_utils_google.hpp

23.173 TypeWithSize< 4 > Class Template Reference

`#include <opentrep/basic/float_utils_google.hpp>`

Public Types

- typedef int Int
- typedef unsigned int UInt

23.173.1 Detailed Description

template<>class TypeWithSize< 4 >

Definition at line 69 of file float_utils_google.hpp.

23.173.2 Member Typedef Documentation

23.173.2.1 typedef int TypeWithSize< 4 >::Int

Definition at line 75 of file float_utils_google.hpp.

23.173.2.2 typedef unsigned int TypeWithSize< 4 >::UInt

Definition at line 76 of file float_utils_google.hpp.

The documentation for this class was generated from the following file:

- opentrep/basic/float_utils_google.hpp

23.174 TypeWithSize< 8 > Class Template Reference

`#include <opentrep/basic/float_utils_google.hpp>`

Public Types

- typedef long long Int
- typedef unsigned long long UInt

23.174.1 Detailed Description

template<>class TypeWithSize< 8 >

Definition at line 81 of file float_utils_google.hpp.

23.174.2 Member Typedef Documentation

23.174.2.1 typedef long long TypeWithSize< 8 >::Int

Definition at line 87 of file float_utils_google.hpp.

23.174.2.2 typedef unsigned long long TypeWithSize< 8 >::UInt

Definition at line 88 of file float_utils_google.hpp.

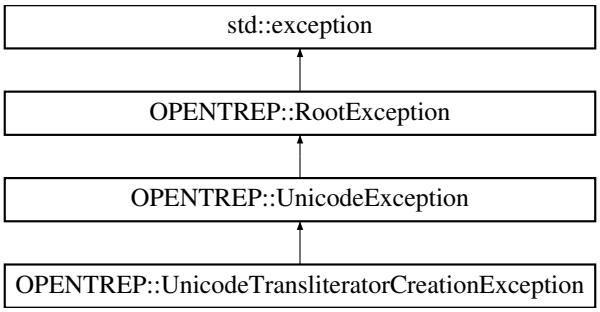
The documentation for this class was generated from the following file:

- opentrep/basic/float_utils_google.hpp

23.175 OPENTREP::UnicodeException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::UnicodeException:



Public Member Functions

- UnicodeException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.175.1 Detailed Description

Unicode failure.

Definition at line 65 of file OPENTREP_exceptions.hpp.

23.175.2 Constructor & Destructor Documentation

23.175.2.1 OPENTREP::UnicodeException::UnicodeException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 70 of file OPENTREP_exceptions.hpp.

23.175.3 Member Function Documentation

23.175.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.175.4 Member Data Documentation

23.175.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

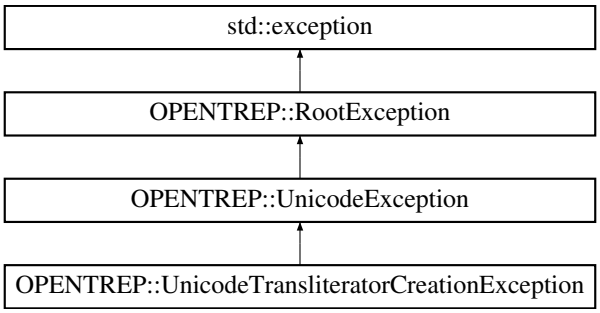
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.176 OPENTREP::UnicodeTransliteratorCreationException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::UnicodeTransliteratorCreationException:



Public Member Functions

- UnicodeTransliteratorCreationException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.176.1 Detailed Description

Unicode transliterator failure.

Definition at line 76 of file OPENTREP_exceptions.hpp.

23.176.2 Constructor & Destructor Documentation

23.176.2.1 OPENTREP::UnicodeTransliteratorCreationException::UnicodeTransliteratorCreationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 81 of file OPENTREP_exceptions.hpp.

23.176.3 Member Function Documentation

23.176.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.176.4 Member Data Documentation

23.176.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.177 UnitTestConfig Struct Reference

Public Member Functions

- UnitTestConfig ()
- ~UnitTestConfig ()
- UnitTestConfig ()
- ~UnitTestConfig ()
- UnitTestConfig ()
- ~UnitTestConfig ()

23.177.1 Detailed Description

Configuration for the Boost Unit Test Framework (UTF)

Definition at line 30 of file PartitionTestSuite.cpp.

23.177.2 Constructor & Destructor Documentation

23.177.2.1 UnitTestConfig::UnitTestConfig ( ) [inline]

Constructor.

Definition at line 32 of file PartitionTestSuite.cpp.

References utfReportStream().

23.177.2.2 UnitTestConfig::~UnitTestConfig ( ) [inline]

Destructor.

Definition at line 40 of file PartitionTestSuite.cpp.

23.177.2.3 UnitTestConfig::UnitTestConfig ( ) [inline]

Constructor.

Definition at line 36 of file SliceTestSuite.cpp.

References utfReportStream().

23.177.2.4 UnitTestConfig::~UnitTestConfig ( ) [inline]

Destructor.

Definition at line 44 of file SliceTestSuite.cpp.

23.177.2.5 UnitTestConfig::UnitTestConfig ( ) [inline]

Constructor.

Definition at line 32 of file UnicodeTestSuite.cpp.

References utfReportStream().

23.177.2.6 UnitTestConfig::~UnitTestConfig ( ) [inline]

Destructor.

Definition at line 40 of file UnicodeTestSuite.cpp.

The documentation for this struct was generated from the following files:

- test/opentrep/PartitionTestSuite.cpp
- test/opentrep/SliceTestSuite.cpp
- test/opentrep/UnicodeTestSuite.cpp

23.178 OPENTREP::UTF8Handler Class Reference

Utility class for basic handling of UTF-8 encoded strings.

`#include <opentrep/basic/UTF8Handler.hpp>`

Static Public Member Functions

- static std::wstring toWideString (const std::string &iSrc)
- static std::string toSimpleString (const std::wstring &iStr)
- static std::string displayCharString (const char *iString)
- static std::string displaySTLWString (const std::wstring &iString)

23.178.1 Detailed Description

Utility class for basic handling of UTF-8 encoded strings.

Most of the methods have taken their inspiration from Jeff Bezanson's work in the Wikix project (see http://meta.wikimedia.org/wiki/Wikix for further details), and have been "C++-ified".

Definition at line 19 of file UTF8Handler.hpp.

23.178.2 Member Function Documentation

23.178.2.1 std::wstring OPENTREP::UTF8Handler::toWideString (const std::string & *iSrc*) [static]

Definition at line 37 of file UTF8Handler.cpp.

References OPENTREP::offsetsFromUTF8, and OPENTREP::trailingBytesForUTF8.

23.178.2.2 std::string OPENTREP::UTF8Handler::toSimpleString (const std::wstring & *iStr*) [static]

Definition at line 98 of file UTF8Handler.cpp.

23.178.2.3 std::string OPENTREP::UTF8Handler::displayCharString (const char * *iString*) [static]

Display the sequence of characters for the simple C-string.

Definition at line 151 of file UTF8Handler.cpp.

23.178.2.4 std::string OPENTREP::UTF8Handler::displaySTLWString (const std::wstring & *iString*) [static]

Display the sequence of characters (one by one) for the given STL wide character string.

Definition at line 172 of file UTF8Handler.cpp.

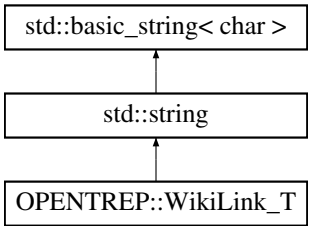
The documentation for this class was generated from the following files:

- opentrep/basic/UTF8Handler.hpp
- opentrep/basic/UTF8Handler.cpp

23.179 OPENTREP::WikiLink_T Struct Reference

`#include <opentrep/OPENTREP_Types.hpp>`

Inheritance diagram for OPENTREP::WikiLink_T:



Public Member Functions

- WikiLink_T (const std::string &iValue)

23.179.1 Detailed Description

Wikipedia link (e.g., `http://en.wikipedia.org/wiki/Chicago`).

Definition at line 361 of file OPENTREP_Types.hpp.

23.179.2 Constructor & Destructor Documentation

23.179.2.1 OPENTREP::WikiLink_T::WikiLink_T (const std::string & *iValue*) [inline], [explicit]

Definition at line 363 of file OPENTREP_Types.hpp.

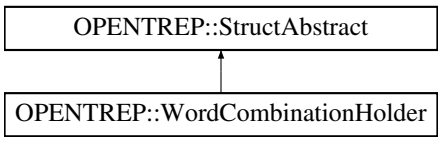
The documentation for this struct was generated from the following file:

- opentrep/OPENTREP_Types.hpp

23.180 OPENTREP::WordCombinationHolder Struct Reference

`#include <opentrep/bom/WordCombinationHolder.hpp>`

Inheritance diagram for OPENTREP::WordCombinationHolder:



Public Types

- typedef std::list< std::string > StringList_T

Public Member Functions

- void push_back (const std::string &)
- size_t size () const
- void clear ()
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- std::string describeKey () const
- std::string describe () const
- WordCombinationHolder (const std::string &)
- ~WordCombinationHolder ()

Public Attributes

- StringList_T _list

23.180.1 Detailed Description

Class holding a list of some specific word combinations within a string.

The list contains all the ways to combine all the words of a given string, while respecting the initial order.

Note

The 2- and 3-letter words (such as 'de' and 'san') are usually not to be indexed by Xapian. Idem with the 'airport' word.

A way to derive that list is simply by:

1. Calculating all the partitions of the initial (full) string
2. Extract from those partitions all the unique word combinations
3. Add all the word combinations, obtained from removing any group of words in the middle of the initial (full) string

For instance, "san francisco international airport" will give:

- Derived from the partitions of the initial string:
- 1. "san francisco international airport"
  2. "san francisco international"
  3. "san francisco"
  4. "francisco international airport"
  5. "international airport"

- Added word combinations:
- 1. "san international airport"
  2. "san francisco airport"
  3. "san airport"

Definition at line 51 of file WordCombinationHolder.hpp.

23.180.2 Member Typedef Documentation

23.180.2.1 typedef std::list<std::string> OPENTREP::WordCombinationHolder::StringList_T

List of strings.

Definition at line 56 of file WordCombinationHolder.hpp.

23.180.3 Constructor & Destructor Documentation

23.180.3.1 OPENTREP::WordCombinationHolder::WordCombinationHolder (const std::string & *iString*)

Constructor.

Definition at line 18 of file WordCombinationHolder.cpp.

23.180.3.2 OPENTREP::WordCombinationHolder::~WordCombinationHolder ( )

Default destructor.

Definition at line 23 of file WordCombinationHolder.cpp.

23.180.4 Member Function Documentation

23.180.4.1 void OPENTREP::WordCombinationHolder::push_back (const std::string & *iString*)

Add an item (string) into the list.

Definition at line 27 of file WordCombinationHolder.cpp.

References _list.

23.180.4.2 size_t OPENTREP::WordCombinationHolder::size ( ) const

Return the size of the list.

Definition at line 32 of file WordCombinationHolder.cpp.

References _list.

23.180.4.3 void OPENTREP::WordCombinationHolder::clear ( )

Empty the list.

Definition at line 37 of file WordCombinationHolder.cpp.

References _list.

23.180.4.4 void OPENTREP::WordCombinationHolder::toStream (std::ostream & *ioOut*) const

Dump the structure into an output stream.

Parameters

ostream&	the output stream.

Definition at line 78 of file WordCombinationHolder.cpp.

References describe().

23.180.4.5 void OPENTREP::WordCombinationHolder::fromStream (std::istream & *ioIn*) [virtual]

Read a structure from an input stream.

Parameters

istream&	the input stream.

Reimplemented from OPENTREP::StructAbstract.

Definition at line 83 of file WordCombinationHolder.cpp.

23.180.4.6 std::string OPENTREP::WordCombinationHolder::describeKey ( ) const

Get a string describing the whole key (differentiating two objects at any level).

Definition at line 42 of file WordCombinationHolder.cpp.

Referenced by describe().

23.180.4.7 std::string OPENTREP::WordCombinationHolder::describe ( ) const [virtual]

Get the serialised version of the structure.

Implements OPENTREP::StructAbstract.

Definition at line 49 of file WordCombinationHolder.cpp.

References _list, and describeKey().

Referenced by toStream().

23.180.5 Member Data Documentation

23.180.5.1 StringList_T OPENTREP::WordCombinationHolder::_list

List of word combinations (strings).

Definition at line 132 of file WordCombinationHolder.hpp.

Referenced by OPENTREP::Place::buildIndexSets(), clear(), describe(), push_back(), and size().

The documentation for this struct was generated from the following files:

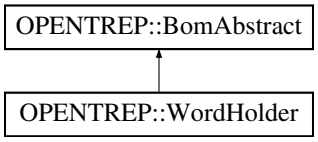
- opentrep/bom/WordCombinationHolder.hpp
- opentrep/bom/WordCombinationHolder.cpp

23.181 OPENTREP::WordHolder Class Reference

Class wrapping utility functions to transform back and forth strings from and into a list of words.

`#include <opentrep/bom/WordHolder.hpp>`

Inheritance diagram for OPENTREP::WordHolder:



Public Member Functions

- virtual void toStream (std::ostream &) const =0
- virtual void fromStream (std::istream &)=0
- virtual std::string toString () const =0
- virtual std::string describeKey () const =0
- virtual std::string describeShortKey () const =0

Static Public Member Functions

- static void tokeniseStringIntoWordList (const TravelQuery_T &, WordList_T &)
- static void tokeniseDocIntoWordList (const TravelQuery_T &, WordList_T &)
- static std::string createStringFromWordList (const WordList_T &iWordList)

Friends

- class FacWordHolder

23.181.1 Detailed Description

Class wrapping utility functions to transform back and forth strings from and into a list of words.

Definition at line 17 of file WordHolder.hpp.

23.181.2 Member Function Documentation

23.181.2.1 void OPENTREP::WordHolder::tokeniseStringIntoWordList (const TravelQuery_T & *iPhrase,* WordList_T & *ioWordList*) [static]

Tokenise a string into a list of words (STL strings).

The Boost.Tokenizer library is used.

Definition at line 37 of file WordHolder.cpp.

References OPENTREP::AllSepatorList(), and OPENTREP::baseTokeniseStringIntoWordList().

Referenced by OPENTREP::addUnmatchedWord(), OPENTREP::Result::calculateCodeMatches(), and OPENTREP::Result::calculateCombinedWeights().

23.181.2.2 void OPENTREP::WordHolder::tokeniseDocIntoWordList (const TravelQuery_T & *iPhrase,* WordList_T & *ioWordList*) [static]

Tokenise a Xapian document data into a list of words (STL strings).

The Boost.Tokenizer library is used.

Definition at line 44 of file WordHolder.cpp.

References OPENTREP::baseTokeniseStringIntoWordList(), and OPENTREP::DocSepatorList().

23.181.2.3 std::string OPENTREP::WordHolder::createStringFromWordList (const WordList_T & *iWordList*) [static]

Serialise a list of words (STL strings) into a single (STL) string.

Definition at line 52 of file WordHolder.cpp.

23.181.2.4 virtual void OPENTREP::BomAbstract::toStream (std::ostream &) const [pure virtual], [inherited]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::World, and OPENTREP::PlaceHolder.

23.181.2.5 virtual void OPENTREP::BomAbstract::fromStream (std::istream &) [pure virtual], [inherited]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::World, and OPENTREP::PlaceHolder.

Referenced by operator>>().

23.181.2.6 virtual std::string OPENTREP::BomAbstract::toString ( ) const [pure virtual], [inherited]

Get the serialised version of the Business Object.

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::World, and OPENTREP::PlaceHolder.

23.181.2.7 virtual std::string OPENTREP::BomAbstract::describeKey ( ) const [pure virtual], [inherited]

Get a string describing the whole key (differentiating two objects at any level).

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::PlaceHolder, and OPENTREP::World.

23.181.2.8 virtual std::string OPENTREP::BomAbstract::describeShortKey ( ) const [pure virtual], [inherited]

Get a string describing the short key (differentiating two objects at the same level).

Implemented in OPENTREP::Place, OPENTREP::Result, OPENTREP::ResultCombination, OPENTREP::ResultHolder, OPENTREP::PlaceHolder, and OPENTREP::World.

23.181.3 Friends And Related Function Documentation

23.181.3.1 friend class FacWordHolder [friend]

Definition at line 18 of file WordHolder.hpp.

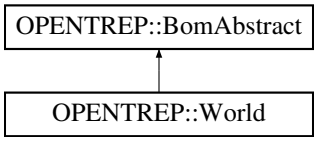
The documentation for this class was generated from the following files:

- opentrep/bom/WordHolder.hpp
- opentrep/bom/WordHolder.cpp

23.182 OPENTREP::World Class Reference

`#include <opentrep/bom/World.hpp>`

Inheritance diagram for OPENTREP::World:



Public Member Functions

- const PlaceOrderedList_T & getSimplePlaceList () const
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- std::string toString () const
- std::string describeKey () const
- std::string describeShortKey () const
- std::string display () const
- std::string shortDisplay () const
- GenericBom_T getGenericBom (const XapianDocID_T &iDocID) const

Friends

- class FacWorld
- class DbaWorld

23.182.1 Detailed Description

Structure modelling an world.

Definition at line 20 of file World.hpp.

23.182.2 Member Function Documentation

23.182.2.1 const PlaceOrderedList_T& OPENTREP::World::getSimplePlaceList ( ) const [inline]

Get the list of Place objects. const PlaceDirectList_T& getPlaceList () const { return _placeList; }Get the list of Place objects.

Definition at line 32 of file World.hpp.

23.182.2.2 void OPENTREP::World::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Implements OPENTREP::BomAbstract.

Definition at line 44 of file World.cpp.

References toString().

23.182.2.3 void OPENTREP::World::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Implements OPENTREP::BomAbstract.

Definition at line 49 of file World.cpp.

23.182.2.4 std::string OPENTREP::World::toString ( ) const [virtual]

Get the serialised version of the World object.

Implements OPENTREP::BomAbstract.

Definition at line 39 of file World.cpp.

References describeKey().

Referenced by toStream().

23.182.2.5 std::string OPENTREP::World::describeKey ( ) const [virtual]

Get a string describing the whole key (differentiating two objects at any level).

Implements OPENTREP::BomAbstract.

Definition at line 34 of file World.cpp.

References describeShortKey().

Referenced by OPENTREP::FacWorld::linkPlaceToWorld(), shortDisplay(), and toString().

23.182.2.6 std::string OPENTREP::World::describeShortKey ( ) const [virtual]

Get a string describing the short key (differentiating two objects at the same level).

Implements OPENTREP::BomAbstract.

Definition at line 28 of file World.cpp.

Referenced by describeKey().

23.182.2.7 std::string OPENTREP::World::display ( ) const

Display the full World context.

Definition at line 60 of file World.cpp.

References OPENTREP::Place::display(), and shortDisplay().

Referenced by OPENTREP::OPENTREP_ServiceContext::display().

23.182.2.8 std::string OPENTREP::World::shortDisplay ( ) const

Display a short World context.

Definition at line 53 of file World.cpp.

References describeKey().

Referenced by display().

23.182.2.9 GenericBom_T OPENTREP::World::getGenericBom (const XapianDocID_T & *iDocID*) const

Retrieve a generic BOM object from the dedicated list.

Definition at line 77 of file World.cpp.

References OPENTREP::BomType::LAST_VALUE.

23.182.3 Friends And Related Function Documentation

23.182.3.1 friend class FacWorld [friend]

Definition at line 21 of file World.hpp.

23.182.3.2 friend class DbaWorld [friend]

Definition at line 22 of file World.hpp.

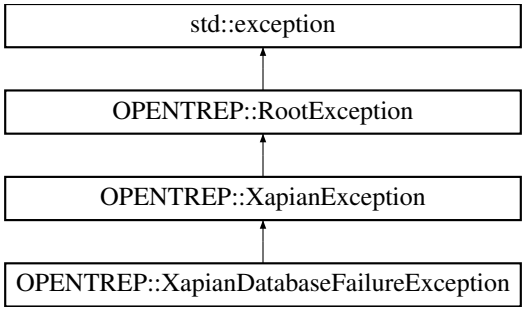
The documentation for this class was generated from the following files:

- opentrep/bom/World.hpp
- opentrep/bom/World.cpp

23.183 OPENTREP::XapianDatabaseFailureException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::XapianDatabaseFailureException:



Public Member Functions

- XapianDatabaseFailureException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.183.1 Detailed Description

Xapian database.

Definition at line 238 of file OPENTREP_exceptions.hpp.

23.183.2 Constructor & Destructor Documentation

23.183.2.1 OPENTREP::XapianDatabaseFailureException::XapianDatabaseFailureException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 243 of file OPENTREP_exceptions.hpp.

23.183.3 Member Function Documentation

23.183.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.183.4 Member Data Documentation

23.183.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

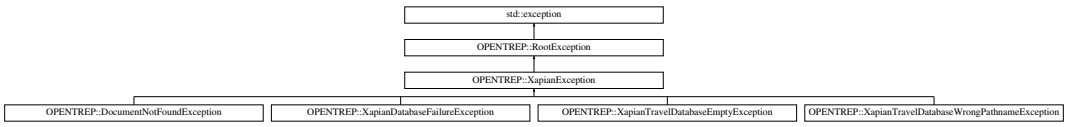
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.184 OPENTREP::XapianException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::XapianException:



Public Member Functions

- XapianException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.184.1 Detailed Description

Xapian root exception.

Definition at line 215 of file OPENTREP_exceptions.hpp.

23.184.2 Constructor & Destructor Documentation

23.184.2.1 OPENTREP::XapianException::XapianException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 220 of file OPENTREP_exceptions.hpp.

23.184.3 Member Function Documentation

23.184.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.184.4 Member Data Documentation

23.184.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.185 OPENTREP::XapianIndexManager Class Reference

Command wrapping utilities for the management of the Xapian (database) index.

`#include <opentrep/command/XapianIndexManager.hpp>`

Friends

- class OPENTREP_Service

23.185.1 Detailed Description

Command wrapping utilities for the management of the Xapian (database) index.

Definition at line 17 of file XapianIndexManager.hpp.

23.185.2 Friends And Related Function Documentation

23.185.2.1 friend class OPENTREP_Service [friend]

Definition at line 18 of file XapianIndexManager.hpp.

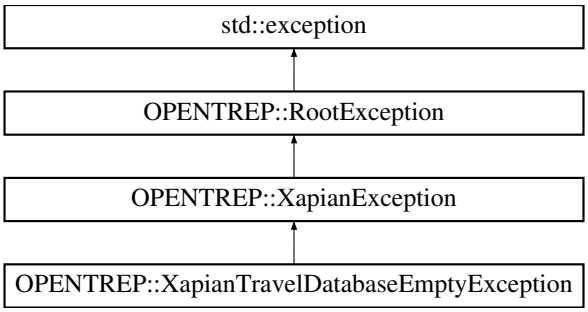
The documentation for this class was generated from the following files:

- opentrep/command/XapianIndexManager.hpp
- opentrep/command/XapianIndexManager.cpp

23.186 OPENTREP::XapianTravelDatabaseEmptyException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::XapianTravelDatabaseEmptyException:



Public Member Functions

- XapianTravelDatabaseEmptyException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.186.1 Detailed Description

Xapian travel database empty.

Definition at line 250 of file OPENTREP_exceptions.hpp.

23.186.2 Constructor & Destructor Documentation

23.186.2.1 OPENTREP::XapianTravelDatabaseEmptyException::XapianTravelDatabaseEmptyException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 255 of file OPENTREP_exceptions.hpp.

23.186.3 Member Function Documentation

23.186.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.186.4 Member Data Documentation

23.186.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

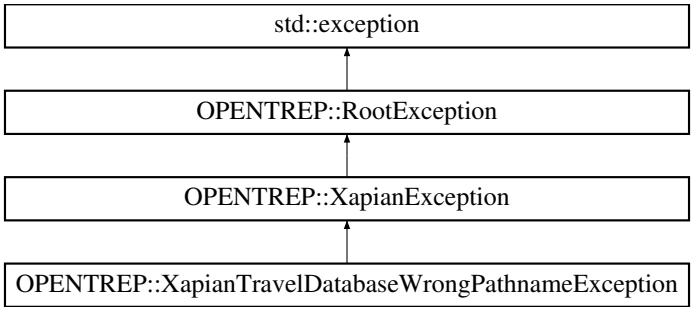
The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

23.187 OPENTREP::XapianTravelDatabaseWrongPathnameException Class Reference

`#include <opentrep/OPENTREP_exceptions.hpp>`

Inheritance diagram for OPENTREP::XapianTravelDatabaseWrongPathnameException:



Public Member Functions

- XapianTravelDatabaseWrongPathnameException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

23.187.1 Detailed Description

The Xapian travel database file-path has not been set correctly.

Definition at line 262 of file OPENTREP_exceptions.hpp.

23.187.2 Constructor & Destructor Documentation

23.187.2.1 OPENTREP::XapianTravelDatabaseWrongPathnameException::XapianTravelDatabaseWrongPathnameException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 267 of file OPENTREP_exceptions.hpp.

23.187.3 Member Function Documentation

23.187.3.1 const char* OPENTREP::RootException::what ( ) const throw) [inline], [inherited]

Give the details of the exception.

Definition at line 39 of file OPENTREP_exceptions.hpp.

References OPENTREP::RootException::_what.

Referenced by OPENTREP::OpenTrepSearcher::init().

23.187.4 Member Data Documentation

23.187.4.1 std::string OPENTREP::RootException::_what [protected], [inherited]

Details for the exception.

Definition at line 47 of file OPENTREP_exceptions.hpp.

Referenced by OPENTREP::RootException::what().

The documentation for this class was generated from the following file:

- opentrep/OPENTREP_exceptions.hpp

24 File Documentation

24.1 doc/local/authors.doc File Reference

24.2 doc/local/codingrules.doc File Reference

24.3 doc/local/copyright.doc File Reference

24.4 doc/local/documentation.doc File Reference

24.5 doc/local/features.doc File Reference

24.6 doc/local/help_wanted.doc File Reference

24.7 doc/local/howto_release.doc File Reference

24.8 doc/local/index.doc File Reference

24.9 doc/local/installation.doc File Reference

24.10 doc/local/linking.doc File Reference

24.11 doc/local/test.doc File Reference

24.12 doc/local/users_guide.doc File Reference

24.13 doc/local/verification.doc File Reference

24.14 doc/tutorial/tutorial.doc File Reference

24.15 opentrep/basic/BasChronometer.cpp File Reference

```
#include <cassert>
#include <opentrep/basic/BasChronometer.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.16 BasChronometer.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // OpenTrep
00007 #include <opentrep/basic/BasChronometer.hpp>
00008 #include <opentrep/service/Logger.hpp>
00009
00010 namespace OPENTREP {
00011
00012   // ////////////////////////////////////////////////////////////////////
00013   BasChronometer::BasChronometer () : _startTimeLaunched (false) {
00014   }
00015
00016   // ////////////////////////////////////////////////////////////////////
00017   void BasChronometer::start () {
00018     // Get the time-stamp of now, and store it for later use
00019     _startTime = boost::posix_time::microsec_clock::local_time();
00020
00021     // Update the boolean which states whether the chronometer
00022     // is launched
00023     _startTimeLaunched = true;
00024   }
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   double BasChronometer::elapsed () const {
00028     assert (_startTimeLaunched == true);
00029
00030     // Get the time-stamp of now
00031     const boost::posix_time::ptime lStopTime =
00032       boost::posix_time::microsec_clock::local_time();
00033
00034     // Calculate the time elapsed since the last time-stamp
00035     const boost::posix_time::time_duration lElapsedTime =
00036       lStopTime - _startTime;
00037
00038     // Derived the corresponding number of milliseconds
00039     const double lElapsedTimeInMicroSeconds =
00040       static_cast<const double> (lElapsedTime.total_microseconds());
00041
00042     /*OPENTREP_LOG_DEBUG ("Elapsed: " << lElapsedTime
00043       << "; (micros): "
00044       << lElapsedTimeInMicroSeconds / 1e6);*/
00045
00046     // The elapsed time given in return is expressed in seconds
00047     return (lElapsedTimeInMicroSeconds / 1e6);
00048   }
00049
00050 }
```

24.17 opentrep/basic/BasChronometer.hpp File Reference

```
#include <boost/date_time/posix_time/posix_time.hpp>
```

Classes

- struct OPENTREP::BasChronometer

  Structure allowing measuring the time elapsed between two events.

Namespaces

- OPENTREP

24.18 BasChronometer.hpp

```
00001 #ifndef __OPENTREP_COM_BAS_BASCHRONOMETER_HPP
00002 #define __OPENTREP_COM_BAS_BASCHRONOMETER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
```

```
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // Boost Date-Time (http://boost.org/doc/html/date_time/posix_time.html)
00008 #include <boost/date_time/posix_time/posix_time.hpp>
00009
00010 namespace OPENTREP {
00011
00015   struct BasChronometer {
00019     BasChronometer();
00020
00026     void start ();
00027
00031     std::string getStart () const {
00032       return boost::posix_time::to_simple_string (_startTime);
00033     }
00034
00040     double elapsed () const;
00041
00042   private:
00046     boost::posix_time::ptime _startTime;
00047
00051     bool _startTimeLaunched;
00052   };
00053
00054 }
00055 #endif // __OPENTREP_COM_BAS_BASCHRONOMETER_HPP
```

24.19 opentrep/basic/BasConst.cpp File Reference

```
#include <opentrep/config/opentrep-paths.hpp>
#include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
#include <opentrep/basic/BasConst_Unicode.hpp>
#include <opentrep/basic/BasConst_General.hpp>
```

Namespaces

- OPENTREP

Functions

- const std::string OPENTREP::DEFAULT_OPENTREP_POR_FILEPATH (OPENTREP_POR_DATA_DIR"/ori_por_public.csv")
- const std::string OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH ("/tmp/opentrep/traveldb")
- const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_USER ("geo")
- const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_PASSWD ("geo")
- const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_DBNAME ("geo_ori")
- const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_HOST ("localhost")
- const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_PORT ("3306")
- const std::string OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH (OPENTREP_POR_DATA_DIR"/ori_por_public.db")
- const std::string OPENTREP::DEFAULT_DATE_STR ("1970-01-01")
- const NbOfMatches_T OPENTREP::K_DEFAULT_XAPIAN_MATCHING_SET_SIZE (30)
- const Weight_T OPENTREP::K_DEFAULT_INDEXING_STD_WEIGHT (1)
- const Weight_T OPENTREP::K_DEFAULT_INDEXING_EXTRA_WEIGHT (2)
- const Percentage_T OPENTREP::K_DEFAULT_MODIFIED_MATCHING_PCT (100.0)
- const Percentage_T OPENTREP::K_DEFAULT_PAGE_RANK (0.10)
- const Percentage_T OPENTREP::K_DEFAULT_ENVELOPE_PCT (K_DEFAULT_PAGE_RANK)
- const Percentage_T OPENTREP::K_DEFAULT_FULL_CODE_MATCH_PCT (11000/K_DEFAULT_PAGE_RANK)
- const Percentage_T OPENTREP::K_DEFAULT_ATTENUATION_FCTR (34.0)
- const NbOfErrors_T OPENTREP::K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT (4)

Variables

- const std::tm OPENTREP::DEFAULT_DATE_TM
- const char * OPENTREP::K_ICU_ACCENT_REMOVAL_RULE = "NFD; [:M:] Remove; NFC;"
- const char * OPENTREP::K_ICU_QUOTATION_REMOVAL_RULE
- const char * OPENTREP::K_ICU_PUNCTUATION_REMOVAL_RULE = "[:P:] Remove;"
- const char * OPENTREP::K_ICU_GENERIC_TRANSLITERATOR_RULE
- const BlackList_T OPENTREP::K_BLACK_LIST = ListGenerator::createBlackList()

24.20 BasConst.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 #include <opentrep/config/opentrep-paths.hpp>
00005 #include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
00006 #include <opentrep/basic/BasConst_Unicode.hpp>
00007 #include <opentrep/basic/BasConst_General.hpp>
00008
00009 namespace OPENTREP {
00010
00011   // ///////////// OPENTREP_Service ///////////////
00015   const std::string DEFAULT_OPENTREP_POR_FILEPATH (OPENTREP_POR_DATA_DIR
00016                                                    "/ori_por_public.csv");
00017
00021   const std::string
00022   DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH ("/tmp/opentrep/traveldb");
00023
00027   const std::string DEFAULT_OPENTREP_MYSQL_DB_USER ("geo");
00028   const std::string DEFAULT_OPENTREP_MYSQL_DB_PASSWD ("geo");
00029   const std::string DEFAULT_OPENTREP_MYSQL_DB_DBNAME ("geo_ori");
00030   const std::string DEFAULT_OPENTREP_MYSQL_DB_HOST ("localhost");
00031   const std::string DEFAULT_OPENTREP_MYSQL_DB_PORT ("3306");
00032
00036   const std::string DEFAULT_OPENTREP_SQLITE_DB_FILEPATH (
      OPENTREP_POR_DATA_DIR
00037                                                          "/ori_por_public.db");
00038
00042   const std::tm DEFAULT_DATE_TM =
00043     boost::gregorian::to_tm (Date_T (1970, boost::gregorian::Jan, 01));
00044
00048   const std::string DEFAULT_DATE_STR ("1970-01-01");
00049
00050   // ///////////////// Unicode (ICU) ////////////////
00054   const char* K_ICU_ACCENT_REMOVAL_RULE = "NFD; [:M:] Remove; NFC;";
00055
00060   const char* K_ICU_QUOTATION_REMOVAL_RULE =
00061     "[\\u02B9] > \\u0027; [\\u002D] > \\u0020; [\\u0027] > \\u0020;";
00062
00067   const char* K_ICU_PUNCTUATION_REMOVAL_RULE = "[:P:] Remove;";
00068
00073   const char* K_ICU_GENERIC_TRANSLITERATOR_RULE =
00074     "Any-Latin; NFD; [:M:] Remove; NFC; Lower;";
00075
00076
00077   // ///////////////// General /////////////////
00081   const NbOfMatches_T K_DEFAULT_XAPIAN_MATCHING_SET_SIZE (30
      );
00082
00086   const Weight_T K_DEFAULT_INDEXING_STD_WEIGHT (1);
00087
00091   const Weight_T K_DEFAULT_INDEXING_EXTRA_WEIGHT (2);
00092
00096   const Percentage_T K_DEFAULT_MODIFIED_MATCHING_PCT (100.0);
00097
00101   const Percentage_T K_DEFAULT_PAGE_RANK (0.10);
00102
00107   const Percentage_T K_DEFAULT_ENVELOPE_PCT (
      K_DEFAULT_PAGE_RANK);
00108
00114   const Percentage_T K_DEFAULT_FULL_CODE_MATCH_PCT (11000 /
      K_DEFAULT_PAGE_RANK);
00115
00119   const Percentage_T K_DEFAULT_ATTENUATION_FCTR (34.0);
00120
00124   const NbOfErrors_T K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT
       (4);
00125
00130   const BlackList_T K_BLACK_LIST =
      ListGenerator::createBlackList();
```

```
00131
00132   // ////////////////////////////////////////////////////////////////////
00133   BlackList_T ListGenerator::createBlackList() {
00134     BlackList_T oBL;
00135     oBL.insert ("City"); oBL.insert ("city");
00136     oBL.insert ("Ville"); oBL.insert ("ville");
00137     oBL.insert ("Stadt"); oBL.insert ("stadt");
00138     oBL.insert ("Gorod"); oBL.insert ("gorod");
00139     oBL.insert ("Airport"); oBL.insert ("airport");
00140     oBL.insert ("Airfield"); oBL.insert ("airfield");
00141     oBL.insert ("Airbase"); oBL.insert ("airbase");
00142     oBL.insert ("Aeroport"); oBL.insert ("aeroport");
00143     oBL.insert ("Aéroport"); oBL.insert ("aéroport");
00144     oBL.insert ("Flughafen"); oBL.insert ("flughafen");
00145     oBL.insert ("Flughafe"); oBL.insert ("flughafe");
00146     oBL.insert ("Flughaveno"); oBL.insert ("flughaveno");
00147     oBL.insert ("Lufthavn"); oBL.insert ("lufthavn");
00148     oBL.insert ("Lentoasema"); oBL.insert ("lentoasema");
00149     oBL.insert ("Lapangan"); oBL.insert ("lapangan");
00150     oBL.insert ("Terbang"); oBL.insert ("terbang");
00151     oBL.insert ("Repülőtér"); oBL.insert ("repülőtér");
00152     oBL.insert ("Aerodrome"); oBL.insert ("aerodrome");
00153     oBL.insert ("Aerodromo"); oBL.insert ("aerodromo");
00154     oBL.insert ("Aeroporto"); oBL.insert ("aeroporto");
00155     oBL.insert ("Aeropuerto"); oBL.insert ("aeropuerto");
00156     oBL.insert (""); oBL.insert ("");
00157     oBL.insert ("Sân"); oBL.insert ("sân");
00158     oBL.insert ("Bay"); oBL.insert ("bay");
00159     oBL.insert ("Quc"); oBL.insert ("quc");
00160     oBL.insert ("T"); oBL.insert ("t");
00161     oBL.insert ("International"); oBL.insert ("international");
00162     oBL.insert ("Internacional"); oBL.insert ("internacional");
00163     oBL.insert ("Internasional"); oBL.insert ("internasional");
00164     oBL.insert ("Internazionale"); oBL.insert ("internazionale");
00165     oBL.insert ("Internasjonale"); oBL.insert ("internasjonale");
00166     oBL.insert (""); oBL.insert ("");
00167     oBL.insert (""); oBL.insert ("");
00168     oBL.insert ("Lotniczy"); oBL.insert ("lotniczy");
00169     oBL.insert ("Kansainvälinen"); oBL.insert ("kansainvälinen");
00170     oBL.insert ("Nemzetközi"); oBL.insert ("nemzetközi");
00171     oBL.insert ("Intl"); oBL.insert ("intl");
00172     oBL.insert ("Saint"); oBL.insert ("saint");
00173     oBL.insert ("Sankt"); oBL.insert ("sankt");
00174     oBL.insert ("Municipal"); oBL.insert ("municipal");
00175     oBL.insert ("County"); oBL.insert ("county");
00176     oBL.insert ("Local"); oBL.insert ("local");
00177     oBL.insert ("Ferry"); oBL.insert ("ferry");
00178     oBL.insert ("Port"); oBL.insert ("port");
00179     oBL.insert ("Harbour"); oBL.insert ("harbour");
00180     oBL.insert ("Harbor"); oBL.insert ("harbor");
00181     return oBL;
00182   }
00183
00184 }
```

24.21 opentrep/basic/BasConst_General.hpp File Reference

```
#include <ctime>
#include <opentrep/OPENTREP_Types.hpp>
```

Classes

- struct OPENTREP::ListGenerator

Namespaces

- OPENTREP

Variables

- const NbOfMatches_T OPENTREP::K_DEFAULT_XAPIAN_MATCHING_SET_SIZE
- const Weight_T OPENTREP::K_DEFAULT_INDEXING_STD_WEIGHT

- const Weight_T OPENTREP::K_DEFAULT_INDEXING_EXTRA_WEIGHT
- const Percentage_T OPENTREP::K_DEFAULT_MODIFIED_MATCHING_PCT
- const Percentage_T OPENTREP::K_DEFAULT_PAGE_RANK
- const Percentage_T OPENTREP::K_DEFAULT_ENVELOPE_PCT
- const Percentage_T OPENTREP::K_DEFAULT_FULL_CODE_MATCH_PCT
- const Percentage_T OPENTREP::K_DEFAULT_ATTENUATION_FCTR
- const NbOfErrors_T OPENTREP::K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT
- const std::string OPENTREP::DEFAULT_DATE_STR

24.22 BasConst_General.hpp

```
00001 #ifndef __OPENTREP_BAS_BASCONST_GENERAL_HPP
00002 #define __OPENTREP_BAS_BASCONST_GENERAL_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <ctime>
00009 // OpenTrep
00010 #include <opentrep/OPENTREP_Types.hpp>
00011
00012 namespace OPENTREP {
00013
00014
00018   extern const NbOfMatches_T K_DEFAULT_XAPIAN_MATCHING_SET_SIZE
      ;
00019
00023   extern const Weight_T K_DEFAULT_INDEXING_STD_WEIGHT;
00024
00028   extern const Weight_T K_DEFAULT_INDEXING_EXTRA_WEIGHT;
00029
00033   extern const Percentage_T K_DEFAULT_MODIFIED_MATCHING_PCT;
00034
00038   extern const Percentage_T K_DEFAULT_PAGE_RANK;
00039
00044   extern const Percentage_T K_DEFAULT_ENVELOPE_PCT;
00045
00051   extern const Percentage_T K_DEFAULT_FULL_CODE_MATCH_PCT;
00052
00056   extern const Percentage_T K_DEFAULT_ATTENUATION_FCTR;
00057
00061   extern const NbOfErrors_T K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT
      ;
00062
00066   struct ListGenerator {
00067     static BlackList_T createBlackList();
00068   };
00069
00074   extern const BlackList_T K_BLACK_LIST;
00075
00079   extern const std::tm DEFAULT_DATE_TM;
00080
00084   extern const std::string DEFAULT_DATE_STR;
00085 }
00086 #endif // __OPENTREP_BAS_BASCONST_GENERAL_HPP
```

24.23 opentrep/basic/BasConst_OPENTREP_Service.hpp File Reference

```
#include <string>
```

Namespaces

- OPENTREP

Variables

- const std::string OPENTREP::DEFAULT_OPENTREP_POR_FILEPATH
- const std::string OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH

- const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_USER
- const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_PASSWD
- const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_DBNAME
- const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_HOST
- const std::string OPENTREP::DEFAULT_OPENTREP_MYSQL_DB_PORT
- const std::string OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH

24.24 BasConst_OPENTREP_Service.hpp

```
00001 #ifndef __OPENTREP_BAS_BASCONST_OPENTREP_SERVICE_HPP
00002 #define __OPENTREP_BAS_BASCONST_OPENTREP_SERVICE_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 #include <string>
00008
00009 namespace OPENTREP {
00010
00022   extern const std::string DEFAULT_OPENTREP_POR_FILEPATH;
00023
00030   extern const std::string DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH;
00031
00045   extern const std::string DEFAULT_OPENTREP_MYSQL_DB_USER;
00046   extern const std::string DEFAULT_OPENTREP_MYSQL_DB_PASSWD;
00047   extern const std::string DEFAULT_OPENTREP_MYSQL_DB_DBNAME;
00048   extern const std::string DEFAULT_OPENTREP_MYSQL_DB_HOST;
00049   extern const std::string DEFAULT_OPENTREP_MYSQL_DB_PORT;
00050
00062   extern const std::string DEFAULT_OPENTREP_SQLITE_DB_FILEPATH;
00063 }
00064 #endif // __OPENTREP_BAS_BASCONST_OPENTREP_SERVICE_HPP
```

24.25 opentrep/basic/BasConst_Unicode.hpp File Reference

Namespaces

- OPENTREP

24.26 BasConst_Unicode.hpp

```
00001 #ifndef __OPENTREP_BAS_BASCONST_UNICODE_HPP
00002 #define __OPENTREP_BAS_BASCONST_UNICODE_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007
00008 namespace OPENTREP {
00009
00014   extern const char* K_ICU_ACCENT_REMOVAL_RULE;
00015
00020   extern const char* K_ICU_QUOTATION_REMOVAL_RULE;
00021
00026   extern const char* K_ICU_PUNCTUATION_REMOVAL_RULE;
00027
00032   extern const char* K_ICU_GENERIC_TRANSLITERATOR_RULE;
00033
00034 }
00035 #endif // __OPENTREP_BAS_BASCONST_UNICODE_HPP
```

24.27 opentrep/basic/BasFileMgr.cpp File Reference

```
#include <cassert>
#include <boost/version.hpp>
#include <boost/filesystem/path.hpp>
#include <boost/filesystem/operations.hpp>
#include <opentrep/basic/BasFileMgr.hpp>
```

Namespaces

- OPENTREP

24.28 BasFileMgr.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost (STL Extension)
00007 // Boost Filesystem (http://www.boost.org/doc/libs/1_48_0/libs/filesystem/doc/index.htm)
00008 #include <boost/version.hpp>
00009 #if BOOST_VERSION >= 103500
00010 #include <boost/filesystem.hpp>
00011 #else // BOOST_VERSION >= 103500
00012 #include <boost/filesystem/path.hpp>
00013 #include <boost/filesystem/operations.hpp>
00014 #endif // BOOST_VERSION >= 103500
00015 // Opentrep
00016 #include <opentrep/basic/BasFileMgr.hpp>
00017
00018 namespace boostfs = boost::filesystem;
00019
00020 namespace OPENTREP {
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   bool BasFileMgr::doesExistAndIsReadable (const std::string& iFilepath)
    {
00024     bool oFine = false;
00025
00026     boostfs::path lPath (iFilepath);
00027
00028     if (boostfs::exists (lPath) == false) {
00029       return oFine;
00030     }
00031
00032 #if BOOST_VERSION >= 103500
00033     if (boostfs::is_regular (lPath) == true) {
00034       oFine = true;
00035     }
00036 #endif // BOOST_VERSION >= 103500
00037
00038     return oFine;
00039   }
00040
00041 }
```

24.29 opentrep/basic/BasFileMgr.hpp File Reference

```
#include <string>
```

Classes

- struct OPENTREP::BasFileMgr

  Helper class for operations on files and on the file-system.

Namespaces

- OPENTREP

24.30 BasFileMgr.hpp

```
00001 #ifndef __OPENTREP_BAS_BASFILEMGR_HPP
00002 #define __OPENTREP_BAS_BASFILEMGR_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
```

```
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace OPENTREP {
00011
00015   struct BasFileMgr {
00016   public:
00017
00018     // ////////////// Functional Support Methods ////////////
00019     static bool doesExistAndIsReadable (const std::string& iFilepath);
00020   };
00021
00022 }
00023 #endif // __OPENTREP_BAS_BASFILEMGR_HPP
```

24.31 opentrep/basic/BasParserTypes.hpp File Reference

```
#include <string>
#include <boost/spirit/include/qi.hpp>
#include <boost/spirit/include/phoenix_core.hpp>
#include <boost/spirit/include/phoenix_operator.hpp>
#include <boost/spirit/include/support_multi_pass.hpp>
#include <boost/spirit/include/classic_position_iterator.hpp>
#include <opentrep/OPENTREP_ParserHelperTypes.hpp>
```

Classes

- struct OPENTREP::short_bool_policies

Namespaces

- OPENTREP

Typedefs

- typedef
  std::istreambuf_iterator< char > OPENTREP::base_iterator_t
- typedef
  boost::spirit::multi_pass
  < base_iterator_t > OPENTREP::iterator_t
- typedef
  boost::spirit::classic::position_iterator2
  < iterator_t > OPENTREP::pos_iterator_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 1, 1 > OPENTREP::uint1_p_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 2, 2 > OPENTREP::uint2_p_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 1, 2 > OPENTREP::int1_2_p_t
- typedef
  boost::spirit::qi::uint_parser
  < int, 10, 4, 4 > OPENTREP::uint4_p_t
- typedef
  boost::spirit::qi::int_parser
  < int, 10, 1, 5 > OPENTREP::int1_5_p_t

- typedef boost::spirit::qi::int_parser < int, 10, 0, 5 > OPENTREP::int0_5_p_t
- typedef boost::spirit::qi::uint_parser < int, 10, 1, 4 > OPENTREP::uint1_4_p_t
- typedef boost::spirit::qi::uint_parser < int, 10, 1, 9 > OPENTREP::uint1_9_p_t
- typedef boost::spirit::qi::uint_parser < int, 10, 0, 9 > OPENTREP::uint0_9_p_t
- typedef boost::spirit::qi::uint_parser < hour_t, 10, 2, 2 > OPENTREP::hour_p_t
- typedef boost::spirit::qi::uint_parser < minute_t, 10, 2, 2 > OPENTREP::minute_p_t
- typedef boost::spirit::qi::uint_parser < second_t, 10, 2, 2 > OPENTREP::second_p_t
- typedef boost::spirit::qi::uint_parser < year_t, 10, 4, 4 > OPENTREP::year_p_t
- typedef boost::spirit::qi::uint_parser < month_t, 10, 2, 2 > OPENTREP::month_p_t
- typedef boost::spirit::qi::uint_parser < day_t, 10, 2, 2 > OPENTREP::day_p_t
- typedef boost::spirit::qi::bool_parser < bool, short_bool_policies > OPENTREP::boolean_p_t

24.32 BasParserTypes.hpp

```
00001 #ifndef __OPENTREP_BAS_BASPARSERTYPES_HPP
00002 #define __OPENTREP_BAS_BASPARSERTYPES_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost Spirit (Parsing)
00010 #include <boost/spirit/include/qi.hpp>
00011 #include <boost/spirit/include/phoenix_core.hpp>
00012 #include <boost/spirit/include/phoenix_operator.hpp>
00013 #include <boost/spirit/include/support_multi_pass.hpp>
00014 #include <boost/spirit/include/classic_position_iterator.hpp>
00015 // OpenTREP
00016 #include <opentrep/OPENTREP_ParserHelperTypes.hpp>
00017
00018 namespace OPENTREP {
00019
00020   // ////////////////////////////////////////////////////////////////////
00021   //
00022   //  Definition of Basic Types
00023   //
00024   // ////////////////////////////////////////////////////////////////////
00025   // The types of iterator, scanner and rule are then derived from
00026   // the parsing unit.
00027   typedef std::istreambuf_iterator<char> base_iterator_t;
00028   typedef boost::spirit::multi_pass<base_iterator_t> iterator_t;
00029
00030   // Wrap the forward iterator with position iterator, to record the position
00031   typedef boost::spirit::classic::position_iterator2<iterator_t> pos_iterator_t;
```

```
00032
00033
00034   // ////////////////////////////////////////////////////////////////////
00035   //
00036   //  Parser related types
00037   //
00038   // ////////////////////////////////////////////////////////////////////
00040   typedef boost::spirit::qi::uint_parser<int, 10, 1, 1> uint1_p_t;
00041
00043   typedef boost::spirit::qi::uint_parser<int, 10, 2, 2> uint2_p_t;
00044
00046   typedef boost::spirit::qi::uint_parser<int, 10, 1, 2> int1_2_p_t;
00047
00049   typedef boost::spirit::qi::uint_parser<int, 10, 4, 4> uint4_p_t;
00050
00052   typedef boost::spirit::qi::int_parser<int, 10, 1, 5> int1_5_p_t;
00053   typedef boost::spirit::qi::int_parser<int, 10, 0, 5> int0_5_p_t;
00054
00056   typedef boost::spirit::qi::uint_parser<int, 10, 1, 4> uint1_4_p_t;
00057
00059   typedef boost::spirit::qi::uint_parser<int, 10, 1, 9> uint1_9_p_t;
00060   typedef boost::spirit::qi::uint_parser<int, 10, 0, 9> uint0_9_p_t;
00061
00063   typedef boost::spirit::qi::uint_parser<hour_t, 10, 2, 2> hour_p_t;
00064   typedef boost::spirit::qi::uint_parser<minute_t, 10, 2, 2> minute_p_t;
00065   typedef boost::spirit::qi::uint_parser<second_t, 10, 2, 2> second_p_t;
00066   typedef boost::spirit::qi::uint_parser<year_t, 10, 4, 4> year_p_t;
00067   typedef boost::spirit::qi::uint_parser<month_t, 10, 2, 2> month_p_t;
00068   typedef boost::spirit::qi::uint_parser<day_t, 10, 2, 2> day_p_t;
00069
00071   struct short_bool_policies : boost::spirit::qi::bool_policies<> {
00072     template <typename Iterator, typename Attribute>
00073     static bool parse_true (Iterator& first, Iterator const& last,
00074                             Attribute& attr) {
00075       if (boost::spirit::qi::detail::string_parse ("Y", first, last,
00076                                                    boost::spirit::qi::unused)) {
00077         // Result is true
00078         boost::spirit::traits::assign_to (true, attr);
00079         return true;
00080       }
00081       return false;
00082     }
00083
00084     // 'Y' will return true and 'N' false
00085     template <typename Iterator, typename Attribute>
00086     static bool parse_false (Iterator& first, Iterator const& last,
00087                              Attribute& attr) {
00088       if (boost::spirit::qi::detail::string_parse ("N", first, last,
00089                                                    boost::spirit::qi::unused)
00090           ||
00091           boost::spirit::qi::detail::string_parse ("Z", first, last,
00092                                                    boost::spirit::qi::unused)) {
00093         // Result is false
00094         boost::spirit::traits::assign_to (false, attr);
00095         return true;
00096       }
00097       return false;
00098     }
00099   };
00100
00101   typedef boost::spirit::qi::bool_parser<bool, short_bool_policies> boolean_p_t;
00102
00103 }
00104 #endif // __OPENTREP_BAS_BASPARSERTYPES_HPP
```

24.33 opentrep/basic/float_utils.hpp File Reference

```
#include <opentrep/basic/float_utils_google.hpp>
```

24.34 float_utils.hpp

```
00001 #ifndef __OPENTREP_BAS_FLOAT_UTILS_HPP
00002 #define __OPENTREP_BAS_FLOAT_UTILS_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/basic/float_utils_google.hpp>
```

```
00009
00021 #endif // __OPENTREP_BAS_FLOAT_UTILS_HPP
```

24.35 opentrep/basic/float_utils_google.hpp File Reference

```
#include <limits>
```

Classes

- class TypeWithSize< size >
- class TypeWithSize< 4 >
- class TypeWithSize< 8 >
- class FloatingPoint< RawType >

24.36 float_utils_google.hpp

```
00001 #ifndef __OPENTREP_BAS_FLOAT_UTILS_GOOGLE_HPP
00002 #define __OPENTREP_BAS_FLOAT_UTILS_GOOGLE_HPP
00003
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //     * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //     * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //     * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // Authors: wan@google.com (Zhanyong Wan), eefacm@gmail.com (Sean Mcafee)
00031 //
00032 // The Google C++ Testing Framework (Google Test)
00033
00034 // ////////////////////////////////////////////////////////////////////////
00035 // Import section
00036 // ////////////////////////////////////////////////////////////////////////
00037 // STL
00038 #include <limits>
00039
00040
00041 // This template class serves as a compile-time function from size to
00042 // type.  It maps a size in bytes to a primitive type with that
00043 // size. e.g.
00044 //
00045 //   TypeWithSize<4>::UInt
00046 //
00047 // is typedef-ed to be unsigned int (unsigned integer made up of 4
00048 // bytes).
00049 //
00050 // Such functionality should belong to STL, but I cannot find it
00051 // there.
00052 //
00053 // Google Test uses this class in the implementation of floating-point
00054 // comparison.
00055 //
00056 // For now it only handles UInt (unsigned int) as that's all Google Test
00057 // needs.  Other types can be easily added in the future if need
00058 // arises.
00059 template <size_t size>
```

```
00060 class TypeWithSize {
00061  public:
00062   // This prevents the user from using TypeWithSize<N> with incorrect
00063   // values of N.
00064   typedef void UInt;
00065 };
00066
00067 // The specialization for size 4.
00068 template <>
00069 class TypeWithSize<4> {
00070  public:
00071   // unsigned int has size 4 in both gcc and MSVC.
00072   //
00073   // As base/basictypes.h doesn't compile on Windows, we cannot use
00074   // uint32, uint64, and etc here.
00075   typedef int Int;
00076   typedef unsigned int UInt;
00077 };
00078
00079 // The specialization for size 8.
00080 template <>
00081 class TypeWithSize<8> {
00082  public:
00083 #if GTEST_OS_WINDOWS
00084   typedef __int64 Int;
00085   typedef unsigned __int64 UInt;
00086 #else
00087   typedef long long Int;  // NOLINT
00088   typedef unsigned long long UInt;  // NOLINT
00089 #endif  // GTEST_OS_WINDOWS
00090 };
00091
00092
00093 // This template class represents an IEEE floating-point number
00094 // (either single-precision or double-precision, depending on the
00095 // template parameters).
00096 //
00097 // The purpose of this class is to do more sophisticated number
00098 // comparison.  (Due to round-off error, etc, it's very unlikely that
00099 // two floating-points will be equal exactly.  Hence a naive
00100 // comparison by the == operation often doesn't work.)
00101 //
00102 // Format of IEEE floating-point:
00103 //
00104 //   The most-significant bit being the leftmost, an IEEE
00105 //   floating-point looks like
00106 //
00107 //     sign_bit exponent_bits fraction_bits
00108 //
00109 //   Here, sign_bit is a single bit that designates the sign of the
00110 //   number.
00111 //
00112 //   For float, there are 8 exponent bits and 23 fraction bits.
00113 //
00114 //   For double, there are 11 exponent bits and 52 fraction bits.
00115 //
00116 //   More details can be found at
00117 //   http://en.wikipedia.org/wiki/IEEE_floating-point_standard.
00118 //
00119 // Template parameter:
00120 //
00121 //   RawType: the raw floating-point type (either float or double)
00122 template <typename RawType>
00123 class FloatingPoint {
00124  public:
00125   // Defines the unsigned integer type that has the same size as the
00126   // floating point number.
00127   typedef typename TypeWithSize<sizeof(RawType)>::UInt
      Bits;
00128
00129   // Constants.
00130
00131   // # of bits in a number.
00132   static const size_t kBitCount = 8*sizeof(RawType);
00133
00134   // # of fraction bits in a number.
00135   static const size_t kFractionBitCount =
00136     std::numeric_limits<RawType>::digits - 1;
00137
00138   // # of exponent bits in a number.
00139   static const size_t kExponentBitCount = kBitCount - 1 -
      kFractionBitCount;
00140
00141   // The mask for the sign bit.
00142   static const Bits kSignBitMask = static_cast<Bits>(1) << (
      kBitCount - 1);
00143
```

```
00144   // The mask for the fraction bits.
00145   static const Bits kFractionBitMask =
00146     ~static_cast<Bits>(0) >> (kExponentBitCount + 1);
00147
00148   // The mask for the exponent bits.
00149   static const Bits kExponentBitMask = ~(kSignBitMask |
      kFractionBitMask);
00150
00151   // How many ULP's (Units in the Last Place) we want to tolerate when
00152   // comparing two numbers.  The larger the value, the more error we
00153   // allow.  A 0 value means that two numbers must be exactly the same
00154   // to be considered equal.
00155   //
00156   // The maximum error of a single floating-point operation is 0.5
00157   // units in the last place.  On Intel CPU's, all floating-point
00158   // calculations are done with 80-bit precision, while double has 64
00159   // bits.  Therefore, 4 should be enough for ordinary use.
00160   //
00161   // See the following article for more details on ULP:
00162   // http://www.cygnus-software.com/papers/comparingfloats/comparingfloats.htm.
00163   static const size_t kMaxUlps = 4;
00164
00165   // Constructs a FloatingPoint from a raw floating-point number.
00166   //
00167   // On an Intel CPU, passing a non-normalized NAN (Not a Number)
00168   // around may change its bits, although the new value is guaranteed
00169   // to be also a NAN.  Therefore, don't expect this constructor to
00170   // preserve the bits in x when x is a NAN.
00171   explicit FloatingPoint (const RawType& x) { u_.value_ = x; }
00172
00173   // Static methods
00174
00175   // Reinterprets a bit pattern as a floating-point number.
00176   //
00177   // This function is needed to test the AlmostEquals() method.
00178   static RawType ReinterpretBits(const Bits bits) {
00179     FloatingPoint fp(0);
00180     fp.u_.bits_ = bits;
00181     return fp.u_.value_;
00182   }
00183
00184   // Returns the floating-point number that represent positive infinity.
00185   static RawType Infinity() {
00186     return ReinterpretBits(kExponentBitMask);
00187   }
00188
00189   // Non-static methods
00190
00191   // Returns the bits that represents this number.
00192   const Bits &bits() const { return u_.bits_; }
00193
00194   // Returns the exponent bits of this number.
00195   Bits exponent_bits() const { return kExponentBitMask & u_.bits_; }
00196
00197   // Returns the fraction bits of this number.
00198   Bits fraction_bits() const { return kFractionBitMask & u_.bits_; }
00199
00200   // Returns the sign bit of this number.
00201   Bits sign_bit() const { return kSignBitMask & u_.bits_; }
00202
00203   // Returns true iff this is NAN (not a number).
00204   bool is_nan() const {
00205     // It's a NAN if the exponent bits are all ones and the fraction
00206     // bits are not entirely zeros.
00207     return (exponent_bits() == kExponentBitMask) && (
      fraction_bits() != 0);
00208   }
00209
00210   // Returns true iff this number is at most kMaxUlps ULP's away from
00211   // rhs.  In particular, this function:
00212   //
00213   //   - returns false if either number is (or both are) NAN.
00214   //   - treats really large numbers as almost equal to infinity.
00215   //   - thinks +0.0 and -0.0 are 0 DLP's apart.
00216   bool AlmostEquals(const FloatingPoint& rhs) const {
00217     // The IEEE standard says that any comparison operation involving
00218     // a NAN must return false.
00219     if (is_nan() || rhs.is_nan()) return false;
00220
00221     return DistanceBetweenSignAndMagnitudeNumbers(u_.bits_, rhs.u_.bits_)
00222         <= kMaxUlps;
00223   }
00224
00225  private:
00226   // The data type used to store the actual floating-point number.
00227   union FloatingPointUnion {
00228     RawType value_;  // The raw floating-point number.
```

```
00229     Bits bits_;      // The bits that represent the number.
00230   };
00231
00232   // Converts an integer from the sign-and-magnitude representation to
00233   // the biased representation.  More precisely, let N be 2 to the
00234   // power of (kBitCount - 1), an integer x is represented by the
00235   // unsigned number x + N.
00236   //
00237   // For instance,
00238   //
00239   //   -N + 1 (the most negative number representable using
00240   //          sign-and-magnitude) is represented by 1;
00241   //   0      is represented by N; and
00242   //   N - 1  (the biggest number representable using
00243   //          sign-and-magnitude) is represented by 2N - 1.
00244   //
00245   // Read http://en.wikipedia.org/wiki/Signed_number_representations
00246   // for more details on signed number representations.
00247   static Bits SignAndMagnitudeToBiased(const Bits &sam) {
00248     if (kSignBitMask & sam) {
00249       // sam represents a negative number.
00250       return ~sam + 1;
00251     } else {
00252       // sam represents a positive number.
00253       return kSignBitMask | sam;
00254     }
00255   }
00256
00257   // Given two numbers in the sign-and-magnitude representation,
00258   // returns the distance between them as an unsigned number.
00259   static Bits DistanceBetweenSignAndMagnitudeNumbers(const Bits &sam1,
00260                                                      const Bits &sam2) {
00261     const Bits biased1 = SignAndMagnitudeToBiased(sam1);
00262     const Bits biased2 = SignAndMagnitudeToBiased(sam2);
00263     return (biased1 >= biased2) ? (biased1 - biased2) : (biased2 - biased1);
00264   }
00265
00266   FloatingPointUnion u_;
00267 };
00268
00269 #endif // __OPENTREP_BAS_FLOAT_UTILS_GOOGLE_HPP
```

24.37 opentrep/basic/icu_util.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <sstream>
#include <unicode/ucnv.h>
#include <opentrep/service/Logger.hpp>
#include "icu_util.hpp"
```

Namespaces

- OPENTREP

Functions

- bool OPENTREP::check (const UErrorCode &iStatus, const char *iMsg)
- UnicodeString & OPENTREP::appendHex (const uint32_t iNumber, int8_t ioDigits, UnicodeString &ioTarget)
- UnicodeString OPENTREP::escape (const UnicodeString &iSource)
- std::string OPENTREP::getUTF8 (const UnicodeString &iString)

24.38 icu_util.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
```

```
00006 #include <iostream>
00007 #include <sstream>
00008 // ICU
00009 #include <unicode/ucnv.h> // Converter
00010 // OpenTrep
00011 #include <opentrep/service/Logger.hpp>
00012 // Local
00013 #include "icu_util.hpp"
00014
00015 namespace OPENTREP {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   bool check (const UErrorCode& iStatus, const char* iMsg) {
00019     bool oSuccessful = true;
00020     if (U_FAILURE (iStatus)) {
00021       oSuccessful = false;
00022       OPENTREP_LOG_ERROR ("Unicode error: " << u_errorName (iStatus)
00023                           << " (" << iMsg << ")");
00024     }
00025
00026     // DEBUG
00027     // OPENTREP_LOG_DEBUG ("No Unicode operation error: " << iMsg);
00028
00029     return oSuccessful;
00030   }
00031
00032   // Append a hex string to the target
00033   // ////////////////////////////////////////////////////////////////////
00034   UnicodeString& appendHex (const uint32_t iNumber, int8_t ioDigits,
00035                             UnicodeString& ioTarget) {
00036     static const UnicodeString DIGIT_STRING ("0123456789ABCDEF");
00037     while (ioDigits > 0) {
00038       ioTarget += DIGIT_STRING[(iNumber >> ((--ioDigits) * 4)) & 0xF];
00039     }
00040
00041     return ioTarget;
00042   }
00043
00044   // ////////////////////////////////////////////////////////////////////
00045   UnicodeString escape (const UnicodeString& iSource) {
00046     int32_t i;
00047     UnicodeString target ("\"");
00048     for (i = 0; i < iSource.length(); ++i) {
00049       UChar ch = iSource[i];
00050
00051       if (ch < 0x09 || (ch > 0x0A && ch < 0x20) || ch > 0x7E) {
00052         target += "\\u";
00053         appendHex (ch, 4, target);
00054
00055       } else {
00056         target += ch;
00057       }
00058     }
00059     target += "\"";
00060
00061     return target;
00062   }
00063
00064   // ////////////////////////////////////////////////////////////////////
00065   std::string getUTF8 (const UnicodeString& iString) {
00066     std::ostringstream oStr;
00067
00068     const int32_t len = iString.length();
00069     // int32_t bufLen = iString.extract(0, len, buf); // Preflight
00070     /* Preflighting seems to be broken now, so assume 1-1 conversion,
00071        plus some slop. */
00072     const int32_t bufLen = len + 256;
00073
00074     // Default codepage conversion
00075     char* buf = new char[bufLen + 1];
00076     const int32_t actualLen = iString.extract (0, len, buf/*, bufLen*/);
00077     buf[actualLen] = '\0';
00078
00079     oStr << buf;
00080     delete buf; buf = NULL;
00081
00082     return oStr.str();
00083   }
00084
00085 }
```

24.39 opentrep/basic/icu_util.hpp File Reference

```
#include <string>
```

```
#include <unicode/unistr.h>
#include <unicode/translit.h>
```

Namespaces

- OPENTREP

Functions

- bool OPENTREP::check (const UErrorCode &iStatus, const char *iMsg)
- UnicodeString OPENTREP::escape (const UnicodeString &iSource)
- std::string OPENTREP::getUTF8 (const UnicodeString &iString)

24.40 icu_util.hpp

```
00001 #ifndef __OPENTREP_BAS_ICU_UTIL_HPP
00002 #define __OPENTREP_BAS_ICU_UTIL_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // ICU
00010 #include <unicode/unistr.h> // UnicodeString
00011 #include <unicode/translit.h> // Transliterator
00012
00013 namespace OPENTREP {
00014
00022   bool check (const UErrorCode& iStatus, const char* iMsg);
00023
00030   UnicodeString escape (const UnicodeString&);
00031
00038   std::string getUTF8 (const UnicodeString&);
00039
00040 }
00041 #endif // __OPENTREP_BAS_ICU_UTIL_HPP
```

24.41 opentrep/basic/Location.cpp File Reference

```
#include <sstream>
#include <string>
#include <list>
#include <opentrep/basic/BasConst_General.hpp>
#include <opentrep/IATAType.hpp>
#include <opentrep/Location.hpp>
```

Namespaces

- OPENTREP

24.42 Location.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <string>
00007 #include <list>
00008 // OpenTrep
00009 #include <opentrep/basic/BasConst_General.hpp>
```

```
00010 #include <opentrep/IATAType.hpp>
00011 #include <opentrep/Location.hpp>
00012
00013 namespace OPENTREP {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   Location::Location() :
00017     _key (IATACode_T (""), IATAType::LAST_VALUE, 0),
00018     _icaoCode (ICAOCode_T ("")), _faaCode (FAACode_T ("")),
00019     _commonName (CommonName_T ("NotAvailable")),
00020     _asciiName (ASCIIName_T ("NotAvailable")),
00021     _altNameShortListString (AltNameShortListString_T ("NotAvailable")),
00022     _tvlPORListString (TvlPORListString_T ("")),
00023     _envelopeID (0),
00024     _dateFrom (1970, 01, 01), _dateEnd (2999, 12, 31), _comment (""),
00025     _cityCode (CityCode_T ("")),
00026     _cityUtfName (CityUTFName_T ("")), _cityAsciiName (
      CityASCIIName_T ("")),
00027     _cityGeonamesID (0),
00028     _stateCode (StateCode_T ("NA")), _countryCode (CountryCode_T ("NA")),
00029     _altCountryCode (AltCountryCode_T ("")), _countryName (
      CountryName_T ("NA")),
00030     _continentCode (ContinentCode_T ("NA")),
00031     _continentName (ContinentName_T ("NotAvailable")),
00032     _latitude (0), _longitude (0),
00033     _featClass (FeatureClass_T ("Z")), _featCode (FeatureCode_T ("ZZZZ")),
00034     _admin1Code (Admin1Code_T ("")),
00035     _admin1UtfName (Admin1UTFName_T ("")),
00036     _admin1AsciiName (Admin1ASCIIName_T ("")),
00037     _admin2Code (Admin2Code_T ("")),
00038     _admin2UtfName (Admin2UTFName_T ("")),
00039     _admin2AsciiName (Admin2ASCIIName_T ("")),
00040     _admin3Code (Admin3Code_T ("Z")), _admin4Code (Admin4Code_T ("")),
00041     _population (0), _elevation (0), _gTopo30 (0),
00042     _timeZone (TimeZone_T ("NA")),
00043     _gmtOffset (0), _dstOffset (0), _rawOffset (0),
00044     _modificationDate (2000, 01, 01),
00045     _wikiLink (WikiLink_T ("http://en.wikipedia.org")),
00046     _pageRank (K_DEFAULT_PAGE_RANK),
00047     _originalKeywords ("NA"), _correctedKeywords ("NA"),
00048     _percentage (0), _editDistance (0), _allowableEditDistance (0),
00049     _rawDataString ("NA"),
00050     _itLanguageCodeNum (Language::LAST_VALUE),
00051     _itLanguageCode (LanguageCode_T ("")) {
00052   }
00053
00054   // ////////////////////////////////////////////////////////////////////
00055   Location::Location (const Location& iLocation) :
00056     _key (iLocation._key),
00057     _icaoCode (iLocation._icaoCode), _faaCode (iLocation._faaCode),
00058     _commonName (iLocation._commonName), _asciiName (iLocation._asciiName),
00059     _altNameShortListString (iLocation._altNameShortListString),
00060     _tvlPORListString (iLocation._tvlPORListString),
00061     _envelopeID (iLocation.getEnvelopeID()),
00062     _dateFrom (iLocation.getDateFrom()), _dateEnd (iLocation.getDateEnd()),
00063     _comment (iLocation.getComment()),
00064     _cityCode (iLocation._cityCode),
00065     _cityUtfName (iLocation._cityUtfName),
00066     _cityAsciiName (iLocation._cityAsciiName),
00067     _cityGeonamesID (iLocation._cityGeonamesID),
00068     _stateCode (iLocation._stateCode), _countryCode (iLocation._countryCode),
00069     _altCountryCode (iLocation._altCountryCode),
00070     _countryName (iLocation._countryName),
00071     _continentCode (iLocation._continentCode),
00072     _continentName (iLocation._continentName),
00073     _latitude (iLocation._latitude), _longitude (iLocation._longitude),
00074     _featClass (iLocation._featClass), _featCode (iLocation._featCode),
00075     _admin1Code (iLocation._admin1Code),
00076     _admin1UtfName (iLocation._admin1UtfName),
00077     _admin1AsciiName (iLocation._admin1AsciiName),
00078     _admin2Code (iLocation._admin2Code),
00079     _admin2UtfName (iLocation._admin2UtfName),
00080     _admin2AsciiName (iLocation._admin2AsciiName),
00081     _admin3Code (iLocation._admin3Code), _admin4Code (iLocation._admin4Code),
00082     _population (iLocation._population), _elevation (iLocation._elevation),
00083     _gTopo30 (iLocation._gTopo30),
00084     _timeZone (iLocation._timeZone), _gmtOffset (iLocation._gmtOffset),
00085     _dstOffset (iLocation._dstOffset), _rawOffset (iLocation._rawOffset),
00086     _modificationDate (iLocation._modificationDate),
00087     _wikiLink (iLocation._wikiLink), _pageRank (iLocation._pageRank),
00088     _nameMatrix (iLocation._nameMatrix),
00089     _originalKeywords (iLocation._originalKeywords),
00090     _correctedKeywords (iLocation._correctedKeywords),
00091     _percentage (iLocation._percentage),
00092     _editDistance (iLocation._editDistance),
00093     _allowableEditDistance (iLocation._allowableEditDistance),
00094     _extraLocationList (iLocation._extraLocationList),
```

```
00095     _alternateLocationList (iLocation._alternateLocationList),
00096     _rawDataString (iLocation._rawDataString),
00097     _itLanguageCodeNum (iLocation._itLanguageCodeNum),
00098     _itLanguageCode (LanguageCode_T ("")) {
00099   }
00100
00101   // ////////////////////////////////////////////////////////////////////
00102   Location::Location (const IATACode_T& iIataCode, const
      IATAType& iIataType,
00103                       const GeonamesID_T& iGeonamesID,
00104                       const ICAOCode_T& iIcaoCode, const FAACode_T& iFaaCode,
00105                       const CommonName_T& iCommonName,
00106                       const ASCIIName_T& iASCIIName,
00107                       const EnvelopeID_T& iEnvelopeID,
00108                       const Date_T& iDateFrom, const Date_T& iDateEnd,
00109                       const Comment_T& iComment,
00110                       const CityCode_T& iCityCode,
00111                       const CityUTFName_T& iCityUtfName,
00112                       const CityASCIIName_T& iCityAsciiName,
00113                       const GeonamesID_T& iCityGeonamesID,
00114                       const StateCode_T& iStateCode,
00115                       const CountryCode_T& iCountryCode,
00116                       const AltCountryCode_T& iAltCountryCode,
00117                       const CountryName_T& iCountryName,
00118                       const ContinentName_T& iContinentName,
00119                       const Latitude_T& iLatitude, const Longitude_T& iLongitude,
00120                       const FeatureClass_T& iFeatureClass,
00121                       const FeatureCode_T& iFeatureCode,
00122                       const Admin1Code_T& iAdmin1Code,
00123                       const Admin1UTFName_T& iAdmin1UtfName,
00124                       const Admin1ASCIIName_T& iAdmin1AsciiName,
00125                       const Admin2Code_T& iAdmin2Code,
00126                       const Admin2UTFName_T& iAdmin2UtfName,
00127                       const Admin2ASCIIName_T& iAdmin2AsciiName,
00128                       const Admin3Code_T& iAdmin3Code,
00129                       const Admin4Code_T& iAdmin4Code,
00130                       const Population_T& iPopulation,
00131                       const Elevation_T& iElevation, const GTopo30_T& iGTopo30,
00132                       const TimeZone_T& iTimeZone,
00133                       const GMTOffset_T& iGMTOffset,
00134                       const DSTOffset_T& iDSTOffset,
00135                       const RawOffset_T& iRawOffset,
00136                       const Date_T& iModDate,
00137                       const TvlPORListString_T& iTvlPORListString,
00138                       const WikiLink_T& iWikiLink, const PageRank_T& iPageRank,
00139                       const std::string& iOriginalKeywords,
00140                       const std::string& iCorrectedKeywords,
00141                       const MatchingPercentage_T& iPercentage,
00142                       const NbOfErrors_T& iEditDistance,
00143                       const NbOfErrors_T& iAllowableEditDistance,
00144                       const RawDataString_T& iRawDataString) :
00145     _key (iIataCode, iIataType, iGeonamesID),
00146     _icaoCode (iIcaoCode), _faaCode (iFaaCode),
00147     _commonName (iCommonName),_asciiName (iASCIIName),
00148     _altNameShortListString (AltNameShortListString_T ("NotAvailable")),
00149     _tvlPORListString (iTvlPORListString),
00150     _envelopeID (iEnvelopeID),
00151     _dateFrom (iDateFrom), _dateEnd (iDateEnd),
00152     _comment (iComment),
00153     _cityCode (iCityCode),
00154     _cityUtfName (iCityUtfName), _cityAsciiName (iCityAsciiName),
00155     _cityGeonamesID (iCityGeonamesID),
00156     _stateCode (iStateCode),
00157     _countryCode (iCountryCode), _altCountryCode (iAltCountryCode),
00158     _countryName (iCountryName),
00159     _continentCode ("NA"), _continentName (iContinentName),
00160     _latitude (iLatitude), _longitude (iLongitude),
00161     _featClass (iFeatureClass), _featCode (iFeatureCode),
00162     _admin1Code (iAdmin1Code),
00163     _admin1UtfName (Admin1UTFName_T (iAdmin1UtfName)),
00164     _admin1AsciiName (Admin1ASCIIName_T (iAdmin1AsciiName)),
00165     _admin2Code (iAdmin2Code),
00166     _admin2UtfName (Admin2UTFName_T (iAdmin2UtfName)),
00167     _admin2AsciiName (Admin2ASCIIName_T (iAdmin2AsciiName)),
00168     _admin3Code (iAdmin3Code), _admin4Code (iAdmin4Code),
00169     _population (iPopulation), _elevation (iElevation), _gTopo30 (iGTopo30),
00170     _timeZone (iTimeZone),
00171     _gmtOffset (iGMTOffset), _dstOffset (iDSTOffset), _rawOffset (iRawOffset),
00172     _modificationDate (iModDate),
00173     _wikiLink (iWikiLink), _pageRank (iPageRank),
00174     _originalKeywords (iOriginalKeywords),
00175     _correctedKeywords (iCorrectedKeywords),
00176     _percentage (iPercentage), _editDistance (iEditDistance),
00177     _allowableEditDistance (iAllowableEditDistance),
00178     _rawDataString (iRawDataString),
00179     _itLanguageCodeNum (Language::LAST_VALUE),
00180     _itLanguageCode (LanguageCode_T ("")) {
```

```
00181   }
00182
00183   // ////////////////////////////////////////////////////////////////////
00184   Location::~Location() {
00185   }
00186
00187   // ////////////////////////////////////////////////////////////////////
00188   void Location::toStream (std::ostream& ioOut) const {
00189     ioOut << toString();
00190   }
00191
00192   // ////////////////////////////////////////////////////////////////////
00193   void Location::fromStream (std::istream&) {
00194   }
00195
00196   // ////////////////////////////////////////////////////////////////////
00197   std::string Location::describeShortKey() const {
00198     std::ostringstream oStr;
00199     oStr << _key.describe();
00200     return oStr.str();
00201   }
00202
00203   // ////////////////////////////////////////////////////////////////////
00204   std::string Location::describeKey() const {
00205     return describeShortKey();
00206   }
00207
00208   // ////////////////////////////////////////////////////////////////////
00209   std::string Location::toBasicString() const {
00210     std::ostringstream oStr;
00211     oStr << describeShortKey();
00212
00213     oStr << ", " << _pageRank << "%"
00214          << ", " << _commonName << ", " << _asciiName
00215          << ", " << _icaoCode << ", " << _faaCode
00216          << ", " << _envelopeID
00217          << ", " << _dateFrom << ", " << _dateEnd << ", " << _comment
00218          << ", " << _cityCode << ", " << _cityUtfName << ", " << _cityAsciiName
00219          << ", " << _cityGeonamesID
00220          << ", " << _stateCode
00221          << ", " << _countryCode << ", " << _altCountryCode
00222          << ", " << _countryName
00223          << ", " << _continentCode << ", " << _continentName
00224          << ", " << _latitude << ", " << _longitude
00225          << ", " << _featClass << ", " << _featCode
00226          << ", " << _admin1Code
00227          << ", " << _admin1UtfName << ", " << _admin1AsciiName
00228          << ", " << _admin2Code
00229          << ", " << _admin2UtfName << ", " << _admin2AsciiName
00230          << ", " << _admin3Code << ", " << _admin4Code
00231          << ", " << _population << ", " << _elevation << ", " << _gTopo30
00232          << ", " << _timeZone
00233          << ", " << _gmtOffset << ", " << _dstOffset << ", " << _rawOffset
00234          << ", " << _modificationDate
00235          << ", " << _tvlPORListString
00236          << ", " << _wikiLink
00237          << ", " << _originalKeywords << ", " << _correctedKeywords
00238          << ", " << _percentage << "%"
00239          << ", " << _editDistance << ", " << _allowableEditDistance;
00240
00241     return oStr.str();
00242   }
00243
00244   // ////////////////////////////////////////////////////////////////////
00245   std::string Location::toShortString() const {
00246     std::ostringstream oStr;
00247     oStr << toBasicString();
00248
00249     if (_extraLocationList.empty() == false) {
00250       oStr << " with " << _extraLocationList.size() << " extra match(es)";
00251     }
00252
00253     if (_alternateLocationList.empty() == false) {
00254       oStr << " with " << _alternateLocationList.size()
00255            << " alternate match(es)";
00256     }
00257
00258     return oStr.str();
00259   }
00260
00261   // ////////////////////////////////////////////////////////////////////
00262   std::string Location::toSingleLocationString() const {
00263     std::ostringstream oStr;
00264     oStr << toBasicString();
00265     return oStr.str();
00266   }
00267
```

```
00268   // ////////////////////////////////////////////////////////////////////
00269   std::string Location::toString() const {
00270     std::ostringstream oStr;
00271     oStr << toSingleLocationString();
00272     oStr << "; name matrix {" << _nameMatrix.describe() << "}";
00273
00274     if (_extraLocationList.empty() == false) {
00275       oStr << "; Extra matches: {";
00276       unsigned short idx = 0;
00277       for (LocationList_T::const_iterator itLoc = _extraLocationList.begin();
00278            itLoc != _extraLocationList.end(); ++itLoc, ++idx) {
00279         if (idx != 0) {
00280           oStr << ". ";
00281         }
00282         const Location& lExtraLocation = *itLoc;
00283         oStr << lExtraLocation.toShortString();
00284       }
00285       oStr << "}";
00286     }
00287
00288     if (_alternateLocationList.empty() == false) {
00289       oStr << "; Alternate matches: {";
00290       unsigned short idx = 0;
00291       for (LocationList_T::const_iterator itLoc =
00292              _alternateLocationList.begin();
00293            itLoc != _alternateLocationList.end(); ++itLoc, ++idx) {
00294         if (idx != 0) {
00295           oStr << ". ";
00296         }
00297         const Location& lAlternateLocation = *itLoc;
00298         oStr << lAlternateLocation.toShortString();
00299       }
00300       oStr << "}";
00301     }
00302
00303     return oStr.str();
00304   }
00305
00306   // ////////////////////////////////////////////////////////////////////
00307   Date_T Location::calculateDate() const {
00308     _itYear.check(); _itMonth.check(); _itDay.
      check();
00309     return Date_T (_itYear._value, _itMonth._value,
      _itDay._value);
00310   }
00311
00312   // ////////////////////////////////////////////////////////////////////
00313   void Location::consolidateAltNameShortListString() {
00314     std::ostringstream oStr;
00315     unsigned short idx = 0;
00316     for (AltNameShortList_T::const_iterator itName = _itAltNameShortList.begin();
00317          itName != _itAltNameShortList.end(); ++itName, ++idx) {
00318       if (idx != 0) {
00319         oStr << ",";
00320       }
00321       const AltNameShortListString_T& lAlternateLocation = *itName;
00322       oStr << lAlternateLocation;
00323     }
00324
00325     // Store the short list of alternative names
00326     const std::string lAltNameShortListString (oStr.str());
00327     setAltNameShortListString (lAltNameShortListString);
00328   }
00329
00330   // ////////////////////////////////////////////////////////////////////
00331   void Location::consolidateTvlPORListString() {
00332     std::ostringstream oStr;
00333     unsigned short idx = 0;
00334     for (TvlPORList_T::const_iterator itCode = _itTvlPORList.begin();
00335          itCode != _itTvlPORList.end(); ++itCode, ++idx) {
00336       if (idx != 0) {
00337         oStr << ",";
00338       }
00339       const TvlPORListString_T& lTvlPORCode = *itCode;
00340       oStr << lTvlPORCode;
00341     }
00342
00343     // Store the list of IATA codes of the travel-related POR
00344     const std::string lTvlPORCodeList (oStr.str());
00345     setTvlPORListString (lTvlPORCodeList);
00346   }
00347
00348   // ////////////////////////////////////////////////////////////////////
00349   FeatureNameList_T Location::getFeatureList(const
      FeatureCode_T& iFeatureCode) {
00350     FeatureNameList_T oList;
00351
```

```
00352     // Extract the first two/three characters of the feature code (e.g.,
00353     // 'MN' for 'MNCU', 'PPL' for 'PPLA'/'PPLC' or 'ADM' for 'ADM1'/'ADM2')
00354     const std::string l2CharFeatCode = iFeatureCode.substr (0, 2);
00355     const std::string l3CharFeatCode = iFeatureCode.substr (0, 3);
00356
00357     if (iFeatureCode == "AIRP" || iFeatureCode == "AIRF"
00358         || iFeatureCode == "AIRQ") {
00359       oList.push_back (FeatureName_T ("airport"));
00360       oList.push_back (FeatureName_T ("apt"));
00361       oList.push_back (FeatureName_T ("airdrome"));
00362       oList.push_back (FeatureName_T ("aerodrome"));
00363       oList.push_back (FeatureName_T ("airfield"));
00364       oList.push_back (FeatureName_T ("air field"));
00365       oList.push_back (FeatureName_T ("field"));
00366       oList.push_back (FeatureName_T ("airstrip"));
00367       oList.push_back (FeatureName_T ("air strip"));
00368       oList.push_back (FeatureName_T ("strip"));
00369
00370     } else if (iFeatureCode == "AIRB") {
00371       oList.push_back (FeatureName_T ("airport"));
00372       oList.push_back (FeatureName_T ("apt"));
00373       oList.push_back (FeatureName_T ("airbase"));
00374       oList.push_back (FeatureName_T ("air base"));
00375       oList.push_back (FeatureName_T ("ab"));
00376       oList.push_back (FeatureName_T ("base"));
00377
00378     } else if (iFeatureCode == "AIRS") {
00379       oList.push_back (FeatureName_T ("airport"));
00380       oList.push_back (FeatureName_T ("apt"));
00381       oList.push_back (FeatureName_T ("spb"));
00382       oList.push_back (FeatureName_T ("sea plane base"));
00383       oList.push_back (FeatureName_T ("sea plane"));
00384       oList.push_back (FeatureName_T ("waterdrome"));
00385       oList.push_back (FeatureName_T ("aerodrome"));
00386       oList.push_back (FeatureName_T ("water aerodrome"));
00387
00388     } else if (iFeatureCode == "AIRH") {
00389       oList.push_back (FeatureName_T ("airport"));
00390       oList.push_back (FeatureName_T ("apt"));
00391       oList.push_back (FeatureName_T ("heliport"));
00392       oList.push_back (FeatureName_T ("hpt"));
00393
00394     } else if (iFeatureCode == "RSTN") {
00395       oList.push_back (FeatureName_T ("railway"));
00396       oList.push_back (FeatureName_T ("railroad"));
00397       oList.push_back (FeatureName_T ("train"));
00398       oList.push_back (FeatureName_T ("station"));
00399       oList.push_back (FeatureName_T ("railway station"));
00400       oList.push_back (FeatureName_T ("railroad station"));
00401       oList.push_back (FeatureName_T ("train station"));
00402
00403     } else if (l3CharFeatCode == "BUS") {
00404       oList.push_back (FeatureName_T ("bus"));
00405       oList.push_back (FeatureName_T ("autobus"));
00406       oList.push_back (FeatureName_T ("station"));
00407       oList.push_back (FeatureName_T ("stn"));
00408       oList.push_back (FeatureName_T ("stop"));
00409       oList.push_back (FeatureName_T ("bus station"));
00410       oList.push_back (FeatureName_T ("bus stop"));
00411       oList.push_back (FeatureName_T ("autobus station"));
00412       oList.push_back (FeatureName_T ("autobus stop"));
00413
00414     } else if (iFeatureCode == "MTRO") {
00415       oList.push_back (FeatureName_T ("metro"));
00416       oList.push_back (FeatureName_T ("station"));
00417       oList.push_back (FeatureName_T ("stn"));
00418       oList.push_back (FeatureName_T ("stop"));
00419       oList.push_back (FeatureName_T ("metro station"));
00420       oList.push_back (FeatureName_T ("metro stop"));
00421
00422     } else if (iFeatureCode == "FY" || iFeatureCode == "PRT"
00423                || iFeatureCode == "PORT" || iFeatureCode == "HBR") {
00424       oList.push_back (FeatureName_T ("ferry"));
00425       oList.push_back (FeatureName_T ("maritime"));
00426       oList.push_back (FeatureName_T ("port"));
00427       oList.push_back (FeatureName_T ("prt"));
00428       oList.push_back (FeatureName_T ("harbor"));
00429       oList.push_back (FeatureName_T ("harbour"));
00430       oList.push_back (FeatureName_T ("hbr"));
00431       oList.push_back (FeatureName_T ("ferry port"));
00432       oList.push_back (FeatureName_T ("maritime port"));
00433
00434     } else if (l3CharFeatCode == "PPL" || l3CharFeatCode == "ADM"
00435                || iFeatureCode == "LCTY") {
00436       oList.push_back (FeatureName_T ("city"));
00437       oList.push_back (FeatureName_T ("cty"));
00438       oList.push_back (FeatureName_T ("town"));
```

```
00439       oList.push_back (FeatureName_T ("downtown"));
00440       oList.push_back (FeatureName_T ("down town"));
00441       oList.push_back (FeatureName_T ("center"));
00442       oList.push_back (FeatureName_T ("centre"));
00443
00444     } else if (l3CharFeatCode == "PCL" || iFeatureCode == "AREA") {
00445       oList.push_back (FeatureName_T ("city"));
00446       oList.push_back (FeatureName_T ("cty"));
00447       oList.push_back (FeatureName_T ("country"));
00448       oList.push_back (FeatureName_T ("town"));
00449       oList.push_back (FeatureName_T ("downtown"));
00450       oList.push_back (FeatureName_T ("down town"));
00451       oList.push_back (FeatureName_T ("center"));
00452       oList.push_back (FeatureName_T ("centre"));
00453       oList.push_back (FeatureName_T ("district"));
00454       oList.push_back (FeatureName_T ("prefecture"));
00455
00456     } else if (iFeatureCode == "MT") {
00457       oList.push_back (FeatureName_T ("city"));
00458       oList.push_back (FeatureName_T ("cty"));
00459       oList.push_back (FeatureName_T ("mount"));
00460       oList.push_back (FeatureName_T ("mt"));
00461
00462     } else if (iFeatureCode == "PLAT") {
00463       oList.push_back (FeatureName_T ("city"));
00464       oList.push_back (FeatureName_T ("cty"));
00465       oList.push_back (FeatureName_T ("plateau"));
00466
00467     } else if (iFeatureCode == "OAS") {
00468       oList.push_back (FeatureName_T ("city"));
00469       oList.push_back (FeatureName_T ("cty"));
00470       oList.push_back (FeatureName_T ("oasis"));
00471
00472     } else if (iFeatureCode == "PRK") {
00473       oList.push_back (FeatureName_T ("city"));
00474       oList.push_back (FeatureName_T ("cty"));
00475       oList.push_back (FeatureName_T ("park"));
00476       oList.push_back (FeatureName_T ("prk"));
00477
00478     } else if (l3CharFeatCode == "RES") {
00479       oList.push_back (FeatureName_T ("city"));
00480       oList.push_back (FeatureName_T ("cty"));
00481       oList.push_back (FeatureName_T ("reserve"));
00482       oList.push_back (FeatureName_T ("rsv"));
00483
00484     } else if (iFeatureCode == "CMP") {
00485       oList.push_back (FeatureName_T ("city"));
00486       oList.push_back (FeatureName_T ("cty"));
00487       oList.push_back (FeatureName_T ("camp"));
00488       oList.push_back (FeatureName_T ("cmp"));
00489
00490     } else if (l2CharFeatCode == "MN") {
00491       oList.push_back (FeatureName_T ("city"));
00492       oList.push_back (FeatureName_T ("cty"));
00493       oList.push_back (FeatureName_T ("mine"));
00494
00495     } else if (iFeatureCode == "HMSD") {
00496       oList.push_back (FeatureName_T ("city"));
00497       oList.push_back (FeatureName_T ("cty"));
00498       oList.push_back (FeatureName_T ("homestead"));
00499
00500     } else if (l2CharFeatCode == "RK") {
00501       oList.push_back (FeatureName_T ("city"));
00502       oList.push_back (FeatureName_T ("cty"));
00503       oList.push_back (FeatureName_T ("rock"));
00504
00505     } else if (iFeatureCode == "PT") {
00506       oList.push_back (FeatureName_T ("city"));
00507       oList.push_back (FeatureName_T ("cty"));
00508       oList.push_back (FeatureName_T ("point"));
00509       oList.push_back (FeatureName_T ("pt"));
00510
00511     } else if (l3CharFeatCode == "RSV") {
00512       oList.push_back (FeatureName_T ("city"));
00513       oList.push_back (FeatureName_T ("cty"));
00514       oList.push_back (FeatureName_T ("reservoir"));
00515       oList.push_back (FeatureName_T ("rsv"));
00516
00517     } else if (iFeatureCode == "LK") {
00518       oList.push_back (FeatureName_T ("city"));
00519       oList.push_back (FeatureName_T ("cty"));
00520       oList.push_back (FeatureName_T ("lake"));
00521       oList.push_back (FeatureName_T ("lk"));
00522
00523     } else if (iFeatureCode == "BAY") {
00524       oList.push_back (FeatureName_T ("city"));
00525       oList.push_back (FeatureName_T ("cty"));
```

```
00526       oList.push_back (FeatureName_T ("bay"));
00527
00528     } else if (iFeatureCode == "CHN") {
00529       oList.push_back (FeatureName_T ("city"));
00530       oList.push_back (FeatureName_T ("cty"));
00531       oList.push_back (FeatureName_T ("channel"));
00532       oList.push_back (FeatureName_T ("chn"));
00533
00534     } else if (l3CharFeatCode == "ISL") {
00535       oList.push_back (FeatureName_T ("city"));
00536       oList.push_back (FeatureName_T ("cty"));
00537       oList.push_back (FeatureName_T ("island"));
00538       oList.push_back (FeatureName_T ("isl"));
00539       oList.push_back (FeatureName_T ("country"));
00540
00541     } else if (iFeatureCode == "ATOL") {
00542       oList.push_back (FeatureName_T ("city"));
00543       oList.push_back (FeatureName_T ("cty"));
00544       oList.push_back (FeatureName_T ("island"));
00545       oList.push_back (FeatureName_T ("isl"));
00546       oList.push_back (FeatureName_T ("atoll"));
00547       oList.push_back (FeatureName_T ("atl"));
00548       oList.push_back (FeatureName_T ("country"));
00549     }
00550
00551     //
00552     return oList;
00553   }
00554
00555 }
```

24.43 opentrep/basic/LocationKey.cpp File Reference

```
#include <cassert>
#include <opentrep/LocationKey.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.44 LocationKey.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // OpenTrep
00007 #include <opentrep/LocationKey.hpp>
00008 #include <opentrep/service/Logger.hpp>
00009
00010 namespace OPENTREP {
00011
00012   // //////////////////////////////////////////////////////////////////
00013   LocationKey::LocationKey (const IATACode_T& iIataCode,
00014                             const IATAType& iIataType,
00015                             const GeonamesID_T& iGeonamesID) :
00016     _iataCode (iIataCode), _iataType (iIataType), _geonamesID (iGeonamesID) {
00017     if (_geonamesID == 0) {
00018       _isGeonames = false;
00019     } else {
00020       _isGeonames = true;
00021     }
00022   }
00023
00024   // //////////////////////////////////////////////////////////////////
00025   LocationKey::LocationKey() :
00026     _iataCode (IATACode_T ("")), _iataType (IATAType::
      LAST_VALUE),
00027     _geonamesID (0), _isGeonames (false) {
00028     assert (false);
00029   }
00030
00031   // //////////////////////////////////////////////////////////////////
```

```
00032   LocationKey::LocationKey (const LocationKey& iLocationKey) :
00033     _iataCode (iLocationKey._iataCode), _iataType (iLocationKey._iataType),
00034     _geonamesID (iLocationKey._geonamesID),
00035     _isGeonames (iLocationKey._isGeonames) {
00036   }
00037
00038   // ////////////////////////////////////////////////////////////////////
00039   LocationKey::~LocationKey() {
00040   }
00041
00042   // ////////////////////////////////////////////////////////////////////
00043   bool LocationKey::operator== (const LocationKey& iLocationKey) const {
00044     const bool areEqual = (_iataCode == iLocationKey._iataCode
00045                            && _iataType == iLocationKey._iataType
00046                            && _geonamesID == iLocationKey._geonamesID);
00047     return areEqual;
00048   }
00049
00050   // ////////////////////////////////////////////////////////////////////
00051   std::string LocationKey::describe() const {
00052     std::ostringstream oStr;
00053     oStr << _iataCode << "-" << _iataType.getTypeAsString()
00054          << "-" << _geonamesID;
00055
00056     return oStr.str();
00057   }
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   std::string LocationKey::toString() const {
00061     std::ostringstream oStr;
00062     oStr << describe();
00063     return oStr.str();
00064   }
00065
00066   // ////////////////////////////////////////////////////////////////////
00067   void LocationKey::toStream (std::ostream& ioOut) const {
00068     ioOut << toString();
00069   }
00070
00071   // ////////////////////////////////////////////////////////////////////
00072   void LocationKey::fromStream (std::istream& ioIn) {
00073   }
00074
00075 }
```

24.45 opentrep/basic/NameMatrix.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <opentrep/NameMatrix.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.46 NameMatrix.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // OpenTrep
00008 #include <opentrep/NameMatrix.hpp>
00009 #include <opentrep/service/Logger.hpp>
00010
00011 namespace OPENTREP {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   NameMatrix::NameMatrix() {
00015   }
00016
00017   // ////////////////////////////////////////////////////////////////////
```

```
00018   NameMatrix::NameMatrix (const NameMatrix& iNameMatrix)
00019     : _nameMatrix (iNameMatrix._nameMatrix) {
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   NameMatrix::~NameMatrix() {
00024   }
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   std::string NameMatrix::describeKey() const {
00028     std::ostringstream oStr;
00029     return oStr.str();
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   std::string NameMatrix::describe() const {
00034     std::ostringstream oStr;
00035     oStr << describeKey();
00036
00037     unsigned short idx = 0;
00038     for (NameMatrix_T::const_iterator itNameList = _nameMatrix.begin();
00039          itNameList != _nameMatrix.end(); ++itNameList, ++idx) {
00040       if (idx != 0) {
00041         oStr << ",";
00042       }
00043       const Names& lNameList = itNameList->second;
00044       oStr << lNameList.describe();
00045     }
00046
00047     return oStr.str();
00048   }
00049
00050   // ////////////////////////////////////////////////////////////////////
00051   void NameMatrix::toStream (std::ostream& ioOut) const {
00052     ioOut << describe();
00053   }
00054
00055   // ////////////////////////////////////////////////////////////////////
00056   void NameMatrix::fromStream (std::istream& ioIn) {
00057   }
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   std::string NameMatrix::toString() const {
00061     std::ostringstream oStr;
00062     oStr << describe();
00063     return oStr.str();
00064   }
00065
00066   // ////////////////////////////////////////////////////////////////////
00067   bool NameMatrix::getNameList (const LanguageCode_T& iLanguageCode,
00068                                 NameList_T& ioNameList) const {
00069     bool oFoundNameList = false;
00070
00071     NameMatrix_T::const_iterator itNameList = _nameMatrix.find (iLanguageCode);
00072     if (itNameList != _nameMatrix.end()) {
00073       const Names& lNameList = itNameList->second;
00074       ioNameList = lNameList.getNameList();
00075       oFoundNameList = true;
00076     }
00077
00078     return oFoundNameList;
00079   }
00080
00081   // ////////////////////////////////////////////////////////////////////
00082   void NameMatrix::addName (const LanguageCode_T& iLanguageCode,
00083                             const std::string& iName) {
00084
00085     // Check whether a name in that language has already been recorded
00086     NameMatrix_T::iterator itNameList = _nameMatrix.find (iLanguageCode);
00087     if (itNameList != _nameMatrix.end()) {
00088       // Just add the name for that language
00089       Names& lNameList = itNameList->second;
00090
00091       lNameList.addName (iName);
00092
00093     } else {
00094       // Create a new name list for the given language
00095       Names lNameList (iLanguageCode);
00096       lNameList.addName (iName);
00097
00098       // Insert the name list with the dedicated list
00099       const bool insertSucceeded =
00100         _nameMatrix.insert (NameMatrix_T::value_type (iLanguageCode,
00101                                                       lNameList)).second;
00102       if (insertSucceeded == false) {
00103         OPENTREP_LOG_ERROR ("The " << iName << " name can not be inserted in "
00104                             << "the dedicated list for the "
```

```
00105                           << iLanguageCode << " language");
00106       }
00107       assert (insertSucceeded == true);
00108     }
00109   }
00110
00111   // ////////////////////////////////////////////////////////////////////
00112   void NameMatrix::reset() {
00113     _nameMatrix.clear();
00114   }
00115
00116 }
```

24.47 opentrep/basic/Names.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <opentrep/Names.hpp>
```

Namespaces

- OPENTREP

24.48 Names.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // OpenTrep
00008 #include <opentrep/Names.hpp>
00009
00010 namespace OPENTREP {
00011
00012   // ////////////////////////////////////////////////////////////////////
00013   Names::Names() : _languageCode ("en_US") {
00014     assert (false);
00015   }
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   Names::Names (const LanguageCode_T& iLanguageCode)
00019     : _languageCode (iLanguageCode) {
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   Names::Names (const Names& iName)
00024     : _languageCode (iName._languageCode),
00025       _nameList (iName._nameList) {
00026   }
00027
00028   // ////////////////////////////////////////////////////////////////////
00029   Names::~Names() {
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   std::string Names::getFirstName() const {
00034     if (_nameList.empty() == true) {
00035       return "";
00036     }
00037     NameList_T::const_iterator itName = _nameList.begin();
00038     assert (itName != _nameList.end());
00039     const std::string& lName = *itName;
00040     return lName;
00041   }
00042
00043   // ////////////////////////////////////////////////////////////////////
00044   std::string Names::describeKey() const {
00045     std::ostringstream oStr;
00046     //oStr << "[" << Language::getLongLabel (_languageCode) << "]: ";
00047     return oStr.str();
00048   }
00049
00050   // ////////////////////////////////////////////////////////////////////
```

```
00051   std::string Names::describe() const {
00052     std::ostringstream oStr;
00053     oStr << describeKey();
00054
00055     // The language code is the same for all the items of the list. But it
00056     // is repeated for every item, so that it can be parsed easily.
00057     // const std::string& lLangCode = Language::getShortLabel (_languageCode);
00058
00059     unsigned short idx = 0;
00060     for (NameList_T::const_iterator itName = _nameList.begin();
00061          itName != _nameList.end(); ++itName, ++idx) {
00062       if (idx != 0) {
00063         oStr << ",";
00064       }
00065       const std::string& lName = *itName;
00066       oStr << _languageCode << "," << lName;
00067     }
00068
00069     return oStr.str();
00070   }
00071
00072   // ////////////////////////////////////////////////////////////////////
00073   void Names::toStream (std::ostream& ioOut) const {
00074     ioOut << describe();
00075   }
00076
00077   // ////////////////////////////////////////////////////////////////////
00078   void Names::fromStream (std::istream& ioIn) {
00079   }
00080
00081   // ////////////////////////////////////////////////////////////////////
00082   std::string Names::toString() const {
00083     std::ostringstream oStr;
00084     oStr << describe();
00085     return oStr.str();
00086   }
00087
00088   // ////////////////////////////////////////////////////////////////////
00089   void Names::addName (const std::string& iName) {
00090     _nameList.push_back (iName);
00091   }
00092
00093   // ////////////////////////////////////////////////////////////////////
00094   void Names::resetList() {
00095     _nameList.clear();
00096   }
00097
00098 }
```

24.49 opentrep/basic/OTransliterator.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <opentrep/OPENTREP_exceptions.hpp>
#include <opentrep/basic/icu_util.hpp>
#include <opentrep/basic/BasConst_Unicode.hpp>
#include <opentrep/basic/OTransliterator.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.50 OTransliterator.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // OpenTrep
00008 #include <opentrep/OPENTREP_exceptions.hpp>
```

```
00009 #include <opentrep/basic/icu_util.hpp>
00010 #include <opentrep/basic/BasConst_Unicode.hpp>
00011 #include <opentrep/basic/OTransliterator.hpp>
00012 #include <opentrep/service/Logger.hpp>
00013
00014 namespace OPENTREP {
00015
00016   // ////////////////////////////////////////////////////////////////////
00017   OTransliterator::OTransliterator()
00018     : _normaliser (NULL), _unquoter (NULL), _unpunctuater (NULL),
00019       _tranlist (NULL) {
00020     init();
00021   }
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   OTransliterator::OTransliterator (const
      OTransliterator& iTransliterator)
00025     : _normaliser (NULL), _unquoter (NULL), _unpunctuater (NULL),
00026       _tranlist (NULL) {
00027     assert (iTransliterator._normaliser != NULL);
00028     _normaliser = iTransliterator._normaliser->clone();
00029
00030     assert (iTransliterator._unquoter != NULL);
00031     _unquoter = iTransliterator._unquoter->clone();
00032
00033     assert (iTransliterator._unpunctuater != NULL);
00034     _unpunctuater = iTransliterator._unpunctuater->clone();
00035
00036     assert (iTransliterator._tranlist != NULL);
00037     _tranlist = iTransliterator._tranlist->clone();
00038
00039   }
00040
00041   // ////////////////////////////////////////////////////////////////////
00042   OTransliterator::~OTransliterator() {
00043     finalise();
00044   }
00045
00046   // ////////////////////////////////////////////////////////////////////
00047   void OTransliterator::init() {
00048     // Create a normaliser
00049     UErrorCode lStatus = U_ZERO_ERROR;
00050
00051     _normaliser =
00052       Transliterator::createInstance (K_ICU_ACCENT_REMOVAL_RULE, UTRANS_FORWARD,
00053                                       lStatus);
00054
00055     if (_normaliser == NULL || U_FAILURE (lStatus)) {
00056       std::ostringstream oStr;
00057       oStr << "Unicode error: no Transliterator can be created for the '"
00058            << K_ICU_ACCENT_REMOVAL_RULE << "' rule.";
00059       OPENTREP_LOG_ERROR (oStr.str());
00060       throw UnicodeTransliteratorCreationException (oStr.str());
00061     }
00062     assert (_normaliser != NULL);
00063
00064     // Register the Unicode Transliterator
00065     Transliterator::registerInstance (_normaliser);
00066
00067     // Create a remover of quotation
00068     lStatus = U_ZERO_ERROR;
00069     UParseError pError;
00070     UnicodeString lUnquotedRules (K_ICU_QUOTATION_REMOVAL_RULE);
00071     _unquoter = Transliterator::createFromRules("RBTUnaccent", lUnquotedRules,
00072                                                 UTRANS_FORWARD, pError, lStatus);
00073
00074     if (_unquoter == NULL || U_FAILURE (lStatus)) {
00075       std::ostringstream oStr;
00076       oStr << "Unicode error: no Transliterator can be created for the '"
00077            << K_ICU_QUOTATION_REMOVAL_RULE << "' rule.";
00078       OPENTREP_LOG_ERROR (oStr.str());
00079       throw UnicodeTransliteratorCreationException (oStr.str());
00080     }
00081     assert (_unquoter != NULL);
00082
00083     // Register the Unicode Transliterator
00084     Transliterator::registerInstance (_unquoter);
00085
00086     // Create a remover of punctuation
00087     lStatus = U_ZERO_ERROR;
00088
00089     _unpunctuater =
00090       Transliterator::createInstance (K_ICU_PUNCTUATION_REMOVAL_RULE,
00091                                       UTRANS_FORWARD, lStatus);
00092
00093     if (_unpunctuater == NULL || U_FAILURE (lStatus)) {
00094       std::ostringstream oStr;
```

```
00095       oStr << "Unicode error: no Transliterator can be created for the '"
00096            << K_ICU_PUNCTUATION_REMOVAL_RULE << "' rule.";
00097       OPENTREP_LOG_ERROR (oStr.str());
00098       throw UnicodeTransliteratorCreationException (oStr.str());
00099     }
00100     assert (_unpunctuater != NULL);
00101
00102     // Register the Unicode Transliterator
00103     Transliterator::registerInstance (_unpunctuater);
00104
00105     // Create a generic transliterator
00106     lStatus = U_ZERO_ERROR;
00107
00108     _tranlist =
00109       Transliterator::createInstance (K_ICU_GENERIC_TRANSLITERATOR_RULE,
00110                                       UTRANS_FORWARD, lStatus);
00111
00112     if (_tranlist == NULL || U_FAILURE (lStatus)) {
00113       std::ostringstream oStr;
00114       oStr << "Unicode error: no Transliterator can be created for the '"
00115            << K_ICU_GENERIC_TRANSLITERATOR_RULE << "' rule.";
00116       OPENTREP_LOG_ERROR (oStr.str());
00117       throw UnicodeTransliteratorCreationException (oStr.str());
00118     }
00119     assert (_tranlist != NULL);
00120
00121     // Register the Unicode Transliterator
00122     Transliterator::registerInstance (_tranlist);
00123   }
00124
00125   // ////////////////////////////////////////////////////////////////////
00126   void OTransliterator::finalise() {
00127     delete _normaliser; _normaliser = NULL;
00128     delete _unquoter; _unquoter = NULL;
00129     delete _unpunctuater; _unpunctuater = NULL;
00130     delete _tranlist; _tranlist = NULL;
00131   }
00132
00133   // ////////////////////////////////////////////////////////////////////
00134   std::string OTransliterator::normalise (const std::string& iString) const {
00135     assert (_normaliser != NULL);
00136
00137     // Build a UnicodeString from the STL string
00138     UnicodeString lString (iString.c_str());
00139
00140     // Apply the whole sery of transformators
00141     _normaliser->transliterate (lString);
00142     _unquoter->transliterate (lString);
00143     _unpunctuater->transliterate (lString);
00144     _tranlist->transliterate (lString);
00145
00146     // Convert from UnicodeString to UTF8-encoded STL string
00147     const std::string& lNormalisedString = getUTF8 (lString);
00148
00149     return lNormalisedString;
00150   }
00151
00152 }
```

24.51 opentrep/basic/OTransliterator.hpp File Reference

```
#include <string>
#include <unicode/unistr.h>
#include <unicode/translit.h>
#include <opentrep/basic/StructAbstract.hpp>
```

Classes

- class OPENTREP::OTransliterator

Namespaces

- OPENTREP

24.52 OTransliterator.hpp

```
00001 #ifndef __OPENTREP_BAS_OTRANSLITERATOR_HPP
00002 #define __OPENTREP_BAS_OTRANSLITERATOR_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // ICU
00010 #include <unicode/unistr.h> // UnicodeString
00011 #include <unicode/translit.h> // Transliterator
00012 // OpenTrep
00013 #include <opentrep/basic/StructAbstract.hpp>
00014
00015 namespace OPENTREP {
00016
00020   class OTransliterator {
00021   public:
00022     // ///////////////// Business support methods ///////////////
00029     std::string normalise (const std::string& iString) const;
00030
00031
00032   public:
00033     // ///////////////// Construction and destruction ///////////////
00037     OTransliterator();
00038
00042     OTransliterator (const OTransliterator&);
00043
00047     ~OTransliterator();
00048
00049
00050   private:
00058     void init();
00059
00063     void finalise();
00064
00065
00066   private:
00067     // ////////////////////// Attributes ////////////////////
00071     Transliterator* _normaliser;
00072
00076     Transliterator* _unquoter;
00077
00081     Transliterator* _unpunctuater;
00082
00088     Transliterator* _tranlist;
00089   };
00090
00091 }
00092 #endif // __OPENTREP_BAS_OTRANSLITERATOR_HPP
```

24.53 opentrep/basic/OutputFormat.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <opentrep/OutputFormat.hpp>
```

Namespaces

- OPENTREP

24.54 OutputFormat.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // OpenTREP
00008 #include <opentrep/OutputFormat.hpp>
00009
00010 namespace OPENTREP {
```

```
00011
00012 // ////////////////////////////////////////////////////////////////////
00013 const std::string OutputFormat::_labels[LAST_VALUE] =
00014   { "Short", "Full", "JSON", "PROTOBUF" };
00015
00016 // ////////////////////////////////////////////////////////////////////
00017 const char OutputFormat::_formatLabels[LAST_VALUE] = { 'S', 'F', 'J', 'P' };
00018
00019
00020 // ////////////////////////////////////////////////////////////////////
00021 OutputFormat::OutputFormat() : _format (LAST_VALUE) {
00022   assert (false);
00023 }
00024
00025 // ////////////////////////////////////////////////////////////////////
00026 OutputFormat::
00027 OutputFormat (const OutputFormat& iOutputFormat)
00028   : _format (iOutputFormat._format) {
00029 }
00030
00031 // ////////////////////////////////////////////////////////////////////
00032 OutputFormat::
00033 OutputFormat (const EN_OutputFormat& iOutputFormat)
00034   : _format (iOutputFormat) {
00035 }
00036
00037 // ////////////////////////////////////////////////////////////////////
00038 OutputFormat::EN_OutputFormat
00039 OutputFormat::getFormat (const char iFormatChar) {
00040   EN_OutputFormat oFormat;
00041   switch (iFormatChar) {
00042   case 'S': oFormat = SHORT; break;
00043   case 'F': oFormat = FULL; break;
00044   case 'J': oFormat = JSON; break;
00045   case 'P': oFormat = PROTOBUF; break;
00046   default: oFormat = LAST_VALUE; break;
00047   }
00048
00049   if (oFormat == LAST_VALUE) {
00050     const std::string& lLabels = describeLabels();
00051     std::ostringstream oMessage;
00052     oMessage << "The output format '" << iFormatChar
00053              << "' is not known. Known output formats: " << lLabels;
00054     throw CodeConversionException (oMessage.str());
00055   }
00056
00057   return oFormat;
00058 }
00059
00060 // ////////////////////////////////////////////////////////////////////
00061 OutputFormat::OutputFormat (const char iFormatChar)
00062   : _format (getFormat (iFormatChar)) {
00063 }
00064
00065 // ////////////////////////////////////////////////////////////////////
00066 OutputFormat::
00067 OutputFormat (const std::string& iFormatStr) {
00068   //
00069   const size_t lSize = iFormatStr.size();
00070   assert (lSize == 1);
00071   const char lFormatChar = iFormatStr[0];
00072   _format = getFormat (lFormatChar);
00073 }
00074
00075 // ////////////////////////////////////////////////////////////////////
00076 const std::string& OutputFormat::
00077 getLabel (const EN_OutputFormat& iFormat) {
00078   return _labels[iFormat];
00079 }
00080
00081 // ////////////////////////////////////////////////////////////////////
00082 char OutputFormat::
00083 getFormatLabel (const EN_OutputFormat& iFormat) {
00084   return _formatLabels[iFormat];
00085 }
00086
00087 // ////////////////////////////////////////////////////////////////////
00088 std::string OutputFormat::
00089 getFormatLabelAsString (const EN_OutputFormat& iFormat) {
00090   std::ostringstream oStr;
00091   oStr << _formatLabels[iFormat];
00092   return oStr.str();
00093 }
00094
00095 // ////////////////////////////////////////////////////////////////////
00096 std::string OutputFormat::describeLabels() {
00097   std::ostringstream ostr;
```

```
00098     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00099       if (idx != 0) {
00100         ostr << ", ";
00101       }
00102       ostr << _labels[idx];
00103     }
00104     return ostr.str();
00105   }
00106
00107   // ////////////////////////////////////////////////////////////////////
00108   OutputFormat::EN_OutputFormat
00109   OutputFormat::getFormat() const {
00110     return _format;
00111   }
00112
00113   // ////////////////////////////////////////////////////////////////////
00114   char OutputFormat::getFormatAsChar() const {
00115     const char oFormatChar = _formatLabels[_format];
00116     return oFormatChar;
00117   }
00118
00119   // ////////////////////////////////////////////////////////////////////
00120   std::string OutputFormat::getFormatAsString() const {
00121     std::ostringstream oStr;
00122     oStr << _formatLabels[_format];
00123     return oStr.str();
00124   }
00125
00126   // ////////////////////////////////////////////////////////////////////
00127   const std::string OutputFormat::describe() const {
00128     std::ostringstream ostr;
00129     ostr << _labels[_format];
00130     return ostr.str();
00131   }
00132
00133   // ////////////////////////////////////////////////////////////////////
00134   bool OutputFormat::
00135   operator== (const EN_OutputFormat& iFormat) const {
00136     return (_format == iFormat);
00137   }
00138
00139 }
```

24.55 opentrep/basic/ScoreType.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <opentrep/OPENTREP_exceptions.hpp>
#include <opentrep/basic/ScoreType.hpp>
```

Namespaces

- OPENTREP

24.56 ScoreType.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // OpenTrep
00008 #include <opentrep/OPENTREP_exceptions.hpp>
00009 #include <opentrep/basic/ScoreType.hpp>
00010
00011 namespace OPENTREP {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   const std::string ScoreType::_labels[LAST_VALUE] =
00015         { "Combination", "Xapian Percentage", "Page Rank", "Passenger Number",
00016           "Heuristic", "Envelope ID", "IATA/ICAO Code Full Match"};
00017
00018   const char ScoreType::_typeLabels[LAST_VALUE] = { 'C', 'X', 'R', 'N', 'H',
00019                                                     'E', 'F' };
```

```
00020
00021
00022   // //////////////////////////////////////////////////////////////////
00023   ScoreType::ScoreType (const EN_ScoreType& iScoreType)
00024     : _type (iScoreType) {
00025   }
00026
00027   // //////////////////////////////////////////////////////////////////
00028   ScoreType::ScoreType (const char iType) {
00029     switch (iType) {
00030     case 'C': _type = COMBINATION; break;
00031     case 'X': _type = XAPIAN_PCT; break;
00032     case 'R': _type = PAGE_RANK; break;
00033     case 'N': _type = PAX_NB; break;
00034     case 'H': _type = HEURISTIC; break;
00035     case 'E': _type = ENV_ID; break;
00036     case 'F': _type = CODE_FULL_MATCH; break;
00037     default: _type = LAST_VALUE; break;
00038     }
00039
00040     if (_type == LAST_VALUE) {
00041       const std::string& lLabels = describeLabels();
00042       std::ostringstream oMessage;
00043       oMessage << "The score type '" << iType
00044                << "' is not known. Known score types: " << lLabels;
00045       throw CodeConversionException (oMessage.str());
00046     }
00047   }
00048
00049   // //////////////////////////////////////////////////////////////////
00050   const std::string& ScoreType::getLongLabel() const {
00051     return _labels[_type];
00052   }
00053
00054   // //////////////////////////////////////////////////////////////////
00055   const std::string& ScoreType::getLabel (const EN_ScoreType& iType) {
00056     return _labels[iType];
00057   }
00058
00059   // //////////////////////////////////////////////////////////////////
00060   char ScoreType::getTypeLabel (const EN_ScoreType& iType) {
00061     return _typeLabels[iType];
00062   }
00063
00064   // //////////////////////////////////////////////////////////////////
00065   std::string ScoreType::
00066   getTypeLabelAsString (const EN_ScoreType& iType) {
00067     std::ostringstream oStr;
00068     oStr << _typeLabels[iType];
00069     return oStr.str();
00070   }
00071
00072   // //////////////////////////////////////////////////////////////////
00073   std::string ScoreType::describeLabels() {
00074     std::ostringstream ostr;
00075     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00076       if (idx != 0) {
00077         ostr << ", ";
00078       }
00079       ostr << _labels[idx];
00080     }
00081     return ostr.str();
00082   }
00083
00084   // //////////////////////////////////////////////////////////////////
00085   ScoreType::EN_ScoreType ScoreType::getType() const {
00086     return _type;
00087   }
00088
00089   // //////////////////////////////////////////////////////////////////
00090   std::string ScoreType::getTypeAsString() const {
00091     std::ostringstream oStr;
00092     oStr << _typeLabels[_type];
00093     return oStr.str();
00094   }
00095
00096   // //////////////////////////////////////////////////////////////////
00097   std::string ScoreType::describe() const {
00098     std::ostringstream ostr;
00099     ostr << _labels[_type];
00100     return ostr.str();
00101   }
00102
00103   // //////////////////////////////////////////////////////////////////
00104   bool ScoreType::isIndividualScore() const {
00105     bool oIsIndividual = true;
00106     if (_type == COMBINATION || _type == LAST_VALUE) {
```

```
00107       oIsIndividual = false;
00108     }
00109     return oIsIndividual;
00110   }
00111
00112   // ////////////////////////////////////////////////////////////////////
00113   bool ScoreType::isIndividualScore (const
      EN_ScoreType& iTypeEnum) {
00114     ScoreType lType (iTypeEnum);
00115     return lType.isIndividualScore();
00116   }
00117
00118   // ////////////////////////////////////////////////////////////////////
00119   bool ScoreType::operator== (const EN_ScoreType& iType) const {
00120     return (_type == iType);
00121   }
00122
00123 }
```

24.57 opentrep/basic/ScoreType.hpp File Reference

```
#include <string>
#include <opentrep/basic/StructAbstract.hpp>
```

Classes

- struct OPENTREP::ScoreType

  Enumeration of score types.

Namespaces

- OPENTREP

24.58 ScoreType.hpp

```
00001 #ifndef __OPENTREP_BAS_SCORETYPE_HPP
00002 #define __OPENTREP_BAS_SCORETYPE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // OpenTrep
00010 #include <opentrep/basic/StructAbstract.hpp>
00011
00012 namespace OPENTREP {
00013
00025   struct ScoreType : public StructAbstract {
00026   public:
00027     typedef enum {
00028       COMBINATION = 0,
00029       XAPIAN_PCT,
00030       PAGE_RANK,
00031       PAX_NB,
00032       HEURISTIC,
00033       ENV_ID,
00034       CODE_FULL_MATCH,
00035       LAST_VALUE
00036     } EN_ScoreType;
00037
00041     static const std::string& getLabel (const EN_ScoreType&);
00042
00046     const std::string& getLongLabel() const;
00047
00051     static char getTypeLabel (const EN_ScoreType&);
00052
00056     static std::string getTypeLabelAsString (const
      EN_ScoreType&);
00057
00061     static std::string describeLabels();
00062
```

```
00066     EN_ScoreType getType() const;
00067
00071     std::string getTypeAsString() const;
00072
00077     std::string describe() const;
00078
00083     bool isIndividualScore() const;
00084
00089     static bool isIndividualScore (const EN_ScoreType&);
00090
00091
00092   public:
00096     bool operator== (const EN_ScoreType&) const;
00097
00098   public:
00102     ScoreType (const EN_ScoreType&);
00103
00107     ScoreType (const char iType);
00108
00109
00110   private:
00114     static const std::string _labels[LAST_VALUE];
00115
00119     static const char _typeLabels[LAST_VALUE];
00120
00121
00122   private:
00123     // ******** Attributes *********
00127     EN_ScoreType _type;
00128   };
00129
00130 }
00131 #endif // __OPENTREP_BAS_SCORETYPE_HPP
```

24.59 opentrep/basic/StructAbstract.hpp File Reference

```
#include <iosfwd>
#include <string>
```

Classes

- struct OPENTREP::StructAbstract

  Base class for the light structures.

Namespaces

- OPENTREP

Functions

- template<class charT , class traits >
  std::basic_ostream< charT,
  traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const OPENTREP::StructAbstract &iStruct)
- template<class charT , class traits >
  std::basic_istream< charT,
  traits > & operator>> (std::basic_istream< charT, traits > &ioIn, OPENTREP::StructAbstract &ioStruct)

24.59.1 Function Documentation

24.59.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut,* const OPENTREP::StructAbstract & *iStruct*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 63 of file StructAbstract.hpp.

24.59.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn,* OPENTREP::StructAbstract & *ioStruct*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 91 of file StructAbstract.hpp.

References OPENTREP::StructAbstract::fromStream().

24.60 StructAbstract.hpp

```
00001 #ifndef __OPENTREP_BAS_STRUCTABSTRACT_HPP
00002 #define __OPENTREP_BAS_STRUCTABSTRACT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010
00011 namespace OPENTREP {
00012
00016   struct StructAbstract {
00017   public:
00018
00022     virtual ~StructAbstract() {}
00023
00029     void toStream (std::ostream& ioOut) const {
00030       ioOut << describe();
00031     }
00032
00038     virtual void fromStream (std::istream& ioIn) {}
00039
00045     virtual std::string describe() const = 0;
00046
00047   protected:
00051     StructAbstract() {}
00052   };
00053 }
00054
00060 template <class charT, class traits>
00061 inline
00062 std::basic_ostream<charT, traits>&
00063 operator<< (std::basic_ostream<charT, traits>& ioOut,
00064             const OPENTREP::StructAbstract& iStruct) {
00070   std::basic_ostringstream<charT,traits> ostr;
00071   ostr.copyfmt (ioOut);
00072   ostr.width (0);
00073
00074   // Fill string stream
00075   iStruct.toStream (ostr);
00076
00077   // Print string stream
00078   ioOut << ostr.str();
00079
00080   return ioOut;
00081 }
00082
00088 template <class charT, class traits>
00089 inline
00090 std::basic_istream<charT, traits>&
00091 operator>> (std::basic_istream<charT, traits>& ioIn,
00092             OPENTREP::StructAbstract& ioStruct) {
00093   // Fill the Structure object with the input stream.
00094   ioStruct.fromStream (ioIn);
00095   return ioIn;
00096
00097 }
00098 #endif // __OPENTREP_BAS_STRUCTABSTRACT_HPP
```

24.61 opentrep/basic/UTF8Handler.cpp File Reference

```
#include <cassert>
```

```
#include <sstream>
#include <string>
#include <opentrep/basic/UTF8Handler.hpp>
```

Namespaces

- OPENTREP

Typedefs

- typedef long unsigned int OPENTREP::u_int32_t

Variables

- static const u_int32_t OPENTREP::offsetsFromUTF8 [6]
- static const char OPENTREP::trailingBytesForUTF8 [256]

24.62 UTF8Handler.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <string>
00008 // OpenTrep
00009 #include <opentrep/basic/UTF8Handler.hpp>
00010
00011 namespace OPENTREP {
00012
00016   typedef long unsigned int u_int32_t;
00017
00018   // //////////////////////////////////////////////////////////////////////
00019   static const u_int32_t offsetsFromUTF8[6] = {
00020     0x00000000UL, 0x00003080UL, 0x000E2080UL,
00021     0x03C82080UL, 0xFA082080UL, 0x82082080UL
00022   };
00023
00024   // //////////////////////////////////////////////////////////////////////
00025   static const char trailingBytesForUTF8[256] = {
00026     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
00027     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
00028     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
00029     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
00030     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
00031     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
00032     1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
00033     2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2, 3,3,3,3,3,3,3,4,4,4,4,5,5,5,5
00034   };
00035
00036   // //////////////////////////////////////////////////////////////////////
00037   std::wstring UTF8Handler::toWideString (const std::string& iSrc) {
00038     std::basic_ostringstream<wchar_t> oStr;
00039
00040     // Length of the source string
00041     const size_t lStringSize = iSrc.size();
00042
00043     // Transform the source string in a regular C-string (char*)
00044     const char* src = iSrc.c_str();
00045
00046     //
00047     typedef unsigned char uchar_t;
00048
00049     size_t idx = 0;
00050     while (idx != lStringSize) {
00051
00052       uchar_t lCurrentChar = static_cast<uchar_t> (src[idx]);
00053
00062       if (lCurrentChar == '\0') {
00063         break;
00064       }
```

```
00065
00066       const int nb = trailingBytesForUTF8[lCurrentChar];
00067
00068       wchar_t tmpChar = 0;
00069       switch (nb) {
00070         // These fall through deliberately
00071       case 3: {
00072         lCurrentChar = static_cast<uchar_t> (src[idx]); ++idx;
00073         tmpChar += lCurrentChar; tmpChar <<= 6;
00074       }
00075       case 2: {
00076         lCurrentChar = static_cast<uchar_t> (src[idx]); ++idx;
00077         tmpChar += lCurrentChar; tmpChar <<= 6;
00078       }
00079       case 1: {
00080         lCurrentChar = static_cast<uchar_t> (src[idx]); ++idx;
00081         tmpChar += lCurrentChar; tmpChar <<= 6;
00082       }
00083       case 0: {
00084         lCurrentChar = static_cast<uchar_t> (src[idx]); ++idx;
00085         tmpChar += lCurrentChar;
00086       }
00087       }
00088
00089       tmpChar -= offsetsFromUTF8[nb];
00090       oStr << tmpChar;
00091     }
00092
00093     oStr << '\0';
00094     return oStr.str();
00095   }
00096
00097   // ////////////////////////////////////////////////////////////////////
00098   std::string UTF8Handler::toSimpleString (const std::wstring& iStr) {
00099     std::ostringstream oStr;
00100
00101     const wchar_t* src = iStr.c_str();
00102     size_t idx = 0;
00103     size_t i = 0;
00104
00105     while (src[i] != 0) {
00106       wchar_t ch = src[i];
00107
00108       if (ch < 0x80) {
00109         const char tmpChar = static_cast<const char> (ch);
00110         oStr << tmpChar; ++idx;
00111
00112       } else if (ch < 0x800) {
00113         char tmpChar = static_cast<const char> ((ch >> 6) | 0xC0);
00114         oStr << tmpChar; ++idx;
00115
00116         tmpChar = static_cast<const char> ((ch & 0x3F) | 0x80);
00117         oStr << tmpChar; ++idx;
00118
00119       } else if (ch < 0x10000) {
00120         char tmpChar = static_cast<const char> ((ch>>12) | 0xE0);
00121         oStr << tmpChar; ++idx;
00122
00123         tmpChar = static_cast<const char> (((ch>>6) & 0x3F) | 0x80);
00124         oStr << tmpChar; ++idx;
00125
00126         tmpChar = static_cast<const char> ((ch & 0x3F) | 0x80);
00127         oStr << tmpChar; ++idx;
00128
00129       } else if (ch < 0x110000) {
00130         char tmpChar = static_cast<const char> ((ch>>18) | 0xF0);
00131         oStr << tmpChar; ++idx;
00132
00133         tmpChar = static_cast<const char> (((ch>>12) & 0x3F) | 0x80);
00134         oStr << tmpChar; ++idx;
00135
00136         tmpChar = static_cast<const char> (((ch>>6) & 0x3F) | 0x80);
00137         oStr << tmpChar; ++idx;
00138
00139         tmpChar = static_cast<const char> ((ch & 0x3F) | 0x80);
00140         oStr << tmpChar; ++idx;
00141       }
00142       i++;
00143     }
00144
00145     oStr << '\0';
00146
00147     return oStr.str();
00148   }
00149
00150   // ////////////////////////////////////////////////////////////////////
00151   std::string UTF8Handler::displayCharString (const char* iString) {
```

```
00152     std::ostringstream oStr;
00153
00154     bool hasReachedEnd = false;
00155     for (size_t idx = 0; hasReachedEnd == false; ++idx) {
00156       if (idx != 0) {
00157         oStr << "; ";
00158       }
00159       const unsigned char lChar = iString[idx];
00160       // const wchar_t lChar = iString[idx];
00161       if (lChar == '\0') {
00162         hasReachedEnd = true;
00163       }
00164       oStr << "[" << idx << "]: " << std::hex << lChar;
00165     }
00166     oStr << std::endl;
00167
00168     return oStr.str();
00169   }
00170
00171   // ////////////////////////////////////////////////////////////////////
00172   std::string UTF8Handler::displaySTLWString (const std::wstring& iString) {
00173     std::ostringstream oStr;
00174
00175     size_t idx = 0;
00176     for (std::wstring::const_iterator itChar = iString.begin();
00177          itChar != iString.end(); ++itChar, ++idx) {
00178       if (idx != 0) {
00179         oStr << "; ";
00180       }
00181       const wchar_t lChar = *itChar;
00182       oStr << "[" << idx << "]: " << std::hex << lChar;
00183     }
00184     oStr << std::endl;
00185
00186     return oStr.str();
00187   }
00188
00189 }
00190
```

24.63 opentrep/basic/UTF8Handler.hpp File Reference

```
#include <string>
```

Classes

- class OPENTREP::UTF8Handler

  Utility class for basic handling of UTF-8 encoded strings.

Namespaces

- OPENTREP

24.64 UTF8Handler.hpp

```
00001 #ifndef __OPENTREP_BAS_UTF8HANDLER_HPP
00002 #define __OPENTREP_BAS_UTF8HANDLER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace OPENTREP {
00011
00019   class UTF8Handler {
00020   public:
00021     /*
00022      * Conversion from a UTF-8-encoded "simple character" (though
00023      * potentially multi-byte) STL string into a wide character STL
00024      * string.
```

```
00025      *
00026      * Note that as there is no checks of appropriate encoding, it
00027      * only works for valid UTF-8, i.e. no 5- or 6-byte sequences.
00028      *
00029      * Note that the "simple characters", within a STL string, may be
00030      * multi-byte (e.g., if they are UTF-8-encoded).
00031      *
00032      * @param std::string The "simple character" (though potentially
00033      *        multi-byte) STL string.
00034      * @return std::wstring The wide character STL string.
00035      */
00036     static std::wstring toWideString (const std::string& iSrc);
00037
00038     /*
00039      * Conversion from a wide character STL string into a UTF-8-encoded
00040      * "simple character" (though potentially multi-byte) STL string.
00041      *
00042      * Note that as there is no checks of appropriate encoding, it
00043      * only works for valid UTF-8, i.e. no 5- or 6-byte sequences.
00044      *
00045      * Note that the "simple characters", within a STL string, may be
00046      * multi-byte (e.g., if they are UTF-8-encoded).
00047      *
00048      * @param std::wstring The wide character STL string.
00049      * @return std::string The "simple character" (though potentially
00050      *         multi-byte) STL string.
00051      */
00052     static std::string toSimpleString (const std::wstring& iStr);
00053
00057     static std::string displayCharString (const char* iString);
00058
00063     static std::string displaySTLWString (const std::wstring& iString);
00064   };
00065
00066 }
00067 #endif // __OPENTREP_BAS_UTF8HANDLER_HPP
```

24.65 opentrep/basic/Utilities.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/tokenizer.hpp>
#include <opentrep/basic/Utilities.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

Functions

- void OPENTREP::tokeniseStringIntoWordList (const std::string &iPhrase, WordList_T &ioWordList)
- std::string OPENTREP::createStringFromWordList (const WordList_T &iWordList, const unsigned short iSplitIdx, const bool iFromBeginningFlag)

24.66 Utilities.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost (Extended STL)
00008 #include <boost/tokenizer.hpp>
00009 // OpenTrep
00010 #include <opentrep/basic/Utilities.hpp>
00011 #include <opentrep/service/Logger.hpp>
00012
00013 namespace OPENTREP {
```

```
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   void tokeniseStringIntoWordList (const std::string& iPhrase,
00017                                    WordList_T& ioWordList) {
00018     // Empty the word list
00019     ioWordList.clear();
00020
00021     // Boost Tokeniser
00022     typedef boost::tokenizer<boost::char_separator<char> > Tokeniser_T;
00023
00024     // Define the separators
00025     const boost::char_separator<char>
00026       lSepatorList(" .,;:|+-*/_=!@#$%`~^&(){}[]?'<>\"");
00027
00028     // Initialise the phrase to be tokenised
00029     Tokeniser_T lTokens (iPhrase, lSepatorList);
00030     for (Tokeniser_T::const_iterator tok_iter = lTokens.begin();
00031          tok_iter != lTokens.end(); ++tok_iter) {
00032       const std::string& lTerm = *tok_iter;
00033       ioWordList.push_back (lTerm);
00034     }
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   std::string createStringFromWordList (const WordList_T& iWordList,
00039                                         const unsigned short iSplitIdx,
00040                                         const bool iFromBeginningFlag) {
00041     std::ostringstream oStr;
00042
00043     // Browse the left-hand side of the string
00044     unsigned short idx = 0;
00045     WordList_T::const_iterator itWord = iWordList.begin();
00046     for ( ; itWord != iWordList.end(); ++itWord, ++idx) {
00047
00048       if (iFromBeginningFlag == true) {
00049         // The beginning of the word list is needed
00050
00051         // Check whether the sub-list has reached the expected split point,
00052         // if any
00053         if (iSplitIdx != 0 && idx >= iSplitIdx) {
00054           break;
00055         }
00056
00057         //
00058         if (idx > 0) {
00059           oStr << " ";
00060         }
00061         //
00062         const std::string& lWord = *itWord;
00063         oStr << lWord;
00064
00065       } else {
00066         // The end of the word list is needed
00067
00068         // Check whether the sub-list has reached the expected split point,
00069         // if any
00070         if (iSplitIdx == 0 || idx >= iSplitIdx) {
00071           break;
00072         }
00073       }
00074     }
00075
00076     // The beginning of the word list is needed
00077     if (iFromBeginningFlag == true) {
00078       return oStr.str();
00079     }
00080
00081     // The end of the word list is needed
00082     assert (iFromBeginningFlag == false);
00083
00084     // Browse the right-hand side of the string
00085     for ( ; itWord != iWordList.end(); ++itWord, ++idx) {
00086       // The end of the word list is needed
00087
00088       //
00089       if (idx > iSplitIdx) {
00090         oStr << " ";
00091       }
00092       //
00093       const std::string& lWord = *itWord;
00094       oStr << lWord;
00095     }
00096
00097     return oStr.str();
00098   }
00099
00100 }
```

24.67 opentrep/basic/Utilities.hpp File Reference

```
#include <string>
#include <vector>
#include <opentrep/OPENTREP_Types.hpp>
```

Namespaces

- OPENTREP

Functions

- void OPENTREP::tokeniseStringIntoWordList (const std::string &iPhrase, WordList_T &ioWordList)
- std::string OPENTREP::createStringFromWordList (const WordList_T &iWordList, const unsigned short iSplitIdx, const bool iFromBeginningFlag)

24.68 Utilities.hpp

```
00001 #ifndef __OPENTREP_BAS_UTILITIES_HPP
00002 #define __OPENTREP_BAS_UTILITIES_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // OpenTrep
00011 #include <opentrep/OPENTREP_Types.hpp>
00012
00013 namespace OPENTREP {
00014
00018   void tokeniseStringIntoWordList (const std::string& iPhrase,
      WordList_T&);
00019
00030   std::string createStringFromWordList (const WordList_T&,
00031                                         const unsigned short iSplitIdx = 0,
00032                                         const bool iFromBeginningFlag = true);
00033
00034 }
00035 #endif // __OPENTREP_BAS_UTILITIES_HPP
```

24.69 opentrep/batches/opentrep-dbmgr.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include <string>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/program_options.hpp>
#include <opentrep/OPENTREP_Service.hpp>
#include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
#include <opentrep/config/opentrep-paths.hpp>
```

Typedefs

- typedef std::vector< std::string > WordList_T

Functions

- const std::string K_OPENTREP_DEFAULT_LOG_FILENAME ("opentrep-dbmgr.log")
- int readConfiguration (int argc, char ∗argv[], std::string &ioPORFilepath, std::string &ioXapianDBFilepath, std::string &ioSQLiteDBFilepath, std::string &ioLogFilename)
- int main (int argc, char ∗argv[])

Variables

- const int K_OPENTREP_EARLY_RETURN_STATUS = 99

24.69.1 Typedef Documentation

24.69.1.1 typedef std::vector<std::string> WordList_T

Definition at line 19 of file opentrep-dbmgr.cpp.

24.69.2 Function Documentation

24.69.2.1 const std::string K_OPENTREP_DEFAULT_LOG_FILENAME ("opentrep-dbmgr.log")

Default name and location for the log file.

Referenced by readConfiguration().

24.69.2.2 int readConfiguration (int *argc*, char ∗ *argv[]*, std::string & *ioPORFilepath*, std::string & *ioXapianDBFilepath*, std::string & *ioSQLiteDBFilepath*, std::string & *ioLogFilename*)

Read and parse the command line options.

Definition at line 34 of file opentrep-dbmgr.cpp.

References OPENTREP::DEFAULT_OPENTREP_POR_FILEPATH, OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH, OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH, K_OPENTREP_DEFAULT_LOG_FILENAME(), and K_OPENTREP_EARLY_RETURN_STATUS.

Referenced by main().

24.69.2.3 int main (int *argc*, char ∗ *argv[]*)

Definition at line 137 of file opentrep-dbmgr.cpp.

References OPENTREP::OPENTREP_Service::buildSQLDB(), K_OPENTREP_EARLY_RETURN_STATUS, and readConfiguration().

24.69.3 Variable Documentation

24.69.3.1 const int K_OPENTREP_EARLY_RETURN_STATUS = 99

Early return status (so that it can be differentiated from an error).

Definition at line 31 of file opentrep-dbmgr.cpp.

Referenced by main(), and readConfiguration().

24.70 opentrep-dbmgr.cpp

```
00001 // STL
00002 #include <cassert>
00003 #include <iostream>
```

```
00004 #include <sstream>
00005 #include <fstream>
00006 #include <vector>
00007 #include <string>
00008 // Boost (Extended STL)
00009 #include <boost/date_time/posix_time/posix_time.hpp>
00010 #include <boost/date_time/gregorian/gregorian.hpp>
00011 #include <boost/program_options.hpp>
00012 // OpenTREP
00013 #include <opentrep/OPENTREP_Service.hpp>
00014 #include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
00015 #include <opentrep/config/opentrep-paths.hpp>
00016
00017
00018 // //////// Type definitions ///////
00019 typedef std::vector<std::string> WordList_T;
00020
00021
00022 // //////// Constants //////
00026 const std::string K_OPENTREP_DEFAULT_LOG_FILENAME ("opentrep-dbmgr.log");
00027
00028
00029 // ///////// Parsing of Options & Configuration /////////
00031 const int K_OPENTREP_EARLY_RETURN_STATUS = 99;
00032
00034 int readConfiguration (int argc, char* argv[],
00035                        std::string& ioPORFilepath,
00036                        std::string& ioXapianDBFilepath,
00037                        std::string& ioSQLiteDBFilepath,
00038                        std::string& ioLogFilename) {
00039
00040   // Declare a group of options that will be allowed only on command line
00041   boost::program_options::options_description generic ("Generic options");
00042   generic.add_options()
00043     ("prefix", "print installation prefix")
00044     ("version,v", "print version string")
00045     ("help,h", "produce help message");
00046
00047   // Declare a group of options that will be allowed both on command
00048   // line and in config file
00049   boost::program_options::options_description config ("Configuration");
00050   config.add_options()
00051     ("porfile,p",
00052      boost::program_options::value< std::string >(&ioPORFilepath)->default_value(
      OPENTREP::DEFAULT_OPENTREP_POR_FILEPATH),
00053      "POR file-path (e.g., ori_por_public.csv)")
00054     ("xapiandb,d",
00055      boost::program_options::value< std::string >(&ioXapianDBFilepath)->default_value(
      OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH),
00056      "Xapian database filepath (e.g., /tmp/opentrep/traveldb)")
00057     ("sqlite,s",
00058      boost::program_options::value< std::string >(&ioSQLiteDBFilepath)->default_value(
      OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH),
00059      "SQLite3 database filepath (e.g., ~/tmp/opentrep/traveldb/ori_por_public.db)")
00060     ("log,l",
00061      boost::program_options::value< std::string >(&ioLogFilename)->default_value(
      K_OPENTREP_DEFAULT_LOG_FILENAME),
00062      "Filepath for the logs")
00063     ;
00064
00065   // Hidden options, will be allowed both on command line and
00066   // in config file, but will not be shown to the user.
00067   boost::program_options::options_description hidden ("Hidden options");
00068   hidden.add_options()
00069     ("copyright",
00070      boost::program_options::value< std::vector<std::string> >(),
00071      "Show the copyright (license)");
00072
00073   boost::program_options::options_description cmdline_options;
00074   cmdline_options.add(generic).add(config).add(hidden);
00075
00076   boost::program_options::options_description config_file_options;
00077   config_file_options.add(config).add(hidden);
00078
00079   boost::program_options::options_description visible ("Allowed options");
00080   visible.add(generic).add(config);
00081
00082   boost::program_options::positional_options_description p;
00083   p.add ("copyright", -1);
00084
00085   boost::program_options::variables_map vm;
00086   boost::program_options::
00087     store (boost::program_options::command_line_parser (argc, argv).
00088            options (cmdline_options).positional(p).run(), vm);
00089
00090   std::ifstream ifs ("opentrep-dbmgr.cfg");
00091   boost::program_options::store (parse_config_file (ifs, config_file_options),
```

```
00092                                 vm);
00093   boost::program_options::notify (vm);
00094
00095   if (vm.count ("help")) {
00096     std::cout << visible << std::endl;
00097     return K_OPENTREP_EARLY_RETURN_STATUS;
00098   }
00099
00100   if (vm.count ("version")) {
00101     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00102     return K_OPENTREP_EARLY_RETURN_STATUS;
00103   }
00104
00105   if (vm.count ("prefix")) {
00106     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00107     return K_OPENTREP_EARLY_RETURN_STATUS;
00108   }
00109
00110   if (vm.count ("porfile")) {
00111     ioPORFilepath = vm["porfile"].as< std::string >();
00112     std::cout << "POR file-path is: " << ioPORFilepath << std::endl;
00113   }
00114
00115   if (vm.count ("xapiandb")) {
00116     ioXapianDBFilepath = vm["xapiandb"].as< std::string >();
00117     std::cout << "Xapian database filepath is: " << ioXapianDBFilepath
00118               << std::endl;
00119   }
00120
00121   if (vm.count ("sqlitedb")) {
00122     ioSQLiteDBFilepath = vm["sqlitedb"].as< std::string >();
00123     std::cout << "SQLite3 database filepath is: " << ioSQLiteDBFilepath
00124               << std::endl;
00125   }
00126
00127   if (vm.count ("log")) {
00128     ioLogFilename = vm["log"].as< std::string >();
00129     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00130   }
00131
00132   return 0;
00133 }
00134
00135
00136 // ///////////////// M A I N /////////////////
00137 int main (int argc, char* argv[]) {
00138
00139   // Output log File
00140   std::string lLogFilename;
00141
00142   // File-path of POR (points of reference)
00143   std::string lPORFilepathStr;
00144
00145   // Xapian database name (directory of the index)
00146   std::string lXapianDBNameStr;
00147
00148   // SQLite3 database file-path
00149   std::string lSQLiteDBFilePathStr;
00150
00151   // Call the command-line option parser
00152   const int lOptionParserStatus =
00153     readConfiguration (argc, argv, lPORFilepathStr,
00154                        lXapianDBNameStr, lSQLiteDBFilePathStr, lLogFilename);
00155
00156   if (lOptionParserStatus == K_OPENTREP_EARLY_RETURN_STATUS) {
00157     return 0;
00158   }
00159
00160   // Set the log parameters
00161   std::ofstream logOutputFile;
00162   // open and clean the log outputfile
00163   logOutputFile.open (lLogFilename.c_str());
00164   logOutputFile.clear();
00165
00166   //
00167   std::cout << "Creating the SQLite3 database may take a few minutes "
00168             << "on some architectures (and a few seconds on fastest ones)..."
00169             << std::endl;
00170
00171   // Initialise the context
00172   const OPENTREP::PORFilePath_T lPORFilepath (lPORFilepathStr);
00173   const OPENTREP::TravelDBFilePath_T lXapianDBName (lXapianDBNameStr);
00174   const OPENTREP::SQLiteDBFilePath_T lSQLiteDBFilePath (lSQLiteDBFilePathStr);
00175   OPENTREP::OPENTREP_Service opentrepService (logOutputFile, lPORFilepath,
00176                                               lXapianDBName, lSQLiteDBFilePath);
00177
00178   // Launch the indexation
```

```
00179   const OPENTREP::NbOfDBEntries_T lNbOfEntries = opentrepService.
      buildSQLDB();
00180
00181   // Close the Log outputFile
00182   logOutputFile.close();
00183
00184   //
00185   std::cout << lNbOfEntries << " entries have been processed" << std::endl;
00186
00187   return 0;
00188 }
```

24.71 opentrep/batches/opentrep-indexer.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include <string>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/program_options.hpp>
#include <opentrep/OPENTREP_Service.hpp>
#include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
#include <opentrep/config/opentrep-paths.hpp>
```

Typedefs

- typedef std::vector< std::string > WordList_T

Functions

- const std::string K_OPENTREP_DEFAULT_LOG_FILENAME ("opentrep-indexer.log")
- int readConfiguration (int argc, char *argv[], std::string &ioPORFilepath, std::string &ioXapianDBFilepath, std::string &ioSQLiteDBFilepath, std::string &ioLogFilename)
- int main (int argc, char *argv[])

Variables

- const int K_OPENTREP_EARLY_RETURN_STATUS = 99

24.71.1 Typedef Documentation

24.71.1.1 typedef std::vector<std::string> WordList_T

Definition at line 19 of file opentrep-indexer.cpp.

24.71.2 Function Documentation

24.71.2.1 const std::string K_OPENTREP_DEFAULT_LOG_FILENAME ("opentrep-indexer.log")

Default name and location for the log file.

Referenced by readConfiguration().

24.71.2.2 int readConfiguration (int *argc*, char * *argv[]*, std::string & *ioPORFilepath*, std::string & *ioXapianDBFilepath*, std::string & *ioSQLiteDBFilepath*, std::string & *ioLogFilename*)

Read and parse the command line options.

Definition at line 34 of file opentrep-indexer.cpp.

References OPENTREP::DEFAULT_OPENTREP_POR_FILEPATH, OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH, OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH, K_OPENTREP_DEFAULT_LOG_FILENAME(), and K_OPENTREP_EARLY_RETURN_STATUS.

Referenced by main().

24.71.2.3 int main (int *argc*, char * *argv[]*)

Definition at line 137 of file opentrep-indexer.cpp.

References OPENTREP::OPENTREP_Service::buildSearchIndex(), K_OPENTREP_EARLY_RETURN_STATUS, and readConfiguration().

24.71.3 Variable Documentation

24.71.3.1 const int K_OPENTREP_EARLY_RETURN_STATUS = 99

Early return status (so that it can be differentiated from an error).

Definition at line 31 of file opentrep-indexer.cpp.

Referenced by main(), and readConfiguration().

24.72 opentrep-indexer.cpp

```
00001 // STL
00002 #include <cassert>
00003 #include <iostream>
00004 #include <sstream>
00005 #include <fstream>
00006 #include <vector>
00007 #include <string>
00008 // Boost (Extended STL)
00009 #include <boost/date_time/posix_time/posix_time.hpp>
00010 #include <boost/date_time/gregorian/gregorian.hpp>
00011 #include <boost/program_options.hpp>
00012 // OpenTREP
00013 #include <opentrep/OPENTREP_Service.hpp>
00014 #include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
00015 #include <opentrep/config/opentrep-paths.hpp>
00016
00017
00018 // //////// Type definitions ///////
00019 typedef std::vector<std::string> WordList_T;
00020
00021
00022 // //////// Constants //////
00026 const std::string K_OPENTREP_DEFAULT_LOG_FILENAME ("opentrep-indexer.log");
00027
00028
00029 // ///////// Parsing of Options & Configuration /////////
00031 const int K_OPENTREP_EARLY_RETURN_STATUS = 99;
00032
00034 int readConfiguration (int argc, char* argv[],
00035                        std::string& ioPORFilepath,
00036                        std::string& ioXapianDBFilepath,
00037                        std::string& ioSQLiteDBFilepath,
00038                        std::string& ioLogFilename) {
00039
00040   // Declare a group of options that will be allowed only on command line
00041   boost::program_options::options_description generic ("Generic options");
00042   generic.add_options()
00043     ("prefix", "print installation prefix")
00044     ("version,v", "print version string")
00045     ("help,h", "produce help message");
00046
00047   // Declare a group of options that will be allowed both on command
00048   // line and in config file
```

```
00049   boost::program_options::options_description config ("Configuration");
00050   config.add_options()
00051     ("porfile,p",
00052      boost::program_options::value< std::string >(&ioPORFilepath)->default_value(
      OPENTREP::DEFAULT_OPENTREP_POR_FILEPATH),
00053      "POR file-path (e.g., ori_por_public.csv)")
00054     ("xapiandb,d",
00055      boost::program_options::value< std::string >(&ioXapianDBFilepath)->default_value(
      OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH),
00056      "Xapian database filepath (e.g., /tmp/opentrep/traveldb)")
00057     ("sqlite,s",
00058      boost::program_options::value< std::string >(&ioSQLiteDBFilepath)->default_value(
      OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH),
00059      "SQLite3 database filepath (e.g., ~/tmp/opentrep/traveldb/ori_por_public.db)")
00060     ("log,l",
00061      boost::program_options::value< std::string >(&ioLogFilename)->default_value(
      K_OPENTREP_DEFAULT_LOG_FILENAME),
00062      "Filepath for the logs")
00063     ;
00064
00065   // Hidden options, will be allowed both on command line and
00066   // in config file, but will not be shown to the user.
00067   boost::program_options::options_description hidden ("Hidden options");
00068   hidden.add_options()
00069     ("copyright",
00070      boost::program_options::value< std::vector<std::string> >(),
00071      "Show the copyright (license)");
00072
00073   boost::program_options::options_description cmdline_options;
00074   cmdline_options.add(generic).add(config).add(hidden);
00075
00076   boost::program_options::options_description config_file_options;
00077   config_file_options.add(config).add(hidden);
00078
00079   boost::program_options::options_description visible ("Allowed options");
00080   visible.add(generic).add(config);
00081
00082   boost::program_options::positional_options_description p;
00083   p.add ("copyright", -1);
00084
00085   boost::program_options::variables_map vm;
00086   boost::program_options::
00087     store (boost::program_options::command_line_parser (argc, argv).
00088            options (cmdline_options).positional(p).run(), vm);
00089
00090   std::ifstream ifs ("opentrep-indexer.cfg");
00091   boost::program_options::store (parse_config_file (ifs, config_file_options),
00092                                  vm);
00093   boost::program_options::notify (vm);
00094
00095   if (vm.count ("help")) {
00096     std::cout << visible << std::endl;
00097     return K_OPENTREP_EARLY_RETURN_STATUS;
00098   }
00099
00100   if (vm.count ("version")) {
00101     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00102     return K_OPENTREP_EARLY_RETURN_STATUS;
00103   }
00104
00105   if (vm.count ("prefix")) {
00106     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00107     return K_OPENTREP_EARLY_RETURN_STATUS;
00108   }
00109
00110   if (vm.count ("porfile")) {
00111     ioPORFilepath = vm["porfile"].as< std::string >();
00112     std::cout << "POR file-path is: " << ioPORFilepath << std::endl;
00113   }
00114
00115   if (vm.count ("xapiandb")) {
00116     ioXapianDBFilepath = vm["xapiandb"].as< std::string >();
00117     std::cout << "Xapian database filepath is: " << ioXapianDBFilepath
00118               << std::endl;
00119   }
00120
00121   if (vm.count ("sqlitedb")) {
00122     ioSQLiteDBFilepath = vm["sqlitedb"].as< std::string >();
00123     std::cout << "SQLite3 database filepath is: " << ioSQLiteDBFilepath
00124               << std::endl;
00125   }
00126
00127   if (vm.count ("log")) {
00128     ioLogFilename = vm["log"].as< std::string >();
00129     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00130   }
00131
```

```
00132   return 0;
00133 }
00134
00135
00136 // /////////////// M A I N /////////////////
00137 int main (int argc, char* argv[]) {
00138
00139   // Output log File
00140   std::string lLogFilename;
00141
00142   // File-path of POR (points of reference)
00143   std::string lPORFilepathStr;
00144
00145   // Xapian database name (directory of the index)
00146   std::string lXapianDBNameStr;
00147
00148   // SQLite3 database file-path
00149   std::string lSQLiteDBFilePathStr;
00150
00151   // Call the command-line option parser
00152   const int lOptionParserStatus =
00153     readConfiguration (argc, argv, lPORFilepathStr,
00154                        lXapianDBNameStr, lSQLiteDBFilePathStr, lLogFilename);
00155
00156   if (lOptionParserStatus == K_OPENTREP_EARLY_RETURN_STATUS) {
00157     return 0;
00158   }
00159
00160   // Set the log parameters
00161   std::ofstream logOutputFile;
00162   // open and clean the log outputfile
00163   logOutputFile.open (lLogFilename.c_str());
00164   logOutputFile.clear();
00165
00166   //
00167   std::cout << "Creating the Xapian index/database may take a few minutes "
00168             << "on some architectures (and a few seconds on fastest ones)..."
00169             << std::endl;
00170
00171   // Initialise the context
00172   const OPENTREP::PORFilePath_T lPORFilepath (lPORFilepathStr);
00173   const OPENTREP::TravelDBFilePath_T lXapianDBName (lXapianDBNameStr);
00174   const OPENTREP::SQLiteDBFilePath_T lSQLiteDBFilePath (lSQLiteDBFilePathStr);
00175   OPENTREP::OPENTREP_Service opentrepService (logOutputFile, lPORFilepath,
00176                                               lXapianDBName, lSQLiteDBFilePath);
00177
00178   // Launch the indexation
00179   const OPENTREP::NbOfDBEntries_T lNbOfEntries =
00180     opentrepService.buildSearchIndex();
00181
00182   // Close the Log outputFile
00183   logOutputFile.close();
00184
00185   //
00186   std::cout << lNbOfEntries << " entries have been processed" << std::endl;
00187
00188   return 0;
00189 }
```

24.73 opentrep/batches/opentrep-searcher.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include <string>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/tokenizer.hpp>
#include <boost/program_options.hpp>
#include <opentrep/OPENTREP_Service.hpp>
#include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
#include <opentrep/Location.hpp>
#include <opentrep/config/opentrep-paths.hpp>
```

Typedefs

- typedef std::vector< std::string > WordList_T

Functions

- const std::string K_OPENTREP_DEFAULT_LOG_FILENAME ("opentrep-searcher.log")
- const std::string K_OPENTREP_DEFAULT_QUERY_STRING ("sna francicso rio de janero lso angles reykyavki")
- void tokeniseStringIntoWordList (const std::string &iPhrase, WordList_T &ioWordList)
- std::string createStringFromWordList (const WordList_T &iWordList)
- template<class T >
  std::ostream & operator<< (std::ostream &os, const std::vector< T > &v)
- int readConfiguration (int argc, char *argv[], unsigned short &ioSpellingErrorDistance, std::string &ioQueryString, std::string &ioXapianDBFilepath, std::string &ioSQLiteDBFilepath, std::string &ioLogFilename, unsigned short &ioSearchType)
- std::string parseQuery (OPENTREP::OPENTREP_Service &ioOpentrepService, const OPENTREP::TravelQuery_T &iTravelQuery)
- int main (int argc, char *argv[])

Variables

- const unsigned short K_OPENTREP_DEFAULT_SEARCH_TYPE = 0
- const unsigned short K_OPENTREP_DEFAULT_SPELLING_ERROR_DISTANCE = 3
- const int K_OPENTREP_EARLY_RETURN_STATUS = 99

24.73.1 Typedef Documentation

24.73.1.1 typedef std::vector<std::string> WordList_T

Definition at line 21 of file opentrep-searcher.cpp.

24.73.2 Function Documentation

24.73.2.1 const std::string K_OPENTREP_DEFAULT_LOG_FILENAME ("opentrep-searcher.log")

Default name and location for the log file.

Referenced by readConfiguration().

24.73.2.2 const std::string K_OPENTREP_DEFAULT_QUERY_STRING ("sna francicso rio de janero lso angles reykyavki")

Default travel query string, to be seached against the Xapian database.

Referenced by readConfiguration().

24.73.2.3 void tokeniseStringIntoWordList (const std::string & *iPhrase,* WordList_T & *ioWordList*)

Definition at line 52 of file opentrep-searcher.cpp.

Referenced by readConfiguration().

24.73.2.4 std::string createStringFromWordList (const WordList_T & *iWordList*)

Definition at line 73 of file opentrep-searcher.cpp.

Referenced by readConfiguration().

24.73.2.5 template<class T > std::ostream& operator<< (std::ostream & *os,* const std::vector< T > & *v*)

Definition at line 92 of file opentrep-searcher.cpp.

24.73.2.6 int readConfiguration (int *argc,* char * *argv[],* unsigned short & *ioSpellingErrorDistance,* std::string & *ioQueryString,* std::string & *ioXapianDBFilepath,* std::string & *ioSQLiteDBFilepath,* std::string & *ioLogFilename,* unsigned short & *ioSearchType*)

Read and parse the command line options.

Definition at line 102 of file opentrep-searcher.cpp.

References createStringFromWordList(), OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH, OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH, K_OPENTREP_DEFAULT_LOG_FILENAME(), K_OPENTREP_DEFAULT_QUERY_STRING(), K_OPENTREP_DEFAULT_SEARCH_TYPE, K_OPENTREP_DEFAULT_SPELLING_ERROR_DISTANCE, K_OPENTREP_EARLY_RETURN_STATUS, and tokeniseStringIntoWordList().

Referenced by main().

24.73.2.7 std::string parseQuery (OPENTREP::OPENTREP_Service & *ioOpentrepService,* const OPENTREP::TravelQuery_T & *iTravelQuery*)

Helper function

Definition at line 226 of file opentrep-searcher.cpp.

References OPENTREP::OPENTREP_Service::interpretTravelRequest().

Referenced by main().

24.73.2.8 int main (int *argc,* char * *argv[]*)

Definition at line 268 of file opentrep-searcher.cpp.

References K_OPENTREP_EARLY_RETURN_STATUS, parseQuery(), and readConfiguration().

24.73.3 Variable Documentation

24.73.3.1 const unsigned short K_OPENTREP_DEFAULT_SEARCH_TYPE = 0

Default type for the search request.

- 0 = Full text
- 1 = Coordinates

Definition at line 43 of file opentrep-searcher.cpp.

Referenced by readConfiguration().

24.73.3.2 const unsigned short K_OPENTREP_DEFAULT_SPELLING_ERROR_DISTANCE = 3

Default error distance for spelling corrections.

Definition at line 48 of file opentrep-searcher.cpp.

Referenced by readConfiguration().

24.73.3.3 const int K_OPENTREP_EARLY_RETURN_STATUS = 99

Early return status (so that it can be differentiated from an error).

Definition at line 99 of file opentrep-searcher.cpp.

Referenced by main(), and readConfiguration().

24.74 opentrep-searcher.cpp

```
00001 // STL
00002 #include <cassert>
00003 #include <iostream>
00004 #include <sstream>
00005 #include <fstream>
00006 #include <vector>
00007 #include <string>
00008 // Boost (Extended STL)
00009 #include <boost/date_time/posix_time/posix_time.hpp>
00010 #include <boost/date_time/gregorian/gregorian.hpp>
00011 #include <boost/tokenizer.hpp>
00012 #include <boost/program_options.hpp>
00013 // OpenTREP
00014 #include <opentrep/OPENTREP_Service.hpp>
00015 #include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
00016 #include <opentrep/Location.hpp>
00017 #include <opentrep/config/opentrep-paths.hpp>
00018
00019
00020 // //////// Type definitions ///////
00021 typedef std::vector<std::string> WordList_T;
00022
00023
00024 // //////// Constants //////
00028 const std::string K_OPENTREP_DEFAULT_LOG_FILENAME ("opentrep-searcher.log");
00029
00033 const std::string K_OPENTREP_DEFAULT_QUERY_STRING ("sna francicso rio de
       janero lso angles reykyavki");
00034
00043 const unsigned short K_OPENTREP_DEFAULT_SEARCH_TYPE = 0;
00044
00048 const unsigned short K_OPENTREP_DEFAULT_SPELLING_ERROR_DISTANCE =
       3;
00049
00050
00051 // ////////////////////////////////////////////////////////////////////
00052 void tokeniseStringIntoWordList (const std::string& iPhrase,
00053                                  WordList_T& ioWordList) {
00054   // Empty the word list
00055   ioWordList.clear();
00056
00057   // Boost Tokeniser
00058   typedef boost::tokenizer<boost::char_separator<char> > Tokeniser_T;
00059
00060   // Define the separators
00061   const boost::char_separator<char> lSepatorList(" .,;:|+-*/_=!@#$%`~^&(){}[]?'<>\"");
00062
00063   // Initialise the phrase to be tokenised
00064   Tokeniser_T lTokens (iPhrase, lSepatorList);
00065   for (Tokeniser_T::const_iterator tok_iter = lTokens.begin();
00066        tok_iter != lTokens.end(); ++tok_iter) {
00067     const std::string& lTerm = *tok_iter;
00068     ioWordList.push_back (lTerm);
00069   }
00070 }
00071
00072 // ////////////////////////////////////////////////////////////////////
00073 std::string createStringFromWordList (const WordList_T& iWordList) {
00074   std::ostringstream oStr;
00075
00076   unsigned short idx = iWordList.size();
00077   for (WordList_T::const_iterator itWord = iWordList.begin();
00078        itWord != iWordList.end(); ++itWord, --idx) {
00079     const std::string& lWord = *itWord;
00080     oStr << lWord;
00081     if (idx > 1) {
00082       oStr << " ";
00083     }
00084   }
00085
00086   return oStr.str();
00087 }
00088
00089
00090 // ///////// Parsing of Options & Configuration /////////
00091 // A helper function to simplify the main part.
00092 template<class T> std::ostream& operator<< (std::ostream& os,
00093                                             const std::vector<T>& v) {
00094   std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00095   return os;
00096 }
00097
00099 const int K_OPENTREP_EARLY_RETURN_STATUS = 99;
00100
00102 int readConfiguration (int argc, char* argv[],
```

```
00103                       unsigned short& ioSpellingErrorDistance,
00104                       std::string& ioQueryString,
00105                       std::string& ioXapianDBFilepath,
00106                       std::string& ioSQLiteDBFilepath,
00107                       std::string& ioLogFilename,
00108                       unsigned short& ioSearchType) {
00109
00110   // Initialise the travel query string, if that one is empty
00111   if (ioQueryString.empty() == true) {
00112     ioQueryString = K_OPENTREP_DEFAULT_QUERY_STRING;
00113   }
00114
00115   // Transform the query string into a list of words (STL strings)
00116   WordList_T lWordList;
00117   tokeniseStringIntoWordList (ioQueryString, lWordList);
00118
00119   // Declare a group of options that will be allowed only on command line
00120   boost::program_options::options_description generic ("Generic options");
00121   generic.add_options()
00122     ("prefix", "print installation prefix")
00123     ("version,v", "print version string")
00124     ("help,h", "produce help message");
00125
00126   // Declare a group of options that will be allowed both on command
00127   // line and in config file
00128   boost::program_options::options_description config ("Configuration");
00129   config.add_options()
00130     ("error,e",
00131      boost::program_options::value< unsigned short >(&ioSpellingErrorDistance)->default_value(
      K_OPENTREP_DEFAULT_SPELLING_ERROR_DISTANCE),
00132      "Spelling error distance (e.g., 3)")
00133     ("xapiandb,d",
00134      boost::program_options::value< std::string >(&ioXapianDBFilepath)->default_value(
      OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH),
00135      "Xapian database filepath (e.g., /tmp/opentrep/traveldb)")
00136     ("sqlite,s",
00137      boost::program_options::value< std::string >(&ioSQLiteDBFilepath)->default_value(
      OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH),
00138      "SQLite3 database filepath (e.g., ~/tmp/opentrep/traveldb/ori_por_public.db)")
00139     ("log,l",
00140      boost::program_options::value< std::string >(&ioLogFilename)->default_value(
      K_OPENTREP_DEFAULT_LOG_FILENAME),
00141      "Filepath for the logs")
00142     ("type,t",
00143      boost::program_options::value<unsigned short>(&ioSearchType)->default_value(
      K_OPENTREP_DEFAULT_SEARCH_TYPE),
00144      "Type of search request (0 = full text, 1 = coordinates)")
00145     ("query,q",
00146      boost::program_options::value< WordList_T >(&lWordList)->multitoken(),
00147      "Travel query word list (e.g. sna francicso rio de janero lso anglese reykyavki), which sould be
       located at the end of the command line (otherwise, the other options would be interpreted as part of that travel
       query word list)")
00148     ;
00149
00150   // Hidden options, will be allowed both on command line and
00151   // in config file, but will not be shown to the user.
00152   boost::program_options::options_description hidden ("Hidden options");
00153   hidden.add_options()
00154     ("copyright",
00155      boost::program_options::value< std::vector<std::string> >(),
00156      "Show the copyright (license)");
00157
00158   boost::program_options::options_description cmdline_options;
00159   cmdline_options.add(generic).add(config).add(hidden);
00160
00161   boost::program_options::options_description config_file_options;
00162   config_file_options.add(config).add(hidden);
00163
00164   boost::program_options::options_description visible ("Allowed options");
00165   visible.add(generic).add(config);
00166
00167   boost::program_options::positional_options_description p;
00168   p.add ("copyright", -1);
00169
00170   boost::program_options::variables_map vm;
00171   boost::program_options::
00172     store (boost::program_options::command_line_parser (argc, argv).
00173            options (cmdline_options).positional(p).run(), vm);
00174
00175   std::ifstream ifs ("opentrep-searcher.cfg");
00176   boost::program_options::store (parse_config_file (ifs, config_file_options),
00177                                  vm);
00178   boost::program_options::notify (vm);
00179
00180   if (vm.count ("help")) {
00181     std::cout << visible << std::endl;
00182     return K_OPENTREP_EARLY_RETURN_STATUS;
```

```
00183   }
00184
00185   if (vm.count ("version")) {
00186     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00187     return K_OPENTREP_EARLY_RETURN_STATUS;
00188   }
00189
00190   if (vm.count ("prefix")) {
00191     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00192     return K_OPENTREP_EARLY_RETURN_STATUS;
00193   }
00194
00195   if (vm.count ("xapiandb")) {
00196     ioXapianDBFilepath = vm["xapiandb"].as< std::string >();
00197     std::cout << "Xapian database filepath is: " << ioXapianDBFilepath
00198               << std::endl;
00199   }
00200
00201   if (vm.count ("sqlitedb")) {
00202     ioSQLiteDBFilepath = vm["sqlitedb"].as< std::string >();
00203     std::cout << "SQLite3 database filepath is: " << ioSQLiteDBFilepath
00204               << std::endl;
00205   }
00206
00207   if (vm.count ("log")) {
00208     ioLogFilename = vm["log"].as< std::string >();
00209     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00210   }
00211
00212   std::cout << "The type of search is: " << ioSearchType << std::endl;
00213
00214   std::cout << "The spelling error distance is: " << ioSpellingErrorDistance
00215             << std::endl;
00216
00217   ioQueryString = createStringFromWordList (lWordList);
00218   std::cout << "The travel query string is: " << ioQueryString << std::endl;
00219
00220   return 0;
00221 }
00222
00226 std::string parseQuery (OPENTREP::OPENTREP_Service& ioOpentrepService,
00227                         const OPENTREP::TravelQuery_T& iTravelQuery) {
00228   std::ostringstream oStr;
00229
00230   // Query the Xapian database (index)
00231   OPENTREP::WordList_T lNonMatchedWordList;
00232   OPENTREP::LocationList_T lLocationList;
00233   const OPENTREP::NbOfMatches_T nbOfMatches =
00234     ioOpentrepService.interpretTravelRequest (iTravelQuery, lLocationList,
00235                                               lNonMatchedWordList);
00236
00237   oStr << nbOfMatches << " (geographical) location(s) have been found "
00238        << "matching your query (`" << iTravelQuery << "'). "
00239        << lNonMatchedWordList.size() << " words were left unmatched."
00240        << std::endl;
00241
00242   if (nbOfMatches != 0) {
00243     OPENTREP::NbOfMatches_T idx = 1;
00244     for (OPENTREP::LocationList_T::const_iterator itLocation =
00245            lLocationList.begin();
00246          itLocation != lLocationList.end(); ++itLocation, ++idx) {
00247       const OPENTREP::Location& lLocation = *itLocation;
00248       oStr << " [" << idx << "]: " << lLocation << std::endl;
00249     }
00250   }
00251
00252   if (lNonMatchedWordList.empty() == false) {
00253     oStr << "List of unmatched words:" << std::endl;
00254
00255     OPENTREP::NbOfMatches_T idx = 1;
00256     for (OPENTREP::WordList_T::const_iterator itWord =
00257            lNonMatchedWordList.begin();
00258          itWord != lNonMatchedWordList.end(); ++itWord, ++idx) {
00259       const OPENTREP::Word_T& lWord = *itWord;
00260       oStr << " [" << idx << "]: " << lWord << std::endl;
00261     }
00262   }
00263
00264   return oStr.str();
00265 }
00266
00267 // ///////////////// M A I N //////////////////
00268 int main (int argc, char* argv[]) {
00269
00270   /*
00271     const OPENTREP::NbOfLetters_T lScaleArray[5] = {3, 6, 9, 14, 19};
00272
```

```
00273     const OPENTREP::DistanceErrorScaleArray_T lScaleBoostArray =
00274     { {3, 6, 9, 14, 19} };
00275
00276     OPENTREP::DistanceErrorRule lScale (5, lScaleArray);
00277     OPENTREP::DistanceErrorRule lScaleBoost (lScaleBoostArray);
00278
00279     std::cout << "Standard array: " << lScale << std::endl;
00280     std::cout << "Boost array: " << lScaleBoost << std::endl;
00281
00282     for (int idx = 0; idx != 20; ++idx) {
00283     std::cout << "For " << idx << " letters => "
00284     << lScale.getAllowedDistanceError(idx) << std::endl;
00285     }
00286
00287     return 0;
00288   */
00289
00290   // Travel query
00291   OPENTREP::TravelQuery_T lTravelQuery;
00292
00293   // Output log File
00294   std::string lLogFilename;
00295
00296   // Xapian database name (directory of the index)
00297   std::string lXapianDBNameStr;
00298
00299   // SQLite3 database file-path
00300   std::string lSQLiteDBFilePathStr;
00301
00302   // Type of search
00303   unsigned short lSearchType;
00304
00305   // Xapian spelling error distance
00306   unsigned short lSpellingErrorDistance;
00307
00308   // Call the command-line option parser
00309   const int lOptionParserStatus =
00310     readConfiguration (argc, argv, lSpellingErrorDistance, lTravelQuery,
00311                        lXapianDBNameStr, lSQLiteDBFilePathStr,
00312                        lLogFilename, lSearchType);
00313
00314   if (lOptionParserStatus == K_OPENTREP_EARLY_RETURN_STATUS) {
00315     return 0;
00316   }
00317
00318   // Set the log parameters
00319   std::ofstream logOutputFile;
00320   // open and clean the log outputfile
00321   logOutputFile.open (lLogFilename.c_str());
00322   logOutputFile.clear();
00323
00324   if (lSearchType == 0) {
00325     // Initialise the context
00326     const OPENTREP::TravelDBFilePath_T lXapianDBName (lXapianDBNameStr);
00327     const OPENTREP::SQLiteDBFilePath_T lSQLiteDBFilePath (lSQLiteDBFilePathStr)
      ;
00328     OPENTREP::OPENTREP_Service opentrepService(logOutputFile,
00329                                                lXapianDBName, lSQLiteDBFilePath);
00330
00331     // Parse the query and retrieve the places from Xapian only
00332     const std::string& lOutput = parseQuery (opentrepService, lTravelQuery);
00333     std::cout << lOutput;
00334
00335   } else {
00336     std::cout << "Finding the airports closest to: " << lTravelQuery
00337               << std::endl;
00338   }
00339
00340   // Close the Log outputFile
00341   logOutputFile.close();
00342
00343   return 0;
00344 }
```

24.75 opentrep/bom/Airline.cpp File Reference

24.76 Airline.cpp

24.77 opentrep/bom/Airline.hpp File Reference

24.78 Airline.hpp

24.79 opentrep/bom/BomAbstract.hpp File Reference

```
#include <iosfwd>
#include <sstream>
#include <string>
```

Classes

- class OPENTREP::BomAbstract

  Base class for the Business Object Model (BOM) layer.

Namespaces

- OPENTREP

Functions

- template<class charT , class traits >
  std::basic_ostream< charT,
  traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const OPENTREP::BomAbstract &iBom)
- template<class charT , class traits >
  std::basic_istream< charT,
  traits > & operator>> (std::basic_istream< charT, traits > &ioIn, OPENTREP::BomAbstract &ioBom)

24.79.1 Function Documentation

24.79.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut,* const OPENTREP::BomAbstract & *iBom*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 75 of file BomAbstract.hpp.

24.79.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn,* OPENTREP::BomAbstract & *ioBom*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 103 of file BomAbstract.hpp.

References OPENTREP::BomAbstract::fromStream().

24.80 BomAbstract.hpp

```
00001 #ifndef __OPENTREP_BOM_BOMABSTRACT_HPP
00002 #define __OPENTREP_BOM_BOMABSTRACT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
```

```
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <sstream>
00010 #include <string>
00011
00012 namespace OPENTREP {
00013
00017   class BomAbstract {
00018     friend class FacBomAbstract;
00019   public:
00020     // ////////// Display support methods /////////
00026     virtual void toStream (std::ostream&) const = 0;
00027
00033     virtual void fromStream (std::istream&) = 0;
00034
00038     virtual std::string toString() const = 0;
00039
00044     virtual std::string describeKey() const = 0;
00045
00050     virtual std::string describeShortKey() const = 0;
00051
00052
00053   protected:
00057     BomAbstract() {}
00058     BomAbstract (const BomAbstract&) {}
00059
00063     virtual ~BomAbstract() {}
00064   };
00065 }
00066
00072 template <class charT, class traits>
00073 inline
00074 std::basic_ostream<charT, traits>&
00075 operator<< (std::basic_ostream<charT, traits>& ioOut,
00076             const OPENTREP::BomAbstract& iBom) {
00082   std::basic_ostringstream<charT,traits> ostr;
00083   ostr.copyfmt (ioOut);
00084   ostr.width (0);
00085
00086   // Fill string stream
00087   iBom.toStream (ostr);
00088
00089   // Print string stream
00090   ioOut << ostr.str();
00091
00092   return ioOut;
00093 }
00094
00100 template <class charT, class traits>
00101 inline
00102 std::basic_istream<charT, traits>&
00103 operator>> (std::basic_istream<charT, traits>& ioIn,
00104             OPENTREP::BomAbstract& ioBom) {
00105   // Fill Bom object with input stream
00106   ioBom.fromStream (ioIn);
00107   return ioIn;
00108 }
00109
00110 #endif // __OPENTREP_BOM_BOMABSTRACT_HPP
```

24.81 opentrep/bom/BomJSONExport.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <opentrep/Location.hpp>
#include <opentrep/bom/BomJSONExport.hpp>
```

Namespaces

- OPENTREP

24.82 BomJSONExport.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <ostream>
00007 // Boost ForEach
00008 //#include <boost/foreach.hpp>
00009 // OpenTREP
00010 #include <opentrep/Location.hpp>
00011 #include <opentrep/bom/BomJSONExport.hpp>
00012
00013 namespace OPENTREP {
00014
00015   // //////////////////////////////////////////////////////////////////
00016   void BomJSONExport::
00017   jsonExportLocationList (std::ostream& oStream,
00018                           const LocationList_T& iLocationList) {
00019
00020     // Create empty Boost.Property_Tree objects
00021     bpt::ptree lPT;
00022     bpt::ptree lPTLocationList;
00023
00024     for (LocationList_T::const_iterator itLocation = iLocationList.begin();
00025          itLocation != iLocationList.end(); ++itLocation) {
00026       const Location& lLocation = *itLocation;
00027       //
00028       bpt::ptree lPTLocation;
00029       jsonExportLocation (lPTLocation, lLocation);
00030
00031       // List of extra matching locations (those with the same matching
00032       // weight/percentage)
00033       const LocationList_T& lExtraLocationList= lLocation.
    getExtraLocationList();
00034       if (lExtraLocationList.empty() == false) {
00035         //
00036         bpt::ptree lPTExtraLocationList;
00037
00038         NbOfMatches_T idxExtra = 0;
00039         for (LocationList_T::const_iterator itLoc = lExtraLocationList.begin();
00040              itLoc != lExtraLocationList.end(); ++itLoc, ++idxExtra) {
00041           const Location& lExtraLocation = *itLoc;
00042           //
00043           bpt::ptree lPTExtraLocation;
00044           jsonExportLocation (lPTExtraLocation, lExtraLocation);
00045
00046           // Put the current location tree in the location list
00047           lPTExtraLocationList.push_back (std::make_pair("", lPTExtraLocation));
00048         }
00049
00050         // Add the extra location list to the Boost.Property_Tree representing
00051         // the current Location instance.
00052         lPTLocation.add_child ("extras", lPTExtraLocationList);
00053       }
00054
00055       // List of alternate matching locations (those with a lower matching
00056       // weight/percentage)
00057       const LocationList_T& lAltLocationList =
00058         lLocation.getAlternateLocationList();
00059       if (lAltLocationList.empty() == false) {
00060         //
00061         bpt::ptree lPTAltLocationList;
00062
00063         NbOfMatches_T idxAlter = 0;
00064         for (LocationList_T::const_iterator itLoc = lAltLocationList.begin();
00065              itLoc != lAltLocationList.end(); ++itLoc, ++idxAlter) {
00066           const Location& lAltLocation = *itLoc;
00067           //
00068           bpt::ptree lPTAltLocation;
00069           jsonExportLocation (lPTAltLocation, lAltLocation);
00070
00071           // Put the current location tree in the location list
00072           lPTAltLocationList.push_back (std::make_pair ("", lPTAltLocation));
00073         }
00074
00075         // Add the extra location list to the Boost.Property_Tree representing
00076         // the current Location instance.
00077         lPTLocation.add_child ("alternates", lPTAltLocationList);
00078       }
00079
00080       // Add the current location property tree to the location list
00081       lPTLocationList.push_back (std::make_pair ("", lPTLocation));
00082     }
00083
00084     // Add the location list to the root of the Boost.Property_Tree
```

```
00085     lPT.add_child ("locations", lPTLocationList);
00086
00087     // Write the property tree into a JSON string
00088     write_json (oStream, lPT);
00089   }
00090
00091   // ////////////////////////////////////////////////////////////////////
00092   void BomJSONExport::jsonExportLocation (bpt::ptree& ioPTLocation,
00093                                           const Location& iLocation) {
00094     // Fill all the fields of the JSON instance
00095     ioPTLocation.put ("iata_code", iLocation.getIataCode());
00096     ioPTLocation.put ("icao_code", iLocation.getIcaoCode());
00097     ioPTLocation.put ("geonames_id", iLocation.getGeonamesID());
00098     ioPTLocation.put ("faa_code", iLocation.getFaaCode());
00099     ioPTLocation.put ("city_code", iLocation.getCityCode());
00100     ioPTLocation.put ("city_name_utf", iLocation.getCityUtfName());
00101     ioPTLocation.put ("city_name_ascii", iLocation.getCityAsciiName());
00102     ioPTLocation.put ("state_code", iLocation.getStateCode());
00103     ioPTLocation.put ("country_code", iLocation.getCountryCode());
00104     ioPTLocation.put ("country_name", iLocation.getCountryName());
00105     ioPTLocation.put ("continent_name", iLocation.getContinentName());
00106     ioPTLocation.put ("adm1_code", iLocation.getAdmin1Code());
00107     ioPTLocation.put ("adm1_name_utf", iLocation.getAdmin1UtfName());
00108     ioPTLocation.put ("adm1_name_ascii", iLocation.getAdmin1AsciiName());
00109     ioPTLocation.put ("adm2_code", iLocation.getAdmin2Code());
00110     ioPTLocation.put ("adm2_name_utf", iLocation.getAdmin2UtfName());
00111     ioPTLocation.put ("adm2_name_ascii", iLocation.getAdmin2AsciiName());
00112     ioPTLocation.put ("adm3_code", iLocation.getAdmin3Code());
00113     ioPTLocation.put ("adm4_code", iLocation.getAdmin4Code());
00114     ioPTLocation.put ("tvl_por_list", iLocation.getTvlPORListString());
00115     ioPTLocation.put ("time_zone", iLocation.getTimeZone());
00116     ioPTLocation.put ("lat", iLocation.getLatitude());
00117     ioPTLocation.put ("lon", iLocation.getLongitude());
00118     ioPTLocation.put ("page_rank", iLocation.getPageRank());
00119     ioPTLocation.put ("wiki_link", iLocation.getWikiLink());
00120     ioPTLocation.put ("original_keywords", iLocation.getOriginalKeywords());
00121     ioPTLocation.put ("corrected_keywords", iLocation.getCorrectedKeywords());
00122     ioPTLocation.put ("matching_percentage", iLocation.getPercentage());
00123     ioPTLocation.put ("edit_distance", iLocation.getEditDistance());
00124     ioPTLocation.put ("allowable_distance", iLocation.getAllowableEditDistance());
00125
00126     bpt::ptree ptLocationNameList;
00127     // Retrieve the place names in all the available languages
00128     const NameMatrix& lNameMatrixFull = iLocation.getNameMatrix();
00129     const NameMatrix_T& lNameMatrix = lNameMatrixFull.getNameMatrix();
00130     for (NameMatrix_T::const_iterator itNameList = lNameMatrix.begin();
00131          itNameList != lNameMatrix.end(); ++itNameList) {
00132       // Retrieve the language code and locale
00133       // const Language::EN_Language& lLanguage = itNameList->first;
00134       const Names& lNames = itNameList->second;
00135
00136       // For a given language, retrieve the list of place names
00137       const NameList_T& lNameList = lNames.getNameList();
00138
00139       for (NameList_T::const_iterator itName = lNameList.begin();
00140            itName != lNameList.end(); ++itName) {
00141         const std::string& lName = *itName;
00142
00143         if (lName.empty() == false) {
00144           bpt::ptree ptLocationName;
00145           ptLocationName.put ("name", lName);
00146           ptLocationNameList.push_back (std::make_pair ("", ptLocationName));
00147         }
00148       }
00149     }
00150
00151     // Add the name list to the Boost.Property_Tree representing the location
00152     ioPTLocation.add_child ("names", ptLocationNameList);
00153   }
00154
00155 }
```

24.83 opentrep/bom/BomJSONExport.hpp File Reference

```
#include <iosfwd>
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#include <opentrep/LocationList.hpp>
```

Classes

- class OPENTREP::BomJSONExport

  Utility class to export Opentrep structures in a JSON format.

Namespaces

- OPENTREP

24.84 BomJSONExport.hpp

```
00001 #ifndef __OPENTREP_BOM_BOMJSONEXPORT_HPP
00002 #define __OPENTREP_BOM_BOMJSONEXPORT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // Boost Property Tree (PT)
00010 #include <boost/property_tree/ptree.hpp>
00011 #include <boost/property_tree/json_parser.hpp>
00012 // OpenTrep
00013 #include <opentrep/LocationList.hpp>
00014
00015 namespace bpt = boost::property_tree;
00016
00017 namespace OPENTREP {
00018
00019   // Forward declarations
00020   struct Location;
00021
00025   class BomJSONExport {
00026   public:
00027     // ///////////////// Export support methods /////////////////
00028
00037     static void jsonExportLocationList (std::ostream&, const
      LocationList_T&);
00038
00047     static void jsonExportLocation (bpt::ptree&, const
      Location&);
00048   };
00049
00050 }
00051 #endif // __OPENTREP_BOM_BOMJSONEXPORT_HPP
```

24.85 opentrep/bom/BomType.cpp File Reference

```
#include <assert.h>
#include <opentrep/bom/BomType.hpp>
```

Namespaces

- OPENTREP

24.86 BomType.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // C
00005 #include <assert.h>
00006 // OpenTrep
00007 #include <opentrep/bom/BomType.hpp>
00008
00009 namespace OPENTREP {
00010
00011   // //////////////////////////////////////////////////////////////////
```

```
00012   const std::string BomType::_labels[LAST_VALUE] = {
00013     "World", "Place", "Airline", "Country", "Day", "Month", "Passenger"};
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   const std::string& BomType::getLabel (const
      BomType::EN_BomType& iBomType) {
00017     return BomType::_labels[iBomType];
00018   }
00019
00020 }
```

24.87 opentrep/bom/BomType.hpp File Reference

```
#include <string>
```

Classes

- struct OPENTREP::BomType

Namespaces

- OPENTREP

24.88 BomType.hpp

```
00001 #ifndef __OPENTREP_BOM_BOMTYPE_HPP
00002 #define __OPENTREP_BOM_BOMTYPE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace OPENTREP {
00011
00013   struct BomType {
00014   public:
00015     typedef enum {
00016       WORLD = 0,
00017       PLACE,
00018       AIRLINE,
00019       COUNTRY,
00020       DAY,
00021       MONTH,
00022       PASSENGER,
00023       LAST_VALUE
00024     } EN_BomType;
00025
00027     static const std::string& getLabel (const EN_BomType&);
00028
00029   private:
00031     static const std::string _labels[LAST_VALUE];
00032   };
00033
00034 }
00035 #endif // __OPENTREP_BOM_BOMTYPE_HPP
```

24.89 opentrep/bom/Country.cpp File Reference

24.90 Country.cpp

24.91 opentrep/bom/Country.hpp File Reference

24.92 Country.hpp

24.93 opentrep/bom/Day.cpp File Reference

24.94 Day.cpp

24.95 opentrep/bom/Day.hpp File Reference

24.96 Day.hpp

24.97 opentrep/bom/DocumentList.hpp File Reference

```
#include <list>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/bom/MatchingDocuments.hpp>
#include <xapian.h>
```

Namespaces

- OPENTREP

Typedefs

- typedef std::list
  < MatchingDocuments > OPENTREP::DocumentList_T

24.98 DocumentList.hpp

```
00001 #ifndef __OPENTREP_BOM_DOCUMENTLIST_HPP
00002 #define __OPENTREP_BOM_DOCUMENTLIST_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <list>
00009 // OpenTREP
00010 #include <opentrep/OPENTREP_Types.hpp>
00011 #include <opentrep/bom/MatchingDocuments.hpp>
00012 // Xapian
00013 #include <xapian.h>
00014
00015 namespace OPENTREP {
00016
00020   typedef std::list<MatchingDocuments> DocumentList_T;
00021
00022 }
00023 #endif // __OPENTREP_BOM_DOCUMENTLIST_HPP
```

24.99 opentrep/bom/Filter.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <opentrep/basic/BasConst_General.hpp>
#include <opentrep/basic/Utilities.hpp>
#include <opentrep/bom/Filter.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

Functions

- bool OPENTREP::hasGoodSize (const std::string &iWord, const NbOfLetters_T &iMinWordLength)
- bool OPENTREP::isBlackListed (const std::string &iWord)
- void OPENTREP::rtrim (WordList_T &ioWordList, const NbOfLetters_T &iMinWordLength)
- void OPENTREP::ltrim (WordList_T &ioWordList, const NbOfLetters_T &iMinWordLength)
- void OPENTREP::trim (WordList_T &ioWordList, const NbOfLetters_T &iMinWordLength)

24.100 Filter.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // OpenTrep
00008 #include <opentrep/basic/BasConst_General.hpp>
00009 #include <opentrep/basic/Utilities.hpp>
00010 #include <opentrep/bom/Filter.hpp>
00011 #include <opentrep/service/Logger.hpp>
00012
00013 namespace OPENTREP {
00014
00015   // //////////////////////////////////////////////////////////////////
00016   Filter::Filter() {
00017     assert (false);
00018   }
00019
00020   // //////////////////////////////////////////////////////////////////
00021   Filter::Filter (const Filter& iFilter) {
00022     assert (false);
00023   }
00024
00025   // //////////////////////////////////////////////////////////////////
00026   Filter::~Filter() {
00027   }
00028
00029
00038   // //////////////////////////////////////////////////////////////////
00039   bool hasGoodSize (const std::string& iWord, const NbOfLetters_T& iMinWordLength)
      {
00040     bool hasGoodSizeFlag = true;
00041     //
00042     const size_t lWordLength = iWord.size();
00043     if (lWordLength < iMinWordLength) {
00044       hasGoodSizeFlag = false;
00045     }
00046     return hasGoodSizeFlag;
00047   }
00048
00052   // //////////////////////////////////////////////////////////////////
00053   bool isBlackListed (const std::string& iWord) {
00054     // When the word is part of the "black list", it should obviously be
00055     // filtered out.
00056     BlackList_T::const_iterator itWord = K_BLACK_LIST.find (iWord);
00057     const bool isBlackListedFlag = (itWord != K_BLACK_LIST.end());
00058
```

```
00059     // DEBUG
00060     // const std::string areEqualStr = (isBlackListedFlag)?"Yes":"No";
00061     // const std::string& lWord = *itWord;
00062     // OPENTREP_LOG_DEBUG ("Word: '" << iWord << "', black-list word: '"
00063     //                     << lWord << "', Equals: " << areEqualStr);
00064
00065     return isBlackListedFlag;
00066   }
00067
00071   // ////////////////////////////////////////////////////////////////////
00072   void rtrim (WordList_T& ioWordList, const NbOfLetters_T& iMinWordLength) {
00073     // If the list is empty, obviously nothing can be done at that stage.
00074     if (ioWordList.empty() == true) {
00075       return;
00076     }
00077
00078     // Take the first right outer word
00079     WordList_T::reverse_iterator itWord = ioWordList.rbegin();
00080     assert (itWord != ioWordList.rend());
00081     const std::string& lWord = *itWord;
00082
00083     // Check whether that word has the good size (>= iMinWordLength) and whether it is
00084     // black-listed.
00085     const bool hasGoodSizeFlag = hasGoodSize (lWord, iMinWordLength);
00086     const bool isBlackListedFlag = isBlackListed (lWord);
00087     if (hasGoodSizeFlag == false || isBlackListedFlag == true) {
00088       ioWordList.erase (--itWord.base());
00089       rtrim (ioWordList, iMinWordLength);
00090     }
00091   }
00092
00096   // ////////////////////////////////////////////////////////////////////
00097   void ltrim (WordList_T& ioWordList, const NbOfLetters_T& iMinWordLength) {
00098     // If the list is empty, obviously nothing can be done at that stage.
00099     if (ioWordList.empty() == true) {
00100       return;
00101     }
00102
00103     // Take the first left outer word
00104     WordList_T::iterator itWord = ioWordList.begin();
00105     assert (itWord != ioWordList.end());
00106     const std::string& lWord = *itWord;
00107
00108     // Check whether that word has the good size (>= iMinWordLength) and whether it is
00109     // black-listed.
00110     const bool hasGoodSizeFlag = hasGoodSize (lWord, iMinWordLength);
00111     const bool isBlackListedFlag = isBlackListed (lWord);
00112     if (hasGoodSizeFlag == false || isBlackListedFlag == true) {
00113       ioWordList.erase (itWord);
00114       ltrim (ioWordList, iMinWordLength);
00115     }
00116   }
00117
00121   // ////////////////////////////////////////////////////////////////////
00122   void trim (WordList_T& ioWordList, const NbOfLetters_T& iMinWordLength) {
00123     // Trim the non-relevant left outer words
00124     ltrim (ioWordList, iMinWordLength);
00125
00126     // Trim the non-relevant right outer words
00127     rtrim (ioWordList, iMinWordLength);
00128   }
00129
00130   // ////////////////////////////////////////////////////////////////////
00131   void Filter::trim (std::string& ioPhrase, const NbOfLetters_T& iMinWordLength) {
00132     // Create a list of words from the given phrase
00133     WordList_T lWordList;
00134     tokeniseStringIntoWordList (ioPhrase, lWordList);
00135
00136     // Trim the non-relevant left and right outer words
00137     OPENTREP::trim (lWordList, iMinWordLength);
00138
00139     // Re-create the phrase from the (potentially altered) list of words
00140     ioPhrase = createStringFromWordList (lWordList);
00141   }
00142
00143   // ////////////////////////////////////////////////////////////////////
00144   bool Filter::shouldKeep (const std::string& iPhrase,
00145                            const std::string& iWord) {
00146     bool isToBeKept = true;
00147
00148     // If both the phrase and the word are empty, the word should obviously
00149     // be filtered out.
00150     if (iPhrase.empty() == true && iWord.empty() == true) {
00151       isToBeKept = false;
00152       return isToBeKept;
00153     }
00154
```

```
00155     // If the term to be added is equal to the whole phrase (e.g., 'san'),
00156     // it should be kept (not filtered out). Indeed, three-letter words
00157     // often correspond to IATA codes, and should obviously be kept for
00158     // indexation/searching.
00159     if (iPhrase == iWord) {
00160       return isToBeKept;
00161     }
00162
00163     // Now, the word is part of the phrase, and not equal to it (and not empty).
00164
00165     // If the word has no more than two letters (e.g., 'de'), it should be
00166     // filtered out. Indeed, when 'de' is part of 'charles de gaulle',
00167     // for instance, it should not be indexed/searched alone (in a search,
00168     // the resulting match score will be zero).
00169     isToBeKept = hasGoodSize (iWord, 3);
00170     if (isToBeKept == false) {
00171       return isToBeKept;
00172     }
00173
00174     // Check whether the word is black-listed
00175     isToBeKept = !isBlackListed (iWord);
00176
00177     //
00178     return isToBeKept;
00179   }
00180
00181 }
```

24.101 opentrep/bom/Filter.hpp File Reference

```
#include <opentrep/OPENTREP_Types.hpp>
```

Classes

- struct OPENTREP::Filter

  Class filtering out the words not suitable for indexing and/or searching, when part of greater strings. Hence, most of the methods take as parameter the "initial"/greater string.

Namespaces

- OPENTREP

24.102 Filter.hpp

```
00001 #ifndef __OPENTREP_BOM_FILTER_HPP
00002 #define __OPENTREP_BOM_FILTER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/OPENTREP_Types.hpp>
00009
00010 namespace OPENTREP {
00011
00021   struct Filter {
00022   public:
00023     // ////////// Business support methods /////////
00042     static void trim (std::string& ioPhrase, const NbOfLetters_T& iMinWordLength = 4);
00043
00064     static bool shouldKeep (const std::string& iPhrase,
00065                             const std::string& iWord);
00066
00067
00068   private:
00069     // ///////////////// Constructors and Destructors /////////////
00073     Filter();
00074
00078     Filter (const Filter&);
00079
00083     ~Filter();
00084   };
```

```
00085
00086 }
00087 #endif // __OPENTREP_BOM_FILTER_HPP
```

24.103 opentrep/bom/GenericBom.hpp File Reference

```
#include <map>
#include <opentrep/bom/BomType.hpp>
```

Namespaces

- OPENTREP

Typedefs

- typedef std::pair
  < BomType::EN_BomType,
  BomAbstract * > OPENTREP::GenericBom_T
- typedef std::map< std::size_t,
  GenericBom_T > OPENTREP::GenericBomList_T

24.104 GenericBom.hpp

```
00001 #ifndef __OPENTREP_BOM_GENERICBOM_HPP
00002 #define __OPENTREP_BOM_GENERICBOM_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <map>
00009 // OpenTrep
00010 #include <opentrep/bom/BomType.hpp>
00011
00012 namespace OPENTREP {
00013
00014   // Forward declarations
00015   class BomAbstract;
00016
00019   typedef std::pair<BomType::EN_BomType, BomAbstract*> GenericBom_T;
00020
00022   typedef std::map<std::size_t, GenericBom_T> GenericBomList_T;
00023
00024 }
00025 #endif // __OPENTREP_BOM_GENERICBOM_HPP
```

24.105 opentrep/bom/IATAType.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <opentrep/Travel.pb.h>
#include <opentrep/IATAType.hpp>
```

Namespaces

- OPENTREP

24.106 IATAType.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // OpenTrep Protobuf
00008 #include <opentrep/Travel.pb.h>
00009 // OpenTREP
00010 #include <opentrep/IATAType.hpp>
00011
00012 namespace OPENTREP {
00013
00014   // //////////////////////////////////////////////////////////////////////
00015   const std::string IATAType::_labels[LAST_VALUE] =
00016     { "CTY_AIRP", "CTY_HPT", "CTY_RSTN", "CTY_BSTN", "CTY_FERRY",
00017       "CITY", "AIRP", "HPT", "RSTN", "BSTN", "FERRY", "OFF" };
00018
00019   // //////////////////////////////////////////////////////////////////////
00020   const char IATAType::_typeLabels[LAST_VALUE] =
00021     { 'a', 'h', 'r', 'b', 'p', 'C', 'A', 'H', 'R', 'B', 'P', 'O' };
00022
00023   // //////////////////////////////////////////////////////////////////////
00024   IATAType::IATAType() : _type (LAST_VALUE) {
00025     assert (false);
00026   }
00027
00028   // //////////////////////////////////////////////////////////////////////
00029   IATAType::IATAType (const IATAType& iIATAType)
00030     : _type (iIATAType._type) {
00031   }
00032
00033   // //////////////////////////////////////////////////////////////////////
00034   IATAType::IATAType (const EN_IATAType& iIATAType)
00035     : _type (iIATAType) {
00036   }
00037
00038   // //////////////////////////////////////////////////////////////////////
00039   IATAType::EN_IATAType IATAType::getType (const char iTypeChar) {
00040     EN_IATAType oType;
00041     switch (iTypeChar) {
00042     case 'a': oType = CTY_AIRP; break;
00043     case 'h': oType = CTY_HPT; break;
00044     case 'r': oType = CTY_RSTN; break;
00045     case 'b': oType = CTY_BSTN; break;
00046     case 'p': oType = CTY_FERRY; break;
00047     case 'C': oType = CITY; break;
00048     case 'A': oType = AIRP; break;
00049     case 'H': oType = HPT; break;
00050     case 'R': oType = RSTN; break;
00051     case 'B': oType = BSTN; break;
00052     case 'P': oType = FERRY; break;
00053     case 'O': oType = OFF; break;
00054     default: oType = LAST_VALUE; break;
00055     }
00056
00057     if (oType == LAST_VALUE) {
00058       const std::string& lLabels = describeLabels();
00059       std::ostringstream oMessage;
00060       oMessage << "The output type '" << iTypeChar
00061                << "' is not known. Known output types: " << lLabels;
00062       throw CodeConversionException (oMessage.str());
00063     }
00064
00065     return oType;
00066   }
00067
00068   // //////////////////////////////////////////////////////////////////////
00069   IATAType::IATAType (const char iTypeChar)
00070     : _type (getType (iTypeChar)) {
00071   }
00072
00073   // //////////////////////////////////////////////////////////////////////
00074   IATAType::IATAType (const std::string& iTypeStr) : _type (LAST_VALUE) {
00075     if (iTypeStr == "CA") {
00076       _type = CTY_AIRP;
00077     } else if (iTypeStr == "CH") {
00078       _type = CTY_HPT;
00079     } else if (iTypeStr == "CR") {
00080       _type = CTY_RSTN;
00081     } else if (iTypeStr == "CB") {
00082       _type = CTY_BSTN;
00083     } else if (iTypeStr == "CP") {
00084       _type = CTY_FERRY;
00085     } else if (iTypeStr == "C") {
```

```
00086       _type = CITY;
00087     } else if (iTypeStr == "A") {
00088       _type = AIRP;
00089     } else if (iTypeStr == "H") {
00090       _type = HPT;
00091     } else if (iTypeStr == "R") {
00092       _type = RSTN;
00093     } else if (iTypeStr == "B") {
00094       _type = BSTN;
00095     } else if (iTypeStr == "P") {
00096       _type = FERRY;
00097     } else if (iTypeStr == "O") {
00098       _type = OFF;
00099     } else {
00100       _type = LAST_VALUE;
00101     }
00102
00103     if (_type == LAST_VALUE) {
00104       const std::string& lLabels = describeLabels();
00105       std::ostringstream oMessage;
00106       oMessage << "The output type '" << iTypeStr
00107                << "' is not known. Known output types: " << lLabels;
00108       throw CodeConversionException (oMessage.str());
00109     }
00110   }
00111
00112   // ////////////////////////////////////////////////////////////////////
00113   const std::string& IATAType::getLabel (const EN_IATAType& iType) {
00114     return _labels[iType];
00115   }
00116
00117   // ////////////////////////////////////////////////////////////////////
00118   char IATAType::getTypeLabel (const EN_IATAType& iType) {
00119     return _typeLabels[iType];
00120   }
00121
00122   // ////////////////////////////////////////////////////////////////////
00123   std::string IATAType::getTypeLabelAsString (const
      EN_IATAType& iType) {
00124     std::ostringstream oStr;
00125     oStr << _typeLabels[iType];
00126     return oStr.str();
00127   }
00128
00129   // ////////////////////////////////////////////////////////////////////
00130   treppb::PlaceType IATAType::getTypeLabelAsPB (const
      EN_IATAType& iType) {
00131     treppb::PlaceType oLocationType;
00132     switch (iType) {
00133     case CTY_AIRP: oLocationType.set_type (treppb::PlaceType::CTY_AIRP); break;
00134     case CTY_HPT: oLocationType.set_type (treppb::PlaceType::CTY_HPT); break;
00135     case CTY_RSTN: oLocationType.set_type (treppb::PlaceType::CTY_RSTN); break;
00136     case CTY_BSTN: oLocationType.set_type (treppb::PlaceType::CTY_BSTN); break;
00137     case CTY_FERRY: oLocationType.set_type (treppb::PlaceType::CTY_FERRY); break;
00138     case CITY: oLocationType.set_type (treppb::PlaceType::CITY); break;
00139     case AIRP: oLocationType.set_type (treppb::PlaceType::AIRP); break;
00140     case HPT: oLocationType.set_type (treppb::PlaceType::HPT); break;
00141     case RSTN: oLocationType.set_type (treppb::PlaceType::RSTN); break;
00142     case BSTN: oLocationType.set_type (treppb::PlaceType::BSTN); break;
00143     case FERRY: oLocationType.set_type (treppb::PlaceType::FERRY); break;
00144     case OFF: oLocationType.set_type (treppb::PlaceType::OFF); break;
00145     default: oLocationType.set_type (treppb::PlaceType::UNKNOWN); break;
00146     }
00147     return oLocationType;
00148   }
00149
00150   // ////////////////////////////////////////////////////////////////////
00151   std::string IATAType::describeLabels() {
00152     std::ostringstream ostr;
00153     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00154       if (idx != 0) {
00155         ostr << ", ";
00156       }
00157       ostr << _labels[idx];
00158     }
00159     return ostr.str();
00160   }
00161
00162   // ////////////////////////////////////////////////////////////////////
00163   IATAType::EN_IATAType IATAType::getType() const {
00164     return _type;
00165   }
00166
00167   // ////////////////////////////////////////////////////////////////////
00168   char IATAType::getTypeAsChar() const {
00169     const char oTypeChar = _typeLabels[_type];
00170     return oTypeChar;
```

```
00171   }
00172
00173   // ////////////////////////////////////////////////////////////////////
00174   std::string IATAType::getTypeAsString() const {
00175     std::ostringstream oStr;
00176     oStr << _typeLabels[_type];
00177     return oStr.str();
00178   }
00179
00180   // ////////////////////////////////////////////////////////////////////
00181   treppb::PlaceType IATAType::getTypeAsPB() const {
00182     return getTypeLabelAsPB (_type);
00183   }
00184
00185   // ////////////////////////////////////////////////////////////////////
00186   const std::string IATAType::describe() const {
00187     std::ostringstream ostr;
00188     ostr << _labels[_type];
00189     return ostr.str();
00190   }
00191
00192   // ////////////////////////////////////////////////////////////////////
00193   bool IATAType::operator== (const EN_IATAType& iType) const {
00194     return (_type == iType);
00195   }
00196
00197   // ////////////////////////////////////////////////////////////////////
00198   bool IATAType::operator== (const IATAType& iIATAType) const {
00199     return (_type == iIATAType._type);
00200   }
00201
00202 }
```

24.107 opentrep/bom/KeyAbstract.hpp File Reference

```
#include <iosfwd>
#include <sstream>
#include <string>
```

Classes

- class OPENTREP::KeyAbstract

  Base class for the Business Object Model (KEY) layer.

Namespaces

- OPENTREP

Functions

- template<class charT , class traits >
  std::basic_ostream< charT,
  traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const OPENTREP::KeyAbstract &iKey)
- template<class charT , class traits >
  std::basic_istream< charT,
  traits > & operator>> (std::basic_istream< charT, traits > &ioIn, OPENTREP::KeyAbstract &ioKey)

24.107.1 Function Documentation

24.107.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut,* const OPENTREP::KeyAbstract & *iKey*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 68 of file KeyAbstract.hpp.

24.107.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, OPENTREP::KeyAbstract & *ioKey*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 96 of file KeyAbstract.hpp.

References OPENTREP::KeyAbstract::fromStream().

24.108 KeyAbstract.hpp

```
00001 #ifndef __OPENTREP_KEY_KEYABSTRACT_HPP
00002 #define __OPENTREP_KEY_KEYABSTRACT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <sstream>
00010 #include <string>
00011
00012 namespace OPENTREP {
00013
00017   class KeyAbstract {
00018   public:
00019     // ////////// Display support methods /////////
00025     virtual void toStream (std::ostream&) const = 0;
00026
00032     virtual void fromStream (std::istream&) = 0;
00033
00037     virtual std::string toString() const = 0;
00038
00043     virtual std::string describe() const = 0;
00044
00045
00046   protected:
00050     KeyAbstract() {}
00051     KeyAbstract (const KeyAbstract&) {}
00052
00056     virtual ~KeyAbstract() {}
00057   };
00058 }
00059
00065 template <class charT, class traits>
00066 inline
00067 std::basic_ostream<charT, traits>&
00068 operator<< (std::basic_ostream<charT, traits>& ioOut,
00069             const OPENTREP::KeyAbstract& iKey) {
00075   std::basic_ostringstream<charT,traits> ostr;
00076   ostr.copyfmt (ioOut);
00077   ostr.width (0);
00078
00079   // Fill string stream
00080   iKey.toStream (ostr);
00081
00082   // Print string stream
00083   ioOut << ostr.str();
00084
00085   return ioOut;
00086 }
00087
00093 template <class charT, class traits>
00094 inline
00095 std::basic_istream<charT, traits>&
00096 operator>> (std::basic_istream<charT, traits>& ioIn,
00097             OPENTREP::KeyAbstract& ioKey) {
00098   // Fill Key object with input stream
00099   ioKey.fromStream (ioIn);
00100   return ioIn;
00101 }
00102
00103 #endif // __OPENTREP_KEY_KEYABSTRACT_HPP
```

24.109 opentrep/bom/Language.cpp File Reference

```
#include <cassert>
#include <opentrep/Language.hpp>
```

Namespaces

- OPENTREP

24.110 Language.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // OpenTrep
00007 #include <opentrep/Language.hpp>
00008
00009 namespace OPENTREP {
00010
00011   // //////////////////////////////////////////////////////////////////
00012   const std::string Language::_longLabels[LAST_VALUE+1] = {
00013     "common",
00014     "ar", "ar_DZ", "ar_MA", "ar_SA", "ar_TN",
00015     "de", "de_DE", "en", "en_GB", "en_US", "es", "es_AR", "es_ES", "es_MX",
00016     "fa", "fa_IR",
00017     "fr", "fr_BE", "fr_CA", "fr_CH", "fr_FR",
00018     "he", "he_IL", "ko", "ko_KR", "pl", "pl_PL", "pt", "pt_BR", "pt_PT",
00019     "sv", "sv_SE",
00020     "ru", "ru_RU", "uk", "uk_UA", "zh", "zh_CN", "zh_HK", "zh_TW",
00021     "out-of-index"};
00022
00023   // //////////////////////////////////////////////////////////////////
00024   const std::string Language::_shortLabels[LAST_VALUE+1] = {
00025     "",
00026     "ar", "ar", "ar", "ar", "ar",
00027     "de", "de", "en", "en", "en", "es", "es", "es", "es",
00028     "fa", "fa",
00029     "fr", "fr", "fr", "fr", "fr",
00030     "he", "he", "ko", "ko", "pl", "pl", "pt", "pt", "pt",
00031     "sv", "sv",
00032     "ru", "ru", "uk", "uk", "zh", "zh", "zh", "zh",
00033     "out-of-index"};
00034
00035   // //////////////////////////////////////////////////////////////////
00036   const std::string& Language::
00037   getLongLabel (const Language::EN_Language& iLanguage) {
00038     return Language::_longLabels[iLanguage];
00039   }
00040
00041   // //////////////////////////////////////////////////////////////////
00042   const std::string& Language::
00043   getShortLabel (const Language::EN_Language& iLanguage) {
00044     return Language::_shortLabels[iLanguage];
00045   }
00046
00047   // //////////////////////////////////////////////////////////////////
00048   Language::EN_Language Language::getCode (const std::string&
      iLanguageString) {
00049     EN_Language oLanguageCode = LAST_VALUE;
00050
00051     // An empty language code means that it is common to all the languages.
00052     if (iLanguageString.empty() == true) {
00053       EN_Language oLanguageCode = lg_com;
00054       return oLanguageCode;
00055     }
00056
00057     for (unsigned int itLanguageCode = 1; itLanguageCode != LAST_VALUE;
00058          ++itLanguageCode) {
00059       const EN_Language lLanguageCode= static_cast<EN_Language> (itLanguageCode);
00060
00061       if (getShortLabel (lLanguageCode) == iLanguageString) {
00062         oLanguageCode = lLanguageCode;
00063         break;
00064       }
00065     }
00066
```

```
00067     return oLanguageCode;
00068   }
00069
00070 }
```

24.111 opentrep/bom/Levenshtein.cpp File Reference

```
#include <string>
#include <vector>
#include <opentrep/bom/Levenshtein.hpp>
```

Namespaces

- OPENTREP

24.112 Levenshtein.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <string>
00006 #include <vector>
00007 // OpenTREP
00008 #include <opentrep/bom/Levenshtein.hpp>
00009
00010 namespace OPENTREP {
00011
00012   // ////////////////////////////////////////////////////////////////////
00013   int Levenshtein::getDistance (const std::string& iSource,
00014                                 const std::string& iTarget) {
00015
00016     // Step 1
00017
00018     const int n = iSource.length();
00019     const int m = iTarget.length();
00020
00021     if (n == 0) {
00022       return m;
00023     }
00024
00025     if (m == 0) {
00026       return n;
00027     }
00028
00029     // Definition of Matrix Type
00030     typedef std::vector<std::vector<int> > Matrix_T;
00031
00032     Matrix_T matrix (n+1);
00033
00034     // Size the vectors in the 2.nd dimension. Unfortunately C++ doesn't
00035     // allow for allocation on declaration of 2.nd dimension of vec of vec
00036
00037     for (int i = 0; i <= n; i++) {
00038       matrix[i].resize(m+1);
00039     }
00040
00041     // Step 2
00042
00043     for (int i = 0; i <= n; i++) {
00044       matrix[i][0]=i;
00045     }
00046
00047     for (int j = 0; j <= m; j++) {
00048       matrix[0][j]=j;
00049     }
00050
00051     // Step 3
00052
00053     for (int i = 1; i <= n; i++) {
00054
00055       const char s_i = iSource[i-1];
00056
00057       // Step 4
00058
```

```
00059       for (int j = 1; j <= m; j++) {
00060
00061         const char t_j = iTarget[j-1];
00062
00063         // Step 5
00064
00065         int cost;
00066         if (s_i == t_j) {
00067           cost = 0;
00068
00069         } else {
00070           cost = 1;
00071         }
00072
00073         // Step 6
00074
00075         const int above = matrix[i-1][j];
00076         const int left = matrix[i][j-1];
00077         const int diag = matrix[i-1][j-1];
00078         int cell = std::min ( above + 1, std::min (left + 1, diag + cost));
00079
00080         // Step 6A: Cover transposition, in addition to deletion,
00081         // insertion and substitution. This step is taken from:
00082         // Berghel, Hal ; Roach, David : "An Extension of Ukkonen's
00083         // Enhanced Dynamic Programming ASM Algorithm"
00084         // (http://www.acm.org/~hlb/publications/asm/asm.html)
00085
00086         if (i>2 && j>2) {
00087           int trans = matrix[i-2][j-2] + 1;
00088
00089           if (iSource[i-2] != t_j) {
00090             trans++;
00091           }
00092
00093           if (s_i != iTarget[j-2]) {
00094             trans++;
00095           }
00096
00097           if (cell > trans) {
00098             cell = trans;
00099           }
00100         }
00101
00102         matrix[i][j] = cell;
00103       }
00104     }
00105
00106     // Step 7
00107
00108     return matrix[n][m];
00109   }
00110
00111 }
```

24.113 opentrep/bom/Levenshtein.hpp File Reference

```
#include <string>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/bom/BomAbstract.hpp>
```

Classes

- class OPENTREP::Levenshtein

Namespaces

- OPENTREP

24.114 Levenshtein.hpp

```
00001 //
00002 // Levenshtein Distance Algorithm: C++ Implementation by Anders Sewerin Johansen
```

```
00003 //
00004 #ifndef __OPENTREP_BOM_LEVENSHTEIN_HPP
00005 #define __OPENTREP_BOM_LEVENSHTEIN_HPP
00006
00007 // //////////////////////////////////////////////////////////////////
00008 // Import section
00009 // //////////////////////////////////////////////////////////////////
00010 // STL
00011 #include <string>
00012 // OpenTREP
00013 #include <opentrep/OPENTREP_Types.hpp>
00014 #include <opentrep/bom/BomAbstract.hpp>
00015
00016 namespace OPENTREP {
00017
00020   class Levenshtein : public BomAbstract {
00021   public:
00023     static int getDistance (const std::string& iSource,
00024                             const std::string& iTarget);
00025   };
00026
00027 }
00028 #endif // __OPENTREP_BOM_LEVENSHTEIN_HPP
```

24.115 opentrep/bom/LocationExchange.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <string>
#include <opentrep/Travel.pb.h>
#include <opentrep/Location.hpp>
#include <opentrep/bom/LocationExchange.hpp>
```

Namespaces

- OPENTREP

24.116 LocationExchange.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STL
00007 #include <ostream>
00008 #include <string>
00009 // OpenTrep Protobuf
00010 #include <opentrep/Travel.pb.h>
00011 // OpenTrep
00012 #include <opentrep/Location.hpp>
00013 #include <opentrep/bom/LocationExchange.hpp>
00014
00015 namespace OPENTREP {
00016
00017   // //////////////////////////////////////////////////////////////////
00018   void LocationExchange::
00019   exportLocationList (std::ostream& oStream,
00020                       const LocationList_T& iLocationList,
00021                       const WordList_T& iNonMatchedWordList) {
00022     // Protobuf structure
00023     treppb::QueryAnswer oQueryAnswer;
00024
00025     // //// 1. Status ////
00026     const bool kOKStatus = true;
00027     oQueryAnswer.set_ok_status (kOKStatus);
00028
00029     // //// 2. Error message ////
00037     // //// 3. List of places ////
00038     treppb::PlaceList* lPlaceListPtr = oQueryAnswer.mutable_place_list();
00039     assert (lPlaceListPtr != NULL);
00040
00041     // Browse the list of Location structures, and fill the Protobuf structure
00042     for (LocationList_T::const_iterator itLocation = iLocationList.begin();
```

```
00043         itLocation != iLocationList.end(); ++itLocation) {
00044       const Location& lLocation = *itLocation;
00045
00046       // Create an instance of a Protobuf Place structure
00047       treppb::Place* lPlacePtr = lPlaceListPtr->add_place();
00048       assert (lPlacePtr != NULL);
00049
00050       // Fill the Protobuf Place structure with the content of
00051       // the Location structure
00052       exportLocation (*lPlacePtr, lLocation);
00053     }
00054
00055     // //// 4. List of un-matched keywords ////
00056     // Create an instance of a Protobuf UnknownKeywordList structure
00057     treppb::UnknownKeywordList* lUnknownKeywordListPtr =
00058       oQueryAnswer.mutable_unmatched_keyword_list();
00059       assert (lUnknownKeywordListPtr != NULL);
00060
00061     // Browse the list of un-matched keywords, and fill the Protobuf structure
00062     for (WordList_T::const_iterator itWord = iNonMatchedWordList.begin();
00063          itWord != iNonMatchedWordList.end(); ++itWord) {
00064       const Word_T& lWord = *itWord;
00065       lUnknownKeywordListPtr->add_word (lWord);
00066     }
00067
00068     // Serialise the Protobuf
00069     oQueryAnswer.SerializeToOstream (&oStream);
00070   }
00071
00072   // ////////////////////////////////////////////////////////////////////
00073   void LocationExchange::exportLocation (treppb::Place& ioPlace,
00074                                          const Location& iLocation) {
00075     // Retrieve the primary key
00076     const LocationKey& lLocationKey = iLocation.getKey();
00077
00078     // Retrieve and set the travel-related IATA code (part of the primary key)
00079     const IATACode_T& lIataCode = lLocationKey.getIataCode();
00080     treppb::IATACode* lIataAirportPtr = ioPlace.mutable_tvl_code();
00081     assert (lIataAirportPtr != NULL);
00082     lIataAirportPtr->set_code (lIataCode);
00083
00084     // Retrieve and set the location type
00085     const IATAType& lLocationType = lLocationKey.getIataType();
00086     const treppb::PlaceType& lPlaceType = lLocationType.getTypeAsPB();
00087     const treppb::PlaceType_LocationType& lPlaceTypeEnum = lPlaceType.type();
00088     treppb::PlaceType* lPlaceTypePtr = ioPlace.mutable_loc_type();
00089     assert (lPlaceTypePtr != NULL);
00090     lPlaceTypePtr->set_type (lPlaceTypeEnum);
00091
00092     // Retrieve and set the Geonames ID
00093     const GeonamesID_T& lGeonamesID = lLocationKey.getGeonamesID();
00094     treppb::GeonameID* lGeonameIDPtr = ioPlace.mutable_geoname_id();
00095     assert (lGeonameIDPtr != NULL);
00096     lGeonameIDPtr->set_id (lGeonamesID);
00097
00098     // Retrieve and set the ICAO code
00099     const ICAOCode_T& lIcaoCode = iLocation.getIcaoCode();
00100     treppb::ICAOCode* lIcaoCodePtr = ioPlace.mutable_icao_code();
00101     assert (lIcaoCodePtr != NULL);
00102     lIcaoCodePtr->set_code (lIcaoCode);
00103
00104     // Retrieve and set the FAA code
00105     const FAACode_T& lFaaCode = iLocation.getFaaCode();
00106     treppb::FAACode* lFaaCodePtr = ioPlace.mutable_faa_code();
00107     assert (lFaaCodePtr != NULL);
00108     lFaaCodePtr->set_code (lFaaCode);
00109
00110     // Retrieve and set the names
00111     const CommonName_T& lUtfName = iLocation.getCommonName();
00112     ioPlace.set_name_utf (lUtfName);
00113     const ASCIIName_T& lAsciiName = iLocation.getAsciiName();
00114     ioPlace.set_name_ascii (lAsciiName);
00115
00116     // Retrieve and set the feature class and code
00117     const FeatureClass_T& lFeatClass = iLocation.getFeatureClass();
00118     const FeatureCode_T& lFeatCode = iLocation.getFeatureCode();
00119     treppb::FeatureType* lFeatTypePtr = ioPlace.mutable_feat_type();
00120     assert (lFeatTypePtr != NULL);
00121     treppb::FeatureClass* lFeatClassPtr = lFeatTypePtr->mutable_fclass();
00122     assert (lFeatClassPtr != NULL);
00123     treppb::FeatureCode* lFeatCodePtr = lFeatTypePtr->mutable_fcode();
00124     assert (lFeatCodePtr != NULL);
00125     lFeatClassPtr->set_code (lFeatClass);
00126     lFeatCodePtr->set_code (lFeatCode);
00127
00128     // Retrieve and set the geographical coordinates
00129     const Latitude_T& lLatitude = iLocation.getLatitude();
```

```
00130     const Longitude_T& lLongitude = iLocation.getLongitude();
00131     treppb::GeoPoint* lPointPtr = ioPlace.mutable_coord();
00132     assert (lPointPtr != NULL);
00133     lPointPtr->set_latitude (lLatitude);
00134     lPointPtr->set_longitude (lLongitude);
00135
00136     // Retrieve and set the city IATA code
00137     const CityCode_T& lCityCode = iLocation.getCityCode();
00138     treppb::IATACode* lCityCodePtr = ioPlace.mutable_city_code();
00139     assert (lCityCodePtr != NULL);
00140     lCityCodePtr->set_code (lCityCode);
00141
00142     // Retrieve and set the city names
00143     const CityUTFName_T& lCityUtfName = iLocation.getCityUtfName();
00144     ioPlace.set_city_name_utf (lCityUtfName);
00145     const CityASCIIName_T& lCityAsciiName = iLocation.
      getCityAsciiName();
00146     ioPlace.set_city_name_ascii (lCityAsciiName);
00147
00148     // Retrieve and set the state code
00149     const StateCode_T& lStateCode = iLocation.getStateCode();
00150     treppb::StateCode* lStateCodePtr = ioPlace.mutable_state_code();
00151     assert (lStateCodePtr != NULL);
00152     lStateCodePtr->set_code (lStateCode);
00153
00154     // Retrieve and set the country code
00155     const CountryCode_T& lCountryCode = iLocation.getCountryCode();
00156     treppb::CountryCode* lCountryCodePtr = ioPlace.mutable_country_code();
00157     assert (lCountryCodePtr != NULL);
00158     lCountryCodePtr->set_code (lCountryCode);
00159
00160     // Retrieve and set the alternative country code
00161     const AltCountryCode_T& lAltCountryCode = iLocation.
      getAltCountryCode();
00162     treppb::AltCountryCode* lAltCountryCodePtr =
00163       ioPlace.mutable_alt_country_code();
00164     assert (lAltCountryCodePtr != NULL);
00165     lAltCountryCodePtr->set_code (lAltCountryCode);
00166
00167     // Retrieve and set the country name
00168     const CountryName_T& lCountryName = iLocation.getCountryName();
00169     ioPlace.set_country_name (lCountryName);
00170
00171     // Retrieve and set the continent code
00172     const ContinentCode_T& lContinentCode = iLocation.
      getContinentCode();
00173     treppb::ContinentCode* lContinentCodePtr = ioPlace.mutable_continent_code();
00174     assert (lContinentCodePtr != NULL);
00175     lContinentCodePtr->set_code (lContinentCode);
00176
00177     // Retrieve and set the continent name
00178     const ContinentName_T& lContinentName = iLocation.
      getContinentName();
00179     ioPlace.set_continent_name (lContinentName);
00180
00181     // Retrieve and set the admin level 1 code
00182     const Admin1Code_T& lAdm1Code = iLocation.getAdmin1Code();
00183     treppb::Admin1Code* lAdm1CodePtr = ioPlace.mutable_adm1_code();
00184     assert (lAdm1CodePtr != NULL);
00185     lAdm1CodePtr->set_code (lAdm1Code);
00186
00187     // Retrieve and set the admin level 1 names
00188     const Admin1UTFName_T& lAdm1UtfName = iLocation.
      getAdmin1UtfName();
00189     ioPlace.set_adm1_name_utf (lAdm1UtfName);
00190     const Admin1ASCIIName_T& lAdm1AsciiName = iLocation.
      getAdmin1AsciiName();
00191     ioPlace.set_adm1_name_ascii (lAdm1AsciiName);
00192
00193     // Retrieve and set the admin level 2 code
00194     const Admin2Code_T& lAdm2Code = iLocation.getAdmin2Code();
00195     treppb::Admin2Code* lAdm2CodePtr = ioPlace.mutable_adm2_code();
00196     assert (lAdm2CodePtr != NULL);
00197     lAdm2CodePtr->set_code (lAdm2Code);
00198
00199     // Retrieve and set the admin level 2 names
00200     const Admin2UTFName_T& lAdm2UtfName = iLocation.
      getAdmin2UtfName();
00201     ioPlace.set_adm2_name_utf (lAdm2UtfName);
00202     const Admin2ASCIIName_T& lAdm2AsciiName = iLocation.
      getAdmin2AsciiName();
00203     ioPlace.set_adm2_name_ascii (lAdm2AsciiName);
00204
00205     // Retrieve and set the admin level 3 code
00206     const Admin3Code_T& lAdm3Code = iLocation.getAdmin3Code();
00207     treppb::Admin3Code* lAdm3CodePtr = ioPlace.mutable_adm3_code();
00208     assert (lAdm3CodePtr != NULL);
```

```
00209     lAdm3CodePtr->set_code (lAdm3Code);
00210
00211     // Retrieve and set the admin level 4 code
00212     const Admin4Code_T& lAdm4Code = iLocation.getAdmin4Code();
00213     treppb::Admin4Code* lAdm4CodePtr = ioPlace.mutable_adm4_code();
00214     assert (lAdm4CodePtr != NULL);
00215     lAdm4CodePtr->set_code (lAdm4Code);
00216
00217     // Retrieve and set the population
00218     const Population_T& lPopulation = iLocation.getPopulation();
00219     treppb::Population* lPopulationPtr = ioPlace.mutable_population();
00220     assert (lPopulationPtr != NULL);
00221     lPopulationPtr->set_value (lPopulation);
00222
00223     // Retrieve and set the elevation
00224     const Elevation_T& lElevation = iLocation.getElevation();
00225     treppb::Elevation* lElevationPtr = ioPlace.mutable_elevation();
00226     assert (lElevationPtr != NULL);
00227     lElevationPtr->set_value (lElevation);
00228
00229     // Retrieve and set the geo topology 30
00230     const GTopo30_T& lGTopo30 = iLocation.getGTopo30();
00231     treppb::GTopo30* lGTopo30Ptr = ioPlace.mutable_gtopo30();
00232     assert (lGTopo30Ptr != NULL);
00233     lGTopo30Ptr->set_value (lGTopo30);
00234
00235     // Retrieve and set the PageRank value
00236     const PageRank_T& lPageRank = iLocation.getPageRank();
00237     treppb::PageRank* lPageRankPtr = ioPlace.mutable_page_rank();
00238     assert (lPageRankPtr != NULL);
00239     lPageRankPtr->set_rank (lPageRank);
00240
00241     // Retrieve and set the time-zone
00242     const TimeZone_T& lTimeZone = iLocation.getTimeZone();
00243     treppb::TimeZone* lTimeZonePtr = ioPlace.mutable_tz();
00244     assert (lTimeZonePtr != NULL);
00245     lTimeZonePtr->set_tz (lTimeZone);
00246
00247     // Retrieve and set the GMT offset
00248     const GMTOffset_T& lGMTOffset = iLocation.getGMTOffset();
00249     treppb::TZOffSet* lGMTOffsetPtr = ioPlace.mutable_gmt_offset();
00250     assert (lGMTOffsetPtr != NULL);
00251     lGMTOffsetPtr->set_offset (lGMTOffset);
00252
00253     // Retrieve and set the DST offset
00254     const DSTOffset_T& lDSTOffset = iLocation.getDSTOffset();
00255     treppb::TZOffSet* lDSTOffsetPtr = ioPlace.mutable_dst_offset();
00256     assert (lDSTOffsetPtr != NULL);
00257     lDSTOffsetPtr->set_offset (lDSTOffset);
00258
00259     // Retrieve and set the RAW offset
00260     const RawOffset_T& lRAWOffset = iLocation.getRawOffset();
00261     treppb::TZOffSet* lRAWOffsetPtr = ioPlace.mutable_raw_offset();
00262     assert (lRAWOffsetPtr != NULL);
00263     lRAWOffsetPtr->set_offset (lRAWOffset);
00264
00265     // Retrieve and set the modification date (within Geonames)
00266     const Date_T& lGeonameModDate = iLocation.getModificationDate();
00267     treppb::Date* lGeonameModDatePtr = ioPlace.mutable_mod_date();
00268     assert (lGeonameModDatePtr != NULL);
00269     lGeonameModDatePtr->set_date (boost::gregorian::to_iso_extended_string(lGeonameModDate));
00270
00271     // Retrieve and set the list of the travel-related POR IATA codes
00272     const TvlPORListString_T& lTvlPORList = iLocation.
    getTvlPORListString();
00273     treppb::TravelRelatedList* lTvlPORListPtr = ioPlace.mutable_tvl_por_list();
00274     assert (lTvlPORListPtr != NULL);
00275     lTvlPORListPtr->add_tvl_code (lTvlPORList);
00276
00277     // Retrieve and set the list of the Wikipedia links (URLs)
00278     const WikiLink_T& lWikiLink = iLocation.getWikiLink();
00279     treppb::WikiLinkList* lWikiLinkListPtr = ioPlace.mutable_link_list();
00280     assert (lWikiLinkListPtr != NULL);
00281     treppb::WikiLink* lWikiLinkPtr = lWikiLinkListPtr->add_link();
00282     assert (lWikiLinkPtr != NULL);
00283     treppb::LanguageCode* lLangCodePtr = lWikiLinkPtr->mutable_lang();
00284     assert (lLangCodePtr != NULL);
00285     lLangCodePtr->set_code ("en");
00286     lWikiLinkPtr->set_link (lWikiLink);
00287
00288     // Retrieve and set the beginning date of the validity period
00289     const Date_T& lDateFrom = iLocation.getDateFrom();
00290     treppb::Date* lDateFromPtr = ioPlace.mutable_date_from();
00291     assert (lDateFromPtr != NULL);
00292     lDateFromPtr->set_date (boost::gregorian::to_iso_extended_string(lDateFrom));
00293
00294     // Retrieve and set the end date of the validity period
```

```
00295     const Date_T& lDateEnd = iLocation.getDateEnd();
00296     treppb::Date* lDateEndPtr = ioPlace.mutable_date_end();
00297     assert (lDateEndPtr != NULL);
00298     lDateEndPtr->set_date (boost::gregorian::to_iso_extended_string(lDateEnd));
00299
00300     // Retrieve and set the commentaries
00301     const Comment_T& lComment = iLocation.getComment();
00302     treppb::Comment* lCommentPtr = ioPlace.mutable_comment();
00303     assert (lCommentPtr != NULL);
00304     lCommentPtr->set_text (lComment);
00305
00306     // Retrieve and set the list of alternate names
00307     const NameMatrix& lNameMatrixRef = iLocation.getNameMatrix();
00308     treppb::AltNameList* lAltNameListPtr = ioPlace.mutable_alt_name_list();
00309     assert (lAltNameListPtr != NULL);
00310     //
00311     const NameMatrix_T lNameMatrix = lNameMatrixRef.getNameMatrix();
00312     for (NameMatrix_T::const_iterator itNameList = lNameMatrix.begin();
00313          itNameList != lNameMatrix.end(); ++itNameList) {
00314       const Names& lNameListRef = itNameList->second;
00315       const LanguageCode_T& lLangCode = lNameListRef.
    getLanguageCode();
00316       const NameList_T& lNameList = lNameListRef.getNameList();
00317       for (NameList_T::const_iterator itName = lNameList.begin();
00318            itName != lNameList.end(); ++itName) {
00319         const std::string& lName = *itName;
00320         //
00321         treppb::AltName* lAltNamePtr = lAltNameListPtr->add_name();
00322         assert (lAltNamePtr != NULL);
00323         //
00324         treppb::LanguageCode* lLangCodePtr = lAltNamePtr->mutable_lang();
00325         assert (lLangCodePtr != NULL);
00326         lLangCodePtr->set_code (lLangCode);
00327         lAltNamePtr->set_name (lName);
00328       }
00329     }
00330
00331     // Retrieve and set the matching percentage value
00332     const MatchingPercentage_T& lPercentage = iLocation.
    getPercentage();
00333     treppb::MatchingPercentage* lPercentagePtr =
00334       ioPlace.mutable_matching_percentage();
00335     assert (lPercentagePtr != NULL);
00336     lPercentagePtr->set_percentage (lPercentage);
00337
00338     // Retrieve and set the list of the original keywords
00343     const std::string& lOriginalKeywords = iLocation.getOriginalKeywords();
00344     treppb::KeywordList* lOriginalKeywordListPtr =
00345       ioPlace.mutable_original_keyword_list();
00346     assert (lOriginalKeywordListPtr != NULL);
00347     lOriginalKeywordListPtr->add_word (lOriginalKeywords);
00348
00349     // Retrieve and set the list of the corrected keywords
00354     const std::string& lCorrectedKeywords = iLocation.getCorrectedKeywords();
00355     treppb::KeywordList* lCorrectedKeywordListPtr =
00356       ioPlace.mutable_corrected_keyword_list();
00357     assert (lCorrectedKeywordListPtr != NULL);
00358     lCorrectedKeywordListPtr->add_word (lCorrectedKeywords);
00359
00360     // Retrieve and set the actual edit distance
00361     const NbOfErrors_T& lEditDistanceActual = iLocation.
    getEditDistance();
00362     treppb::EditDistance* lEditDistanceActualPtr =
00363       ioPlace.mutable_edit_distance_actual();
00364     assert (lEditDistanceActualPtr != NULL);
00365     lEditDistanceActualPtr->set_dist (lEditDistanceActual);
00366
00367     // Retrieve and set the allowable edit distance
00368     const NbOfErrors_T& lEditDistanceAllowable =
00369       iLocation.getAllowableEditDistance();
00370     treppb::EditDistance* lEditDistanceAllowablePtr =
00371       ioPlace.mutable_edit_distance_actual();
00372     assert (lEditDistanceAllowablePtr != NULL);
00373     lEditDistanceAllowablePtr->set_dist (lEditDistanceAllowable);
00374
00375     // Iterate on the extra list of locations
00376     const LocationList_T& lExtraLocationList = iLocation.
    getExtraLocationList();
00377     treppb::PlaceList* lExtraPlaceListPtr = ioPlace.mutable_extra_place_list();
00378     assert (lExtraPlaceListPtr != NULL);
00379     for (LocationList_T::const_iterator itLoc = lExtraLocationList.begin();
00380          itLoc != lExtraLocationList.end(); ++itLoc) {
00381       const Location& lExtraLocation = *itLoc;
00382       //
00383       treppb::Place* lPlacePtr = lExtraPlaceListPtr->add_place();
00384       assert (lPlacePtr != NULL);
00385       //
```

```
00386       exportLocation (*lPlacePtr, lExtraLocation);
00387     }
00388
00389     // Iterate on the alternative list of locations
00390     const LocationList_T& lAltLocationList= iLocation.
      getAlternateLocationList();
00391     treppb::PlaceList* lAltPlaceListPtr = ioPlace.mutable_alt_place_list();
00392     assert (lAltPlaceListPtr != NULL);
00393     for (LocationList_T::const_iterator itLoc = lAltLocationList.begin();
00394          itLoc != lAltLocationList.end(); ++itLoc) {
00395       const Location& lAlternateLocation = *itLoc;
00396       //
00397       treppb::Place* lPlacePtr = lAltPlaceListPtr->add_place();
00398       assert (lPlacePtr != NULL);
00399       //
00400       exportLocation (*lPlacePtr, lAlternateLocation);
00401     }
00402   }
00403
00404 }
00405
```

24.117 opentrep/bom/LocationExchange.hpp File Reference

```
#include <iosfwd>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/LocationList.hpp>
```

Classes

- class OPENTREP::LocationExchange

  Utility class to export Opentrep structures in a Protobuf format.

Namespaces

- treppb
- OPENTREP

24.118 LocationExchange.hpp

```
00001 #ifndef __OPENTREP_BOM_LOCATIONEXCHANGE_HPP
00002 #define __OPENTREP_BOM_LOCATIONEXCHANGE_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // OpenTrep
00010 #include <opentrep/OPENTREP_Types.hpp>
00011 #include <opentrep/LocationList.hpp>
00012
00013 // Forward declarations for the Protobuf structures
00014 namespace treppb {
00015   class Place;
00016 }
00017
00018 namespace OPENTREP {
00019
00020   // Forward declarations
00021   struct Location;
00022
00026   class LocationExchange {
00027   public:
00028
00038     static void exportLocationList (std::ostream&, const
      LocationList_T&,
00039                                     const WordList_T& iNonMatchedWordList);
00040
00049     static void exportLocation (treppb::Place&, const Location&);
00050   };
```

```
00051
00052 }
00053 #endif // __OPENTREP_BOM_LOCATIONEXCHANGE_HPP
```

24.119 opentrep/bom/Month.cpp File Reference

24.120 Month.cpp

24.121 opentrep/bom/Month.hpp File Reference

24.122 Month.hpp

24.123 opentrep/bom/Passenger.cpp File Reference

24.124 Passenger.cpp

24.125 opentrep/bom/Passenger.hpp File Reference

24.126 Passenger.hpp

24.127 opentrep/bom/Place.cpp File Reference

```
#include <cassert>
#include <opentrep/basic/BasConst_General.hpp>
#include <opentrep/basic/OTransliterator.hpp>
#include <opentrep/basic/Utilities.hpp>
#include <opentrep/bom/WordCombinationHolder.hpp>
#include <opentrep/bom/Place.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.128 Place.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // OpenTrep
00007 #include <opentrep/basic/BasConst_General.hpp>
00008 #include <opentrep/basic/OTransliterator.hpp>
00009 #include <opentrep/basic/Utilities.hpp>
00010 #include <opentrep/bom/WordCombinationHolder.hpp>
00011 #include <opentrep/bom/Place.hpp>
00012 #include <opentrep/service/Logger.hpp>
```

```
00013
00014 namespace OPENTREP {
00015
00016   // ////////////////////////////////////////////////////////////////////
00017   Place::Place() :
00018     _world (NULL), _placeHolder (NULL), _mainPlace (NULL),
00019     _location (IATACode_T (""), IATAType::LAST_VALUE, 0,
00020                ICAOCode_T (""), FAACode_T (""),
00021                CommonName_T (""), ASCIIName_T (""), 0,
00022                Date_T (1970, 01, 01), Date_T (2999, 12, 31), Comment_T (""),
00023                CityCode_T (""), CityUTFName_T (""), CityASCIIName_T (""), 0,
00024                StateCode_T (""),
00025                CountryCode_T (""), AltCountryCode_T (""), CountryName_T (""),
00026                ContinentName_T (""),
00027                0.0, 0.0,
00028                FeatureClass_T (""), FeatureCode_T (""),
00029                Admin1Code_T (""), Admin1UTFName_T (""), Admin1ASCIIName_T (""),
00030                Admin2Code_T (""), Admin2UTFName_T (""), Admin2ASCIIName_T (""),
00031                Admin3Code_T (""), Admin4Code_T (""),
00032                0, 0, 0, TimeZone_T (""), 0, 0, 0,
00033                Date_T (2000, 01, 01), TvlPORListString_T (""),
00034                WikiLink_T (""),  K_DEFAULT_PAGE_RANK, "", "", 0, 0, 0,
00035                RawDataString_T ("")),
00036     _docID (0) {
00037   }
00038
00039   // ////////////////////////////////////////////////////////////////////
00040   Place::Place (const LocationKey& iKey) :
00041     _world (NULL), _placeHolder (NULL), _mainPlace (NULL),
00042     _location (iKey.getIataCode(), iKey.getIataType(), iKey.getGeonamesID(),
00043                ICAOCode_T (""), FAACode_T (""),
00044                CommonName_T (""), ASCIIName_T (""), 0,
00045                Date_T (1970, 01, 01), Date_T (2999, 12, 31), Comment_T (""),
00046                CityCode_T (""), CityUTFName_T (""), CityASCIIName_T (""), 0,
00047                StateCode_T (""),
00048                CountryCode_T (""), AltCountryCode_T (""), CountryName_T (""),
00049                ContinentName_T (""),
00050                0.0, 0.0,
00051                FeatureClass_T (""), FeatureCode_T (""),
00052                Admin1Code_T (""), Admin1UTFName_T (""), Admin1ASCIIName_T (""),
00053                Admin2Code_T (""), Admin2UTFName_T (""), Admin2ASCIIName_T (""),
00054                Admin3Code_T (""), Admin4Code_T (""),
00055                0, 0, 0, TimeZone_T (""), 0, 0, 0,
00056                Date_T (2000, 01, 01), TvlPORListString_T (""),
00057                WikiLink_T (""), K_DEFAULT_PAGE_RANK, "", "", 0, 0, 0,
00058                RawDataString_T ("")),
00059     _docID (0) {
00060   }
00061
00062   // ////////////////////////////////////////////////////////////////////
00063   Place::Place (const Location& iLocation) :
00064     _world (NULL), _placeHolder (NULL), _mainPlace (NULL),
00065     _location (iLocation), _docID (0) {
00066   }
00067
00068   // ////////////////////////////////////////////////////////////////////
00069   Place::Place (const Place& iPlace) :
00070     _world (iPlace._world), _placeHolder (iPlace._placeHolder),
00071     _mainPlace (iPlace._mainPlace),
00072     _location (iPlace._location),
00073     _docID (iPlace._docID),
00074     _termSetMap (iPlace._termSetMap), _spellingSet (iPlace._spellingSet),
00075     _stemmingSet (iPlace._stemmingSet), _synonymSet (iPlace._synonymSet) {
00076   }
00077
00078   // ////////////////////////////////////////////////////////////////////
00079   Place::~Place() {
00080   }
00081
00082   // ////////////////////////////////////////////////////////////////////
00083   std::string Place::toString() const {
00084     std::ostringstream oStr;
00085     oStr << _location.toString();
00086     oStr << ", " << _docID
00087          << ". ";
00088
00089     if (_extraPlaceList.empty() == false) {
00090       oStr << "; Extra matches: {";
00091       unsigned short idx = 0;
00092       for (PlaceOrderedList_T::const_iterator itLoc = _extraPlaceList.begin();
00093            itLoc != _extraPlaceList.end(); ++itLoc, ++idx) {
00094         if (idx != 0) {
00095           oStr << "; ";
00096         }
00097         const Place* lExtraPlace_ptr = *itLoc;
00098         assert (lExtraPlace_ptr != NULL);
00099         oStr << lExtraPlace_ptr->toShortString();
```

```
00100       }
00101       oStr << "}";
00102     }
00103
00104     if (_alternatePlaceList.empty() == false) {
00105       oStr << "; Alternate matches: {";
00106       unsigned short idx = 0;
00107       for (PlaceOrderedList_T::const_iterator itLoc =
00108              _alternatePlaceList.begin();
00109            itLoc != _alternatePlaceList.end(); ++itLoc, ++idx) {
00110         if (idx != 0) {
00111           oStr << "; ";
00112         }
00113         const Place* lAlternatePlace_ptr = *itLoc;
00114         assert (lAlternatePlace_ptr != NULL);
00115         oStr << lAlternatePlace_ptr->toShortString();
00116       }
00117       oStr << "}";
00118     }
00119
00120     return oStr.str();
00121   }
00122
00123   // ////////////////////////////////////////////////////////////////////
00124   std::string Place::toShortString() const {
00130     std::ostringstream oStr;
00131     oStr << _location.toShortString();
00132     oStr << ", " << _docID;
00133
00134     if (_extraPlaceList.empty() == false) {
00135       oStr << " " << _extraPlaceList.size() << " extra match(es)";
00136     }
00137
00138     if (_alternatePlaceList.empty() == false) {
00139       oStr << " " << _alternatePlaceList.size() << " alternate match(es)";
00140     }
00141
00142     return oStr.str();
00143   }
00144
00145   // ////////////////////////////////////////////////////////////////////
00146   void Place::toStream (std::ostream& ioOut) const {
00147     ioOut << toString();
00148   }
00149
00150   // ////////////////////////////////////////////////////////////////////
00151   void Place::fromStream (std::istream& ioIn) {
00152   }
00153
00154   // ////////////////////////////////////////////////////////////////////
00155   std::string Place::describeSets() const {
00156     std::ostringstream oStr;
00157
00158     // Xapian index for the current place/POR (point of reference)
00159     oStr << "[index] ";
00160     short idx_map = 0;
00161     for (TermSetMap_T::const_iterator itStringSet = _termSetMap.begin();
00162          itStringSet != _termSetMap.end(); ++itStringSet, ++idx_map) {
00163       if (idx_map != 0) {
00164         oStr << " - ";
00165       }
00166       // Retrieve the weight and display it
00167       const Weight_T& lWeight = itStringSet->first;
00168       oStr << "[" << lWeight << "] ";
00169
00170       // Retrieve and browse the set of strings for that weight
00171       const StringSet_T& lStringSet = itStringSet->second;
00172       short idx_set = 0;
00173       for (StringSet_T::const_iterator itString = lStringSet.begin();
00174            itString != lStringSet.end(); ++itString, ++idx_set) {
00175         if (idx_set != 0) {
00176           oStr << ", ";
00177         }
00178         const std::string& lString = *itString;
00179         oStr << lString;
00180       }
00181     }
00182
00183     // Xapian spelling dictionary
00184     oStr << "; [spelling] ";
00185     short idx = 0;
00186     for (StringSet_T::const_iterator itTerm = _spellingSet.begin();
00187          itTerm != _spellingSet.end(); ++itTerm, ++idx) {
00188       if (idx != 0) {
00189         oStr << ", ";
00190       }
00191       const std::string& lTerm = *itTerm;
```

```
00192       oStr <<  lTerm;
00193     }
00194     oStr << ";";
00195
00196     return oStr.str();
00197   }
00198
00199   // ////////////////////////////////////////////////////////////////////
00200   std::string Place::shortDisplay() const {
00201     /* When the city code is empty, it means that the place is a city and
00202        not an airport. The city code is thus the same as the place code
00203        itself. */
00204     std::ostringstream oStr;
00205     oStr << _location.toBasicString();
00206     oStr << ", " << _docID;
00207     return oStr.str();
00208   }
00209
00210   // ////////////////////////////////////////////////////////////////////
00211   std::string Place::display() const {
00212     std::ostringstream oStr;
00213     oStr << shortDisplay() << std::endl;
00214     return oStr.str();
00215   }
00216
00217   // ////////////////////////////////////////////////////////////////////
00218   void Place::resetIndexSets() {
00219     _termSetMap.clear();
00220     _spellingSet.clear();
00221     _stemmingSet.clear();
00222     _synonymSet.clear();
00223   }
00224
00225   // ////////////////////////////////////////////////////////////////////
00226   Place::StringSet_T Place::getTermSet (const
      Weight_T& iWeight) const {
00227     // Initialise an empty string set
00228     StringSet_T oTermSet;
00229
00230     // If existing, retrieve the string set for that weight.
00231     TermSetMap_T::const_iterator itTermSet = _termSetMap.find (iWeight);
00232     if (itTermSet != _termSetMap.end()) {
00233       oTermSet = itTermSet->second;
00234     }
00235
00236     return oTermSet;
00237   }
00238
00239   // ////////////////////////////////////////////////////////////////////
00240   bool Place::addTermSet (const Weight_T& iWeight, const
      StringSet_T& iTermSet) {
00241     bool hasInsertBeenSuccessful = true;
00242
00243     // If existing, retrieve the string set for that weight.
00244     TermSetMap_T::iterator itTermSet = _termSetMap.find (iWeight);
00245
00246     if (itTermSet == _termSetMap.end()) {
00247       // If there was no string set for that weight yet, insert it
00248       hasInsertBeenSuccessful =
00249         _termSetMap.insert (TermSetMap_T::value_type (iWeight, iTermSet)).second;
00250
00251       // Sanity check
00252       assert (hasInsertBeenSuccessful == true);
00253
00254     } else {
00255       // Otherwise, add the given string set, string by string,
00256       // to the existing one
00257       StringSet_T& lStringSet = itTermSet->second;
00258       for (StringSet_T::const_iterator itString = iTermSet.begin();
00259            itString != iTermSet.end(); ++itString) {
00260         const std::string& lString = *itString;
00261         lStringSet.insert (lString);
00262       }
00263     }
00264
00265     return hasInsertBeenSuccessful;
00266   }
00267
00268   // ////////////////////////////////////////////////////////////////////
00269   void Place::addNameToXapianSets (const Weight_T& iWeight,
00270                                    const std::string& iBaseName,
00271                                    const FeatureCode_T& iFeatureCode) {
00272     // Retrieve the string set for the given weight, if existing
00273     // (empty otherwise)
00274     StringSet_T lTermSet = getTermSet (iWeight);
00275
00282     const FeatureNameList_T& lFeatureNameList =
```

```
00283       Location::getFeatureList (iFeatureCode);
00284     for (FeatureNameList_T::const_iterator itFeatName = lFeatureNameList.begin();
00285         itFeatName != lFeatureNameList.end(); ++itFeatName) {
00286       const FeatureName_T& lFeatureName = *itFeatName;
00287
00288       lTermSet.insert (iBaseName + " " + lFeatureName);
00289     }
00290
00291     // Insert (or replace) the newly created (or just altered) string set
00292     addTermSet (iWeight, lTermSet);
00293   }
00294
00295   // ////////////////////////////////////////////////////////////////////
00296   void Place::addNameToXapianSets (const Weight_T& iWeight,
00297                                    const LocationName_T& iLocationName,
00298                                    const FeatureCode_T& iFeatureCode,
00299                                    const CityUTFName_T& iCityUtfName,
00300                                    const CityASCIIName_T& iCityAsciiName,
00301                                    const Admin1UTFName_T& iAdm1UtfName,
00302                                    const Admin1ASCIIName_T& iAdm1AsciiName,
00303                                    const Admin2UTFName_T& iAdm2UtfName,
00304                                    const Admin2ASCIIName_T& iAdm2AsciiName,
00305                                    const StateCode_T& iStateCode,
00306                                    const CountryCode_T& iCountryCode,
00307                                    const CountryName_T& iCountryName,
00308                                    const ContinentName_T& iContinentName,
00309                                    const OTransliterator& iTransliterator) {
00310     // Retrieve the string set for the given weight, if existing
00311     // (empty otherwise)
00312     StringSet_T lTermSet = getTermSet (iWeight);
00313
00314     // Tokenise the name. Some of the names contain punctuation characters.
00315     // For instance, "Paris/FR/Gare" is transformed into "Paris FR Gare".
00316     WordList_T lWordList;
00317     tokeniseStringIntoWordList (iLocationName, lWordList);
00318     const std::string lTokenisedName = createStringFromWordList (lWordList);
00319
00320     // Add the tokenised name to the Xapian index
00321     lTermSet.insert (lTokenisedName);
00322
00323     // Add the (tokenised name, feature name) pair to the Xapian index
00324     addNameToXapianSets (iWeight, lTokenisedName, iFeatureCode);
00325
00326     // Add the (tokenised name, UTF8 city name) pair to the Xapian index
00327     if (iCityUtfName.empty() == false) {
00328       lTermSet.insert (lTokenisedName + " " + iCityUtfName);
00329     }
00330
00331     // Add the (tokenised name, ASCII city name) pair to the Xapian index
00332     if (iCityAsciiName.empty() == false) {
00333       lTermSet.insert (lTokenisedName + " " + iCityAsciiName);
00334     }
00335
00336     // Add the (tokenised name, UTF8 admin level 1 name) pair to the Xapian index
00337     if (iAdm1UtfName.empty() == false) {
00338       lTermSet.insert (lTokenisedName + " " + iAdm1UtfName);
00339     }
00340
00341     // Add the (tokenised name, ASCII adm level 1 name) pair to the Xapian index
00342     if (iAdm1AsciiName.empty() == false) {
00343       lTermSet.insert (lTokenisedName + " " + iAdm1AsciiName);
00344     }
00345
00346     // Add the (tokenised name, UTF8 admin level 2 name) pair to the Xapian index
00347     if (iAdm2UtfName.empty() == false) {
00348       lTermSet.insert (lTokenisedName + " " + iAdm2UtfName);
00349     }
00350
00351     // Add the (tokenised name, ASCII adm level 2 name) pair to the Xapian index
00352     if (iAdm2AsciiName.empty() == false) {
00353       lTermSet.insert (lTokenisedName + " " + iAdm2AsciiName);
00354     }
00355
00356     // Add the (tokenised name, state code) pair to the Xapian index
00357     lTermSet.insert (lTokenisedName + " " + iStateCode);
00358
00359     // Add the (tokenised name, country code) pair to the Xapian index
00360     lTermSet.insert (lTokenisedName + " " + iCountryCode);
00361
00362     // Add the (tokenised name, country name) pair to the Xapian index
00363     lTermSet.insert (lTokenisedName + " " + iCountryName);
00364
00365     // Add the (tokenised name, continent name) pair to the Xapian index
00366     lTermSet.insert (lTokenisedName + " " + iContinentName);
00367
00368     // Add the tokenised name to the Xapian spelling dictionary
00369     _spellingSet.insert (lTokenisedName);
```

```
00370
00371     // Normalise, according to the Unicode standard, the given name.
00372     // Note that it is important to normalise after the tokenisation process,
00373     // as the punctuation is eliminated (and not replaced by space)
00374     // by that latter.
00375     const std::string& lNormalisedCommonName =
00376       iTransliterator.normalise (lTokenisedName);
00377
00378     // Add the tokenised and normalised name to the Xapian index
00379     lTermSet.insert (lNormalisedCommonName);
00380
00381     // Add the (tokenised and normalised name, feature name) pair
00382     // to the Xapian index
00383     addNameToXapianSets (iWeight, lNormalisedCommonName, iFeatureCode);
00384
00385     // Add the (tokenised and normalised name, UTF8 city name) pair
00386     // to the Xapian index
00387     if (iCityUtfName.empty() == false) {
00388       lTermSet.insert (lNormalisedCommonName + " " + iCityUtfName);
00389     }
00390
00391     // Add the (tokenised and normalised name, ASCII city name) pair
00392     // to the Xapian index
00393     if (iCityAsciiName.empty() == false) {
00394       lTermSet.insert (lNormalisedCommonName + " " + iCityAsciiName);
00395     }
00396
00397     // Add the (tokenised and normalised name, UTF8 admin level 1 name) pair
00398     // to the Xapian index
00399     if (iAdm1UtfName.empty() == false) {
00400       lTermSet.insert (lNormalisedCommonName + " " + iAdm1UtfName);
00401     }
00402
00403     // Add the (tokenised and normalised name, ASCII adm level 1 name) pair
00404     // to the Xapian index
00405     if (iAdm1AsciiName.empty() == false) {
00406       lTermSet.insert (lNormalisedCommonName + " " + iAdm1AsciiName);
00407     }
00408
00409     // Add the (tokenised and normalised name, UTF8 admin level 2 name) pair
00410     // to the Xapian index
00411     if (iAdm2UtfName.empty() == false) {
00412       lTermSet.insert (lNormalisedCommonName + " " + iAdm2UtfName);
00413     }
00414
00415     // Add the (tokenised and normalised name, ASCII adm level 2 name) pair
00416     // to the Xapian index
00417     if (iAdm2AsciiName.empty() == false) {
00418       lTermSet.insert (lNormalisedCommonName + " " + iAdm2AsciiName);
00419     }
00420
00421     // Add the (tokenised and normalised name, state code) pair to the
00422     // Xapian index
00423     lTermSet.insert (lNormalisedCommonName + " " + iStateCode);
00424
00425     // Add the (tokenised and normalised name, country code) pair to the
00426     // Xapian index
00427     lTermSet.insert (lNormalisedCommonName + " " + iCountryCode);
00428
00429     // Add the (tokenised and normalised name, country name) pair to the
00430     // Xapian index
00431     lTermSet.insert (lNormalisedCommonName + " " + iCountryName);
00432
00433     // Add the (tokenised and normalised name, continent name) pair to the
00434     // Xapian index
00435     lTermSet.insert (lNormalisedCommonName + " " + iContinentName);
00436
00437     // Add the tokenised and normalised name to the Xapian spelling dictionary
00438     _spellingSet.insert (lNormalisedCommonName);
00439
00440     // Insert (or replace) the newly created (or just altered) string set
00441     addTermSet (iWeight, lTermSet);
00442   }
00443
00444   // ////////////////////////////////////////////////////////////////////
00445   void Place::buildIndexSets (const OTransliterator& iTransliterator) {
00446
00457     // Retrieve the PageRank
00458     const PageRank_T& lPageRankDouble = _location.getPageRank();
00459     //const Weight_T lPageRank =
00460     //  (lPageRankDouble == K_DEFAULT_PAGE_RANK)?
00461     //  K_DEFAULT_INDEXING_EXTRA_WEIGHT:
00462     //  static_cast<const Weight_T> (lPageRankDouble);
00463     const Weight_T lPageRank = K_DEFAULT_INDEXING_EXTRA_WEIGHT
00464       + static_cast<const Weight_T> (lPageRankDouble / 5.0);
00465
00466     // Retrieve the string set for the given weight, if existing
```

```
00467     // (empty otherwise)
00468     StringSet_T lWeightedTermSet = getTermSet (lPageRank);
00469
00470     // Retrieve the string set for the given weight, if existing
00471     // (empty otherwise)
00472     StringSet_T lStdTermSet = getTermSet (
      K_DEFAULT_INDEXING_STD_WEIGHT);
00473
00474     // Retrieve the feature code
00475     const FeatureCode_T& lFeatureCode = _location.getFeatureCode();
00476
00477     // Add the IATA code
00478     const std::string& lIataCode = _location.getIataCode();
00479     if (lIataCode.empty() == false) {
00480       lWeightedTermSet.insert (lIataCode);
00481       _spellingSet.insert (lIataCode);
00482
00483       // Add the (IATA code, feature name) to the Xapian index, where the
00484       // feature name is derived from the feature code.
00485       addNameToXapianSets (lPageRank, lIataCode, lFeatureCode);
00486     }
00487
00488     // Add the ICAO code
00489     const std::string& lIcaoCode = _location.getIcaoCode();
00490     if (lIcaoCode.empty() == false) {
00491       lWeightedTermSet.insert (lIcaoCode);
00492       _spellingSet.insert (lIcaoCode);
00493
00494       // Add the (ICAO code, feature name) to the Xapian index, where the
00495       // feature name is derived from the feature code.
00496       addNameToXapianSets (lPageRank, lIcaoCode, lFeatureCode);
00497     }
00498
00499     // Add the FAA code
00500     const std::string& lFaaCode = _location.getFaaCode();
00501     if (lFaaCode.empty() == false) {
00502       lWeightedTermSet.insert (lFaaCode);
00503       _spellingSet.insert (lFaaCode);
00504
00505       // Add the (FAA code, feature name) to the Xapian index, where the
00506       // feature name is derived from the feature code.
00507       addNameToXapianSets (lPageRank, lFaaCode, lFeatureCode);
00508     }
00509
00510     // Add the Geonames ID
00511     const GeonamesID_T& lGeonamesID = _location.getGeonamesID();
00512     if (lGeonamesID != 0) {
00513       std::stringstream oStr;
00514       oStr << lGeonamesID;
00515       const std::string lGeonamesIDStr = oStr.str();
00516       lStdTermSet.insert (lGeonamesIDStr);
00517       _spellingSet.insert (lGeonamesIDStr);
00518     }
00519
00520     // Add the feature code
00521     if (lFeatureCode.empty() == false) {
00522       lWeightedTermSet.insert (lFeatureCode);
00523       _spellingSet.insert (lFeatureCode);
00524     }
00525
00526     // Add the city IATA code
00527     const std::string& lCityCode = _location.getCityCode();
00528     if (lCityCode.empty() == false && lCityCode != lIataCode) {
00529       lWeightedTermSet.insert (lCityCode);
00530       _spellingSet.insert (lCityCode);
00531     }
00532
00533     // Add the city UTF8 name
00534     const std::string& lCityUtfName = _location.getCityUtfName();
00535     if (lCityUtfName.empty() == false) {
00536       lWeightedTermSet.insert (lCityUtfName);
00537       _spellingSet.insert (lCityUtfName);
00538     }
00539
00540     // Add the city ASCII name
00541     const std::string& lCityAsciiName = _location.getCityAsciiName();
00542     if (lCityAsciiName.empty() == false) {
00543       lWeightedTermSet.insert (lCityAsciiName);
00544       _spellingSet.insert (lCityAsciiName);
00545     }
00546
00547     // Add the state code
00548     const std::string& lStateCode = _location.getStateCode();
00549     if (lStateCode.empty() == false) {
00550       lWeightedTermSet.insert (lStateCode);
00551       _spellingSet.insert (lStateCode);
00552     }
```

```
00553
00554     // Add the country code
00555     const std::string& lCountryCode = _location.getCountryCode();
00556     lWeightedTermSet.insert (lCountryCode);
00557
00558     // Add the country name
00559     const std::string& lCountryName = _location.getCountryName();
00560     if (lCountryName.empty() == false) {
00561       lWeightedTermSet.insert (lCountryName);
00562       _spellingSet.insert (lCountryName);
00563     }
00564
00565     // Add the administrative level 1 code
00566     const std::string& lAdm1Code = _location.getAdmin1Code();
00567     if (lAdm1Code.empty() == false) {
00568       lWeightedTermSet.insert (lAdm1Code);
00569     }
00570
00571     // Add the administrative level 1 UTF8 name
00572     const std::string& lAdm1UtfName = _location.getAdmin1UtfName();
00573     if (lAdm1UtfName.empty() == false) {
00574       lWeightedTermSet.insert (lAdm1UtfName);
00575       _spellingSet.insert (lAdm1UtfName);
00576     }
00577
00578     // Add the administrative level 1 ASCII name
00579     const std::string& lAdm1AsciiName = _location.getAdmin1AsciiName();
00580     if (lAdm1AsciiName.empty() == false) {
00581       lWeightedTermSet.insert (lAdm1AsciiName);
00582       _spellingSet.insert (lAdm1AsciiName);
00583     }
00584
00585     // Add the administrative level 2 code
00586     const std::string& lAdm2Code = _location.getAdmin1Code();
00587     if (lAdm2Code.empty() == false) {
00588       lWeightedTermSet.insert (lAdm2Code);
00589     }
00590
00591     // Add the administrative level 2 UTF8 name
00592     const std::string& lAdm2UtfName = _location.getAdmin2UtfName();
00593     if (lAdm2UtfName.empty() == false) {
00594       lWeightedTermSet.insert (lAdm2UtfName);
00595       _spellingSet.insert (lAdm2UtfName);
00596     }
00597
00598     // Add the administrative level 2 ASCII name
00599     const std::string& lAdm2AsciiName = _location.getAdmin2AsciiName();
00600     if (lAdm2AsciiName.empty() == false) {
00601       lWeightedTermSet.insert (lAdm2AsciiName);
00602       _spellingSet.insert (lAdm2AsciiName);
00603     }
00604
00605     // Add the continent name
00606     const std::string& lContinentName = _location.getContinentName();
00607     lWeightedTermSet.insert (lContinentName);
00608
00609     // Add the common name (usually in American English, but not necessarily
00610     // in ASCII).
00611     const std::string& lCommonName = _location.getCommonName();
00612     if (lCommonName.empty() == false) {
00613       addNameToXapianSets (K_DEFAULT_INDEXING_STD_WEIGHT,
00614                            LocationName_T (lCommonName),
00615                            FeatureCode_T (lFeatureCode),
00616                            CityUTFName_T (lCityUtfName),
00617                            CityASCIIName_T (lCityAsciiName),
00618                            Admin1UTFName_T (lAdm1UtfName),
00619                            Admin1ASCIIName_T (lAdm1AsciiName),
00620                            Admin2UTFName_T (lAdm2UtfName),
00621                            Admin2ASCIIName_T (lAdm2AsciiName),
00622                            StateCode_T (lStateCode),
00623                            CountryCode_T (lCountryCode),
00624                            CountryName_T (lCountryName),
00625                            ContinentName_T (lContinentName), iTransliterator);
00626     }
00627
00628     // Add the ASCII name (not necessarily in English).
00629     const std::string& lASCIIName = _location.getAsciiName();
00630     if (lASCIIName.empty() == false) {
00631       addNameToXapianSets (K_DEFAULT_INDEXING_STD_WEIGHT,
00632                            LocationName_T (lASCIIName),
00633                            FeatureCode_T (lFeatureCode),
00634                            CityUTFName_T (lCityUtfName),
00635                            CityASCIIName_T (lCityAsciiName),
00636                            Admin1UTFName_T (lAdm1UtfName),
00637                            Admin1ASCIIName_T (lAdm1AsciiName),
00638                            Admin2UTFName_T (lAdm2UtfName),
00639                            Admin2ASCIIName_T (lAdm2AsciiName),
```

```
00640                           StateCode_T (lStateCode),
00641                           CountryCode_T (lCountryCode),
00642                           CountryName_T (lCountryName),
00643                           ContinentName_T (lContinentName), iTransliterator);
00644     }
00645
00646     // Retrieve the place names in all the available languages
00647     const NameMatrix& lNameMatrixFull = _location.getNameMatrix();
00648     const NameMatrix_T& lNameMatrix = lNameMatrixFull.getNameMatrix();
00649     for (NameMatrix_T::const_iterator itNameList = lNameMatrix.begin();
00650          itNameList != lNameMatrix.end(); ++itNameList) {
00651       // Retrieve the language code and locale
00652       // const LanguageCode_T& lLanguage = itNameList->first;
00653       const Names& lNames = itNameList->second;
00654
00655       // For a given language, retrieve the list of place names
00656       const NameList_T& lNameList = lNames.getNameList();
00657
00658       for (NameList_T::const_iterator itName = lNameList.begin();
00659            itName != lNameList.end(); ++itName) {
00660         const std::string& lName = *itName;
00661
00662         // Add the alternate name, which can be made of several words
00663         // (e.g., 'san francisco').
00664         if (lName.empty() == false) {
00665           // Create a list made of all the word combinations of the
00666           // initial string
00667           WordCombinationHolder lWordCombinationHolder (lName);
00668
00669           // Browse the list of unique strings (word combinations)
00670           const WordCombinationHolder::StringList_T& lStringList =
00671             lWordCombinationHolder._list;
00672           for (WordCombinationHolder::StringList_T::const_iterator itString =
00673                  lStringList.begin();
00674                itString != lStringList.end(); ++itString) {
00675             const std::string& lWordCombination = *itString;
00676             const std::string& lNormalisedWordCombination =
00677               iTransliterator.normalise (lWordCombination);
00678
00679             // Add that combination of words into the set of terms
00680             lStdTermSet.insert (lWordCombination);
00681             _spellingSet.insert (lWordCombination);
00682
00683             // Add the normalised combination of words into the set of terms
00684             lStdTermSet.insert (lNormalisedWordCombination);
00685             _spellingSet.insert (lNormalisedWordCombination);
00686           }
00687         }
00688       }
00689     }
00690
00691     // Insert (or replace) the newly created (or just altered) string set
00692     addTermSet (lPageRank, lWeightedTermSet);
00693
00694     // Insert (or replace) the newly created (or just altered) string set
00695     addTermSet (K_DEFAULT_INDEXING_STD_WEIGHT, lStdTermSet);
00696   }
00697
00698   // ////////////////////////////////////////////////////////////////////
00699   const Location& Place::completeLocation() {
00700
00701     NameList_T lNameList;
00702     LanguageCode_T K_DEFAULT_LANGUAGE_CODE ("en_US");
00703     const bool hasFoundNameList = getNameList (K_DEFAULT_LANGUAGE_CODE,
00704                                                lNameList);
00705
00706     if (hasFoundNameList == false) {
00707       //
00708       std::ostringstream errorStr;
00709       errorStr << "No list of names in (American) English ("
00710                << K_DEFAULT_LANGUAGE_CODE
00711                << " locale) can be found for the following place: "
00712                << toShortString();
00713       OPENTREP_LOG_ERROR (errorStr.str());
00714       // throw LanguageCodeNotDefinedInNameTableException (errorStr.str());
00715     }
00716     // assert (hasFoundNameList == true);
00717
00718     // Add extra matching locations, whenever they exist
00719     if (_extraPlaceList.empty() == false) {
00720       for (PlaceOrderedList_T::const_iterator itLoc = _extraPlaceList.begin();
00721            itLoc != _extraPlaceList.end(); ++itLoc) {
00722         const Place* lExtraPlace_ptr = *itLoc;
00723         assert (lExtraPlace_ptr != NULL);
00724
00725         // Add the extra matching location
00726         const Location& lExtraLocation = lExtraPlace_ptr->getLocation();
```

```
00727         _location.addExtraLocation (lExtraLocation);
00728       }
00729     }
00730
00731     // Add alternate matching locations, whenever they exist
00732     if (_alternatePlaceList.empty() == false) {
00733       for (PlaceOrderedList_T::const_iterator itLoc =
00734              _alternatePlaceList.begin();
00735            itLoc != _alternatePlaceList.end(); ++itLoc) {
00736         const Place* lAlternatePlace_ptr = *itLoc;
00737         assert (lAlternatePlace_ptr != NULL);
00738
00739         // Add the alternate matching location
00740         const Location& lAlternateLocation = lAlternatePlace_ptr->
      getLocation();
00741         _location.addAlternateLocation (lAlternateLocation);
00742       }
00743     }
00744
00745     return _location;
00746   }
00747 }
```

24.129 opentrep/bom/Place.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <set>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/NameMatrix.hpp>
#include <opentrep/Location.hpp>
#include <opentrep/bom/BomAbstract.hpp>
#include <opentrep/bom/PlaceList.hpp>
```

Classes

- class OPENTREP::Place

  Class modelling a place/POR (point of reference).

Namespaces

- OPENTREP

24.130 Place.hpp

```
00001 #ifndef __OPENTREP_BOM_PLACE_HPP
00002 #define __OPENTREP_BOM_PLACE_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <set>
00011 // OpenTrep
00012 #include <opentrep/OPENTREP_Types.hpp>
00013 #include <opentrep/NameMatrix.hpp>
00014 #include <opentrep/Location.hpp>
00015 #include <opentrep/bom/BomAbstract.hpp>
00016 #include <opentrep/bom/PlaceList.hpp>
00017
00018 namespace OPENTREP {
00019
00020   // Forward declarations
00021   class World;
00022   class PlaceHolder;
00023   struct OTransliterator;
00024
```

```
00028   class Place : public BomAbstract {
00029     friend class FacWorld;
00030     friend class FacPlace;
00031     friend class FacPlaceHolder;
00032     friend class DbaPlace;
00033   public:
00034     // ////////// Type definitions //////////
00039     typedef std::set<std::string> StringSet_T;
00040     typedef std::map<const Weight_T, StringSet_T> TermSetMap_T;
00041
00042
00043   public:
00044     // ///////////////// Getters ///////////////
00051     const Location& getLocation() const {
00052       return _location;
00053     }
00054
00058     const LocationKey& getKey() const {
00059       return _location.getKey();
00060     }
00061
00065     const IATACode_T& getIataCode() const {
00066       return _location.getIataCode();
00067     }
00068
00072     const IATAType& getIataType() const {
00073       return _location.getIataType();
00074     }
00075
00079     const GeonamesID_T& getGeonamesID() const {
00080       return _location.getGeonamesID();
00081     }
00082
00086     const IsGeonames_T& isGeonames() const {
00087       return _location.isGeonames();
00088     }
00089
00093     const ICAOCode_T& getIcaoCode() const {
00094       return _location.getIcaoCode();
00095     }
00096
00100     const FAACode_T& getFaaCode() const {
00101       return _location.getFaaCode();
00102     }
00103
00108     const CommonName_T& getCommonName() const {
00109       return _location.getCommonName();
00110     }
00111
00115     const ASCIIName_T& getAsciiName() const {
00116       return _location.getAsciiName();
00117     }
00118
00122     const AltNameShortListString_T&
      getAltNameShortListString() const {
00123       return _location.getAltNameShortListString();
00124     }
00125
00129     const EnvelopeID_T& getEnvelopeID() const {
00130       return _location.getEnvelopeID();
00131     }
00132
00136     const Date_T& getDateFrom() const {
00137       return _location.getDateFrom();
00138     }
00139
00143     const Date_T& getDateEnd() const {
00144       return _location.getDateEnd();
00145     }
00146
00150     const Comment_T& getComment() const {
00151       return _location.getComment();
00152     }
00153
00157     const CityCode_T& getCityCode() const {
00158       return _location.getCityCode();
00159     }
00160
00164     const CityUTFName_T& getCityUtfName() const {
00165       return _location.getCityUtfName();
00166     }
00167
00171     const CityASCIIName_T& getCityAsciiName() const {
00172       return _location.getCityAsciiName();
00173     }
00174
00178     const GeonamesID_T& getCityGeonamesID() const {
```

```
00179       return _location.getCityGeonamesID();
00180     }
00181
00185     const StateCode_T& getStateCode() const {
00186       return _location.getStateCode();
00187     }
00188
00192     const CountryCode_T& getCountryCode() const {
00193       return _location.getCountryCode();
00194     }
00195
00199     const AltCountryCode_T& getAltCountryCode() const {
00200       return _location.getAltCountryCode();
00201     }
00202
00206     const CountryName_T& getCountryName() const {
00207       return _location.getCountryName();
00208     }
00209
00213     const ContinentCode_T& getContinentCode() const {
00214       return _location.getContinentCode();
00215     }
00216
00220     const ContinentName_T& getContinentName() const {
00221       return _location.getContinentName();
00222     }
00223
00227     const TimeZone_T& getTimeZone() const {
00228       return _location.getTimeZone();
00229     }
00230
00234     const GMTOffset_T& getGMTOffset() const {
00235       return _location.getGMTOffset();
00236     }
00237
00241     const DSTOffset_T& getDSTOffset() const {
00242       return _location.getDSTOffset();
00243     }
00244
00248     const RawOffset_T& getRawOffset() const {
00249       return _location.getRawOffset();
00250     }
00251
00255     const Latitude_T& getLatitude() const {
00256       return _location.getLatitude();
00257     }
00258
00262     const Longitude_T& getLongitude() const {
00263       return _location.getLongitude();
00264     }
00265
00269     const FeatureClass_T& getFeatureClass() const {
00270       return _location.getFeatureClass();
00271     }
00272
00276     const FeatureCode_T& getFeatureCode() const {
00277       return _location.getFeatureCode();
00278     }
00279
00283     const Admin1Code_T& getAdmin1Code() const {
00284       return _location.getAdmin1Code();
00285     }
00286
00290     const Admin1UTFName_T& getAdmin1UtfName() const {
00291       return _location.getAdmin1UtfName();
00292     }
00293
00297     const Admin1ASCIIName_T& getAdmin1AsciiName() const {
00298       return _location.getAdmin1AsciiName();
00299     }
00300
00304     const Admin2Code_T& getAdmin2Code() const {
00305       return _location.getAdmin2Code();
00306     }
00307
00311     const Admin2UTFName_T& getAdmin2UtfName() const {
00312       return _location.getAdmin2UtfName();
00313     }
00314
00318     const Admin2ASCIIName_T& getAdmin2AsciiName() const {
00319       return _location.getAdmin2AsciiName();
00320     }
00321
00325     const Admin3Code_T& getAdmin3Code() const {
00326       return _location.getAdmin3Code();
00327     }
00328
```

```
00332     const Admin4Code_T& getAdmin4Code() const {
00333       return _location.getAdmin4Code();
00334     }
00335
00339     const Population_T& getPopulation() const {
00340       return _location.getPopulation();
00341     }
00342
00346     const Elevation_T& getElevation() const {
00347       return _location.getElevation();
00348     }
00349
00353     const GTopo30_T& getGTopo30() const {
00354       return _location.getGTopo30();
00355     }
00356
00360     const PageRank_T& getPageRank() const {
00361       return _location.getPageRank();
00362     }
00363
00367     const Date_T& getModificationDate() const {
00368       return _location.getModificationDate();
00369     }
00370
00374     const WikiLink_T& getWikiLink() const {
00375       return _location.getWikiLink();
00376     }
00377
00381     const NameMatrix& getNameMatrix() const {
00382       return _location.getNameMatrix();
00383     }
00384
00394     bool getNameList (const LanguageCode_T& iLanguageCode,
00395                       NameList_T& ioNameList) const {
00396       return _location.getNameList (iLanguageCode, ioNameList);
00397     }
00398
00402     const std::string& getOriginalKeywords() const {
00403       return _location.getOriginalKeywords();
00404     }
00405
00409     const std::string& getCorrectedKeywords() const {
00410       return _location.getCorrectedKeywords();
00411     }
00412
00416     const MatchingPercentage_T& getPercentage() const {
00417       return _location.getPercentage();
00418     }
00419
00423     const NbOfErrors_T& getEditDistance() const {
00424       return _location.getEditDistance();
00425     }
00426
00431     const NbOfErrors_T& getAllowableEditDistance() const {
00432       return _location.getAllowableEditDistance();
00433     }
00434
00438     const RawDataString_T& getRawDataString() const {
00439       return _location.getRawDataString();
00440     }
00441
00445     const XapianDocID_T& getDocID() const {
00446       return _docID;
00447     }
00448
00452     const PlaceOrderedList_T& getExtraPlaceList() const {
00453       return _extraPlaceList;
00454     }
00455
00459     const PlaceOrderedList_T& getAlternatePlaceList() const {
00460       return _alternatePlaceList;
00461     }
00462
00466     const TermSetMap_T& getTermSetMap() const {
00467       return _termSetMap;
00468     }
00469
00475     StringSet_T getTermSet (const Weight_T&) const;
00476
00480     const StringSet_T& getSpellingSet() const {
00481       return _spellingSet;
00482     }
00483
00487     const StringSet_T& getStemmingSet() const {
00488       return _stemmingSet;
00489     }
00490
```

```
00494     const StringSet_T& getSynonymSet() const {
00495       return _synonymSet;
00496     }
00497
00498
00499   public:
00500     // ////////////////// Setters /////////////////
00504     void setLocation (const Location& iLocation) {
00505       _location = iLocation;
00506     }
00507
00511     void setKey (const LocationKey& iKey) {
00512       _location.setKey (iKey);
00513     }
00514
00518     void setIataCode (const std::string& iIataCode) {
00519       _location.setIataCode (iIataCode);
00520     }
00521
00525     void setIataType (const std::string& iIATAType) {
00526       _location.setIataType (iIATAType);
00527     }
00528
00532     void setGeonamesID (const GeonamesID_T& iGeonamesID) {
00533       _location.setGeonamesID (iGeonamesID);
00534     }
00535
00539     void setIcaoCode (const std::string& iIcaoCode) {
00540       _location.setIcaoCode (iIcaoCode);
00541     }
00542
00546     void setFaaCode (const std::string& iFaaCode) {
00547       _location.setFaaCode (iFaaCode);
00548     }
00549
00554     void setCommonName (const std::string& iName) {
00555       _location.setCommonName (iName);
00556     }
00557
00561     void setAsciiName (const std::string& iName) {
00562       _location.setAsciiName (iName);
00563     }
00564
00568     void setAltNameShortListString (const std::string& iNameListString) {
00569       _location.setAltNameShortListString (iNameListString);
00570     }
00571
00575     void setEnvelopeID (const EnvelopeID_T& iEnvelopeID) {
00576       _location.setEnvelopeID (iEnvelopeID);
00577     }
00578
00582     void setDateFrom (const Date_T& iDate) {
00583       _location.setDateFrom (iDate);
00584     }
00585
00589     void setDateEnd (const Date_T& iDate) {
00590       _location.setDateEnd (iDate);
00591     }
00592
00596     void setComment (const std::string& iComment) {
00597       _location.setComment (iComment);
00598     }
00599
00603     void setCityCode (const std::string& iCityCode) {
00604       _location.setCityCode (iCityCode);
00605     }
00606
00610     void setCityUtfName (const std::string& iCityUtfName) {
00611       _location.setCityUtfName (iCityUtfName);
00612     }
00613
00617     void setCityAsciiName (const std::string& iCityAsciiName) {
00618       _location.setCityAsciiName (iCityAsciiName);
00619     }
00620
00624     void setCityGeonamesID (const GeonamesID_T& iGeonamesID) {
00625       _location.setCityGeonamesID (iGeonamesID);
00626     }
00627
00631     void setStateCode (const std::string& iStateCode) {
00632       _location.setStateCode (iStateCode);
00633     }
00634
00638     void setCountryCode (const std::string& iCountryCode) {
00639       _location.setCountryCode (iCountryCode);
00640     }
00641
```

```
00645     void setAltCountryCode (const std::string& iCountryCode) {
00646       _location.setAltCountryCode (iCountryCode);
00647     }
00648
00652     void setCountryName (const std::string& iCountryName) {
00653       _location.setCountryName (iCountryName);
00654     }
00655
00659     void setContinentCode (const std::string& iContinentCode) {
00660       _location.setContinentCode (iContinentCode);
00661     }
00662
00666     void setContinentName (const std::string& iContinentName) {
00667       _location.setContinentName (iContinentName);
00668     }
00669
00673     void setTimeZone (const std::string& iTimeZone) {
00674       _location.setTimeZone (iTimeZone);
00675     }
00676
00680     void setGMTOffset (const GMTOffset_T& iOffset) {
00681       _location.setGMTOffset (iOffset);
00682     }
00683
00687     void setDSTOffset (const DSTOffset_T& iOffset) {
00688       _location.setDSTOffset (iOffset);
00689     }
00690
00694     void setRawOffset (const RawOffset_T& iOffset) {
00695       _location.setRawOffset (iOffset);
00696     }
00697
00701     void setLatitude (const Latitude_T& iLatitude) {
00702       _location.setLatitude (iLatitude);
00703     }
00704
00708     void setLongitude (const Longitude_T& iLongitude) {
00709       _location.setLongitude (iLongitude);
00710     }
00711
00715     void setFeatureClass (const std::string& iFeatClass) {
00716       _location.setFeatureClass (iFeatClass);
00717     }
00718
00722     void setFeatureCode (const std::string& iFeatCode) {
00723       _location.setFeatureCode (iFeatCode);
00724     }
00725
00729     void setAdmin1Code (const std::string& iAdminCode) {
00730       _location.setAdmin1Code (iAdminCode);
00731     }
00732
00736     void setAdmin1UtfName (const std::string& iAdminName) {
00737       _location.setAdmin1UtfName (iAdminName);
00738     }
00739
00743     void setAdmin1AsciiName (const std::string& iAdminName) {
00744       _location.setAdmin1AsciiName (iAdminName);
00745     }
00746
00750     void setAdmin2Code (const std::string& iAdminCode) {
00751       _location.setAdmin2Code (iAdminCode);
00752     }
00753
00757     void setAdmin2UtfName (const std::string& iAdminName) {
00758       _location.setAdmin2UtfName (iAdminName);
00759     }
00760
00764     void setAdmin2AsciiName (const std::string& iAdminName) {
00765       _location.setAdmin2AsciiName (iAdminName);
00766     }
00767
00771     void setAdmin3Code (const std::string& iAdminCode) {
00772       _location.setAdmin3Code (iAdminCode);
00773     }
00774
00778     void setAdmin4Code (const std::string& iAdminCode) {
00779       _location.setAdmin4Code (iAdminCode);
00780     }
00781
00785     void setPopulation (const Population_T& iPopulation) {
00786       _location.setPopulation (iPopulation);
00787     }
00788
00792     void setElevation (const Elevation_T& iElevation) {
00793       _location.setElevation (iElevation);
00794     }
```

```
00795
00799     void setGTopo30 (const GTopo30_T& iGTopo30) {
00800       _location.setGTopo30 (iGTopo30);
00801     }
00802
00806     void setPageRank (const PageRank_T& iPageRank) {
00807       _location.setPageRank (iPageRank);
00808     }
00809
00813     void setModificationDate (const Date_T& iModDate) {
00814       _location.setModificationDate (iModDate);
00815     }
00816
00820     void setWikiLink (const std::string& iWikiLink) {
00821       _location.setWikiLink (iWikiLink);
00822     }
00823
00827     void setOriginalKeywords (const std::string& iOriginalKeywords) {
00828       _location.setOriginalKeywords (iOriginalKeywords);
00829     }
00830
00834     void setCorrectedKeywords (const std::string& iCorrectedKeywords) {
00835       _location.setCorrectedKeywords (iCorrectedKeywords);
00836     }
00837
00841     void setPercentage (const MatchingPercentage_T& iPercentage) {
00842       _location.setPercentage (iPercentage);
00843     }
00844
00848     void setEditDistance (const NbOfErrors_T& iEditDistance) {
00849       _location.setEditDistance (iEditDistance);
00850     }
00851
00856     void setAllowableEditDistance (const NbOfErrors_T&
    iAllowableEditDistance) {
00857       _location.setAllowableEditDistance (iAllowableEditDistance);
00858     }
00859
00863     void setRawDataString (const std::string& iRawDataString) {
00864       _location.setRawDataString (iRawDataString);
00865     }
00866
00870     void setDocID (const XapianDocID_T& iDocID) {
00871       _docID = iDocID;
00872     }
00873
00885     bool addTermSet (const Weight_T&, const StringSet_T&);
00886
00887
00888   public:
00889     // ////////// Setters in underlying names ////////
00896     void addName (const LanguageCode_T& iLanguageCode,
00897                   const std::string& iName) {
00898       _location.addName (iLanguageCode, iName);
00899     }
00900
00904     void resetMatrix() {
00905       _location.resetMatrix();
00906     }
00907
00911     void resetIndexSets();
00912
00913
00914   public:
00915     // /////////// Business methods /////////
00924     const Location& completeLocation();
00925
00952     void addNameToXapianSets (const Weight_T&,
00953                               const LocationName_T&, const
    FeatureCode_T&,
00954                               const CityUTFName_T&, const
    CityASCIIName_T&,
00955                               const Admin1UTFName_T&, const
    Admin1ASCIIName_T&,
00956                               const Admin2UTFName_T&, const
    Admin2ASCIIName_T&,
00957                               const StateCode_T&,
00958                               const CountryCode_T&, const
    CountryName_T&,
00959                               const ContinentName_T&, const
    OTransliterator&);
00960
00967     void buildIndexSets (const OTransliterator&);
00968
00979     void addNameToXapianSets (const Weight_T&, const std::string& iBaseName,
00980                               const FeatureCode_T&);
00981
```

```
00982
00983   public:
00984     // ********* Display methods ********
00990     void toStream (std::ostream&) const;
00991
00997     void fromStream (std::istream&);
00998
01002     std::string describeKey() const {
01003       return _location.describeKey();
01004     }
01005
01009     std::string describeShortKey() const {
01010       return _location.describeShortKey();
01011     }
01012
01016     std::string toString() const;
01017
01021     std::string toShortString() const;
01022
01026     std::string describeSets() const;
01027
01031     std::string display() const;
01032
01036     std::string shortDisplay() const;
01037
01038
01039   private:
01043     Place (const LocationKey&);
01044
01048     Place (const Location&);
01049
01053     Place();
01054
01058     Place (const Place&);
01059
01063     virtual ~Place();
01064
01065
01066   private:
01067     // ********************** Parent objects ********************
01071     World* _world;
01072
01077     PlaceHolder* _placeHolder;
01078
01083     Place* _mainPlace;
01084
01085
01086   private:
01087     // ****************** Attributes ************************
01093     Location _location;
01094
01095
01096   private:
01097     // ************** Full-text matching process support attributes **********
01104     XapianDocID_T _docID;
01105
01109     PlaceOrderedList_T _extraPlaceList;
01110
01114     PlaceOrderedList_T _alternatePlaceList;
01115
01121     TermSetMap_T _termSetMap;
01122
01127     StringSet_T _spellingSet;
01128
01133     StringSet_T _stemmingSet;
01134
01139     StringSet_T _synonymSet;
01140   };
01141
01142 }
01143 #endif // __OPENTREP_BOM_PLACE_HPP
```

24.131 opentrep/bom/PlaceHolder.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <opentrep/bom/Place.hpp>
#include <opentrep/bom/PlaceHolder.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.132 PlaceHolder.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // C
00005 #include <cassert>
00006 #include <sstream>
00007 // OpenTREP
00008 #include <opentrep/bom/Place.hpp>
00009 #include <opentrep/bom/PlaceHolder.hpp>
00010 #include <opentrep/service/Logger.hpp>
00011
00012 namespace OPENTREP {
00013
00014   // //////////////////////////////////////////////////////////////////////
00015   PlaceHolder::PlaceHolder () {
00016     init();
00017   }
00018
00019   // //////////////////////////////////////////////////////////////////////
00020   PlaceHolder::~PlaceHolder () {
00021   }
00022
00023   // //////////////////////////////////////////////////////////////////////
00024   void PlaceHolder::init () {
00025     _placeList.clear();
00026     _placeOrderedList.clear();
00027   }
00028
00029   // //////////////////////////////////////////////////////////////////////
00030   std::string PlaceHolder::describeShortKey() const {
00031     std::ostringstream oStr;
00032     return oStr.str();
00033   }
00034
00035   // //////////////////////////////////////////////////////////////////////
00036   std::string PlaceHolder::describeKey() const {
00037     return describeShortKey();
00038   }
00039
00040   // //////////////////////////////////////////////////////////////////////
00041   std::string PlaceHolder::toString() const {
00042     std::ostringstream oStr;
00043     oStr << describeShortKey() << std::endl;
00044
00045     for (PlaceOrderedList_T::const_iterator itPlace = _placeOrderedList.begin();
00046          itPlace != _placeOrderedList.end(); ++itPlace) {
00047       const Place* lPlace_ptr = *itPlace;
00048       assert (lPlace_ptr != NULL);
00049
00050       oStr << lPlace_ptr->toString() << std::endl;
00051     }
00052
00053     return oStr.str();
00054   }
00055
00056   // //////////////////////////////////////////////////////////////////////
00057   std::string PlaceHolder::toShortString() const {
00058     std::ostringstream oStr;
00059     oStr << describeShortKey() << std::endl;
00060
00061     for (PlaceOrderedList_T::const_iterator itPlace = _placeOrderedList.begin();
00062          itPlace != _placeOrderedList.end(); ++itPlace) {
00063       const Place* lPlace_ptr = *itPlace;
00064       assert (lPlace_ptr != NULL);
00065
00066       oStr << lPlace_ptr->toShortString() << std::endl;
00067     }
00068
00069     return oStr.str();
00070   }
00071
00072   // //////////////////////////////////////////////////////////////////////
00073   void PlaceHolder::toStream (std::ostream& ioOut) const {
00074     ioOut << toString();
00075   }
00076
00077   // //////////////////////////////////////////////////////////////////////
```

```
00078   void PlaceHolder::fromStream (std::istream& ioIn) {
00079   }
00080
00081   // ////////////////////////////////////////////////////////////////////
00082   void PlaceHolder::createLocations (LocationList_T&
      ioLocationList) const {
00083
00084     for (PlaceOrderedList_T::const_iterator itPlace = _placeOrderedList.begin();
00085          itPlace != _placeOrderedList.end(); ++itPlace) {
00086       Place* lPlace_ptr = *itPlace;
00087       assert (lPlace_ptr != NULL);
00088
00089       const Location& lLocation = lPlace_ptr->completeLocation();
00090       ioLocationList.push_back (lLocation);
00091     }
00092   }
00093
00094
00095 }
```

24.133 opentrep/bom/PlaceHolder.hpp File Reference

```
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/Location.hpp>
#include <opentrep/bom/BomAbstract.hpp>
#include <opentrep/bom/PlaceList.hpp>
```

Classes

- class OPENTREP::PlaceHolder

Namespaces

- OPENTREP

24.134 PlaceHolder.hpp

```
00001 #ifndef __OPENTREP_BOM_PLACEHOLDER_HPP
00002 #define __OPENTREP_BOM_PLACEHOLDER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // OpenTREP
00008 #include <opentrep/OPENTREP_Types.hpp>
00009 #include <opentrep/Location.hpp>
00010 #include <opentrep/bom/BomAbstract.hpp>
00011 #include <opentrep/bom/PlaceList.hpp>
00012
00013 namespace OPENTREP {
00014
00016   class PlaceHolder : public BomAbstract {
00017     friend class FacPlaceHolder;
00018   public:
00019     // ////////////// Getters /////////////
00021     const PlaceList_T& getPlaceList() const {
00022       return _placeList;
00023     }
00024
00025
00026     // ////////////// Setters /////////////
00027
00028
00029   public:
00030     // ////////// Business methods /////////
00034     void createLocations (LocationList_T&) const;
00035
00036
00037   public:
00038     // ////////// Display support methods /////////
00041     void toStream (std::ostream& ioOut) const;
00042
```

```
00045     void fromStream (std::istream& ioIn);
00046
00048     std::string toString() const;
00049
00051     std::string toShortString() const;
00052
00055     std::string describeKey() const;
00056
00059     std::string describeShortKey() const;
00060
00061
00062   private:
00063     // /////////////// Constructors and Destructors /////////////
00065     PlaceHolder ();
00067     PlaceHolder (const PlaceHolder&);
00069     ~PlaceHolder ();
00071     void init ();
00072
00073
00074   private:
00075     // ///////////////// Attributes /////////////////
00077     PlaceList_T _placeList;
00078
00081     PlaceOrderedList_T _placeOrderedList;
00082   };
00083
00084 }
00085 #endif // __OPENTREP_BOM_PLACEHOLDER_HPP
```

24.135 opentrep/bom/PlaceList.hpp File Reference

```
#include <map>
#include <list>
```

Namespaces

- OPENTREP

Typedefs

- typedef std::multimap
  < std::string, Place * > OPENTREP::PlaceList_T
- typedef std::list< Place * > OPENTREP::PlaceOrderedList_T

24.136 PlaceList.hpp

```
00001 #ifndef __OPENTREP_BOM_PLACELIST_HPP
00002 #define __OPENTREP_BOM_PLACELIST_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <map>
00009 #include <list>
00010
00011 namespace OPENTREP {
00012
00013   // Forward declarations
00014   class Place;
00015
00016   // ////////////// Type definitions ////////////////////
00017   // typedef std::size_t PlaceID_T;
00018   // typedef std::map<PlaceID_T, Place*> PlaceDirectList_T;
00019
00020   typedef std::multimap<std::string, Place*> PlaceList_T;
00021   typedef std::list<Place*> PlaceOrderedList_T;
00022
00023 }
00024 #endif // __OPENTREP_BOM_PLACELIST_HPP
```

24.137 opentrep/bom/PORParserHelper.cpp File Reference

```
#include <cassert>
#include <vector>
#include <fstream>
#include <boost/lexical_cast.hpp>
#include <opentrep/basic/BasParserTypes.hpp>
#include <opentrep/bom/PORParserHelper.hpp>
#include <opentrep/service/Logger.hpp>
```

Classes

- struct OPENTREP::PorParserHelper::LocationParser< Iterator >

Namespaces

- OPENTREP
- OPENTREP::PorParserHelper

Macros

- #define BOOST_SPIRIT_UNICODE

Variables

- OPENTREP::uint1_p_t OPENTREP::PorParserHelper::uint1_p
- OPENTREP::uint2_p_t OPENTREP::PorParserHelper::uint2_p
- OPENTREP::int1_2_p_t OPENTREP::PorParserHelper::int1_2_p
- OPENTREP::uint4_p_t OPENTREP::PorParserHelper::uint4_p
- OPENTREP::int1_5_p_t OPENTREP::PorParserHelper::int1_5_p
- OPENTREP::int0_5_p_t OPENTREP::PorParserHelper::int0_5_p
- OPENTREP::uint1_4_p_t OPENTREP::PorParserHelper::uint1_4_p
- OPENTREP::uint1_9_p_t OPENTREP::PorParserHelper::uint1_9_p
- OPENTREP::uint0_9_p_t OPENTREP::PorParserHelper::uint0_9_p
- OPENTREP::hour_p_t OPENTREP::PorParserHelper::hour_p
- OPENTREP::minute_p_t OPENTREP::PorParserHelper::minute_p
- OPENTREP::second_p_t OPENTREP::PorParserHelper::second_p
- OPENTREP::year_p_t OPENTREP::PorParserHelper::year_p
- OPENTREP::month_p_t OPENTREP::PorParserHelper::month_p
- OPENTREP::day_p_t OPENTREP::PorParserHelper::day_p
- OPENTREP::boolean_p_t OPENTREP::PorParserHelper::boolean_p

24.137.1 Macro Definition Documentation

24.137.1.1 #define BOOST_SPIRIT_UNICODE

Definition at line 11 of file PORParserHelper.cpp.

24.138 PORParserHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <vector>
00007 #include <fstream>
00008 // Boost
00009 #include <boost/lexical_cast.hpp>
00010 //#define BOOST_SPIRIT_DEBUG
00011 #define BOOST_SPIRIT_UNICODE
00012 // OpenTREP
00013 #include <opentrep/basic/BasParserTypes.hpp>
00014 #include <opentrep/bom/PORParserHelper.hpp>
00015 #include <opentrep/service/Logger.hpp>
00016
00017 namespace OPENTREP {
00018
00020   namespace bsq = boost::spirit::qi;
00021   //namespace bsa = boost::spirit::ascii;
00022   namespace bsu = boost::spirit::unicode;
00023   namespace bsc = boost::spirit::classic;
00024
00025   namespace PorParserHelper {
00026
00027     // ////////////////////////////////////////////////////////////////
00028     //  Semantic actions
00029     // ////////////////////////////////////////////////////////////////
00030
00031     ParserSemanticAction::ParserSemanticAction (
      Location& ioLocation)
00032       : _location (ioLocation) {
00033     }
00034
00035     // ////////////////////////////////////////////////////////////////
00036     storeIataCode::storeIataCode (Location& ioLocation)
00037       : ParserSemanticAction (ioLocation) {
00038     }
00039
00040     // ////////////////////////////////////////////////////////////////
00041     void storeIataCode::operator() (std::vector<uchar_t> iChar,
00042                                     bsq::unused_type, bsq::unused_type) const {
00043
00044       const std::string lIataCodeStr (iChar.begin(), iChar.end());
00045       const OPENTREP::IATACode_T lIataCode (lIataCodeStr);
00046       _location.setIataCode (lIataCode);
00047
00048       // Reset the values
00049       //_location.resetMatrix();
00050       //_location._itAltNameShortList.clear();
00051
00052       // DEBUG
00053       //OPENTREP_LOG_DEBUG ("IATA code: " << _location.getIataCode());
00054     }
00055
00056     // ////////////////////////////////////////////////////////////////
00057     storeIcaoCode::storeIcaoCode (Location& ioLocation)
00058       : ParserSemanticAction (ioLocation) {
00059     }
00060
00061     // ////////////////////////////////////////////////////////////////
00062     void storeIcaoCode::operator() (std::vector<uchar_t> iChar,
00063                                     bsq::unused_type, bsq::unused_type) const {
00064
00065       const std::string lIcaoCodeStr (iChar.begin(), iChar.end());
00066       const OPENTREP::ICAOCode_T lIcaoCode (lIcaoCodeStr);
00067       _location.setIcaoCode (lIcaoCode);
00068
00069       // DEBUG
00070       //OPENTREP_LOG_DEBUG ("ICAO code: " << _location.getIcaoCode());
00071     }
00072
00073     // ////////////////////////////////////////////////////////////////
00074     storeFaaCode::storeFaaCode (Location& ioLocation)
00075       : ParserSemanticAction (ioLocation) {
00076     }
00077
00078     // ////////////////////////////////////////////////////////////////
00079     void storeFaaCode::operator() (std::vector<uchar_t> iChar,
00080                                    bsq::unused_type, bsq::unused_type) const {
00081
00082       const std::string lFaaCodeStr (iChar.begin(), iChar.end());
00083       const OPENTREP::FAACode_T lFaaCode (lFaaCodeStr);
00084       _location.setFaaCode (lFaaCode);
00085
```

```
00086       // DEBUG
00087       //OPENTREP_LOG_DEBUG ("FAA code: " << _location.getFaaCode());
00088     }
00089
00090     // ////////////////////////////////////////////////////////////////////
00091     storeGeonamesID::storeGeonamesID (Location& ioLocation)
00092       : ParserSemanticAction (ioLocation) {
00093     }
00094
00095     // ////////////////////////////////////////////////////////////////////
00096     void storeGeonamesID::operator() (unsigned int iPorId,
00097                                       bsq::unused_type, bsq::unused_type) const {
00098       _location.setGeonamesID (iPorId);
00099
00100       // DEBUG
00101       //OPENTREP_LOG_DEBUG ("POR ID: " << _location.getPorID());
00102     }
00103
00104     // ////////////////////////////////////////////////////////////////////
00105     storeEnvelopeID::storeEnvelopeID (Location& ioLocation)
00106       : ParserSemanticAction (ioLocation) {
00107     }
00108
00109     // ////////////////////////////////////////////////////////////////////
00110     void storeEnvelopeID::operator() (unsigned int iEnvelopeID,
00111                                       bsq::unused_type, bsq::unused_type) const {
00112       _location.setEnvelopeID (iEnvelopeID);
00113
00114       // DEBUG
00115       //OPENTREP_LOG_DEBUG ("Envelope ID: " << _location.getEnvelopeID());
00116     }
00117
00118     // ////////////////////////////////////////////////////////////////////
00119     storeCommonName::storeCommonName (Location& ioLocation)
00120       : ParserSemanticAction (ioLocation) {
00121     }
00122
00123     // ////////////////////////////////////////////////////////////////////
00124     void storeCommonName::operator() (std::vector<uchar_t> iChar,
00125                                       bsq::unused_type, bsq::unused_type) const {
00126
00127       const std::string lCommonNameStr (iChar.begin(), iChar.end());
00128       const OPENTREP::CommonName_T lCommonName (lCommonNameStr);
00129       _location.setCommonName (lCommonName);
00130        // DEBUG
00131        //OPENTREP_LOG_DEBUG ("Common name: " << _location.getCommonName());
00132     }
00133
00134     // ////////////////////////////////////////////////////////////////////
00135     storeAsciiName::storeAsciiName (Location& ioLocation)
00136       : ParserSemanticAction (ioLocation) {
00137     }
00138
00139     // ////////////////////////////////////////////////////////////////////
00140     void storeAsciiName::operator() (std::vector<uchar_t> iChar,
00141                                      bsq::unused_type, bsq::unused_type) const {
00142
00143       const std::string lAsciiNameStr (iChar.begin(), iChar.end());
00144       const OPENTREP::ASCIIName_T lAsciiName (lAsciiNameStr);
00145       _location.setAsciiName (lAsciiName);
00146        // DEBUG
00147        //OPENTREP_LOG_DEBUG ("ASCII name: " << _location.getAsciiName());
00148     }
00149
00150     // ////////////////////////////////////////////////////////////////////
00151     storeAltNameShort::storeAltNameShort (
      Location& ioLocation)
00152       : ParserSemanticAction (ioLocation) {
00153     }
00154
00155     // ////////////////////////////////////////////////////////////////////
00156     void storeAltNameShort::operator() (std::vector<uchar_t> iChar,
00157                                         bsq::unused_type,
00158                                         bsq::unused_type)const {
00159
00160       const std::string lAltNameShortStr (iChar.begin(), iChar.end());
00161       const OPENTREP::AltNameShortListString_T lAltNameShort (
      lAltNameShortStr);
00162       _location._itAltNameShortList.push_back (lAltNameShort);
00163        // DEBUG
00164        //OPENTREP_LOG_DEBUG ("Alt name short: " << lAltNameShortStr);
00165     }
00166
00167     // ////////////////////////////////////////////////////////////////////
00168     storeAltNameShortListString::
00169     storeAltNameShortListString (Location& ioLocation)
00170       : ParserSemanticAction (ioLocation) {
```

```
00171     }
00172
00173     // ////////////////////////////////////////////////////////////////////
00174     void storeAltNameShortListString::
00175     operator() (bsq::unused_type, bsq::unused_type, bsq::unused_type)const {
00176       _location.consolidateAltNameShortListString();
00177        // DEBUG
00178        //OPENTREP_LOG_DEBUG ("Alternative name short list: " << _location.getAltNameShortListString());
00179     }
00180
00181     // ////////////////////////////////////////////////////////////////////
00182     storeTvlPORCode::storeTvlPORCode (Location& ioLocation)
00183       : ParserSemanticAction (ioLocation) {
00184     }
00185
00186     // ////////////////////////////////////////////////////////////////////
00187     void storeTvlPORCode::operator() (std::vector<uchar_t> iChar,
00188                                       bsq::unused_type,
00189                                       bsq::unused_type)const {
00190
00191       const std::string lTvlPORCodeStr (iChar.begin(), iChar.end());
00192       const OPENTREP::TvlPORListString_T lTvlPORCode (lTvlPORCodeStr);
00193       _location._itTvlPORList.push_back (lTvlPORCode);
00194        // DEBUG
00195        //OPENTREP_LOG_DEBUG ("Travel-related IATA code: " << lTvlPORCodeStr);
00196     }
00197
00198     // ////////////////////////////////////////////////////////////////////
00199     storeTvlPORListString::storeTvlPORListString (
    Location& ioLocation)
00200       : ParserSemanticAction (ioLocation) {
00201     }
00202
00203     // ////////////////////////////////////////////////////////////////////
00204     void storeTvlPORListString::operator() (bsq::unused_type,
    bsq::unused_type,
00205                                             bsq::unused_type)const {
00206       _location.consolidateTvlPORListString();
00207        // DEBUG
00208        //OPENTREP_LOG_DEBUG ("Travel-related POR list: " << _location.getTvlPORListString());
00209     }
00210
00211     // ////////////////////////////////////////////////////////////////////
00212     storeLatitude::storeLatitude (Location& ioLocation)
00213       : ParserSemanticAction (ioLocation) {
00214     }
00215
00216     // ////////////////////////////////////////////////////////////////////
00217     void storeLatitude::operator() (double iLatitude,
00218                                     bsq::unused_type, bsq::unused_type) const {
00219       _location.setLatitude (iLatitude);
00220       // DEBUG
00221       //OPENTREP_LOG_DEBUG ("Latitude: " << _location.getLatitude());
00222     }
00223
00224     // ////////////////////////////////////////////////////////////////////
00225     storeLongitude::storeLongitude (Location& ioLocation)
00226       : ParserSemanticAction (ioLocation) {
00227     }
00228
00229     // ////////////////////////////////////////////////////////////////////
00230     void storeLongitude::operator() (double iLongitude,
00231                                      bsq::unused_type, bsq::unused_type) const {
00232       _location.setLongitude (iLongitude);
00233
00234       // DEBUG
00235       //OPENTREP_LOG_DEBUG ("Longitude: " << _location.getLongitude());
00236     }
00237
00238     // ////////////////////////////////////////////////////////////////////
00239     storeFeatureClass::storeFeatureClass (
    Location& ioLocation)
00240       : ParserSemanticAction (ioLocation) {
00241     }
00242
00243     // ////////////////////////////////////////////////////////////////////
00244     void storeFeatureClass::operator() (std::vector<uchar_t> iChar,
00245                                         bsq::unused_type,
00246                                         bsq::unused_type) const {
00247       const std::string lFeatClassStr (iChar.begin(), iChar.end());
00248       const FeatureClass_T lFeatClass (lFeatClassStr);
00249       _location.setFeatureClass (lFeatClass);
00250
00251       // DEBUG
00252       //OPENTREP_LOG_DEBUG ("Feature class: " << _location.getFeatureClass());
00253     }
00254
```

```
00255     // ////////////////////////////////////////////////////////////////////
00256     storeFeatureCode::storeFeatureCode (
      Location& ioLocation)
00257       : ParserSemanticAction (ioLocation) {
00258     }
00259
00260     // ////////////////////////////////////////////////////////////////////
00261     void storeFeatureCode::operator() (std::vector<uchar_t> iChar,
00262                                        bsq::unused_type,
00263                                        bsq::unused_type) const {
00264       const std::string lFeatCodeStr (iChar.begin(), iChar.end());
00265       const FeatureClass_T lFeatCode (lFeatCodeStr);
00266       _location.setFeatureCode (lFeatCode);
00267
00268       // DEBUG
00269       //OPENTREP_LOG_DEBUG ("Feature code: " << _location.getFeatureCode());
00270     }
00271
00272     // ////////////////////////////////////////////////////////////////////
00273     storePageRank::storePageRank (Location& ioLocation)
00274       : ParserSemanticAction (ioLocation) {
00275     }
00276
00277     // ////////////////////////////////////////////////////////////////////
00278     void storePageRank::operator() (double iPageRank,
00279                                     bsq::unused_type, bsq::unused_type) const {
00280       _location.setPageRank (100.0 * iPageRank);
00281
00282       // DEBUG
00283       //OPENTREP_LOG_DEBUG ("PageRank: " << _location.getPageRank());
00284     }
00285
00286     // ////////////////////////////////////////////////////////////////////
00287     storeDateFrom::storeDateFrom (Location& ioLocation)
00288       : ParserSemanticAction (ioLocation) {
00289     }
00290
00291     // ////////////////////////////////////////////////////////////////////
00292     void storeDateFrom::operator() (bsq::unused_type,
00293                                     bsq::unused_type, bsq::unused_type) const {
00294       /*
00295       const OPENTREP::Date_T& lDateFrom = _location.calculateDate();
00296       _location.setDateFrom (lDateFrom);
00297       */
00298
00299       // DEBUG
00300       //OPENTREP_LOG_DEBUG ("Date from: " << _location.getDateFrom());
00301     }
00302
00303     // ////////////////////////////////////////////////////////////////////
00304     storeDateUntil::storeDateUntil (Location& ioLocation)
00305       : ParserSemanticAction (ioLocation) {
00306     }
00307
00308     // ////////////////////////////////////////////////////////////////////
00309     void storeDateUntil::operator() (bsq::unused_type,
00310                                      bsq::unused_type, bsq::unused_type) const {
00311       /*
00312       const OPENTREP::Date_T& lDateUntil = _location.calculateDate();
00313       _location.setDateUntil (lDateUntil);
00314       */
00315
00316       // DEBUG
00317       //OPENTREP_LOG_DEBUG ("Date until: " << _location.getDateUntil());
00318     }
00319
00320     // ////////////////////////////////////////////////////////////////////
00321     storeComments::storeComments (Location& ioLocation)
00322       : ParserSemanticAction (ioLocation) {
00323     }
00324
00325     // ////////////////////////////////////////////////////////////////////
00326     void storeComments::operator() (std::vector<uchar_t> iChar,
00327                                     bsq::unused_type, bsq::unused_type) const {
00328       /*
00329       const std::string lCommentsStr (iChar.begin(), iChar.end());
00330       const CountryCode_T lComments (lCommentsStr);
00331       _location.setComments (lComments);
00332       */
00333
00334       // DEBUG
00335       //OPENTREP_LOG_DEBUG ("Comments: " << _location.getComments());
00336     }
00337
00338     // ////////////////////////////////////////////////////////////////////
00339     storeCountryCode::storeCountryCode (
      Location& ioLocation)
```

```
00340       : ParserSemanticAction (ioLocation) {
00341     }
00342
00343     // ////////////////////////////////////////////////////////////////////
00344     void storeCountryCode::operator() (std::vector<uchar_t> iChar,
00345                                        bsq::unused_type,
00346                                        bsq::unused_type) const {
00347       const std::string lCountryCodeStr (iChar.begin(), iChar.end());
00348       const CountryCode_T lCountryCode (lCountryCodeStr);
00349       _location.setCountryCode (lCountryCode);
00350       // DEBUG
00351       //OPENTREP_LOG_DEBUG ("Country code: " << _location.getCountryCode());
00352     }
00353
00354     // ////////////////////////////////////////////////////////////////////
00355     storeAltCountryCode::storeAltCountryCode (
      Location& ioLocation)
00356       : ParserSemanticAction (ioLocation) {
00357     }
00358
00359     // ////////////////////////////////////////////////////////////////////
00360     void storeAltCountryCode::operator() (std::vector<uchar_t> iChar,
00361                                           bsq::unused_type,
00362                                           bsq::unused_type) const {
00363       const std::string lCountryCodeStr (iChar.begin(), iChar.end());
00364       const CountryCode_T lCountryCode (lCountryCodeStr);
00365       _location.setAltCountryCode (lCountryCode);
00366       // DEBUG
00367       //OPENTREP_LOG_DEBUG ("Alt country code: " << _location.getAltCountryCode());
00368     }
00369
00370     // ////////////////////////////////////////////////////////////////////
00371     storeCountryName::storeCountryName (
      Location& ioLocation)
00372       : ParserSemanticAction (ioLocation) {
00373     }
00374
00375     // ////////////////////////////////////////////////////////////////////
00376     void storeCountryName::operator() (std::vector<uchar_t> iChar,
00377                                        bsq::unused_type,
00378                                        bsq::unused_type) const {
00379       const std::string lCountryNameStr (iChar.begin(), iChar.end());
00380       const CountryName_T lCountryName (lCountryNameStr);
00381       _location.setCountryName (lCountryName);
00382       // DEBUG
00383       //OPENTREP_LOG_DEBUG ("Country name: " << _location.getCountryName());
00384     }
00385
00386     // ////////////////////////////////////////////////////////////////////
00387     storeContinentName::storeContinentName (
      Location& ioLocation)
00388       : ParserSemanticAction (ioLocation) {
00389     }
00390
00391     // ////////////////////////////////////////////////////////////////////
00392     void storeContinentName::operator() (std::vector<uchar_t> iChar,
00393                                          bsq::unused_type,
00394                                          bsq::unused_type) const {
00395       const std::string lContinentNameStr (iChar.begin(), iChar.end());
00396       const ContinentName_T lContinentName (lContinentNameStr);
00397       _location.setContinentName (lContinentName);
00398       // DEBUG
00399       //OPENTREP_LOG_DEBUG ("Continent name: " << _location.getContinentName());
00400     }
00401
00402     // ////////////////////////////////////////////////////////////////////
00403     storeAdm1Code::storeAdm1Code (Location& ioLocation)
00404       : ParserSemanticAction (ioLocation) {
00405     }
00406
00407     // ////////////////////////////////////////////////////////////////////
00408     void storeAdm1Code::operator() (std::vector<uchar_t> iChar,
00409                                     bsq::unused_type,
00410                                     bsq::unused_type) const {
00411       const std::string lAdmCodeStr (iChar.begin(), iChar.end());
00412       const Admin1Code_T lAdmCode (lAdmCodeStr);
00413       _location.setAdmin1Code (lAdmCode);
00414       // DEBUG
00415       //OPENTREP_LOG_DEBUG ("Adm1 code: " << _location.getAdmin1Code());
00416     }
00417
00418     // ////////////////////////////////////////////////////////////////////
00419     storeAdm1UtfName::storeAdm1UtfName (
      Location& ioLocation)
00420       : ParserSemanticAction (ioLocation) {
00421     }
00422
```

```
00423     // ////////////////////////////////////////////////////////////////////
00424     void storeAdm1UtfName::operator() (std::vector<uchar_t> iChar,
00425                                        bsq::unused_type,
00426                                        bsq::unused_type) const {
00427       const std::string lAdmNameStr (iChar.begin(), iChar.end());
00428       const Admin1UTFName_T lAdmName (lAdmNameStr);
00429       _location.setAdmin1UtfName (lAdmName);
00430       // DEBUG
00431       //OPENTREP_LOG_DEBUG ("Adm1 UTF8 name: " << _location.getAdmin1UtfName());
00432     }
00433
00434     // ////////////////////////////////////////////////////////////////////
00435     storeAdm1AsciiName::storeAdm1AsciiName (
      Location& ioLocation)
00436       : ParserSemanticAction (ioLocation) {
00437     }
00438
00439     // ////////////////////////////////////////////////////////////////////
00440     void storeAdm1AsciiName::operator() (std::vector<uchar_t> iChar,
00441                                          bsq::unused_type,
00442                                          bsq::unused_type) const {
00443       const std::string lAdmNameStr (iChar.begin(), iChar.end());
00444       const Admin1ASCIIName_T lAdmName (lAdmNameStr);
00445       _location.setAdmin1AsciiName (lAdmName);
00446       // DEBUG
00447       //OPENTREP_LOG_DEBUG("Adm1 ASCII name: "<< _location.getAdmin1AsciiName());
00448     }
00449
00450     // ////////////////////////////////////////////////////////////////////
00451     storeAdm2Code::storeAdm2Code (Location& ioLocation)
00452       : ParserSemanticAction (ioLocation) {
00453     }
00454
00455     // ////////////////////////////////////////////////////////////////////
00456     void storeAdm2Code::operator() (std::vector<uchar_t> iChar,
00457                                     bsq::unused_type,
00458                                     bsq::unused_type) const {
00459       const std::string lAdmCodeStr (iChar.begin(), iChar.end());
00460       const Admin2Code_T lAdmCode (lAdmCodeStr);
00461       _location.setAdmin2Code (lAdmCode);
00462       // DEBUG
00463       //OPENTREP_LOG_DEBUG ("Adm2 code: " << _location.getAdmin2Code());
00464     }
00465
00466     // ////////////////////////////////////////////////////////////////////
00467     storeAdm2UtfName::storeAdm2UtfName (
      Location& ioLocation)
00468       : ParserSemanticAction (ioLocation) {
00469     }
00470
00471     // ////////////////////////////////////////////////////////////////////
00472     void storeAdm2UtfName::operator() (std::vector<uchar_t> iChar,
00473                                        bsq::unused_type,
00474                                        bsq::unused_type) const {
00475       const std::string lAdmNameStr (iChar.begin(), iChar.end());
00476       const Admin2UTFName_T lAdmName (lAdmNameStr);
00477       _location.setAdmin2UtfName (lAdmName);
00478       // DEBUG
00479       //OPENTREP_LOG_DEBUG ("Adm2 UTF8 name: " << _location.getAdmin2UtfName());
00480     }
00481
00482     // ////////////////////////////////////////////////////////////////////
00483     storeAdm2AsciiName::storeAdm2AsciiName (
      Location& ioLocation)
00484       : ParserSemanticAction (ioLocation) {
00485     }
00486
00487     // ////////////////////////////////////////////////////////////////////
00488     void storeAdm2AsciiName::operator() (std::vector<uchar_t> iChar,
00489                                          bsq::unused_type,
00490                                          bsq::unused_type) const {
00491       const std::string lAdmNameStr (iChar.begin(), iChar.end());
00492       const Admin2ASCIIName_T lAdmName (lAdmNameStr);
00493       _location.setAdmin2AsciiName (lAdmName);
00494       // DEBUG
00495       //OPENTREP_LOG_DEBUG("Adm2 ASCII name: "<< _location.getAdmin2AsciiName());
00496     }
00497
00498     // ////////////////////////////////////////////////////////////////////
00499     storeAdm3Code::storeAdm3Code (Location& ioLocation)
00500       : ParserSemanticAction (ioLocation) {
00501     }
00502
00503     // ////////////////////////////////////////////////////////////////////
00504     void storeAdm3Code::operator() (std::vector<uchar_t> iChar,
00505                                     bsq::unused_type,
00506                                     bsq::unused_type) const {
```

```
00507       const std::string lAdmCodeStr (iChar.begin(), iChar.end());
00508       const Admin3Code_T lAdmCode (lAdmCodeStr);
00509       _location.setAdmin3Code (lAdmCode);
00510       // DEBUG
00511       //OPENTREP_LOG_DEBUG ("Adm3 code: " << _location.getAdmin3Code());
00512     }
00513
00514     // ////////////////////////////////////////////////////////////////////
00515     storeAdm4Code::storeAdm4Code (Location& ioLocation)
00516       : ParserSemanticAction (ioLocation) {
00517     }
00518
00519     // ////////////////////////////////////////////////////////////////////
00520     void storeAdm4Code::operator() (std::vector<uchar_t> iChar,
00521                                     bsq::unused_type,
00522                                     bsq::unused_type) const {
00523       const std::string lAdmCodeStr (iChar.begin(), iChar.end());
00524       const Admin4Code_T lAdmCode (lAdmCodeStr);
00525       _location.setAdmin4Code (lAdmCode);
00526       // DEBUG
00527       //OPENTREP_LOG_DEBUG ("Adm4 code: " << _location.getAdmin4Code());
00528     }
00529
00530     // ////////////////////////////////////////////////////////////////////
00531     storePopulation::storePopulation (Location& ioLocation)
00532       : ParserSemanticAction (ioLocation) {
00533     }
00534
00535     // ////////////////////////////////////////////////////////////////////
00536     void storePopulation::operator() (unsigned int iPopulation,
00537                                       bsq::unused_type, bsq::unused_type) const {
00538       _location.setPopulation (iPopulation);
00539       // DEBUG
00540       //OPENTREP_LOG_DEBUG ("Population: " << _location.getPopulation());
00541     }
00542
00543     // ////////////////////////////////////////////////////////////////////
00544     storeElevation::storeElevation (Location& ioLocation)
00545       : ParserSemanticAction (ioLocation) {
00546     }
00547
00548     // ////////////////////////////////////////////////////////////////////
00549     void storeElevation::operator() (int iElevation,
00550                                      bsq::unused_type, bsq::unused_type) const {
00551       _location.setElevation (iElevation);
00552       // DEBUG
00553       //OPENTREP_LOG_DEBUG ("Elevation: " << _location.getElevation());
00554     }
00555
00556     // ////////////////////////////////////////////////////////////////////
00557     storeGTopo30::storeGTopo30 (Location& ioLocation)
00558       : ParserSemanticAction (ioLocation) {
00559     }
00560
00561     // ////////////////////////////////////////////////////////////////////
00562     void storeGTopo30::operator() (int iGTopo30,
00563                                    bsq::unused_type, bsq::unused_type) const {
00564       _location.setGTopo30 (iGTopo30);
00565       // DEBUG
00566       //OPENTREP_LOG_DEBUG ("GTopo30: " << _location.getGTopo30());
00567     }
00568
00569     // ////////////////////////////////////////////////////////////////////
00570     storeTimeZone::storeTimeZone (Location& ioLocation)
00571       : ParserSemanticAction (ioLocation) {
00572     }
00573
00574     // ////////////////////////////////////////////////////////////////////
00575     void storeTimeZone::operator() (std::vector<uchar_t> iChar,
00576                                     bsq::unused_type,
00577                                     bsq::unused_type) const {
00578       const std::string lTimeZoneStr (iChar.begin(), iChar.end());
00579       const TimeZone_T lTimeZone (lTimeZoneStr);
00580       _location.setTimeZone (lTimeZone);
00581       // DEBUG
00582       //OPENTREP_LOG_DEBUG ("Time-zone code: " << _location.getTimeZone());
00583     }
00584
00585     // ////////////////////////////////////////////////////////////////////
00586     storeGMTOffset::storeGMTOffset (Location& ioLocation)
00587       : ParserSemanticAction (ioLocation) {
00588     }
00589
00590     // ////////////////////////////////////////////////////////////////////
00591     void storeGMTOffset::operator() (float iOffset,
00592                                      bsq::unused_type, bsq::unused_type) const {
00593       _location.setGMTOffset (iOffset);
```

```
00594       // DEBUG
00595       //OPENTREP_LOG_DEBUG ("GMT offset: " << _location.getGMTOffset());
00596     }
00597
00598     // ////////////////////////////////////////////////////////////////////
00599     storeDSTOffset::storeDSTOffset (Location& ioLocation)
00600       : ParserSemanticAction (ioLocation) {
00601     }
00602
00603     // ////////////////////////////////////////////////////////////////////
00604     void storeDSTOffset::operator() (float iOffset,
00605                                      bsq::unused_type, bsq::unused_type) const {
00606       _location.setDSTOffset (iOffset);
00607       // DEBUG
00608       //OPENTREP_LOG_DEBUG ("DST offset: " << _location.getDSTOffset());
00609     }
00610
00611     // ////////////////////////////////////////////////////////////////////
00612     storeRawOffset::storeRawOffset (Location& ioLocation)
00613       : ParserSemanticAction (ioLocation) {
00614     }
00615
00616     // ////////////////////////////////////////////////////////////////////
00617     void storeRawOffset::operator() (float iOffset,
00618                                      bsq::unused_type, bsq::unused_type) const {
00619       _location.setRawOffset (iOffset);
00620       // DEBUG
00621       //OPENTREP_LOG_DEBUG ("Raw offset: " << _location.getRawOffset());
00622     }
00623
00624     // ////////////////////////////////////////////////////////////////////
00625     storeModDate::storeModDate (Location& ioLocation)
00626       : ParserSemanticAction (ioLocation) {
00627     }
00628
00629     // ////////////////////////////////////////////////////////////////////
00630     void storeModDate::operator() (bsq::unused_type,
00631                                    bsq::unused_type, bsq::unused_type) const {
00632       const OPENTREP::Date_T& lModDate = _location.
      calculateDate();
00633       _location.setModificationDate (lModDate);
00634       // DEBUG
00635       //OPENTREP_LOG_DEBUG ("Modification date: " << _location.getModificationDate());
00636     }
00637
00638     // ////////////////////////////////////////////////////////////////////
00639     storeCityCode::storeCityCode (Location& ioLocation)
00640       : ParserSemanticAction (ioLocation) {
00641     }
00642
00643     // ////////////////////////////////////////////////////////////////////
00644     void storeCityCode::operator() (std::vector<uchar_t> iChar,
00645                                     bsq::unused_type, bsq::unused_type) const {
00646
00647       const std::string lCityCodeStr (iChar.begin(), iChar.end());
00648       const OPENTREP::CityCode_T lCityCode (lCityCodeStr);
00649       _location.setCityCode (lCityCode);
00650       // DEBUG
00651       //OPENTREP_LOG_DEBUG ("City code: " << _location.getCityCode());
00652     }
00653
00654     // ////////////////////////////////////////////////////////////////////
00655     storeCityUtfName::storeCityUtfName (
      Location& ioLocation)
00656       : ParserSemanticAction (ioLocation) {
00657     }
00658
00659     // ////////////////////////////////////////////////////////////////////
00660     void storeCityUtfName::operator() (std::vector<uchar_t> iChar,
00661                                        bsq::unused_type,
00662                                        bsq::unused_type) const {
00663
00664       const std::string lCityUtfNameStr (iChar.begin(), iChar.end());
00665       const OPENTREP::CityUTFName_T lCityUtfName (lCityUtfNameStr);
00666       _location.setCityUtfName (lCityUtfName);
00667       // DEBUG
00668       //OPENTREP_LOG_DEBUG ("City UTF8 name: " << _location.getCityUtfName());
00669     }
00670
00671     // ////////////////////////////////////////////////////////////////////
00672     storeCityAsciiName::storeCityAsciiName (
      Location& ioLocation)
00673       : ParserSemanticAction (ioLocation) {
00674     }
00675
00676     // ////////////////////////////////////////////////////////////////////
00677     void storeCityAsciiName::operator() (std::vector<uchar_t> iChar,
```

```
00678                                       bsq::unused_type,
00679                                       bsq::unused_type) const {
00680
00681       const std::string lCityAsciiNameStr (iChar.begin(), iChar.end());
00682       const OPENTREP::CityASCIIName_T lCityAsciiName (lCityAsciiNameStr);
00683       _location.setCityAsciiName (lCityAsciiName);
00684        // DEBUG
00685        //OPENTREP_LOG_DEBUG("City ASCII name: " << _location.getCityAsciiName());
00686     }
00687
00688     // ////////////////////////////////////////////////////////////////////
00689     storeCityGeonamesID::storeCityGeonamesID (
      Location& ioLocation)
00690       : ParserSemanticAction (ioLocation) {
00691     }
00692
00693     // ////////////////////////////////////////////////////////////////////
00694     void storeCityGeonamesID::operator() (unsigned int iCtyId,
00695                                           bsq::unused_type,
00696                                           bsq::unused_type) const {
00697
00698       _location.setCityGeonamesID (iCtyId);
00699        // DEBUG
00700        //OPENTREP_LOG_DEBUG ("City Geonames ID: " << _location.getCityGeonamesID());
00701     }
00702
00703     // ////////////////////////////////////////////////////////////////////
00704     storeStateCode::storeStateCode (Location& ioLocation)
00705       : ParserSemanticAction (ioLocation) {
00706     }
00707
00708     // ////////////////////////////////////////////////////////////////////
00709     void storeStateCode::operator() (std::vector<uchar_t> iChar,
00710                                      bsq::unused_type, bsq::unused_type) const {
00711
00712       const std::string lStateCodeStr (iChar.begin(), iChar.end());
00713       const OPENTREP::StateCode_T lStateCode (lStateCodeStr);
00714       _location.setStateCode (lStateCode);
00715        // DEBUG
00716        //OPENTREP_LOG_DEBUG ("State code: " << _location.getStateCode());
00717     }
00718
00719     // ////////////////////////////////////////////////////////////////////
00720     storePORType::storePORType (Location& ioLocation)
00721       : ParserSemanticAction (ioLocation) {
00722     }
00723
00724     // ////////////////////////////////////////////////////////////////////
00725     void storePORType::operator() (std::vector<uchar_t> iChar,
00726                                    bsq::unused_type, bsq::unused_type) const {
00727       const std::string lIATATypeStr (iChar.begin(), iChar.end());
00728       const IATAType lIATAType (lIATATypeStr);
00729       _location.setIataType (lIATAType);
00730       // DEBUG
00731       //OPENTREP_LOG_DEBUG ("IATA type: " << _location.getIataType());
00732     }
00733
00734     // ////////////////////////////////////////////////////////////////////
00735     storeWikiLink::storeWikiLink (Location& ioLocation)
00736       : ParserSemanticAction (ioLocation) {
00737     }
00738
00739     // ////////////////////////////////////////////////////////////////////
00740     void storeWikiLink::operator() (std::vector<uchar_t> iChar,
00741                                     bsq::unused_type, bsq::unused_type) const {
00742
00743       const std::string lWikiLinkStr (iChar.begin(), iChar.end());
00744       const OPENTREP::WikiLink_T lWikiLink (lWikiLinkStr);
00745       _location.setWikiLink (lWikiLink);
00746        // DEBUG
00747        // OPENTREP_LOG_DEBUG ("Wiki link: " << _location.getWikiLink());
00748     }
00749
00750     // ////////////////////////////////////////////////////////////////////
00751     storeAltLangCodeFull::storeAltLangCodeFull (
      Location& ioLocation)
00752       : ParserSemanticAction (ioLocation) {
00753     }
00754
00755     // ////////////////////////////////////////////////////////////////////
00756     void storeAltLangCodeFull::operator() (std::vector<uchar_t> iChar,
00757                                            bsq::unused_type, bsq::unused_type) const {
00758
00759       const std::string lAltLangCodeStr (iChar.begin(), iChar.end());
00760       const OPENTREP::LanguageCode_T lAltLangCode (lAltLangCodeStr);
00761       _location._itLanguageCode = lAltLangCode;
00762        // DEBUG
```

```
00763         //OPENTREP_LOG_DEBUG ("Alt lang full code: " << _location._itLanguageCode);
00764       }
00765
00766       // ////////////////////////////////////////////////////////////////////
00767       storeAltLangCode2Char::storeAltLangCode2Char (
      Location& ioLocation)
00768         : ParserSemanticAction (ioLocation) {
00769       }
00770
00771       // ////////////////////////////////////////////////////////////////////
00772       void storeAltLangCode2Char::operator() (std::vector<uchar_t> iChar,
00773                                               bsq::unused_type, bsq::unused_type) const {
00774
00775         const std::string lAltLangCodeStr (iChar.begin(), iChar.end());
00776         _location._itLangCode2Char = lAltLangCodeStr;
00777         _location._itLangCodeExt = "";
00778         _location._itLangCodeHist = "";
00779         // DEBUG
00780         //OPENTREP_LOG_DEBUG ("Alt lang 2-char code: " << _location._itLangCode2Char);
00781       }
00782
00783       // ////////////////////////////////////////////////////////////////////
00784       storeAltLangCodeExt::storeAltLangCodeExt (
      Location& ioLocation)
00785         : ParserSemanticAction (ioLocation) {
00786       }
00787
00788       // ////////////////////////////////////////////////////////////////////
00789       void storeAltLangCodeExt::operator() (std::vector<uchar_t> iChar,
00790                                             bsq::unused_type, bsq::unused_type) const {
00791
00792         const std::string lAltLangCodeStr (iChar.begin(), iChar.end());
00793         std::ostringstream oStr;
00794         oStr << _location._itLangCode2Char << "-" << lAltLangCodeStr;
00795         _location._itLangCodeExt = oStr.str();
00796         // DEBUG
00797         //OPENTREP_LOG_DEBUG ("Alt lang 2-char code: " << _location._itLangCodeExt);
00798       }
00799
00800       // ////////////////////////////////////////////////////////////////////
00801       storeAltLangCodeHist::storeAltLangCodeHist (
      Location& ioLocation)
00802         : ParserSemanticAction (ioLocation) {
00803       }
00804
00805       // ////////////////////////////////////////////////////////////////////
00806       void storeAltLangCodeHist::operator() (std::vector<uchar_t> iChar,
00807                                              bsq::unused_type, bsq::unused_type) const {
00808
00809         const std::string lAltLangCodeStr (iChar.begin(), iChar.end());
00810         std::ostringstream oStr;
00811         oStr << _location._itLangCode2Char << "_" << lAltLangCodeStr;
00812         _location._itLangCodeHist = oStr.str();
00813         // DEBUG
00814         //OPENTREP_LOG_DEBUG ("Alt lang 2-char code: " << _location._itLangCodeHist);
00815       }
00816
00817       // ////////////////////////////////////////////////////////////////////
00818       storeAltName::storeAltName (Location& ioLocation)
00819         : ParserSemanticAction (ioLocation) {
00820       }
00821
00822       // ////////////////////////////////////////////////////////////////////
00823       void storeAltName::operator() (std::vector<uchar_t> iChar,
00824                                      bsq::unused_type, bsq::unused_type) const {
00825
00826         const std::string lAltNameStr (iChar.begin(), iChar.end());
00827         const OPENTREP::AltNameShortListString_T lAltName (lAltNameStr);
00828         //_location.addName (_location._itLanguageCodeNum, lAltName);
00829         _location.addName (_location._itLanguageCode, lAltName);
00830         // Reset the values
00831         //_location._itLanguageCodeNum = OPENTREP::Language::LAST_VALUE;
00832         _location._itLanguageCode = LanguageCode_T ("");
00833         // DEBUG
00834         //OPENTREP_LOG_DEBUG ("Alt name: " << _location.getAltNameShortList());
00835       }
00836
00837
00838       // ////////////////////////////////////////////////////////////////////
00839       doEndPor::doEndPor (Location& ioLocation)
00840         : ParserSemanticAction (ioLocation) {
00841       }
00842
00843       // ////////////////////////////////////////////////////////////////////
00844       void doEndPor::operator() (bsq::unused_type,
00845                                  bsq::unused_type, bsq::unused_type) const {
00846         // DEBUG
```

```
00847       //OPENTREP_LOG_DEBUG ("Do End. Location structure: " << _location);
00848     }
00849
00850     // ////////////////////////////////////////////////////////////////////
00851     //
00852     //  Utility Parsers
00853     //
00854     // ////////////////////////////////////////////////////////////////////
00855
00857     OPENTREP::uint1_p_t uint1_p;
00858
00860     OPENTREP::uint2_p_t uint2_p;
00861
00863     OPENTREP::int1_2_p_t int1_2_p;
00864
00866     OPENTREP::uint4_p_t uint4_p;
00867
00869     OPENTREP::int1_5_p_t int1_5_p;
00870     OPENTREP::int0_5_p_t int0_5_p;
00871
00873     OPENTREP::uint1_4_p_t uint1_4_p;
00874
00876     OPENTREP::uint1_9_p_t uint1_9_p;
00877     OPENTREP::uint0_9_p_t uint0_9_p;
00878
00880     OPENTREP::hour_p_t hour_p;
00881     OPENTREP::minute_p_t minute_p;
00882     OPENTREP::second_p_t second_p;
00883
00885     OPENTREP::year_p_t year_p;
00886     OPENTREP::month_p_t month_p;
00887     OPENTREP::day_p_t day_p;
00888
00890     OPENTREP::boolean_p_t boolean_p;
00891
00893     //
00894     //  (Boost Spirit) Grammar Definition
00895     //
00897
01068     template <typename Iterator>
01069     struct LocationParser : public bsq::grammar<Iterator, bsu::blank_type> {
01070
01071       LocationParser (Location& ioPORRule) :
01072         LocationParser::base_type(start), _location(ioPORRule) {
01073
01074         start = bsq::eps
01075           >> *(header | por_rule);
01076
01077         header = bsq::lit("iata_code") >> +(bsu::char_ - bsq::eoi - bsq::eol)
01078                                        >> (bsq::eoi | bsq::eol);
01079
01080         por_rule = por_key
01081           >> '^' >> por_details
01082           >> '^' >> -alt_name_section
01083           >> por_rule_end[doEndPor(_location)];
01084         // >> +( '^' >> segment )
01085
01086         por_rule_end = bsq::eps;
01087
01088         por_key = iata_code
01089           >> '^' >> -icao_code
01090           >> '^' >> -faa_code
01091           >> '^' >> is_geonames
01092           >> '^' >> geoname_id
01093           >> '^' >> -envelope_id
01094           ;
01095
01096         por_details = common_name
01097           >> '^' >> ascii_name
01098           >> '^' >> -latitude
01099           >> '^' >> -longitude
01100           >> '^' >> feat_class
01101           >> '^' >> feat_code
01102           >> '^' >> -page_rank
01103           >> '^' >> -date_from
01104           >> '^' >> -date_until
01105           >> '^' >> -comments
01106           >> '^' >> country_code
01107           >> '^' >> -country_code2
01108           >> '^' >> country_name
01109           >> '^' >> -continent_name
01110           >> '^' >> -adm1_code
01111           >> '^' >> -adm1_name_utf
01112           >> '^' >> -adm1_name_ascii
01113           >> '^' >> -adm2_code
01114           >> '^' >> -adm2_name_utf
01115           >> '^' >> -adm2_name_ascii
```

```
01116           >> '^' >> -adm3_code
01117           >> '^' >> -adm4_code
01118           >> '^' >> -population
01119           >> '^' >> -elevation
01120           >> '^' >> -gtopo30
01121           >> '^' >> -time_zone
01122           >> '^' >> -gmt_offset
01123           >> '^' >> -dst_offset
01124           >> '^' >> -raw_offset
01125           >> '^' >> (mod_date | bsq::lit("-1"))
01126           >> '^' >> city_code_list
01127           >> '^' >> -city_name_list
01128           >> '^' >> -city_detail_list
01129           >> '^' >> -tvl_por_code_list[storeTvlPORListString(
    _location)]
01130           >> '^' >> -state_code
01131           >> '^' >> por_type
01132           >> '^' >> -wiki_link
01133           ;
01134         // >> '^' >> -alt_name_short_list[storeAltNameShortListString(_location)]
01135
01136         iata_code =
01137           bsq::repeat(3)[bsu::char_('A', 'Z')][storeIataCode(
    _location)];
01138
01139         icao_code =
01140           bsq::repeat(4)[bsu::char_("A-Z0-9")][storeIcaoCode(
    _location)];
01141
01142         faa_code =
01143           bsq::repeat(1,4)[bsu::char_("A-Z0-9")][storeFaaCode(
    _location)];
01144
01145         geoname_id = uint1_9_p[storeGeonamesID(
    _location)];
01146
01147         envelope_id = uint1_4_p[storeEnvelopeID(
    _location)];
01148
01149         is_geonames = boolean_p;
01150
01151         common_name =
01152           (bsq::no_skip[+~bsu::char_('^')]
01153            - (bsq::eoi|bsq::eol))[storeCommonName(_location)]
01154           ;
01155
01156         ascii_name =
01157           (bsq::no_skip[+~bsu::char_('^')]
01158            - (bsq::eoi|bsq::eol))[storeAsciiName(_location)]
01159           ;
01160
01161         alt_name_short_list = alt_name_short % ',';
01162
01163         alt_name_short =
01164           (bsq::no_skip[+~bsu::char_("^,")]
01165            - (bsq::eoi|bsq::eol))[storeAltNameShort(_location)]
01166           ;
01167
01168         tvl_por_code_list = tvl_por_code % ',';
01169
01170         tvl_por_code =
01171           (bsq::no_skip[+~bsu::char_("^,")]
01172            - (bsq::eoi|bsq::eol))[storeTvlPORCode(_location)]
01173           ;
01174
01175         latitude = bsq::double_[storeLatitude(_location)];
01176
01177         longitude = bsq::double_[storeLongitude(_location)];
01178
01179         feat_class =
01180           bsq::repeat(1)[bsu::char_("A-Z")][storeFeatureClass(
    _location)]
01181           ;
01182
01183         feat_code =
01184           bsq::repeat(2,5)[bsu::char_("A-Z1-5")][storeFeatureCode(
    _location)]
01185           ;
01186
01187         page_rank = bsq::double_[storePageRank(_location)];
01188
01189         date_from = date[storeDateFrom(_location)];
01190
01191         date_until = date[storeDateUntil(_location)];
01192
01193         comments =
01194           (bsq::no_skip[+~bsu::char_('^')]
```

```
01195           - (bsq::eoi|bsq::eol))[storeComments(_location)]
01196         ;
01197
01198       country_code =
01199         bsq::repeat(2,3)[bsu::char_("A-Z")][storeCountryCode(
    _location)]
01200         ;
01201
01202       country_code2 =
01203         (bsq::no_skip[+~bsu::char_('^')]
01204           - (bsq::eoi|bsq::eol))[storeAltCountryCode(
    _location)]
01205         ;
01206
01207       country_name =
01208         (bsq::no_skip[+~bsu::char_('^')]
01209           - (bsq::eoi|bsq::eol))[storeCountryName(_location)]
01210         ;
01211
01212       continent_name =
01213         (bsq::no_skip[+~bsu::char_('^')]
01214           - (bsq::eoi|bsq::eol))[storeContinentName(_location)]
01215         ;
01216
01217       adm1_code =
01218         (bsq::no_skip[+~bsu::char_('^')]
01219           - (bsq::eoi|bsq::eol))[storeAdm1Code(_location)]
01220         ;
01221
01222       adm1_name_utf =
01223         (bsq::no_skip[+~bsu::char_('^')]
01224           - (bsq::eoi|bsq::eol))[storeAdm1UtfName(_location)]
01225         ;
01226
01227       adm1_name_ascii =
01228         (bsq::no_skip[+~bsu::char_('^')]
01229           - (bsq::eoi|bsq::eol))[storeAdm1AsciiName(_location)]
01230         ;
01231
01232       adm2_code =
01233         (bsq::no_skip[+~bsu::char_('^')]
01234           - (bsq::eoi|bsq::eol))[storeAdm2Code(_location)]
01235         ;
01236
01237       adm2_name_utf =
01238         (bsq::no_skip[+~bsu::char_('^')]
01239           - (bsq::eoi|bsq::eol))[storeAdm2UtfName(_location)]
01240         ;
01241
01242       adm2_name_ascii =
01243         (bsq::no_skip[+~bsu::char_('^')]
01244           - (bsq::eoi|bsq::eol))[storeAdm2AsciiName(_location)]
01245         ;
01246
01247       adm3_code =
01248         (bsq::no_skip[+~bsu::char_('^')]
01249           - (bsq::eoi|bsq::eol))[storeAdm3Code(_location)]
01250         ;
01251
01252       adm4_code =
01253         (bsq::no_skip[+~bsu::char_('^')]
01254           - (bsq::eoi|bsq::eol))[storeAdm4Code(_location)]
01255         ;
01256
01257       population = uint1_9_p[storePopulation(
    _location)];
01258
01259       elevation = int1_5_p[storeElevation(
    _location)];
01260
01261       gtopo30 = int1_5_p[storeGTopo30(_location)];
01262
01263       time_zone =
01264         (bsq::no_skip[+~bsu::char_('^')]
01265           - (bsq::eoi|bsq::eol))[storeTimeZone(_location)]
01266         ;
01267
01268       gmt_offset = bsq::float_[storeGMTOffset(
    _location)];
01269
01270       dst_offset = bsq::float_[storeDSTOffset(
    _location)];
01271
01272       raw_offset = bsq::float_[storeRawOffset(
    _location)];
01273
01274       mod_date = date[storeModDate(_location)];
```

```
01275
01276         date = bsq::lexeme
01277           [year_p[boost::phoenix::ref(_location._itYear) = bsq::labels::_1]
01278            >> '-'
01279            >> month_p[boost::phoenix::ref(_location._itMonth) = bsq::labels::_1]
01280            >> '-'
01281            >> day_p[boost::phoenix::ref(_location._itDay) = bsq::labels::_1] ];
01282
01283         city_code_list = city_code % ',';
01284
01285         city_code =
01286           bsq::repeat(3)[bsu::char_('A', 'Z')][storeCityCode(
      _location)]
01287           ;
01288
01289         city_name_list = city_name_utf % '=';
01290
01291         city_name_utf =
01292           (bsq::no_skip[+~bsu::char_("^|=")]
01293            - (bsq::eoi|bsq::eol))[storeCityUtfName(_location)]
01294           ;
01295
01296         city_name_ascii =
01297           (bsq::no_skip[+~bsu::char_("^|=")]
01298            - (bsq::eoi|bsq::eol))[storeCityAsciiName(_location)]
01299           ;
01300
01301         city_detail_list = city_details % '=';
01302
01303         city_details =
01304           city_code
01305           >> '|' >> city_geoname_id
01306           >> '|' >> city_name_utf
01307           >> '|' >> city_name_ascii
01308           ;
01309
01310         city_geoname_id = uint1_9_p[storeCityGeonamesID(
      _location)];
01311
01312         state_code =
01313           (bsq::no_skip[+~bsu::char_('^')]
01314            - (bsq::eoi|bsq::eol))[storeStateCode(_location)]
01315           ;
01316
01317         por_type =
01318           bsq::repeat(1,3)[bsu::char_("ABCGHOPRZ")][storePORType(
      _location)]
01319           ;
01320
01321         wiki_link =
01322           (bsq::no_skip[+~bsu::char_('^')]
01323            - (bsq::eoi|bsq::eol))[storeWikiLink(_location)]
01324           ;
01325
01326         alt_name_section = alt_name_details % '=';
01327
01328         alt_name_details =
01329           -alt_lang_code
01330           >> '|' >> alt_name
01331           >> '|' >> -alt_name_qualifiers
01332           ;
01333
01334         alt_lang_code =
01335           (+~bsu::char_("|=")
01336            - (bsq::eoi|bsq::eol))[storeAltLangCodeFull(
      _location)]
01337           ;
01338
01339         alt_lang_code_ftd = lang_code_2char >>
      lang_code_opt;
01340
01341         lang_code_opt = -(lang_code_ext | lang_code_hist);
01342
01343         lang_code_2char =
01344           bsq::repeat(2,4)[bsu::char_("a-z")][storeAltLangCode2Char(
      _location)]
01345           ;
01346
01347         lang_code_ext =
01348           '-' >> bsq::repeat(1,4)[bsu::char_('A', 'Z')][storeAltLangCodeExt(
      _location)];
01349
01350         lang_code_hist =
01351           '_' >> bsq::repeat(1,4)[bsu::char_("a-z0-9")][storeAltLangCodeHist(
      _location)];
01352
01353         alt_name =
```

```
01354           (bsq::no_skip[+~bsu::char_("|=")]
01355            - (bsq::eoi|bsq::eol))[storeAltName(_location)]
01356           ;
01357
01358         alt_name_qualifiers =
01359           bsq::repeat(1,4)[bsu::char_("shpc")]
01360           ;
01361
01362         //BOOST_SPIRIT_DEBUG_NODE (LocationParser);
01363         BOOST_SPIRIT_DEBUG_NODE (start);
01364         BOOST_SPIRIT_DEBUG_NODE (header);
01365         BOOST_SPIRIT_DEBUG_NODE (por_rule);
01366         BOOST_SPIRIT_DEBUG_NODE (por_rule_end);
01367         BOOST_SPIRIT_DEBUG_NODE (por_key);
01368         BOOST_SPIRIT_DEBUG_NODE (por_details);
01369         BOOST_SPIRIT_DEBUG_NODE (iata_code);
01370         BOOST_SPIRIT_DEBUG_NODE (icao_code);
01371         BOOST_SPIRIT_DEBUG_NODE (faa_code);
01372         BOOST_SPIRIT_DEBUG_NODE (geoname_id);
01373         BOOST_SPIRIT_DEBUG_NODE (envelope_id);
01374         BOOST_SPIRIT_DEBUG_NODE (is_geonames);
01375         BOOST_SPIRIT_DEBUG_NODE (common_name);
01376         BOOST_SPIRIT_DEBUG_NODE (ascii_name);
01377         BOOST_SPIRIT_DEBUG_NODE (alt_name_short_list);
01378         BOOST_SPIRIT_DEBUG_NODE (alt_name_short);
01379         BOOST_SPIRIT_DEBUG_NODE (alt_name_sep);
01380         BOOST_SPIRIT_DEBUG_NODE (tvl_por_code_list);
01381         BOOST_SPIRIT_DEBUG_NODE (tvl_por_code);
01382         BOOST_SPIRIT_DEBUG_NODE (tvl_por_sep);
01383         BOOST_SPIRIT_DEBUG_NODE (latitude);
01384         BOOST_SPIRIT_DEBUG_NODE (longitude);
01385         BOOST_SPIRIT_DEBUG_NODE (feat_class);
01386         BOOST_SPIRIT_DEBUG_NODE (feat_code);
01387         BOOST_SPIRIT_DEBUG_NODE (page_rank);
01388         BOOST_SPIRIT_DEBUG_NODE (date_from);
01389         BOOST_SPIRIT_DEBUG_NODE (date_until);
01390         BOOST_SPIRIT_DEBUG_NODE (comments);
01391         BOOST_SPIRIT_DEBUG_NODE (country_code);
01392         BOOST_SPIRIT_DEBUG_NODE (country_code2);
01393         BOOST_SPIRIT_DEBUG_NODE (country_name);
01394         BOOST_SPIRIT_DEBUG_NODE (continent_name);
01395         BOOST_SPIRIT_DEBUG_NODE (adm1_code);
01396         BOOST_SPIRIT_DEBUG_NODE (adm1_name_utf);
01397         BOOST_SPIRIT_DEBUG_NODE (adm1_name_ascii);
01398         BOOST_SPIRIT_DEBUG_NODE (adm2_code);
01399         BOOST_SPIRIT_DEBUG_NODE (adm2_name_utf);
01400         BOOST_SPIRIT_DEBUG_NODE (adm2_name_ascii);
01401         BOOST_SPIRIT_DEBUG_NODE (adm3_code);
01402         BOOST_SPIRIT_DEBUG_NODE (adm4_code);
01403         BOOST_SPIRIT_DEBUG_NODE (population);
01404         BOOST_SPIRIT_DEBUG_NODE (elevation);
01405         BOOST_SPIRIT_DEBUG_NODE (gtopo30);
01406         BOOST_SPIRIT_DEBUG_NODE (time_zone);
01407         BOOST_SPIRIT_DEBUG_NODE (gmt_offset);
01408         BOOST_SPIRIT_DEBUG_NODE (raw_offset);
01409         BOOST_SPIRIT_DEBUG_NODE (dst_offset);
01410         BOOST_SPIRIT_DEBUG_NODE (mod_date);
01411         BOOST_SPIRIT_DEBUG_NODE (date);
01412         BOOST_SPIRIT_DEBUG_NODE (city_code_list);
01413         BOOST_SPIRIT_DEBUG_NODE (city_name_list);
01414         BOOST_SPIRIT_DEBUG_NODE (city_detail_list);
01415         BOOST_SPIRIT_DEBUG_NODE (city_details);
01416         BOOST_SPIRIT_DEBUG_NODE (city_geoname_id);
01417         BOOST_SPIRIT_DEBUG_NODE (city_code);
01418         BOOST_SPIRIT_DEBUG_NODE (city_name_utf);
01419         BOOST_SPIRIT_DEBUG_NODE (city_name_ascii);
01420         BOOST_SPIRIT_DEBUG_NODE (state_code);
01421         BOOST_SPIRIT_DEBUG_NODE (por_type);
01422         BOOST_SPIRIT_DEBUG_NODE (wiki_link);
01423         BOOST_SPIRIT_DEBUG_NODE (alt_name_section);
01424         BOOST_SPIRIT_DEBUG_NODE (alt_name_details);
01425         BOOST_SPIRIT_DEBUG_NODE (alt_lang_code);
01426         BOOST_SPIRIT_DEBUG_NODE (alt_lang_code_ftd);
01427         BOOST_SPIRIT_DEBUG_NODE (alt_name);
01428         BOOST_SPIRIT_DEBUG_NODE (alt_name_qualifiers);
01429         BOOST_SPIRIT_DEBUG_NODE (lang_code_opt);
01430         BOOST_SPIRIT_DEBUG_NODE (lang_code_2char);
01431         BOOST_SPIRIT_DEBUG_NODE (lang_code_ext);
01432         BOOST_SPIRIT_DEBUG_NODE (lang_code_hist);
01433       }
01434
01435       // Instantiation of rules
01436       bsq::rule<Iterator, bsu::blank_type>
01437       start, header, por_rule, por_rule_end,
      por_key, por_details,
01438         iata_code, icao_code, faa_code, geoname_id,
      envelope_id, is_geonames,
```

```
01439         common_name, ascii_name,
01440         alt_name_short_list, alt_name_short,
      alt_name_sep,
01441         tvl_por_code_list, tvl_por_code,
      tvl_por_sep,
01442         latitude, longitude, feat_class, feat_code,
01443         page_rank, date_from, date_until, comments,
01444         country_code, country_code2, country_name,
      continent_name,
01445         adm1_code, adm1_name_utf, adm1_name_ascii,
01446         adm2_code, adm2_name_utf, adm2_name_ascii,
      adm3_code, adm4_code,
01447         population, elevation, gtopo30,
01448         time_zone, gmt_offset, dst_offset,
      raw_offset,
01449         mod_date, date,
01450         city_code_list, city_code,
01451         city_name_list, city_name_utf,
      city_name_ascii,
01452         city_detail_list, city_details,
      city_geoname_id,
01453         state_code,
01454         por_type, wiki_link,
01455         alt_name_section, alt_name_details,
01456         alt_lang_code, alt_lang_code_ftd, alt_name,
      alt_name_qualifiers,
01457         lang_code_opt, lang_code_2char,
      lang_code_ext, lang_code_hist,
01458         destination;
01459
01460       // Parser Context
01461       Location& _location;
01462     };
01463   }
01464
01465
01467   //
01468   //  Entry class for the string parser
01469   //
01471
01472   // ////////////////////////////////////////////////////////////////////
01473   PORStringParser::PORStringParser (const std::string& iString)
01474     : _string (iString) {
01475     init();
01476   }
01477
01478   // ////////////////////////////////////////////////////////////////////
01479   void PORStringParser::init() {
01480     // Store the raw data string
01481     _location.setRawDataString (_string);
01482   }
01483
01484   // ////////////////////////////////////////////////////////////////////
01485   PORStringParser::~PORStringParser() {
01486   }
01487
01488   // ////////////////////////////////////////////////////////////////////
01489   const Location& PORStringParser::generateLocation() {
01490     // DEBUG
01491     // OPENTREP_LOG_DEBUG ("Parsing POR string: '" << _string << "'");
01492
01493     // String to be parsed
01494     std::istringstream stringToBeParsed (_string);
01495
01496     // Create an input iterator
01497     OPENTREP::base_iterator_t iStr (stringToBeParsed);
01498
01499     // Convert input iterator to an iterator usable by spirit parser
01500     OPENTREP::iterator_t fwd_start(boost::spirit::make_default_multi_pass(iStr));
01501     OPENTREP::iterator_t fwd_end;
01502
01503     // Initialise the positional iterators
01504     OPENTREP::pos_iterator_t pos_start (fwd_start, fwd_end, _string);
01505     OPENTREP::pos_iterator_t pos_end;
01506
01507     // Initialise the parser (grammar) with the helper/staging structure.
01508     PorParserHelper::LocationParser<OPENTREP::iterator_t>
     lPORParser (_location);
01509
01510     // Launch the parsing of the file and, thanks to the doEndPor
01511     // call-back structure, the building of the whole BomRoot BOM
01512     bool hasParsingBeenSuccesful = false;
01513     try {
01514
01515       hasParsingBeenSuccesful = bsq::phrase_parse (fwd_start, fwd_end,
01516                                                    lPORParser, bsu::blank);
01517
```

```
01518     } catch (const bsq::expectation_failure<pos_iterator_t>& e) {
01519       const bsc::file_position_base<std::string>& pos = e.first.get_position();
01520       std::ostringstream oStr;
01521       oStr << "Parse error on POR string '" << _string
01522            << "', position " << pos.column << std::endl
01523            << "'" << e.first.get_currentline() << "'" << std::endl
01524            << std::setw(pos.column) << " " << "^- here";
01525       OPENTREP_LOG_ERROR (oStr.str());
01526       throw PorFileParsingException (oStr.str());
01527     }
01528
01529     if (hasParsingBeenSuccesful == false) {
01530       OPENTREP_LOG_ERROR ("Parsing of POR input string: '" << _string
01531                           << "' failed");
01532       throw PorFileParsingException ("Parsing of POR input string: '"
01533                                      + _string + "' failed");
01534     }
01535
01536     if  (fwd_start != fwd_end) {
01537       OPENTREP_LOG_ERROR ("Parsing of POR input string: '" << _string
01538                           << "' failed");
01539       throw PorFileParsingException ("Parsing of POR input file: '"
01540                                      + _string + "' failed");
01541     }
01542
01543     //
01544     if (hasParsingBeenSuccesful == true && fwd_start == fwd_end) {
01545       // DEBUG
01546       /*
01547       OPENTREP_LOG_DEBUG ("Parsing of POR input string: '" << _string
01548                           << "' succeeded");
01549       */
01550     }
01551
01552     return _location;
01553   }
01554
01555
01557   //
01558   //  Entry class for the file parser
01559   //
01561
01562   // ////////////////////////////////////////////////////////////////////
01563   PORFileParser::PORFileParser (const PORFilePath_T& iFilename)
01564     : _filename (iFilename) {
01565     init();
01566   }
01567
01568   // ////////////////////////////////////////////////////////////////////
01569   void PORFileParser::init() {
01570   }
01571
01572   // ////////////////////////////////////////////////////////////////////
01573   void PORFileParser::generateLocations() {
01574
01575     OPENTREP_LOG_DEBUG ("Parsing por input file: " << _filename);
01576
01577     // File to be parsed
01578     const char* lFilenameStr = _filename.c_str();
01579     std::ifstream fileToBeParsed (lFilenameStr, std::ios_base::in);
01580
01581     // Check if the filename exist and can be open
01582     if (fileToBeParsed.is_open() == false) {
01583       OPENTREP_LOG_ERROR ("The por file " << _filename << " can not be open."
01584                           << std::endl);
01585
01586       throw FileNotFoundException ("The file " + _filename
01587                                    + " does not exist or can not be read");
01588     }
01589
01590     // Create an input iterator
01591     OPENTREP::base_iterator_t iStr (fileToBeParsed);
01592
01593     // Convert input iterator to an iterator usable by spirit parser
01594     OPENTREP::iterator_t start (boost::spirit::make_default_multi_pass (iStr));
01595     OPENTREP::iterator_t end;
01596
01597     // Initialise the parser (grammar) with the helper/staging structure.
01598     PorParserHelper::LocationParser<OPENTREP::iterator_t>
    lPORParser (_location);
01599
01600     // Launch the parsing of the file and, thanks to the doEndPor
01601     // call-back structure, the building of the whole BomRoot BOM
01602     const bool hasParsingBeenSuccesful =
01603        bsq::phrase_parse (start, end, lPORParser, bsu::blank);
01604
01605     if (hasParsingBeenSuccesful == false) {
```

```
01606       OPENTREP_LOG_ERROR ("Parsing of por input file: " << _filename
01607                           << " failed");
01608       throw PorFileParsingException ("Parsing of por input file: "
01609                                      + _filename + " failed");
01610     }
01611
01612     if  (start != end) {
01613       OPENTREP_LOG_ERROR ("Parsing of por input file: " << _filename
01614                           << " failed");
01615       throw PorFileParsingException ("Parsing of por input file: "
01616                                      + _filename + " failed");
01617     }
01618
01619     if (hasParsingBeenSuccesful == true && start == end) {
01620       OPENTREP_LOG_DEBUG ("Parsing of por input file: " << _filename
01621                           << " succeeded");
01622     }
01623   }
01624
01625 }
```

24.139 opentrep/bom/PORParserHelper.hpp File Reference

```
#include <string>
#include <boost/spirit/include/qi.hpp>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/Location.hpp>
```

Classes

- struct OPENTREP::PorParserHelper::ParserSemanticAction
- struct OPENTREP::PorParserHelper::storeIataCode
- struct OPENTREP::PorParserHelper::storeIcaoCode
- struct OPENTREP::PorParserHelper::storeFaaCode
- struct OPENTREP::PorParserHelper::storeGeonamesID
- struct OPENTREP::PorParserHelper::storeEnvelopeID
- struct OPENTREP::PorParserHelper::storeCommonName
- struct OPENTREP::PorParserHelper::storeAsciiName
- struct OPENTREP::PorParserHelper::storeAltNameShort
- struct OPENTREP::PorParserHelper::storeAltNameShortListString
- struct OPENTREP::PorParserHelper::storeTvlPORCode
- struct OPENTREP::PorParserHelper::storeTvlPORListString
- struct OPENTREP::PorParserHelper::storeLatitude
- struct OPENTREP::PorParserHelper::storeLongitude
- struct OPENTREP::PorParserHelper::storeFeatureClass
- struct OPENTREP::PorParserHelper::storeFeatureCode
- struct OPENTREP::PorParserHelper::storePageRank
- struct OPENTREP::PorParserHelper::storeDateFrom
- struct OPENTREP::PorParserHelper::storeDateUntil
- struct OPENTREP::PorParserHelper::storeComments
- struct OPENTREP::PorParserHelper::storeCountryCode
- struct OPENTREP::PorParserHelper::storeAltCountryCode
- struct OPENTREP::PorParserHelper::storeCountryName
- struct OPENTREP::PorParserHelper::storeContinentName
- struct OPENTREP::PorParserHelper::storeAdm1Code
- struct OPENTREP::PorParserHelper::storeAdm1UtfName
- struct OPENTREP::PorParserHelper::storeAdm1AsciiName
- struct OPENTREP::PorParserHelper::storeAdm2Code
- struct OPENTREP::PorParserHelper::storeAdm2UtfName
- struct OPENTREP::PorParserHelper::storeAdm2AsciiName

- struct OPENTREP::PorParserHelper::storeAdm3Code
- struct OPENTREP::PorParserHelper::storeAdm4Code
- struct OPENTREP::PorParserHelper::storePopulation
- struct OPENTREP::PorParserHelper::storeElevation
- struct OPENTREP::PorParserHelper::storeGTopo30
- struct OPENTREP::PorParserHelper::storeTimeZone
- struct OPENTREP::PorParserHelper::storeGMTOffset
- struct OPENTREP::PorParserHelper::storeDSTOffset
- struct OPENTREP::PorParserHelper::storeRawOffset
- struct OPENTREP::PorParserHelper::storeModDate
- struct OPENTREP::PorParserHelper::storeCityCode
- struct OPENTREP::PorParserHelper::storeCityUtfName
- struct OPENTREP::PorParserHelper::storeCityAsciiName
- struct OPENTREP::PorParserHelper::storeCityGeonamesID
- struct OPENTREP::PorParserHelper::storeStateCode
- struct OPENTREP::PorParserHelper::storePORType
- struct OPENTREP::PorParserHelper::storeWikiLink
- struct OPENTREP::PorParserHelper::storeAltLangCodeFull
- struct OPENTREP::PorParserHelper::storeAltLangCode2Char
- struct OPENTREP::PorParserHelper::storeAltLangCodeExt
- struct OPENTREP::PorParserHelper::storeAltLangCodeHist
- struct OPENTREP::PorParserHelper::storeAltName
- struct OPENTREP::PorParserHelper::doEndPor
- class OPENTREP::PORStringParser
- class OPENTREP::PORFileParser

Namespaces

- OPENTREP
- OPENTREP::PorParserHelper

Typedefs

- typedef boost::uint32_t OPENTREP::PorParserHelper::uchar_t

24.140 PORParserHelper.hpp

```
00001 #ifndef __OPENTREP_CMD_PORPARSERHELPER_HPP
00002 #define __OPENTREP_CMD_PORPARSERHELPER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/spirit/include/qi.hpp>
00011 // Opentrep
00012 #include <opentrep/OPENTREP_Types.hpp>
00013 #include <opentrep/Location.hpp>
00014
00015 namespace OPENTREP {
00016
00017   namespace PorParserHelper {
00018
00020     //typedef char uchar_t;
00021     typedef boost::uint32_t uchar_t;
00022
00023
00024     // ////////////////////////////////////////////////////////////////
00025     //  Semantic actions
00026     // ////////////////////////////////////////////////////
00028
```

```
00029     struct ParserSemanticAction {
00031       ParserSemanticAction (Location&);
00033       Location& _location;
00034     };
00035
00037     struct storeIataCode : public ParserSemanticAction {
00039       storeIataCode (Location&);
00041       void operator() (std::vector<uchar_t>,
00042                        boost::spirit::qi::unused_type,
00043                        boost::spirit::qi::unused_type) const;
00044     };
00045
00047     struct storeIcaoCode : public ParserSemanticAction {
00049       storeIcaoCode (Location&);
00051       void operator() (std::vector<uchar_t>,
00052                        boost::spirit::qi::unused_type,
00053                        boost::spirit::qi::unused_type) const;
00054     };
00055
00057     struct storeFaaCode : public ParserSemanticAction {
00059       storeFaaCode (Location&);
00061       void operator() (std::vector<uchar_t>,
00062                        boost::spirit::qi::unused_type,
00063                        boost::spirit::qi::unused_type) const;
00064     };
00065
00067     struct storeGeonamesID : public ParserSemanticAction {
00069       storeGeonamesID (Location&);
00071       void operator() (unsigned int,
00072                        boost::spirit::qi::unused_type,
00073                        boost::spirit::qi::unused_type) const;
00074     };
00075
00077     struct storeEnvelopeID : public ParserSemanticAction {
00079       storeEnvelopeID (Location&);
00081       void operator() (unsigned int,
00082                        boost::spirit::qi::unused_type,
00083                        boost::spirit::qi::unused_type) const;
00084     };
00085
00087     struct storeCommonName : public ParserSemanticAction {
00089       storeCommonName (Location&);
00091       void operator() (std::vector<uchar_t>,
00092                        boost::spirit::qi::unused_type,
00093                        boost::spirit::qi::unused_type) const;
00094     };
00095
00097     struct storeAsciiName : public ParserSemanticAction {
00099       storeAsciiName (Location&);
00101       void operator() (std::vector<uchar_t>,
00102                        boost::spirit::qi::unused_type,
00103                        boost::spirit::qi::unused_type) const;
00104     };
00105
00107     struct storeAltNameShort : public ParserSemanticAction {
00109       storeAltNameShort (Location&);
00111       void operator() (std::vector<uchar_t>,
00112                        boost::spirit::qi::unused_type,
00113                        boost::spirit::qi::unused_type) const;
00114     };
00115
00117     struct storeAltNameShortListString : public
      ParserSemanticAction {
00119       storeAltNameShortListString (Location&);
00121       void operator() (boost::spirit::qi::unused_type,
00122                        boost::spirit::qi::unused_type,
00123                        boost::spirit::qi::unused_type) const;
00124     };
00125
00127     struct storeTvlPORCode : public ParserSemanticAction {
00129       storeTvlPORCode (Location&);
00131       void operator() (std::vector<uchar_t>,
00132                        boost::spirit::qi::unused_type,
00133                        boost::spirit::qi::unused_type) const;
00134     };
00135
00137     struct storeTvlPORListString : public
      ParserSemanticAction {
00139       storeTvlPORListString (Location&);
00141       void operator() (boost::spirit::qi::unused_type,
00142                        boost::spirit::qi::unused_type,
00143                        boost::spirit::qi::unused_type) const;
00144     };
00145
00147     struct storeLatitude : public ParserSemanticAction {
00149       storeLatitude (Location&);
00151       void operator() (double,
```

```
00152                     boost::spirit::qi::unused_type,
00153                     boost::spirit::qi::unused_type) const;
00154     };
00155
00157     struct storeLongitude : public ParserSemanticAction {
00159       storeLongitude (Location&);
00161       void operator() (double,
00162                     boost::spirit::qi::unused_type,
00163                     boost::spirit::qi::unused_type) const;
00164     };
00165
00167     struct storeFeatureClass : public ParserSemanticAction {
00169       storeFeatureClass (Location&);
00171       void operator() (std::vector<uchar_t>,
00172                     boost::spirit::qi::unused_type,
00173                     boost::spirit::qi::unused_type) const;
00174     };
00175
00177     struct storeFeatureCode : public ParserSemanticAction {
00179       storeFeatureCode (Location&);
00181       void operator() (std::vector<uchar_t>,
00182                     boost::spirit::qi::unused_type,
00183                     boost::spirit::qi::unused_type) const;
00184     };
00185
00187     struct storePageRank : public ParserSemanticAction {
00189       storePageRank (Location&);
00191       void operator() (double,
00192                     boost::spirit::qi::unused_type,
00193                     boost::spirit::qi::unused_type) const;
00194     };
00195
00197     struct storeDateFrom : public ParserSemanticAction {
00199       storeDateFrom (Location&);
00201       void operator() (boost::spirit::qi::unused_type,
00202                     boost::spirit::qi::unused_type,
00203                     boost::spirit::qi::unused_type) const;
00204     };
00205
00207     struct storeDateUntil : public ParserSemanticAction {
00209       storeDateUntil (Location&);
00211       void operator() (boost::spirit::qi::unused_type,
00212                     boost::spirit::qi::unused_type,
00213                     boost::spirit::qi::unused_type) const;
00214     };
00215
00217     struct storeComments : public ParserSemanticAction {
00219       storeComments (Location&);
00221       void operator() (std::vector<uchar_t>,
00222                     boost::spirit::qi::unused_type,
00223                     boost::spirit::qi::unused_type) const;
00224     };
00225
00227     struct storeCountryCode : public ParserSemanticAction {
00229       storeCountryCode (Location&);
00231       void operator() (std::vector<uchar_t>,
00232                     boost::spirit::qi::unused_type,
00233                     boost::spirit::qi::unused_type) const;
00234     };
00235
00237     struct storeAltCountryCode : public ParserSemanticAction {
00239       storeAltCountryCode (Location&);
00241       void operator() (std::vector<uchar_t>,
00242                     boost::spirit::qi::unused_type,
00243                     boost::spirit::qi::unused_type) const;
00244     };
00245
00247     struct storeCountryName : public ParserSemanticAction {
00249       storeCountryName (Location&);
00251       void operator() (std::vector<uchar_t>,
00252                     boost::spirit::qi::unused_type,
00253                     boost::spirit::qi::unused_type) const;
00254     };
00255
00257     struct storeContinentName : public ParserSemanticAction {
00259       storeContinentName (Location&);
00261       void operator() (std::vector<uchar_t>,
00262                     boost::spirit::qi::unused_type,
00263                     boost::spirit::qi::unused_type) const;
00264     };
00265
00267     struct storeAdm1Code : public ParserSemanticAction {
00269       storeAdm1Code (Location&);
00271       void operator() (std::vector<uchar_t>,
00272                     boost::spirit::qi::unused_type,
00273                     boost::spirit::qi::unused_type) const;
00274     };
```

```
00275
00277     struct storeAdm1UtfName : public ParserSemanticAction {
00279       storeAdm1UtfName (Location&);
00281       void operator() (std::vector<uchar_t>,
00282                        boost::spirit::qi::unused_type,
00283                        boost::spirit::qi::unused_type) const;
00284     };
00285
00287     struct storeAdm1AsciiName : public ParserSemanticAction {
00289       storeAdm1AsciiName (Location&);
00291       void operator() (std::vector<uchar_t>,
00292                        boost::spirit::qi::unused_type,
00293                        boost::spirit::qi::unused_type) const;
00294     };
00295
00297     struct storeAdm2Code : public ParserSemanticAction {
00299       storeAdm2Code (Location&);
00301       void operator() (std::vector<uchar_t>,
00302                        boost::spirit::qi::unused_type,
00303                        boost::spirit::qi::unused_type) const;
00304     };
00305
00307     struct storeAdm2UtfName : public ParserSemanticAction {
00309       storeAdm2UtfName (Location&);
00311       void operator() (std::vector<uchar_t>,
00312                        boost::spirit::qi::unused_type,
00313                        boost::spirit::qi::unused_type) const;
00314     };
00315
00317     struct storeAdm2AsciiName : public ParserSemanticAction {
00319       storeAdm2AsciiName (Location&);
00321       void operator() (std::vector<uchar_t>,
00322                        boost::spirit::qi::unused_type,
00323                        boost::spirit::qi::unused_type) const;
00324     };
00325
00327     struct storeAdm3Code : public ParserSemanticAction {
00329       storeAdm3Code (Location&);
00331       void operator() (std::vector<uchar_t>,
00332                        boost::spirit::qi::unused_type,
00333                        boost::spirit::qi::unused_type) const;
00334     };
00335
00337     struct storeAdm4Code : public ParserSemanticAction {
00339       storeAdm4Code (Location&);
00341       void operator() (std::vector<uchar_t>,
00342                        boost::spirit::qi::unused_type,
00343                        boost::spirit::qi::unused_type) const;
00344     };
00345
00347     struct storePopulation : public ParserSemanticAction {
00349       storePopulation (Location&);
00351       void operator() (unsigned int,
00352                        boost::spirit::qi::unused_type,
00353                        boost::spirit::qi::unused_type) const;
00354     };
00355
00357     struct storeElevation : public ParserSemanticAction {
00359       storeElevation (Location&);
00361       void operator() (int,
00362                        boost::spirit::qi::unused_type,
00363                        boost::spirit::qi::unused_type) const;
00364     };
00365
00367     struct storeGTopo30 : public ParserSemanticAction {
00369       storeGTopo30 (Location&);
00371       void operator() (int,
00372                        boost::spirit::qi::unused_type,
00373                        boost::spirit::qi::unused_type) const;
00374     };
00375
00377     struct storeTimeZone : public ParserSemanticAction {
00379       storeTimeZone (Location&);
00381       void operator() (std::vector<uchar_t>,
00382                        boost::spirit::qi::unused_type,
00383                        boost::spirit::qi::unused_type) const;
00384     };
00385
00387     struct storeGMTOffset : public ParserSemanticAction {
00389       storeGMTOffset (Location&);
00391       void operator() (float,
00392                        boost::spirit::qi::unused_type,
00393                        boost::spirit::qi::unused_type) const;
00394     };
00395
00397     struct storeDSTOffset : public ParserSemanticAction {
00399       storeDSTOffset (Location&);
```

```
00401       void operator() (float,
00402                        boost::spirit::qi::unused_type,
00403                        boost::spirit::qi::unused_type) const;
00404     };
00405
00407     struct storeRawOffset : public ParserSemanticAction {
00409       storeRawOffset (Location&);
00411       void operator() (float,
00412                        boost::spirit::qi::unused_type,
00413                        boost::spirit::qi::unused_type) const;
00414     };
00415
00417     struct storeModDate : public ParserSemanticAction {
00419       storeModDate (Location&);
00421       void operator() (boost::spirit::qi::unused_type,
00422                        boost::spirit::qi::unused_type,
00423                        boost::spirit::qi::unused_type) const;
00424     };
00425
00427     struct storeCityCode : public ParserSemanticAction {
00429       storeCityCode (Location&);
00431       void operator() (std::vector<uchar_t>,
00432                        boost::spirit::qi::unused_type,
00433                        boost::spirit::qi::unused_type) const;
00434     };
00435
00437     struct storeCityUtfName : public ParserSemanticAction {
00439       storeCityUtfName (Location&);
00441       void operator() (std::vector<uchar_t>,
00442                        boost::spirit::qi::unused_type,
00443                        boost::spirit::qi::unused_type) const;
00444     };
00445
00447     struct storeCityAsciiName : public ParserSemanticAction {
00449       storeCityAsciiName (Location&);
00451       void operator() (std::vector<uchar_t>,
00452                        boost::spirit::qi::unused_type,
00453                        boost::spirit::qi::unused_type) const;
00454     };
00455
00457     struct storeCityGeonamesID : public ParserSemanticAction {
00459       storeCityGeonamesID (Location&);
00461       void operator() (unsigned int,
00462                        boost::spirit::qi::unused_type,
00463                        boost::spirit::qi::unused_type) const;
00464     };
00465
00467     struct storeStateCode : public ParserSemanticAction {
00469       storeStateCode (Location&);
00471       void operator() (std::vector<uchar_t>,
00472                        boost::spirit::qi::unused_type,
00473                        boost::spirit::qi::unused_type) const;
00474     };
00475
00477     struct storePORType : public ParserSemanticAction {
00479       storePORType (Location&);
00481       void operator() (std::vector<uchar_t>,
00482                        boost::spirit::qi::unused_type,
00483                        boost::spirit::qi::unused_type) const;
00484     };
00485
00486
00488     struct storeWikiLink : public ParserSemanticAction {
00490       storeWikiLink (Location&);
00492       void operator() (std::vector<uchar_t>,
00493                        boost::spirit::qi::unused_type,
00494                        boost::spirit::qi::unused_type) const;
00495     };
00496
00498     struct storeAltLangCodeFull : public
      ParserSemanticAction {
00500       storeAltLangCodeFull (Location&);
00502       void operator() (std::vector<boost::uint32_t>,
00503                        boost::spirit::qi::unused_type,
00504                        boost::spirit::qi::unused_type) const;
00505     };
00506
00508     struct storeAltLangCode2Char : public
      ParserSemanticAction {
00510       storeAltLangCode2Char (Location&);
00512       void operator() (std::vector<boost::uint32_t>,
00513                        boost::spirit::qi::unused_type,
00514                        boost::spirit::qi::unused_type) const;
00515     };
00516
00518     struct storeAltLangCodeExt : public ParserSemanticAction {
00520       storeAltLangCodeExt (Location&);
```

```
00522       void operator() (std::vector<boost::uint32_t>,
00523                        boost::spirit::qi::unused_type,
00524                        boost::spirit::qi::unused_type) const;
00525     };
00526
00528     struct storeAltLangCodeHist : public
      ParserSemanticAction {
00530       storeAltLangCodeHist (Location&);
00532       void operator() (std::vector<boost::uint32_t>,
00533                        boost::spirit::qi::unused_type,
00534                        boost::spirit::qi::unused_type) const;
00535     };
00536
00538     struct storeAltName : public ParserSemanticAction {
00540       storeAltName (Location&);
00542       void operator() (std::vector<boost::uint32_t>,
00543                        boost::spirit::qi::unused_type,
00544                        boost::spirit::qi::unused_type) const;
00545     };
00546
00548     struct doEndPor : public ParserSemanticAction {
00550       doEndPor (Location&);
00552       void operator() (boost::spirit::qi::unused_type,
00553                        boost::spirit::qi::unused_type,
00554                        boost::spirit::qi::unused_type) const;
00555     };
00556
00557   }
00558
00559
00561   //
00562   //  Entry class for the string parser
00563   //
00565
00572   class PORStringParser {
00573   public:
00577     PORStringParser (const std::string& iString);
00578
00582     ~PORStringParser();
00583
00587     const Location& generateLocation();
00588
00589   private:
00593     void init();
00594
00595   private:
00596     // Attributes
00600     std::string _string;
00601
00605     Location _location;
00606   };
00607
00608
00610   //
00611   //  Entry class for the file parser
00612   //
00614
00621   class PORFileParser {
00622   public:
00626     PORFileParser (const PORFilePath_T& iFilename);
00627
00631     void generateLocations();
00632
00633   private:
00637     void init();
00638
00639   private:
00640     // Attributes
00644     PORFilePath_T _filename;
00645
00649     Location _location;
00650   };
00651
00652 }
00653 #endif // __OPENTREP_CMD_PORPARSERHELPER_HPP
```

24.141 opentrep/bom/QuerySlices.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <set>
#include <opentrep/basic/BasConst_General.hpp>
#include <opentrep/basic/Utilities.hpp>
#include <opentrep/bom/Levenshtein.hpp>
#include <opentrep/bom/Filter.hpp>
#include <opentrep/bom/QuerySlices.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

Functions

- static unsigned int OPENTREP::calculateEditDistance (const TravelQuery_T &iPhrase)

  Helper function.
- bool OPENTREP::doesMatch (const Xapian::Database &iDatabase, const std::string &iWord1, const std::string &iWord2)

  Helper function to query for a Xapian-based full text match.

24.142 QuerySlices.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <set>
00008 // OpenTrep
00009 #include <opentrep/basic/BasConst_General.hpp>
00010 #include <opentrep/basic/Utilities.hpp>
00011 #include <opentrep/bom/Levenshtein.hpp>
00012 #include <opentrep/bom/Filter.hpp>
00013 #include <opentrep/bom/QuerySlices.hpp>
00014 #include <opentrep/service/Logger.hpp>
00015
00016 namespace OPENTREP {
00017
00018   // //////////////////////////////////////////////////////////////////
00019   QuerySlices::QuerySlices (const Xapian::Database& iDatabase,
00020                             const TravelQuery_T& iQueryString)
00021     : _database (iDatabase), _queryString (iQueryString) {
00022     init();
00023   }
00024
00025   // //////////////////////////////////////////////////////////////////
00026   QuerySlices::~QuerySlices() {
00027   }
00028
00029   // //////////////////////////////////////////////////////////////////
00030   void QuerySlices::push_back (const StringPartition& iStringPartition
      ) {
00031     if (iStringPartition.empty() == false) {
00032       _slices.push_back (iStringPartition);
00033     }
00034   }
00035
00036   // //////////////////////////////////////////////////////////////////
00037   size_t QuerySlices::size() const {
00038     return _slices.size();
00039   }
00040
00041   // //////////////////////////////////////////////////////////////////
00042   bool QuerySlices::empty() const {
```

```
00043     return _slices.empty();
00044   }
00045
00046   // ////////////////////////////////////////////////////////////////////
00047   void QuerySlices::clear() {
00048     _slices.clear();
00049   }
00050
00051   // ////////////////////////////////////////////////////////////////////
00052   std::string QuerySlices::describeKey() const {
00053     std::ostringstream oStr;
00054     oStr << "";
00055     return oStr.str();
00056   }
00057
00058   // ////////////////////////////////////////////////////////////////////
00059   std::string QuerySlices::describe() const {
00060     std::ostringstream oStr;
00061     oStr << describeKey();
00062
00063     //
00064     oStr << "[ ";
00065
00066     short idx_sublist = 0;
00067     for (StringPartitionList_T::const_iterator itSlice = _slices.begin();
00068          itSlice != _slices.end(); ++itSlice, ++idx_sublist) {
00069       //
00070       if (idx_sublist != 0) {
00071         oStr << "; ";
00072       }
00073
00074       //
00075       const StringPartition& lStringPartition = *itSlice;
00076
00077       //
00078       oStr << idx_sublist << ". " << lStringPartition;
00079     }
00080
00081     //
00082     oStr << " ]";
00083
00084     return oStr.str();
00085   }
00086
00087   // ////////////////////////////////////////////////////////////////////
00088   void QuerySlices::toStream (std::ostream& ioOut) const {
00089     ioOut << describe();
00090   }
00091
00092   // ////////////////////////////////////////////////////////////////////
00093   void QuerySlices::fromStream (std::istream& ioIn) {
00094   }
00095
00107   // ////////////////////////////////////////////////////////////////////
00108   static unsigned int calculateEditDistance (const
      TravelQuery_T& iPhrase) {
00109     NbOfErrors_T oEditDistance = 2;
00110
00111     const NbOfErrors_T lQueryStringSize = iPhrase.size();
00112
00113     oEditDistance = lQueryStringSize / K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT
      ;
00114     return oEditDistance;
00115   }
00116
00120   // ////////////////////////////////////////////////////////////////////
00121   bool doesMatch (const Xapian::Database& iDatabase,
00122                   const std::string& iWord1, const std::string& iWord2) {
00123     bool oDoesMatch = false;
00124
00125     //
00126     std::ostringstream oStr;
00127     oStr << iWord1 << " " << iWord2;
00128     const std::string lQueryString (oStr.str());
00129
00130     // Catch any Xapian::Error exceptions thrown
00131     Xapian::MSet lMatchingSet;
00132     try {
00133
00134       // Build the query object
00135       Xapian::QueryParser lQueryParser;
00136       lQueryParser.set_database (iDatabase);
00137
00143       // lQueryParser.set_default_op (Xapian::Query::OP_ADJ);
00144       lQueryParser.set_default_op (Xapian::Query::OP_PHRASE);
00145
00146       // DEBUG
```

```
00147       /*
00148         OPENTREP_LOG_DEBUG ("Query parser `" << lQueryParser.get_description()
00149         << "'");
00150       */
00151
00152       // DEBUG
00153       // OPENTREP_LOG_DEBUG ("        --------");
00154
00155       // Start an enquire session
00156       Xapian::Enquire enquire (iDatabase);
00157
00164       const Xapian::Query& lXapianQuery =
00165         lQueryParser.parse_query (lQueryString,
00166                                   Xapian::QueryParser::FLAG_BOOLEAN
00167                                   | Xapian::QueryParser::FLAG_PHRASE
00168                                   | Xapian::QueryParser::FLAG_LOVEHATE);
00169
00170       // Give the query object to the enquire session
00171       enquire.set_query (lXapianQuery);
00172
00173       // Get the top 20 results of the query
00174       lMatchingSet = enquire.get_mset (0, 20);
00175
00176       // Display the results
00177       int nbMatches = lMatchingSet.size();
00178
00179       // DEBUG
00180       /*
00181       OPENTREP_LOG_DEBUG ("      Query string: `" << lQueryString
00182                           << "', i.e.: `" << lXapianQuery.get_description()
00183                           << "' => " << nbMatches << " result(s) found");
00184       */
00185
00186       if (nbMatches != 0) {
00187         // There has been a matching
00188         oDoesMatch = true;
00189
00190         // DEBUG
00191         /*
00192         OPENTREP_LOG_DEBUG ("        Query string: `" << lQueryString
00193                             << "' provides " << nbMatches << " exact matches.");
00194         */
00195
00196         return oDoesMatch;
00197       }
00198       assert (lMatchingSet.empty() == true);
00199
00205       const NbOfErrors_T& lAllowableEditDistance =
00206         calculateEditDistance (lQueryString);
00207
00208       // Let Xapian find a spelling correction (if any)
00209       const std::string& lCorrectedString =
00210         iDatabase.get_spelling_suggestion (lQueryString, lAllowableEditDistance);
00211
00212       // If the correction is no better than the original string, there is
00213       // no need to go further: there is no match.
00214       if (lCorrectedString.empty() == true || lCorrectedString == lQueryString) {
00215         // DEBUG
00216         /*
00217         OPENTREP_LOG_DEBUG ("        Query string: `"
00218                             << lQueryString << "' provides no match, "
00219                             << "and there is no spelling suggestion, "
00220                             << "even with an edit distance of "
00221                             << lAllowableEditDistance);
00222         */
00223
00224         // No match
00225         return oDoesMatch;
00226       }
00227       assert (lCorrectedString.empty() == false
00228               && lCorrectedString != lQueryString);
00229
00230       // Calculate the effective (Levenshtein) edit distance/error
00231       const NbOfErrors_T& lEditDistance =
00232         Levenshtein::getDistance (lQueryString, lCorrectedString);
00233
00241       const Xapian::Query& lCorrectedXapianQuery =
00242         lQueryParser.parse_query (lCorrectedString,
00243                                   Xapian::QueryParser::FLAG_BOOLEAN
00244                                   | Xapian::QueryParser::FLAG_PHRASE
00245                                   | Xapian::QueryParser::FLAG_LOVEHATE);
00246
00247       enquire.set_query (lCorrectedXapianQuery);
00248       lMatchingSet = enquire.get_mset (0, 20);
00249
00250       // Display the results
00251       nbMatches = lMatchingSet.size();
```

```
00252
00253       // DEBUG
00254       /*
00255       OPENTREP_LOG_DEBUG ("      Corrected query string: `" << lCorrectedString
00256                           << "', i.e.: `"
00257                           << lCorrectedXapianQuery.get_description()
00258                           << "' => " << nbMatches << " result(s) found");
00259       */
00260
00261       if (nbMatches != 0) {
00262         // DEBUG
00263         /*
00264         OPENTREP_LOG_DEBUG ("        Query string: `"
00265                             << lQueryString << "', spelling suggestion: `"
00266                             << lCorrectedString
00267                             << "', with a Levenshtein edit distance of "
00268                             << lEditDistance
00269                             << " over an allowable edit distance of "
00270                             << lAllowableEditDistance << ", provides "
00271                             << nbMatches << " matches.");
00272         */
00273
00274         //
00275         oDoesMatch = true;
00276         return oDoesMatch;
00277       }
00278
00279       // Error
00280       OPENTREP_LOG_ERROR ("        Query string: `"
00281                           << lQueryString << "', spelling suggestion: `"
00282                           << lCorrectedString
00283                           << "', with a Levenshtein edit distance of "
00284                           << lEditDistance
00285                           << " over an allowable edit distance of "
00286                           << lAllowableEditDistance << ", provides no match, "
00287                           << "which is not consistent with the existence of "
00288                           << "the spelling correction.");
00289       assert (false);
00290
00291     } catch (const Xapian::Error& error) {
00292       // Error
00293       OPENTREP_LOG_ERROR ("Exception: "  << error.get_msg());
00294       throw XapianException (error.get_msg());
00295     }
00296
00297     return oDoesMatch;
00298   }
00299
00300   // //////////////////////////////////////////////////////////////////////
00301   void QuerySlices::init() {
00302     // 0. Initialisation
00303     // 0.1. Initialisation of the tokenizer
00304     WordList_T lWordList;
00305     tokeniseStringIntoWordList (_queryString, lWordList);
00306     const unsigned short nbOfWords = lWordList.size();
00307
00308     // When the query has a single word, stop here, as there is a single slice
00309     if (nbOfWords <= 1) {
00310       _slices.push_back (_queryString);
00311       return;
00312     }
00313
00314     // 0.2. Re-create the initial phrase, without any (potential) seperator
00315     const std::string lPhrase = createStringFromWordList (lWordList);
00316
00317     // 1. Browse the words, two by two, and check whether their association
00318     //    matches with the Xapian index
00319     WordList_T::const_iterator itWord = lWordList.begin();
00320     WordList_T::const_iterator itNextWord = lWordList.begin(); ++itNextWord;
00321     for (unsigned short idx = 1, idx_rel = 1; itNextWord != lWordList.end();
00322          ++itWord, ++itNextWord, ++idx, ++idx_rel) {
00323       const std::string& leftWord = *itWord;
00324       const std::string& rightWord = *itNextWord;
00325
00326       // Store the left word in the staging string
00327       if (idx_rel >= 2) {
00328         _itLeftWords += " ";
00329       }
00330       _itLeftWords += leftWord;
00331
00332       // Check whether the juxtaposition of the two contiguous words matches
00333       const bool lDoesMatch =
00334         OPENTREP::doesMatch (_database, leftWord, rightWord);
00335
00336       if (lDoesMatch == true) {
00337         // When the two words give a match, do nothing now, as at the next turn,
00338         // the right word will become the left word and thus be added to the
```

```
00339         // staging string
00340
00341         // DEBUG
00342         /*
00343         OPENTREP_LOG_DEBUG ("[" << idx << "][" << idx_rel
00344                             << "] Match - staging string: '"
00345                             << _itLeftWords << "'");
00346         */
00347
00348       } else {
00349         // DEBUG
00350         /*
00351         OPENTREP_LOG_DEBUG ("[" << idx << "][" << idx_rel
00352                             << "] No match - staging string: '"
00353                             << _itLeftWords << "'");
00354         */
00355
00356         // When the two words give no match, add the content of the staging
00357         // list to the list of slices. Then, empty the staging string.
00358         _slices.push_back (_itLeftWords);
00359         _itLeftWords = "";
00360         idx_rel = 0;
00361       }
00362     }
00363
00364     // 2.
00365     const std::string& leftWord = *itWord;
00366     if (_itLeftWords.empty() == false) {
00367       _itLeftWords += " ";
00368     }
00369     _itLeftWords += leftWord;
00370     _slices.push_back (_itLeftWords);
00371
00372     // DEBUG
00373     // OPENTREP_LOG_DEBUG ("Last staging string: '" << _itLeftWords << "'");
00374     // OPENTREP_LOG_DEBUG ("Slices: " << *this);
00375   }
00376
00377 }
```

24.143 opentrep/bom/QuerySlices.hpp File Reference

```
#include <string>
#include <list>
#include <xapian.h>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/basic/StructAbstract.hpp>
#include <opentrep/bom/StringPartition.hpp>
```

Classes

- struct OPENTREP::QuerySlices

Namespaces

- OPENTREP

24.144 QuerySlices.hpp

```
00001 #ifndef __OPENTREP_BOM_QUERYSLICES_HPP
00002 #define __OPENTREP_BOM_QUERYSLICES_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <list>
00010 // Xapian
00011 #include <xapian.h>
00012 // OpenTrep
```

```
00013 #include <opentrep/OPENTREP_Types.hpp>
00014 #include <opentrep/basic/StructAbstract.hpp>
00015 #include <opentrep/bom/StringPartition.hpp>
00016
00017 namespace OPENTREP {
00018
00044   struct QuerySlices : public StructAbstract {
00045   public:
00049     const StringPartitionList_T& getStringPartitionList() const
      {
00050       return _slices;
00051     }
00052
00059     void push_back (const StringPartition& iStringPartition);
00060
00064     size_t size() const;
00065
00069     bool empty() const;
00070
00074     void clear();
00075
00076
00077   private:
00084     void init();
00085
00086
00087   public:
00088     // ////////// Display support methods /////////
00094     void toStream (std::ostream& ioOut) const;
00095
00101     void fromStream (std::istream& ioIn);
00102
00107     std::string describeKey() const;
00108
00112     std::string describe() const;
00113
00114
00115   public:
00116     // ///////////////// Constructors and Destructors /////////////
00123     QuerySlices (const Xapian::Database&, const TravelQuery_T&);
00124
00128     ~QuerySlices();
00129
00130
00131   public:
00132     // ///////////////// Attributes ///////////////
00136     const Xapian::Database& _database;
00137
00141     TravelQuery_T _queryString;
00142
00146     StringPartitionList_T _slices;
00147
00151     std::string _itLeftWords;
00152   };
00153
00154 }
00155 #endif // __OPENTREP_BOM_QUERYSLICES_HPP
```

24.145 opentrep/bom/Result.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <algorithm>
#include <boost/tokenizer.hpp>
#include <opentrep/LocationKey.hpp>
#include <opentrep/basic/BasConst_General.hpp>
#include <opentrep/bom/Filter.hpp>
#include <opentrep/bom/WordHolder.hpp>
#include <opentrep/bom/StringPartition.hpp>
#include <opentrep/bom/Levenshtein.hpp>
#include <opentrep/bom/Place.hpp>
#include <opentrep/bom/Result.hpp>
#include <opentrep/bom/PORParserHelper.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

Functions

- static unsigned int OPENTREP::calculateEditDistance (const TravelQuery_T &iPhrase)

  Helper function.

24.146 Result.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <algorithm>
00008 // Boost
00009 #include <boost/tokenizer.hpp>
00010 // OpenTREP
00011 #include <opentrep/LocationKey.hpp>
00012 #include <opentrep/basic/BasConst_General.hpp>
00013 #include <opentrep/bom/Filter.hpp>
00014 #include <opentrep/bom/WordHolder.hpp>
00015 #include <opentrep/bom/StringPartition.hpp>
00016 #include <opentrep/bom/Levenshtein.hpp>
00017 #include <opentrep/bom/Place.hpp>
00018 #include <opentrep/bom/Result.hpp>
00019 #include <opentrep/bom/PORParserHelper.hpp>
00020 #include <opentrep/service/Logger.hpp>
00021
00022 namespace OPENTREP {
00023
00024   // //////////////////////////////////////////////////////////////////////
00025   Result::Result (const TravelQuery_T& iQueryString,
00026                   const Xapian::Database& iDatabase)
00027     : _resultHolder (NULL), _database (iDatabase),
00028       _queryString (iQueryString), _hasFullTextMatched (false),
00029       _bestDocData (RawDataString_T ("")) {
00030     init();
00031   }
00032
00033   // //////////////////////////////////////////////////////////////////////
00034   Result::~Result() {
00035   }
00036
00037   // //////////////////////////////////////////////////////////////////////
00038   void Result::init() {
00039   }
00040
00041   // //////////////////////////////////////////////////////////////////////
00042   std::string Result::describeShortKey() const {
00043     std::ostringstream oStr;
00044     oStr << _queryString;
00045     return oStr.str();
00046   }
00047
00048   // //////////////////////////////////////////////////////////////////////
00049   std::string Result::describeKey() const {
00050     std::ostringstream oStr;
00051     oStr << "'" << describeShortKey() << "' ";
00052     if (_correctedQueryString.empty() == false
00053         && _correctedQueryString != _queryString) {
00054       oStr << "(corrected into '" << _correctedQueryString
00055            << "' with an edit distance/error of " << _editDistance
00056            << " over an allowable distance of " << _allowableEditDistance
00057            << ") - ";
00058     } else {
00059       oStr << "- ";
00060     }
00061     return oStr.str();
00062   }
00063
00064   // //////////////////////////////////////////////////////////////////////
00065   std::string Result::toString() const {
00066     std::ostringstream oStr;
00067     oStr << describeKey();
00068
00069     if (_documentList.empty() == true) {
```

```
00070       oStr << "No match" << std::endl;
00071       return oStr.str();
00072     }
00073     assert (_hasFullTextMatched == true);
00074
00075     unsigned short idx = 0;
00076     for (DocumentList_T::const_iterator itDoc = _documentList.begin();
00077          itDoc != _documentList.end(); ++itDoc, ++idx) {
00078       const XapianDocumentPair_T& lDocumentPair = *itDoc;
00079
00080       const Xapian::Document& lXapianDoc = lDocumentPair.first;
00081       const Xapian::docid& lDocID = lXapianDoc.get_docid();
00082
00083       const ScoreBoard& lScoreBoard = lDocumentPair.second;
00084
00085       if (idx != 0) {
00086         oStr << ", ";
00087       }
00088       oStr << "Doc ID: " << lDocID << ", matching with ("
00089            << lScoreBoard.describe() << "), containing: '"
00090            << lXapianDoc.get_data() << "'";
00091     }
00092
00093     return oStr.str();
00094   }
00095
00096   // ////////////////////////////////////////////////////////////////////
00097   void Result::toStream (std::ostream& ioOut) const {
00098     ioOut << toString();
00099   }
00100
00101   // ////////////////////////////////////////////////////////////////////
00102   void Result::fromStream (std::istream& ioIn) {
00103   }
00104
00105   // ////////////////////////////////////////////////////////////////////
00106   const XapianDocumentPair_T& Result::
00107   getDocumentPair (const Xapian::docid& iDocID) const {
00108     // Retrieve the Xapian document and associated ScoreBoard structure
00109     // corresponding to the doc ID of the best matching document
00110     DocumentMap_T::const_iterator itDoc = _documentMap.find (iDocID);
00111
00112     if (itDoc == _documentMap.end()) {
00113       OPENTREP_LOG_ERROR ("The Xapian document (ID = " << iDocID
00114                           << ") can not be found in the Result object "
00115                           << describeKey());
00116     }
00117     assert (itDoc != _documentMap.end());
00118
00119     //
00120     const XapianDocumentPair_T& oDocumentPair = itDoc->second;
00121
00122     //
00123     return oDocumentPair;
00124   }
00125
00126   // ////////////////////////////////////////////////////////////////////
00127   const Xapian::Document& Result::
00128   getDocument (const Xapian::docid& iDocID) const {
00129     // First, retrieve the pair made of Xapian document and associated
00130     // ScoreBoard structure
00131     const XapianDocumentPair_T& lDocumentPair =
      getDocumentPair (iDocID);
00132
00133     // Then, take the Xapian document (and leave the ScoreBoard out)
00134     const Xapian::Document& oXapianDocument = lDocumentPair.first;
00135
00136     //
00137     return oXapianDocument;
00138   }
00139
00140   // ////////////////////////////////////////////////////////////////////
00141   void Result::addDocument (const Xapian::Document& iDocument,
00142                             const Score_T& iScore) {
00151     Score_T lCorrectedScore = iScore;
00152     if (_editDistance > 0) {
00153       lCorrectedScore = iScore / (_editDistance * _editDistance * _editDistance);
00154     }
00155
00156     // The document is created at the time of (Xapian-based) full-text matching
00157     const ScoreType lXapianScoreType (ScoreType::XAPIAN_PCT);
00158
00159     // Create a ScoreBoard structure
00160     const ScoreBoard lScoreBoard (_queryString,
00161                                   lXapianScoreType, lCorrectedScore);
00162
00163     // Retrieve the ID of the Xapian document
```

```
00164     const Xapian::docid& lDocID = iDocument.get_docid();
00165
00175     // Create a (Xapian document, score board) pair, so as to store
00176     // the document along with its corresponding score board
00177     const XapianDocumentPair_T lDocumentPair (iDocument, lScoreBoard);
00178
00179     // Insert the just created pair into the dedicated (STL) list
00180     _documentList.push_back (lDocumentPair);
00181
00182     // Insert the just created pair into the dedicated (STL) map
00183     const bool hasInsertBeenSuccessful =
00184       _documentMap.insert (DocumentMap_T::value_type (lDocID,
00185                                                       lDocumentPair)).second;
00186     // Sanity check
00187     assert (hasInsertBeenSuccessful == true);
00188   }
00189
00190   // ////////////////////////////////////////////////////////////////////
00191   void Result::fillResult (const Xapian::MSet& iMatchingSet) {
00196     for (Xapian::MSetIterator itDoc = iMatchingSet.begin();
00197          itDoc != iMatchingSet.end(); ++itDoc) {
00198       const Xapian::percent& lXapianPercentage = itDoc.get_percent();
00199       const Xapian::Document& lDocument = itDoc.get_document();
00200       addDocument (lDocument, lXapianPercentage);
00201     }
00202   }
00203
00204   // ////////////////////////////////////////////////////////////////////
00205   void Result::fillPlace (Place& ioPlace) const {
00206     // Set the original and corrected/suggested keywords
00207     ioPlace.setOriginalKeywords (_queryString);
00208     ioPlace.setCorrectedKeywords (_correctedQueryString);
00209
00210     // Set the effective (Levenshtein) edit distance/error, as
00211     // well as the allowable edit distance/error
00212     ioPlace.setEditDistance (_editDistance);
00213     ioPlace.setAllowableEditDistance (_allowableEditDistance);
00214
00215     // Set the Xapian document ID
00216     ioPlace.setDocID (_bestDocID);
00217
00218     // Set the matching percentage
00219     ioPlace.setPercentage (_bestCombinedWeight);
00220
00221     // Retrieve the parameters of the best matching document
00222     const LocationKey& lKey = ioPlace.getKey();
00223
00224     // DEBUG
00225     OPENTREP_LOG_DEBUG ("Place key: " << lKey << " - Xapian ID " << _bestDocID
00226                         << ", " << _bestCombinedWeight << "% [" << _bestDocData
00227                         << "]");
00228   }
00229
00241   // ////////////////////////////////////////////////////////////////////
00242   static unsigned int calculateEditDistance (const
      TravelQuery_T& iPhrase) {
00243     NbOfErrors_T oEditDistance = 2;
00244
00245     const NbOfErrors_T lQueryStringSize = iPhrase.size();
00246
00247     oEditDistance = lQueryStringSize / K_DEFAULT_SIZE_FOR_SPELLING_ERROR_UNIT
      ;
00248     return oEditDistance;
00249   }
00250
00251   // ////////////////////////////////////////////////////////////////////
00252   Location Result::retrieveLocation (const
      RawDataString_T& iRawDataString) {
00253     // Initialise the POR (point of reference) parser
00254     PORStringParser lStringParser (iRawDataString);
00255
00256     // Parse the raw data
00257     const Location& oLocation = lStringParser.generateLocation();
00258
00259     // DEBUG
00260     //OPENTREP_LOG_DEBUG ("Location: " << oLocation);
00261
00262     return oLocation;
00263   }
00264
00265   // ////////////////////////////////////////////////////////////////////
00266   Location Result::retrieveLocation (const Xapian::Document& iDocument) {
00267     // Retrieve the Xapian document data
00268     const std::string& lDocumentDataStr = iDocument.get_data();
00269     const RawDataString_T& lDocumentData = RawDataString_T (lDocumentDataStr)
      ;
00270
```

```
00271     // Parse the POR details and create the corresponding Location structure
00272     const Location& oLocation = retrieveLocation (lDocumentData);
00273
00274     return oLocation;
00275   }
00276
00277   // ////////////////////////////////////////////////////////////////////
00278   LocationKey Result::getPrimaryKey (const Xapian::Document& iDocument) {
00279     // Parse the POR (point of reference) details held by the Xapian document
00280     const Location& lLocation = retrieveLocation (iDocument);
00281
00282     // Get the key (IATA and ICAO codes, GeonamesID)
00283     const LocationKey& oLocationKey = lLocation.getKey();
00284
00285     return oLocationKey;
00286   }
00287
00288   // ////////////////////////////////////////////////////////////////////
00289   Score_T Result::getEnvelopeID (const Xapian::Document& iDocument) {
00290     // Parse the POR (point of reference) details held by the Xapian document
00291     const Location& lLocation = retrieveLocation (iDocument);
00292
00293     // Get the envelope ID (it is an integer value in the Location structure)
00294     const EnvelopeID_T& lEnvelopeIDInt = lLocation.getEnvelopeID();
00295
00296     // Convert the envelope ID value, from an integer to a floating point one
00297     const Score_T oEnvelopeID = static_cast<const Score_T> (lEnvelopeIDInt);
00298
00299     return oEnvelopeID;
00300   }
00301
00302   // ////////////////////////////////////////////////////////////////////
00303   PageRank_T Result::getPageRank (const Xapian::Document& iDocument) {
00304     // Parse the POR (point of reference) details held by the Xapian document
00305     const Location& lLocation = retrieveLocation (iDocument);
00306
00307     // Get the PageRank value
00308     const PageRank_T& oPageRank = lLocation.getPageRank();
00309
00310     return oPageRank;
00311   }
00312
00313   // ////////////////////////////////////////////////////////////////////
00314   std::string Result::fullTextMatch (const Xapian::Database& iDatabase,
00315                                      const TravelQuery_T& iQueryString,
00316                                      Xapian::MSet& ioMatchingSet) {
00317     std::string oMatchedString;
00318
00319     // Catch any Xapian::Error exceptions thrown
00320     try {
00321
00322       // Build the query object
00323       Xapian::QueryParser lQueryParser;
00324       lQueryParser.set_database (iDatabase);
00325
00331       // lQueryParser.set_default_op (Xapian::Query::OP_ADJ);
00332       lQueryParser.set_default_op (Xapian::Query::OP_PHRASE);
00333
00334       // DEBUG
00335       /*
00336         OPENTREP_LOG_DEBUG ("Query parser `" << lQueryParser.get_description()
00337         << "'");
00338       */
00339
00340       // DEBUG
00341       OPENTREP_LOG_DEBUG ("        --------");
00342
00343       // Start an enquire session
00344       Xapian::Enquire enquire (iDatabase);
00345
00352       const Xapian::Query& lXapianQuery =
00353         lQueryParser.parse_query (iQueryString,
00354                                   Xapian::QueryParser::FLAG_BOOLEAN
00355                                   | Xapian::QueryParser::FLAG_PHRASE
00356                                   | Xapian::QueryParser::FLAG_LOVEHATE);
00357
00358       // Give the query object to the enquire session
00359       enquire.set_query (lXapianQuery);
00360
00361       // Get the top K_DEFAULT_XAPIAN_MATCHING_SET_SIZE (normally, 30)
00362       // results of the query
00363       ioMatchingSet = enquire.get_mset (0, K_DEFAULT_XAPIAN_MATCHING_SET_SIZE
      );
00364
00365       // Display the results
00366       int nbMatches = ioMatchingSet.size();
00367
```

```
00368       // DEBUG
00369       OPENTREP_LOG_DEBUG ("      Query string: `" << iQueryString
00370                           << "', i.e.: `" << lXapianQuery.get_description()
00371                           << "' => " << nbMatches << " result(s) found");
00372
00373       if (nbMatches != 0) {
00374         // Store the effective (Levenshtein) edit distance/error
00375         const NbOfErrors_T lEditDistance = 0;
00376         setEditDistance (lEditDistance);
00377
00378         // Store the allowable edit distance/error
00379         setAllowableEditDistance (lEditDistance);
00380
00381         //
00382         oMatchedString = iQueryString;
00383
00384         // Store the fact that there has been a full-text match
00385         setHasFullTextMatched (true);
00386
00387         // Store the corrected string (the same as the given string, here,
00388         // as that latter directly gave full-text matches).
00389         setCorrectedQueryString (oMatchedString);
00390
00391         // DEBUG
00392         OPENTREP_LOG_DEBUG ("        Query string: `" << iQueryString
00393                             << "' provides " << nbMatches << " exact matches.");
00394
00395         return oMatchedString;
00396       }
00397       assert (ioMatchingSet.empty() == true);
00398
00404       const NbOfErrors_T& lAllowableEditDistance =
00405         calculateEditDistance (iQueryString);
00406
00407       // Let Xapian find a spelling correction (if any)
00408       const std::string& lCorrectedString =
00409         iDatabase.get_spelling_suggestion (iQueryString, lAllowableEditDistance);
00410
00411       // If the correction is no better than the original string, there is
00412       // no need to go further: there is no match.
00413       if (lCorrectedString.empty() == true || lCorrectedString == iQueryString) {
00414         // DEBUG
00415         OPENTREP_LOG_DEBUG ("        Query string: `"
00416                             << iQueryString << "' provides no match, "
00417                             << "and there is no spelling suggestion, "
00418                             << "even with an edit distance of "
00419                             << lAllowableEditDistance);
00420
00421         // Store the fact that there has not been any full-text match
00422         setHasFullTextMatched (false);
00423
00424         // Leave the string empty
00425         return oMatchedString;
00426       }
00427       assert (lCorrectedString.empty() == false
00428               && lCorrectedString != iQueryString);
00429
00430       // Calculate the effective (Levenshtein) edit distance/error
00431       const NbOfErrors_T& lEditDistance =
00432         Levenshtein::getDistance (iQueryString, lCorrectedString);
00433
00441       const Xapian::Query& lCorrectedXapianQuery =
00442         lQueryParser.parse_query (lCorrectedString,
00443                                   Xapian::QueryParser::FLAG_BOOLEAN
00444                                   | Xapian::QueryParser::FLAG_PHRASE
00445                                   | Xapian::QueryParser::FLAG_LOVEHATE);
00446
00447       // Retrieve a maximum of K_DEFAULT_XAPIAN_MATCHING_SET_SIZE (normally,
00448       // 30) entries
00449       enquire.set_query (lCorrectedXapianQuery);
00450       ioMatchingSet = enquire.get_mset (0, K_DEFAULT_XAPIAN_MATCHING_SET_SIZE
    );
00451
00452       // Display the results
00453       nbMatches = ioMatchingSet.size();
00454
00455       // DEBUG
00456       OPENTREP_LOG_DEBUG ("      Corrected query string: `" << lCorrectedString
00457                           << "', i.e.: `"
00458                           << lCorrectedXapianQuery.get_description()
00459                           << "' => " << nbMatches << " result(s) found");
00460
00461       if (nbMatches != 0) {
00462         // Store the effective (Levenshtein) edit distance/error
00463         setEditDistance (lEditDistance);
00464
00465         // Store the allowable edit distance/error
```

```
00466         setAllowableEditDistance (lAllowableEditDistance);
00467
00468         //
00469         oMatchedString = lCorrectedString;
00470
00471         // Store the fact that there has been a full-text match
00472         setHasFullTextMatched (true);
00473
00474         // Store the corrected string
00475         setCorrectedQueryString (oMatchedString);
00476
00477         // DEBUG
00478         OPENTREP_LOG_DEBUG ("        Query string: `"
00479                             << iQueryString << "', spelling suggestion: `"
00480                             << lCorrectedString
00481                             << "', with a Levenshtein edit distance of "
00482                             << lEditDistance
00483                             << " over an allowable edit distance of "
00484                             << lAllowableEditDistance << ", provides "
00485                             << nbMatches << " matches.");
00486
00487         //
00488         return oMatchedString;
00489       }
00490
00491       // Error
00492       OPENTREP_LOG_ERROR ("        Query string: `"
00493                           << iQueryString << "', spelling suggestion: `"
00494                           << lCorrectedString
00495                           << "', with a Levenshtein edit distance of "
00496                           << lEditDistance
00497                           << " over an allowable edit distance of "
00498                           << lAllowableEditDistance << ", provides no match, "
00499                           << "which is not consistent with the existence of "
00500                           << "the spelling correction.");
00501       assert (false);
00502
00503     } catch (const Xapian::Error& error) {
00504       OPENTREP_LOG_ERROR ("Exception: "  << error.get_msg());
00505       throw XapianException (error.get_msg());
00506     }
00507
00508     // Store the fact that there has not been any full-text match
00509     setHasFullTextMatched (false);
00510
00511     return oMatchedString;
00512   }
00513
00514   // ////////////////////////////////////////////////////////////////////
00515   std::string Result::fullTextMatch (const Xapian::Database& iDatabase,
00516                                      const TravelQuery_T& iQueryString) {
00517     std::string oMatchedString;
00518
00519     // Catch any Xapian::Error exceptions thrown
00520     try {
00521
00522       // DEBUG
00523       OPENTREP_LOG_DEBUG ("      ----------------");
00524       OPENTREP_LOG_DEBUG ("      Current query string: '"<< iQueryString << "'");
00525
00526       // Check whether the string should be filtered out
00527       const bool isToBeAdded = Filter::shouldKeep ("", iQueryString);
00528       //const bool isToBeAdded = true;
00529
00530       Xapian::MSet lMatchingSet;
00531       if (isToBeAdded == true) {
00532         oMatchedString = fullTextMatch (iDatabase, iQueryString, lMatchingSet);
00533       }
00534
00535       // Create the corresponding documents (from the Xapian MSet object)
00536       fillResult (lMatchingSet);
00537
00538       // DEBUG
00539       if (isToBeAdded == false) {
00540         OPENTREP_LOG_DEBUG ("      No full text search performed as '"
00541                             << iQueryString
00542                             << "' is not made of searchable words");
00543       }
00544       OPENTREP_LOG_DEBUG ("      ==> " << toString());
00545       OPENTREP_LOG_DEBUG ("      ----------------");
00546
00547     } catch (const Xapian::Error& error) {
00548       OPENTREP_LOG_ERROR ("Xapian-related error: "  << error.get_msg());
00549       throw XapianException (error.get_msg());
00550     }
00551
00552     return oMatchedString;
```

```
00553   }
00554
00555   // ////////////////////////////////////////////////////////////////////
00556   void Result::displayXapianPercentages() const {
00557     // Browse the list of Xapian documents
00558     for (DocumentList_T::const_iterator itDoc = _documentList.begin();
00559          itDoc != _documentList.end(); ++itDoc) {
00560       const XapianDocumentPair_T& lDocumentPair = *itDoc;
00561
00562       // Retrieve the Xapian document
00563       const Xapian::Document& lXapianDoc = lDocumentPair.first;
00564
00565       // Extract the Xapian document ID
00566       const Xapian::docid& lDocID = lXapianDoc.get_docid();
00567
00568       // Extract the envelope ID from the document data
00569       const LocationKey& lLocationKey = getPrimaryKey (lXapianDoc);
00570
00571       // Retrieve the score board for that Xapian document
00572       const ScoreBoard& lScoreBoard = lDocumentPair.second;
00573
00574       // Extract the Xapian matching percentage
00575       const Score_T& lXapianPct = lScoreBoard.getScore (
    ScoreType::XAPIAN_PCT);
00576
00577       // DEBUG
00578       OPENTREP_LOG_NOTIFICATION ("        [xapian] '" <<
    describeShortKey()
00579                                  << "' with (" << lLocationKey << ", doc ID = "
00580                                  << lDocID << ") matches at " << lXapianPct
00581                                  << "%");
00582     }
00583   }
00584
00585   // ////////////////////////////////////////////////////////////////////
00586   void Result::setScoreOnDocMap (const Xapian::docid& iDocID,
00587                                  const ScoreType& iType, const Score_T& iScore) {
00588     // Retrieve the Xapian document and associated ScoreBoard structure
00589     // corresponding to the given doc ID
00590     DocumentMap_T::iterator itDoc = _documentMap.find (iDocID);
00591
00592     if (itDoc == _documentMap.end()) {
00593       OPENTREP_LOG_ERROR ("The Xapian document (ID = " << iDocID
00594                           << ") can not be found in the Result object "
00595                           << describeKey());
00596     }
00597     assert (itDoc != _documentMap.end());
00598
00599     // Retrieve the associated ScoreBoard structure
00600     XapianDocumentPair_T& lXapianDocPair = itDoc->second;
00601     ScoreBoard& lScoreBoard = lXapianDocPair.second;
00602
00603     // Update the score/weight
00604     lScoreBoard.setScore (iType, iScore);
00605   }
00606
00607   // ////////////////////////////////////////////////////////////////////
00608   void Result::calculateEnvelopeWeights() {
00609     // Browse the list of Xapian documents
00610     for (DocumentList_T::iterator itDoc = _documentList.begin();
00611          itDoc != _documentList.end(); ++itDoc) {
00612       XapianDocumentPair_T& lDocumentPair = *itDoc;
00613
00614       // Retrieve the Xapian document
00615       const Xapian::Document& lXapianDoc = lDocumentPair.first;
00616
00617       // Extract the Xapian document ID
00618       const Xapian::docid& lDocID = lXapianDoc.get_docid();
00619
00620       // Extract the envelope ID from the document data
00621       const LocationKey& lLocationKey = getPrimaryKey (lXapianDoc);
00622
00623       // Extract the envelope ID from the document data
00624       const EnvelopeID_T& lEnvelopeIDInt = getEnvelopeID (lXapianDoc);
00625
00626       // DEBUG
00627       if (lEnvelopeIDInt != 0) {
00628         OPENTREP_LOG_NOTIFICATION ("        [env][" <<
    describeShortKey()
00629                                    << "] (" << lLocationKey << ", doc ID = "
00630                                    << lDocID << ") has a non-null envelope ID ("
00631                                    << lEnvelopeIDInt << ") => match of 0.10%");
00632       }
00633
00634       // Convert the envelope ID value, from an integer to a floating point one
00635       const Score_T lEnvelopeID = static_cast<const Score_T> (lEnvelopeIDInt);
00636
```

```
00637       // Retrieve the score board for that Xapian document
00638       ScoreBoard& lScoreBoard = lDocumentPair.second;
00639
00640       // Store the envelope-related weight
00641       lScoreBoard.setScore (ScoreType::ENV_ID, lEnvelopeID);
00642       setScoreOnDocMap (lDocID, ScoreType::ENV_ID, lEnvelopeID);
00643     }
00644   }
00645
00646   // ////////////////////////////////////////////////////////////////////
00647   void Result::calculateCodeMatches() {
00648     // Browse the list of Xapian documents
00649     for (DocumentList_T::iterator itDoc = _documentList.begin();
00650          itDoc != _documentList.end(); ++itDoc) {
00651       XapianDocumentPair_T& lDocumentPair = *itDoc;
00652
00653       // Retrieve the Xapian document
00654       const Xapian::Document& lXapianDoc = lDocumentPair.first;
00655
00656       // Extract the Xapian document ID
00657       const Xapian::docid& lDocID = lXapianDoc.get_docid();
00658
00659       // Extract the envelope ID from the document data
00660       const LocationKey& lLocationKey = getPrimaryKey (lXapianDoc);
00661
00662       // Initialisation of the IATA/ICAO code full matching percentage
00663       Score_T lCodeMatchPct = 0.0;
00664       bool hasCodeFullyMatched = false;
00665
00666       // Filter out "standard" words such as "airport", "international",
00667       // "city", as well as words having a length strictly less than
00668       // 3 letters.
00669       std::string lFilteredString (_queryString);
00670       const NbOfLetters_T kMinWordLength = 3;
00671       Filter::trim (lFilteredString, kMinWordLength);
00672
00673       // Check whether or not the filtered query string is made of
00674       // a single word
00675       WordList_T lFilteredQueryWordList;
00676       WordHolder::tokeniseStringIntoWordList (lFilteredString,
00677                                               lFilteredQueryWordList);
00678       const NbOfWords_T nbOfFilteredQueryWords = lFilteredQueryWordList.size();
00679
00680       //
00681       if (_hasFullTextMatched == true) {
00688         const size_t lNbOfLetters = lFilteredString.size();
00689         if (nbOfFilteredQueryWords == 1
00690             && lNbOfLetters >= 3 && lNbOfLetters <= 4
00691             && _correctedQueryString == _queryString) {
00692           // Convert the query string (made of one word of 3 or 4 letters)
00693           // to uppercase letters
00694           std::string lUpperQueryWord;
00695           lUpperQueryWord.resize (lNbOfLetters);
00696           std::transform (lFilteredString.begin(), lFilteredString.end(),
00697                           lUpperQueryWord.begin(), ::toupper);
00698
00699           // Retrieve with the IATA code
00700           const IATACode_T& lIataCode = lLocationKey.getIataCode();
00701
00702           // Compare the 3/4-letter-word query string with the IATA
00703           // and ICAO codes
00704           if (lUpperQueryWord == lIataCode) {
00708             lCodeMatchPct = 1.0;
00709             hasCodeFullyMatched = true;
00710           }
00711         }
00712
00713         if (hasCodeFullyMatched == true) {
00714           // DEBUG
00715           OPENTREP_LOG_NOTIFICATION ("        [code] '" <<
      describeShortKey()
00716                                      << "' matches the IATA/ICAO code ("
00717                                      << lLocationKey << ", doc ID = "
00718                                      << lDocID << ") => match of "
00719                                      << K_DEFAULT_FULL_CODE_MATCH_PCT << "%");
00720         } else {
00721           // DEBUG
00722           OPENTREP_LOG_NOTIFICATION ("        [code] '" <<
      describeShortKey()
00723                                      << "' does not match with the IATA/ICAO "
00724                                      << "code (" << lLocationKey << ", doc ID = "
00725                                      << lDocID << ") => match of "
00726                                      << K_DEFAULT_MODIFIED_MATCHING_PCT << "
      %");
00727         }
00728       }
00729
```

```
00730       // Retrieve the score board for that Xapian document
00731       ScoreBoard& lScoreBoard = lDocumentPair.second;
00732
00733       // Store the IATA/ICAO code match percentage/weight
00734       lScoreBoard.setScore (ScoreType::CODE_FULL_MATCH, lCodeMatchPct);
00735       setScoreOnDocMap (lDocID, ScoreType::CODE_FULL_MATCH,
    lCodeMatchPct);
00736     }
00737   }
00738
00739   // ////////////////////////////////////////////////////////////////////
00740   void Result::calculatePageRanks() {
00741     // Browse the list of Xapian documents
00742     for (DocumentList_T::iterator itDoc = _documentList.begin();
00743          itDoc != _documentList.end(); ++itDoc) {
00744       XapianDocumentPair_T& lDocumentPair = *itDoc;
00745
00746       // Retrieve the Xapian document
00747       const Xapian::Document& lXapianDoc = lDocumentPair.first;
00748
00749       // Extract the Xapian document ID
00750       const Xapian::docid& lDocID = lXapianDoc.get_docid();
00751
00752       // Extract the envelope ID from the document data
00753       const LocationKey& lLocationKey = getPrimaryKey (lXapianDoc);
00754
00755       // Extract the PageRank from the document data
00756       const Score_T& lPageRank = getPageRank (lXapianDoc);
00757
00758       // DEBUG
00759       OPENTREP_LOG_NOTIFICATION ("        [pr][" <<
    describeShortKey()
00760                                  << "] (" << lLocationKey << ", doc ID = "
00761                                  << lDocID << ") has a PageRank of "
00762                                  << lPageRank << "%");
00763
00764       // Retrieve the score board for that Xapian document
00765       ScoreBoard& lScoreBoard = lDocumentPair.second;
00766
00767       // Store the PageRank weight
00768       lScoreBoard.setScore (ScoreType::PAGE_RANK, lPageRank);
00769       setScoreOnDocMap (lDocID, ScoreType::PAGE_RANK, lPageRank);
00770     }
00771   }
00772
00773   // ////////////////////////////////////////////////////////////////////
00774   void Result::calculateHeuristicWeights() {
00780   }
00781
00782   // ////////////////////////////////////////////////////////////////////
00783   void Result::calculateCombinedWeights() {
00784     Percentage_T lMaxPercentage = 0.0;
00785     std::string lBestDocData;
00786
00787     // Browse the list of Xapian documents
00788     Xapian::docid lBestDocID = 0;
00789     for (DocumentList_T::iterator itDoc = _documentList.begin();
00790          itDoc != _documentList.end(); ++itDoc) {
00791       XapianDocumentPair_T& lDocumentPair = *itDoc;
00792
00793       // Retrieve the Xapian document ID
00794       const Xapian::Document& lXapianDoc = lDocumentPair.first;
00795       const Xapian::docid& lDocID = lXapianDoc.get_docid();
00796       const std::string& lDocData = lXapianDoc.get_data();
00797
00802       ScoreBoard& lScoreBoard = lDocumentPair.second;
00803       const Percentage_T& lPercentage = lScoreBoard.
    calculateCombinedWeight();
00804
00814       // Register the document, if it is the best matching until now
00815       if (lPercentage > lMaxPercentage) {
00816         lMaxPercentage = lPercentage;
00817         lBestDocID = lDocID;
00818         lBestDocData = lDocData;
00819       }
00820     }
00821
00822     // Check whether or not the (original) query string is made of a single word
00823     WordList_T lOriginalQueryWordList;
00824     WordHolder::tokeniseStringIntoWordList (_queryString,
00825                                             lOriginalQueryWordList);
00826     const NbOfWords_T nbOfOriginalQueryWords = lOriginalQueryWordList.size();
00827
00828     //
00829     if (_hasFullTextMatched == true) {
00830       // Retrieve the primary key (IATA, location type, Geonames ID) of
00831       // the place corresponding to the document
```

```
00832       const XapianDocumentPair_T& lXapianDocPair =
      getDocumentPair (lBestDocID);
00833       const Xapian::Document& lXapianDoc = lXapianDocPair.first;
00834       const ScoreBoard& lScoreBoard = lXapianDocPair.second;
00835       const LocationKey& lLocationKey = getPrimaryKey (lXapianDoc);
00836
00837       // DEBUG
00838       OPENTREP_LOG_DEBUG ("        [pct] '" <<
      describeShortKey()
00839                           << "' matches at " << lMaxPercentage
00840                           << "% for " << lLocationKey << " (doc ID = "
00841                           << lBestDocID << "). Score calculation: "
00842                           << lScoreBoard.describe());
00843
00844     } else {
00849       const bool shouldBeKept = Filter::shouldKeep ("", _queryString);
00850
00851       if (nbOfOriginalQueryWords == 1 && shouldBeKept == true) {
00857         lMaxPercentage = 100.0;
00858
00859         // DEBUG
00860         OPENTREP_LOG_DEBUG ("        [pct] '" <<
      describeShortKey()
00861                             << "' does not match, but it is a non black-listed "
00862                             << "single-word string; hence, the weight is "
00863                             << lMaxPercentage << "%");
00864
00865       } else {
00874         lMaxPercentage = std::pow (10.0, -3*nbOfOriginalQueryWords);
00875
00876         // DEBUG
00877         OPENTREP_LOG_DEBUG("        [pct] '" <<
      describeShortKey()
00878                            << "' does not match, and is either a multiple-word "
00879                            << "string or black-listed; hence, the weight is "
00880                            << lMaxPercentage << "%");
00881       }
00882     }
00883
00884     // Store the doc ID of the best matching document
00885     setBestDocID (lBestDocID);
00886
00887     // Store the best weight
00888     setBestCombinedWeight (lMaxPercentage);
00889
00890     // Store all the details of the Xapian document
00891     setBestDocData (lBestDocData);
00892   }
00893
00894 }
```

24.147 opentrep/bom/Result.hpp File Reference

```
#include <list>
#include <map>
#include <xapian.h>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/bom/BomAbstract.hpp>
#include <opentrep/bom/ScoreBoard.hpp>
```

Classes

- class OPENTREP::Result

  Class wrapping a set of Xapian documents having matched a given query string.

Namespaces

- OPENTREP

Typedefs

- typedef std::pair
  < Xapian::Document, ScoreBoard > OPENTREP::XapianDocumentPair_T
- typedef std::map
  < Xapian::docid,
  XapianDocumentPair_T > OPENTREP::DocumentMap_T

24.148 Result.hpp

```
00001 #ifndef __OPENTREP_BOM_RESULT_HPP
00002 #define __OPENTREP_BOM_RESULT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <list>
00009 #include <map>
00010 // Xapian
00011 #include <xapian.h>
00012 // OpenTREP
00013 #include <opentrep/OPENTREP_Types.hpp>
00014 #include <opentrep/bom/BomAbstract.hpp>
00015 #include <opentrep/bom/ScoreBoard.hpp>
00016
00017 namespace OPENTREP {
00018
00019   // Forward declarations
00020   class ResultHolder;
00021   struct LocationKey;
00022   struct Location;
00023   class Place;
00024
00025
00026   // ////////////////////// Type definitions ///////////////////////
00030   typedef std::pair<Xapian::Document, ScoreBoard> XapianDocumentPair_T;
00031
00035   typedef std::list<XapianDocumentPair_T> DocumentList_T;
00036
00040   typedef std::map<Xapian::docid, XapianDocumentPair_T> DocumentMap_T;
00041
00042
00043   // ////////////////////////// Main Class ////////////////////////////
00048   class Result : public BomAbstract {
00049     friend class FacResultHolder;
00050     friend class FacResult;
00051   public:
00052     // ////////////////////// Getters ///////////////////////
00056     const TravelQuery_T& getQueryString() const {
00057       return _queryString;
00058     }
00059
00064     const TravelQuery_T& getCorrectedTravelQuery() const {
00065      return _correctedQueryString;
00066     }
00067
00071     bool hasFullTextMatched() const {
00072       return _hasFullTextMatched;
00073     }
00074
00078     const NbOfErrors_T& getEditDistance() const {
00079       return _editDistance;
00080     }
00081
00086     const NbOfErrors_T& getAllowableEditDistance() const {
00087       return _allowableEditDistance;
00088     }
00089
00093     const DocumentList_T& getDocumentList() const {
00094      return _documentList;
00095     }
00096
00100     const DocumentMap_T& getDocumentMap() const {
00101      return _documentMap;
00102     }
00103
00108     const XapianDocumentPair_T& getDocumentPair (const Xapian::docid&)
      const;
00109
00113     const Xapian::Document& getDocument (const Xapian::docid&) const;
```

```
00114
00118     const Xapian::docid& getBestDocID() const {
00119       return _bestDocID;
00120     }
00121
00125     const Percentage_T& getBestCombinedWeight() const {
00126       return _bestCombinedWeight;
00127     }
00128
00132     const RawDataString_T& getBestDocData() const {
00133       return _bestDocData;
00134     }
00135
00139     const Xapian::Document& getBestXapianDocument() const {
00140       return getDocument (_bestDocID);
00141     }
00142
00143
00144   public:
00145     // ////////////////////// Setters //////////////////////
00149     void setQueryString (const TravelQuery_T& iQueryString) {
00150       _queryString = iQueryString;
00151     }
00152
00156     void setCorrectedQueryString (const TravelQuery_T&
      iCorrectedQueryString) {
00157       _correctedQueryString = iCorrectedQueryString;
00158     }
00159
00163     void setHasFullTextMatched (const bool iHasFullTextMatched) {
00164       _hasFullTextMatched = iHasFullTextMatched;
00165     }
00166
00170     void setEditDistance (const NbOfErrors_T& iEditDistance) {
00171       _editDistance = iEditDistance;
00172     }
00173
00178     void setAllowableEditDistance (const NbOfErrors_T&
      iAllowableEditDistance) {
00179       _allowableEditDistance = iAllowableEditDistance;
00180     }
00181
00202     void addDocument (const Xapian::Document&, const Score_T&);
00203
00207     void setBestDocID (const Xapian::docid& iDocID) {
00208       _bestDocID = iDocID;
00209     }
00210
00215     void setBestCombinedWeight (const Percentage_T& iPercentage) {
00216       _bestCombinedWeight = iPercentage;
00217     }
00218
00222     void setBestDocData (const std::string& iDocData) {
00223       _bestDocData = RawDataString_T (iDocData);
00224     }
00225
00233     void fillResult (const Xapian::MSet& iMatchingSet);
00234
00241     void fillPlace (Place&) const;
00242
00243
00244   public:
00245     // ////////////////// Business support methods /////////////////
00256     std::string fullTextMatch (const Xapian::Database&, const
      TravelQuery_T&);
00257
00266     static Location retrieveLocation (const Xapian::Document&);
00267
00276     static Location retrieveLocation (const
      RawDataString_T&);
00277
00289     static LocationKey getPrimaryKey (const Xapian::Document&);
00290
00300     static Score_T getEnvelopeID (const Xapian::Document&);
00301
00310     static PageRank_T getPageRank (const Xapian::Document&);
00311
00318     void displayXapianPercentages() const;
00319
00328     void setScoreOnDocMap (const Xapian::docid&,
00329                            const ScoreType&, const Score_T&);
00330
00339     void calculateEnvelopeWeights();
00340
00345     void calculateCodeMatches();
00346
00350     void calculatePageRanks();
```

```
00351
00355     void calculateHeuristicWeights();
00356
00361     void calculateCombinedWeights();
00362
00363   private:
00375     std::string fullTextMatch (const Xapian::Database&, const
      TravelQuery_T&,
00376                                Xapian::MSet&);
00377
00378
00379   public:
00380     // ////////////////////// Display support methods /////////////////
00386     void toStream (std::ostream& ioOut) const;
00387
00393     void fromStream (std::istream& ioIn);
00394
00398     std::string toString() const;
00399
00404     std::string describeKey() const;
00405
00410     std::string describeShortKey() const;
00411
00412
00413   private:
00414     // ///////////////// Constructors and Destructors //////////////
00418     Result (const TravelQuery_T&, const Xapian::Database&);
00419
00423     Result();
00424
00428     Result (const Result&);
00429
00433     ~Result();
00434
00438     void init();
00439
00440
00441   private:
00442     // ///////////////// Attributes /////////////////
00446     ResultHolder* _resultHolder;
00447
00451     const Xapian::Database& _database;
00452
00456     TravelQuery_T _queryString;
00457
00462     TravelQuery_T _correctedQueryString;
00463
00470     bool _hasFullTextMatched;
00471
00475     NbOfErrors_T _editDistance;
00476
00481     NbOfErrors_T _allowableEditDistance;
00482
00486     Xapian::docid _bestDocID;
00487
00493     Percentage_T _bestCombinedWeight;
00494
00498     RawDataString_T _bestDocData;
00499
00503     DocumentList_T _documentList;
00504
00508     DocumentMap_T _documentMap;
00509   };
00510
00511 }
00512 #endif // __OPENTREP_BOM_RESULT_HPP
```

24.149 opentrep/bom/ResultCombination.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <xapian.h>
#include <opentrep/bom/StringSet.hpp>
#include <opentrep/bom/ResultHolder.hpp>
#include <opentrep/bom/ResultCombination.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.150 ResultCombination.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Xapian
00008 #include <xapian.h>
00009 // OpenTrep
00010 #include <opentrep/bom/StringSet.hpp>
00011 #include <opentrep/bom/ResultHolder.hpp>
00012 #include <opentrep/bom/ResultCombination.hpp>
00013 #include <opentrep/service/Logger.hpp>
00014
00015 namespace OPENTREP {
00016
00017   // //////////////////////////////////////////////////////////////////////
00018   ResultCombination::ResultCombination()
00019     : _travelQuery (""), _bestMatchingResultHolder (NULL) {
00020     assert (false);
00021   }
00022
00023   // //////////////////////////////////////////////////////////////////////
00024   ResultCombination::ResultCombination (const ResultCombination&)
00025     : _travelQuery (""), _bestMatchingResultHolder (NULL) {
00026     assert (false);
00027   }
00028
00029   // //////////////////////////////////////////////////////////////////////
00030   ResultCombination::ResultCombination (const TravelQuery_T& iQueryString)
00031     : _travelQuery (iQueryString), _bestMatchingResultHolder (NULL) {
00032     init();
00033   }
00034
00035   // //////////////////////////////////////////////////////////////////////
00036   ResultCombination::~ResultCombination() {
00037     _bestMatchingResultHolder = NULL;
00038   }
00039
00040   // //////////////////////////////////////////////////////////////////////
00041   void ResultCombination::init () {
00042     _resultHolderList.clear();
00043   }
00044
00045   // //////////////////////////////////////////////////////////////////////
00046   const ResultHolder& ResultCombination::getBestMatchingResultHolder
      () const {
00047     if (_bestMatchingResultHolder == NULL) {
00048       std::ostringstream oStr;
00049       oStr << "There is no best matching ResultHolder object for the given "
00050            << "query string: '" << _travelQuery << "'";
00051       OPENTREP_LOG_ERROR (oStr.str());
00052       throw NoBestMatchingResultHolderException (oStr.str());
00053     }
00054     assert (_bestMatchingResultHolder != NULL);
00055     return *_bestMatchingResultHolder;
00056   }
00057
00058   // //////////////////////////////////////////////////////////////////////
00059   const Percentage_T& ResultCombination::getBestMatchingWeight
      () const {
00060     const ResultHolder& lResultHolder = getBestMatchingResultHolder(
      );
00061     const Percentage_T& oPercentage = lResultHolder.
      getCombinedWeight();
00062     return oPercentage;
00063   }
00064
00065   // //////////////////////////////////////////////////////////////////////
00066   StringSet ResultCombination::getCorrectedStringSet()
       const {
00067     const ResultHolder& lResultHolder = getBestMatchingResultHolder(
      );
00068     const StringSet& oStringSet = lResultHolder.getCorrectedStringSet();
00069     return oStringSet;
00070   }
00071
```

```
00072 // ////////////////////////////////////////////////////////////////////
00073 std::string ResultCombination::describeShortKey() const {
00074   std::ostringstream oStr;
00075   oStr << _travelQuery;
00076   return oStr.str();
00077 }
00078
00079 // ////////////////////////////////////////////////////////////////////
00080 std::string ResultCombination::describeKey() const {
00081   return describeShortKey();
00082 }
00083
00084 // ////////////////////////////////////////////////////////////////////
00085 std::string ResultCombination::toString() const {
00086   std::ostringstream oStr;
00087   oStr << describeShortKey() << std::endl;
00088
00089   for (ResultHolderList_T::const_iterator itResultHolder =
00090          _resultHolderList.begin();
00091        itResultHolder != _resultHolderList.end(); ++itResultHolder) {
00092     const ResultHolder* lResultHolder_ptr = *itResultHolder;
00093     assert (lResultHolder_ptr != NULL);
00094
00095     oStr << " ==> " << std::endl << lResultHolder_ptr->toString();
00096   }
00097
00098   return oStr.str();
00099 }
00100
00101 // ////////////////////////////////////////////////////////////////////
00102 void ResultCombination::toStream (std::ostream& ioOut) const {
00103   ioOut << toString();
00104 }
00105
00106 // ////////////////////////////////////////////////////////////////////
00107 void ResultCombination::fromStream (std::istream& ioIn) {
00108 }
00109
00110 // ////////////////////////////////////////////////////////////////////
00111 void ResultCombination::displayXapianPercentages() const {
00112   // Browse the ResultHolder objects
00113   for (ResultHolderList_T::const_iterator itResultHolder =
00114          _resultHolderList.begin();
00115        itResultHolder != _resultHolderList.end(); ++itResultHolder) {
00116     ResultHolder* lResultHolder_ptr = *itResultHolder;
00117     assert (lResultHolder_ptr != NULL);
00118
00119     //
00120     lResultHolder_ptr->displayXapianPercentages();
00121   }
00122 }
00123
00124 // ////////////////////////////////////////////////////////////////////
00125 void ResultCombination::calculateEnvelopeWeights() const {
00126   // Browse the ResultHolder objects
00127   for (ResultHolderList_T::const_iterator itResultHolder =
00128          _resultHolderList.begin();
00129        itResultHolder != _resultHolderList.end(); ++itResultHolder) {
00130     ResultHolder* lResultHolder_ptr = *itResultHolder;
00131     assert (lResultHolder_ptr != NULL);
00132
00133     //
00134     lResultHolder_ptr->calculateEnvelopeWeights();
00135   }
00136 }
00137
00138 // ////////////////////////////////////////////////////////////////////
00139 void ResultCombination::calculateCodeMatches() const {
00140   // Browse the ResultHolder objects
00141   for (ResultHolderList_T::const_iterator itResultHolder =
00142          _resultHolderList.begin();
00143        itResultHolder != _resultHolderList.end(); ++itResultHolder) {
00144     ResultHolder* lResultHolder_ptr = *itResultHolder;
00145     assert (lResultHolder_ptr != NULL);
00146
00147     //
00148     lResultHolder_ptr->calculateCodeMatches();
00149   }
00150 }
00151
00152 // ////////////////////////////////////////////////////////////////////
00153 void ResultCombination::calculatePageRanks() const {
00154   // Browse the ResultHolder objects
00155   for (ResultHolderList_T::const_iterator itResultHolder =
00156          _resultHolderList.begin();
00157        itResultHolder != _resultHolderList.end(); ++itResultHolder) {
00158     ResultHolder* lResultHolder_ptr = *itResultHolder;
```

```
00159       assert (lResultHolder_ptr != NULL);
00160
00161       //
00162       lResultHolder_ptr->calculatePageRanks();
00163     }
00164   }
00165
00166   // ////////////////////////////////////////////////////////////////////
00167   void ResultCombination::calculateHeuristicWeights() const {
00168     // Browse the ResultHolder objects
00169     for (ResultHolderList_T::const_iterator itResultHolder =
00170            _resultHolderList.begin();
00171          itResultHolder != _resultHolderList.end(); ++itResultHolder) {
00172       ResultHolder* lResultHolder_ptr = *itResultHolder;
00173       assert (lResultHolder_ptr != NULL);
00174
00175       //
00176       lResultHolder_ptr->calculateHeuristicWeights();
00177     }
00178   }
00179
00180   // ////////////////////////////////////////////////////////////////////
00181   void ResultCombination::calculateCombinedWeights() const {
00182     // Browse the ResultHolder objects
00183     for (ResultHolderList_T::const_iterator itResultHolder =
00184            _resultHolderList.begin();
00185          itResultHolder != _resultHolderList.end(); ++itResultHolder) {
00186       ResultHolder* lResultHolder_ptr = *itResultHolder;
00187       assert (lResultHolder_ptr != NULL);
00188
00189       //
00190       lResultHolder_ptr->calculateCombinedWeights();
00191     }
00192   }
00193
00194   // ////////////////////////////////////////////////////////////////////
00195   bool ResultCombination::chooseBestMatchingResultHolder()
      {
00196
00197     // Identify the ResultHolder object corresponding to the best
00198     // matching percentage
00199     Percentage_T lMaxMatchingPercentage = 0.0;
00200     for (ResultHolderList_T::const_iterator itResultHolder =
00201            _resultHolderList.begin();
00202          itResultHolder != _resultHolderList.end(); ++itResultHolder) {
00203       ResultHolder* lResultHolder_ptr = *itResultHolder;
00204       assert (lResultHolder_ptr != NULL);
00205
00206       // Retrieve the weight from the all the matching processes: full-text,
00207       // envelope ID, IATA/ICAO code, heuristic PageRank.
00208       const Percentage_T& lMatchingPercentage =
00209         lResultHolder_ptr->getCombinedWeight();
00210
00211       // Override the maximum percentage, if needed
00212       if (lMatchingPercentage > lMaxMatchingPercentage) {
00213         lMaxMatchingPercentage = lMatchingPercentage;
00214         _bestMatchingResultHolder = lResultHolder_ptr;
00215       }
00216     }
00217
00218     // DEBUG
00219     if (_bestMatchingResultHolder != NULL) {
00220       OPENTREP_LOG_DEBUG ("    [pct] The best match for the '" <<
      describeKey()
00221                           << "' string has a weight of "
00222                           << lMaxMatchingPercentage
00223                           << "%. It is the following string partition: "
00224                           << _bestMatchingResultHolder->describeKey());
00225
00226     } else {
00227       OPENTREP_LOG_DEBUG ("    [pct] There is no match for the '"
00228                           << describeKey() << "' string");
00229     }
00230
00231     //
00232     const bool doesBestMatchingResultHolderExist =
00233       (_bestMatchingResultHolder != NULL);
00234     return doesBestMatchingResultHolderExist;
00235   }
00236
00237 }
```

24.151 opentrep/bom/ResultCombination.hpp File Reference

```
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/bom/BomAbstract.hpp>
#include <opentrep/bom/ResultHolderList.hpp>
```

Classes

- class OPENTREP::ResultCombination

  Class wrapping functions on a list of ResultHolder objects.

Namespaces

- Xapian
- OPENTREP

24.152 ResultCombination.hpp

```
00001 #ifndef __OPENTREP_BOM_RESULTCOMBINATION_HPP
00002 #define __OPENTREP_BOM_RESULTCOMBINATION_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/OPENTREP_Types.hpp>
00009 #include <opentrep/bom/BomAbstract.hpp>
00010 #include <opentrep/bom/ResultHolderList.hpp>
00011
00012 // Forward declarations
00013 namespace Xapian {
00014   class Database;
00015 }
00016
00017 namespace OPENTREP {
00018
00019   // Forward declarations
00020   struct StringSet;
00021
00025   class ResultCombination : public BomAbstract {
00026     friend class FacResultCombination;
00027   public:
00028     // ////////////// Getters /////////////
00032     const TravelQuery_T& getQueryString() const {
00033       return _travelQuery;
00034     }
00035
00039     const ResultHolderList_T& getResultHolderList() const {
00040       return _resultHolderList;
00041     }
00042
00046     bool hasFullTextMatched() const {
00047       return (_bestMatchingResultHolder != NULL);
00048     }
00049
00055     const ResultHolder& getBestMatchingResultHolder() const;
00056
00062     const Percentage_T& getBestMatchingWeight() const;
00063
00070     StringSet getCorrectedStringSet() const;
00071
00072
00073   public:
00074     // ////////// Business methods //////////
00078     void displayXapianPercentages() const;
00079
00083     void calculateEnvelopeWeights() const;
00084
00089     void calculateCodeMatches() const;
00090
00094     void calculatePageRanks() const;
00095
00099     void calculateHeuristicWeights() const;
```

```
00100
00104     void calculateCombinedWeights() const;
00105
00111     bool chooseBestMatchingResultHolder();
00112
00113
00114   public:
00115     // /////////// Display support methods /////////
00121     void toStream (std::ostream&) const;
00122
00128     void fromStream (std::istream&);
00129
00133     std::string toString() const;
00134
00139     std::string describeKey() const;
00140
00145     std::string describeShortKey() const;
00146
00147
00148   private:
00149     // /////////////// Constructors and Destructors /////////////
00153     ResultCombination (const TravelQuery_T&);
00157     ~ResultCombination ();
00161     void init ();
00162
00163   private:
00167     ResultCombination();
00171     ResultCombination (const ResultCombination&);
00172
00173
00174   private:
00175     // ///////////////// Attributes /////////////////
00179     const TravelQuery_T _travelQuery;
00180
00184     ResultHolderList_T _resultHolderList;
00185
00189     const ResultHolder* _bestMatchingResultHolder;
00190   };
00191
00192 }
00193 #endif // __OPENTREP_BOM_RESULTCOMBINATION_HPP
```

24.153 opentrep/bom/ResultHolder.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <xapian.h>
#include <opentrep/basic/BasConst_General.hpp>
#include <opentrep/bom/StringSet.hpp>
#include <opentrep/bom/Result.hpp>
#include <opentrep/bom/ResultHolder.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.154 ResultHolder.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Xapian
00008 #include <xapian.h>
00009 // OpenTrep
00010 #include <opentrep/basic/BasConst_General.hpp>
00011 #include <opentrep/bom/StringSet.hpp>
00012 #include <opentrep/bom/Result.hpp>
00013 #include <opentrep/bom/ResultHolder.hpp>
00014 #include <opentrep/service/Logger.hpp>
```

```
00015
00016 namespace OPENTREP {
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   ResultHolder::ResultHolder (const TravelQuery_T& iQueryString,
00020                               const Xapian::Database& iDatabase)
00021     : _resultCombination (NULL),
00022       _queryString (iQueryString), _database (iDatabase) {
00023     init();
00024   }
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   ResultHolder::~ResultHolder () {
00028   }
00029
00030   // ////////////////////////////////////////////////////////////////////
00031   void ResultHolder::init () {
00032     _resultCombination = NULL;
00033     _resultList.clear();
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   std::string ResultHolder::describeShortKey() const {
00038     std::ostringstream oStr;
00039     oStr << _queryString;
00040     return oStr.str();
00041   }
00042
00043   // ////////////////////////////////////////////////////////////////////
00044   std::string ResultHolder::describeKey() const {
00045     return describeShortKey();
00046   }
00047
00048   // ////////////////////////////////////////////////////////////////////
00049   std::string ResultHolder::toString() const {
00050     std::ostringstream oStr;
00051     oStr << describeShortKey() << std::endl;
00052
00053     unsigned short idx = 0;
00054     for (ResultList_T::const_iterator itResult = _resultList.begin();
00055          itResult != _resultList.end(); ++itResult, ++idx) {
00056       const Result* lResult_ptr = *itResult;
00057       assert (lResult_ptr != NULL);
00058
00059       if (idx != 0) {
00060         oStr << std::endl;
00061       }
00062       oStr << " ==> " << std::endl << lResult_ptr->toString();
00063     }
00064
00065     return oStr.str();
00066   }
00067
00068   // ////////////////////////////////////////////////////////////////////
00069   void ResultHolder::toStream (std::ostream& ioOut) const {
00070     ioOut << toString();
00071   }
00072
00073   // ////////////////////////////////////////////////////////////////////
00074   void ResultHolder::fromStream (std::istream& ioIn) {
00075   }
00076
00077   // ////////////////////////////////////////////////////////////////////
00078   StringSet ResultHolder::getCorrectedStringSet() const {
00079     StringSet oStringSet;
00080     for (ResultList_T::const_iterator itResult = _resultList.begin();
00081          itResult != _resultList.end(); ++itResult) {
00082       const Result* lResult_ptr = *itResult;
00083       assert (lResult_ptr != NULL);
00084
00085       // Retrieve the corrected string
00086       const TravelQuery_T& lCorrectedQueryString =
00087         lResult_ptr->getCorrectedTravelQuery();
00088       oStringSet.push_back (lCorrectedQueryString);
00089     }
00090
00091     //
00092     return oStringSet;
00093   }
00094
00095   // ////////////////////////////////////////////////////////////////////
00096   void ResultHolder::displayXapianPercentages() const {
00097     // Browse the Result objects
00098     for (ResultList_T::const_iterator itResult = _resultList.begin();
00099          itResult != _resultList.end(); ++itResult) {
00100       Result* lResult_ptr = *itResult;
00101       assert (lResult_ptr != NULL);
```

```
00102
00103       //
00104       lResult_ptr->displayXapianPercentages();
00105     }
00106   }
00107
00108   // ////////////////////////////////////////////////////////////////////
00109   void ResultHolder::calculateEnvelopeWeights() const {
00110     // Browse the Result objects
00111     for (ResultList_T::const_iterator itResult = _resultList.begin();
00112          itResult != _resultList.end(); ++itResult) {
00113       Result* lResult_ptr = *itResult;
00114       assert (lResult_ptr != NULL);
00115
00116       //
00117       lResult_ptr->calculateEnvelopeWeights();
00118     }
00119   }
00120
00121   // ////////////////////////////////////////////////////////////////////
00122   void ResultHolder::calculateCodeMatches() const {
00123     // Browse the Result objects
00124     for (ResultList_T::const_iterator itResult = _resultList.begin();
00125          itResult != _resultList.end(); ++itResult) {
00126       Result* lResult_ptr = *itResult;
00127       assert (lResult_ptr != NULL);
00128
00129       //
00130       lResult_ptr->calculateCodeMatches();
00131     }
00132   }
00133
00134   // ////////////////////////////////////////////////////////////////////
00135   void ResultHolder::calculatePageRanks() const {
00136     // Browse the Result objects
00137     for (ResultList_T::const_iterator itResult = _resultList.begin();
00138          itResult != _resultList.end(); ++itResult) {
00139       Result* lResult_ptr = *itResult;
00140       assert (lResult_ptr != NULL);
00141
00142       //
00143       lResult_ptr->calculatePageRanks();
00144     }
00145   }
00146
00147   // ////////////////////////////////////////////////////////////////////
00148   void ResultHolder::calculateHeuristicWeights() const {
00149     // Browse the Result objects
00150     for (ResultList_T::const_iterator itResult = _resultList.begin();
00151          itResult != _resultList.end(); ++itResult) {
00152       Result* lResult_ptr = *itResult;
00153       assert (lResult_ptr != NULL);
00154
00155       //
00156       lResult_ptr->calculateHeuristicWeights();
00157     }
00158   }
00159
00160   // ////////////////////////////////////////////////////////////////////
00161   void ResultHolder::calculateCombinedWeights() {
00162     Percentage_T oCombinedPercentage = 100.0;
00163
00164     // When there is no result, the weight is obviously 0%
00165     if (_resultList.empty() == true) {
00166       oCombinedPercentage = 0.0;
00167     }
00168
00169     // Browse the Result objects
00170     for (ResultList_T::const_iterator itResult = _resultList.begin();
00171          itResult != _resultList.end(); ++itResult) {
00172       Result* lResult_ptr = *itResult;
00173       assert (lResult_ptr != NULL);
00174
00175       // Calculate the combined weights, for all the matching documents,
00176       // and set the best combined weight to the greatest one
00177       lResult_ptr->calculateCombinedWeights();
00178
00179       // Retrieve the just calculated combined weight
00180       const Percentage_T& lPercentage = lResult_ptr->
    getBestCombinedWeight();
00181
00182       // Take into account the current weight into the total
00183       oCombinedPercentage *= lPercentage / 100.0;
00184     }
00185
00192     unsigned short nbOfResults = _resultList.size();
00193     if (nbOfResults > 1) {
```

```
00194       oCombinedPercentage /= std::pow (K_DEFAULT_ATTENUATION_FCTR, nbOfResults);
00195     }
00196
00197     // DEBUG
00198     OPENTREP_LOG_DEBUG ("      [pct] The " << describeKey()
00199                         << " string partition overall matches at "
00200                         << oCombinedPercentage << "%");
00201
00202     // Store the combined weight
00203     setCombinedWeight (oCombinedPercentage);
00204   }
00205
00206 }
```

24.155 opentrep/bom/ResultHolder.hpp File Reference

```
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/bom/BomAbstract.hpp>
#include <opentrep/bom/ResultList.hpp>
```

Classes

- class OPENTREP::ResultHolder

  Class wrapping functions on a list of Result objects.

Namespaces

- Xapian
- OPENTREP

24.156 ResultHolder.hpp

```
00001 #ifndef __OPENTREP_BOM_RESULTHOLDER_HPP
00002 #define __OPENTREP_BOM_RESULTHOLDER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/OPENTREP_Types.hpp>
00009 #include <opentrep/bom/BomAbstract.hpp>
00010 #include <opentrep/bom/ResultList.hpp>
00011
00012 // Forward declarations
00013 namespace Xapian {
00014   class Database;
00015 }
00016
00017 namespace OPENTREP {
00018
00019   // Forward declarations
00020   class ResultCombination;
00021   struct StringSet;
00022
00026   class ResultHolder : public BomAbstract {
00027     friend class FacResultCombination;
00028     friend class FacResultHolder;
00029   public:
00030     // ////////////////////// Getters ////////////////////
00034     const TravelQuery_T& getQueryString() const {
00035       return _queryString;
00036     }
00037
00041     const ResultList_T& getResultList() const {
00042       return _resultList;
00043     }
00044
00048     const Percentage_T& getCombinedWeight() const {
00049       return _combinedWeight;
00050     }
00051
```

```
00055     StringSet getCorrectedStringSet() const;
00056
00057
00058   public:
00059     // ////////////////////// Setters ////////////////////
00063     void setCombinedWeight (const Percentage_T& iPercentage) {
00064       _combinedWeight = iPercentage;
00065     }
00066
00067
00068   public:
00069     // /////////// Business methods ///////////
00073     void displayXapianPercentages() const;
00074
00078     void calculateEnvelopeWeights() const;
00079
00084     void calculateCodeMatches() const;
00085
00089     void calculatePageRanks() const;
00090
00094     void calculateHeuristicWeights() const;
00095
00099     void calculateCombinedWeights();
00100
00101
00102   public:
00103     // /////////// Display support methods /////////
00109     void toStream (std::ostream&) const;
00110
00116     void fromStream (std::istream&);
00117
00121     std::string toString() const;
00122
00127     std::string describeKey() const;
00128
00133     std::string describeShortKey() const;
00134
00135
00136   private:
00137     // ////////////// Constructors and Destructors /////////////
00141     ResultHolder (const TravelQuery_T&, const Xapian::Database&);
00145     ResultHolder();
00149     ResultHolder (const ResultHolder&);
00153     ~ResultHolder();
00157     void init();
00158
00159
00160   private:
00161     // /////////////// Attributes ////////////////
00165     ResultCombination* _resultCombination;
00166
00170     const TravelQuery_T _queryString;
00171
00175     const Xapian::Database& _database;
00176
00180     ResultList_T _resultList;
00181
00185     Percentage_T _combinedWeight;
00186   };
00187
00188 }
00189 #endif // __OPENTREP_BOM_RESULTHOLDER_HPP
```

24.157 opentrep/bom/ResultHolderList.hpp File Reference

```
#include <list>
```

Namespaces

- OPENTREP

Typedefs

- typedef std::list< ResultHolder * > OPENTREP::ResultHolderList_T

24.158 ResultHolderList.hpp

```
00001 #ifndef __OPENTREP_BOM_RESULTHOLDERLIST_HPP
00002 #define __OPENTREP_BOM_RESULTHOLDERLIST_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <list>
00009
00010 namespace OPENTREP {
00011
00012   // Forward declarations
00013   class ResultHolder;
00014
00018   typedef std::list<ResultHolder*> ResultHolderList_T;
00019
00020 }
00021 #endif // __OPENTREP_BOM_RESULTHOLDERLIST_HPP
```

24.159 opentrep/bom/ResultList.hpp File Reference

```
#include <list>
```

Namespaces

- OPENTREP

Typedefs

- typedef std::list< Result * > OPENTREP::ResultList_T

24.160 ResultList.hpp

```
00001 #ifndef __OPENTREP_BOM_RESULTLIST_HPP
00002 #define __OPENTREP_BOM_RESULTLIST_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <list>
00009
00010 namespace OPENTREP {
00011
00012   // Forward declarations
00013   class Result;
00014
00018   typedef std::list<Result*> ResultList_T;
00019
00020 }
00021 #endif // __OPENTREP_BOM_RESULTLIST_HPP
```

24.161 opentrep/bom/ScoreBoard.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <opentrep/basic/BasConst_General.hpp>
#include <opentrep/basic/float_utils.hpp>
#include <opentrep/bom/WordHolder.hpp>
#include <opentrep/bom/ScoreBoard.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.162 ScoreBoard.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // OpenTrep
00008 #include <opentrep/basic/BasConst_General.hpp>
00009 #include <opentrep/basic/float_utils.hpp>
00010 #include <opentrep/bom/WordHolder.hpp>
00011 #include <opentrep/bom/ScoreBoard.hpp>
00012 #include <opentrep/service/Logger.hpp>
00013
00014 namespace OPENTREP {
00015
00016   // //////////////////////////////////////////////////////////////////////
00017   ScoreBoard::ScoreBoard (const TravelQuery_T& iQueryString)
00018     : _queryString (iQueryString) {
00019   }
00020
00021   // //////////////////////////////////////////////////////////////////////
00022   ScoreBoard::ScoreBoard (const ScoreBoard& iScoreBoard)
00023     : _queryString (iScoreBoard._queryString),
00024       _scoreMap (iScoreBoard._scoreMap) {
00025   }
00026
00027   // //////////////////////////////////////////////////////////////////////
00028   ScoreBoard::ScoreBoard (const TravelQuery_T& iQueryString,
00029                           const ScoreType& iType, const Score_T& iScore)
00030     : _queryString (iQueryString) {
00031     setScore (iType, iScore);
00032   }
00033
00034   // //////////////////////////////////////////////////////////////////////
00035   ScoreBoard::~ScoreBoard() {
00036     _scoreMap.clear();
00037   }
00038
00039   // //////////////////////////////////////////////////////////////////////
00040   Score_T ScoreBoard::getScore (const ScoreType& iScoreType) const {
00041     Score_T oScore = 0.0;
00042
00043     // Check whether a score value already exists for that type
00044     const ScoreType::EN_ScoreType& lScoreTypeEnum = iScoreType.
    getType();
00045     ScoreMap_T::const_iterator itScore = _scoreMap.find (lScoreTypeEnum);
00046     if (itScore != _scoreMap.end()) {
00047       oScore = itScore->second;
00048     }
00049
00050     return oScore;
00051   }
00052
00053   // //////////////////////////////////////////////////////////////////////
00054   void ScoreBoard::setScore (const ScoreType& iScoreType,
00055                              const Score_T& iScore) {
00056     Score_T oScore = iScore;
00057
00073     if (iScoreType == ScoreType::CODE_FULL_MATCH) {
00074       const FloatingPoint<Percentage_T> lComparablePct (oScore);
00075       const FloatingPoint<Percentage_T> lCodeFullMatchingPct (1.0);
00076       if (lComparablePct.AlmostEquals (lCodeFullMatchingPct) == true) {
00077         oScore = K_DEFAULT_FULL_CODE_MATCH_PCT;
00078
00079       } else {
00080         // Normally, K_DEFAULT_MODIFIED_MATCHING_PCT == 99.999.
00081         // See basic/BasConst.cpp
00082         oScore = K_DEFAULT_MODIFIED_MATCHING_PCT;
00083       }
00084     }
00085
00091     /*
00092     if (iScoreType == ScoreType::XAPIAN_PCT) {
00093       const FloatingPoint<Percentage_T> lComparablePct (oScore);
00094       const FloatingPoint<Percentage_T> lFullMatchingPct (100.0);
00095       if (lComparablePct.AlmostEquals (lFullMatchingPct) == true) {
00096         // Normally, K_DEFAULT_MODIFIED_MATCHING_PCT == 99.999.
```

```
00097         // See basic/BasConst.cpp
00098         oScore = K_DEFAULT_MODIFIED_MATCHING_PCT;
00099       }
00100     }
00101     */
00102
00113     if (iScoreType == ScoreType::ENV_ID) {
00114       const FloatingPoint<Percentage_T> lComparableValue (oScore);
00115       const FloatingPoint<Percentage_T> lNoEnvelopeIDValue (0.0);
00116       if (lComparableValue.AlmostEquals (lNoEnvelopeIDValue) == true) {
00117         oScore = 100.00;
00118       } else {
00119         // Normally, K_DEFAULT_ENVELOPE_PCT == 0.001. See basic/BasConst.cpp
00120         oScore = K_DEFAULT_ENVELOPE_PCT;
00121       }
00122     }
00123
00124     // Check whether a score value already exists for that type
00125     const ScoreType::EN_ScoreType& lScoreTypeEnum = iScoreType.
      getType();
00126     ScoreMap_T::iterator itScore = _scoreMap.find (lScoreTypeEnum);
00127
00128     if (itScore != _scoreMap.end()) {
00129       // Just replace the score value
00130       Score_T& lScore = itScore->second;
00131       lScore = oScore;
00132
00133     } else {
00134       // Insert the score value for that new type
00135       const bool insertSucceeded =
00136         _scoreMap.insert (ScoreMap_T::value_type (lScoreTypeEnum,
00137                                                   oScore)).second;
00138
00139       // Sanity check
00140       if (insertSucceeded == false) {
00141         OPENTREP_LOG_ERROR ("The " << iScore << " score value can not be "
00142                             << "inserted in the dedicated list for the "
00143                             << iScoreType.describe() << " score type");
00144       }
00145       assert (insertSucceeded == true);
00146     }
00147   }
00148
00149   // ////////////////////////////////////////////////////////////////////
00150   std::string ScoreBoard::describeKey() const {
00151     std::ostringstream oStr;
00152     oStr << _queryString;
00153     return oStr.str();
00154   }
00155
00156   // ////////////////////////////////////////////////////////////////////
00157   std::string ScoreBoard::describe() const {
00158     std::ostringstream oStr;
00159     oStr << describeKey() << " - ";
00160
00161     unsigned short idx = 0;
00162     for (ScoreMap_T::const_iterator itScore = _scoreMap.begin();
00163          itScore != _scoreMap.end(); ++itScore, ++idx) {
00164       if (idx != 0) {
00165         oStr << ", ";
00166       }
00167       const ScoreType::EN_ScoreType& lScoreType = itScore->first;
00168       const Score_T& lScore = itScore->second;
00169       oStr << ScoreType::getTypeLabelAsString (lScoreType) << ": "
00170            << lScore << "%";
00171     }
00172
00173     return oStr.str();
00174   }
00175
00176   // ////////////////////////////////////////////////////////////////////
00177   void ScoreBoard::toStream (std::ostream& ioOut) const {
00178     ioOut << describe();
00179   }
00180
00181   // ////////////////////////////////////////////////////////////////////
00182   void ScoreBoard::fromStream (std::istream& ioIn) {
00183   }
00184
00185   // ////////////////////////////////////////////////////////////////////
00186   Percentage_T ScoreBoard::calculateCombinedWeight() {
00187     Percentage_T oPercentage = 100.0;
00188
00189     // Browse the registered scores
00190     for (ScoreMap_T::iterator itScore = _scoreMap.begin();
00191          itScore != _scoreMap.end(); ++itScore) {
00192       const ScoreType::EN_ScoreType& lScoreType = itScore->first;
```

```
00193       Score_T& lScore = itScore->second;
00194
00200       const bool isIndividual = ScoreType::isIndividualScore (lScoreType);
00201       if (isIndividual == true) {
00202         oPercentage *= lScore / 100.0;
00203       }
00204
00212     }
00213
00214     // Register the combined score
00215     setCombinedWeight (oPercentage);
00216
00217     //
00218     return oPercentage;
00219   }
00220
00221 }
```

24.163 opentrep/bom/ScoreBoard.hpp File Reference

```
#include <map>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/basic/ScoreType.hpp>
#include <opentrep/basic/StructAbstract.hpp>
#include <xapian.h>
```

Classes

- struct OPENTREP::ScoreBoard

  Structure holding a board for all the types of score/matching having been performed.

Namespaces

- OPENTREP

24.164 ScoreBoard.hpp

```
00001 #ifndef __OPENTREP_BOM_SCOREBOARD_HPP
00002 #define __OPENTREP_BOM_SCOREBOARD_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <map>
00009 // OpenTrep
00010 #include <opentrep/OPENTREP_Types.hpp>
00011 #include <opentrep/basic/ScoreType.hpp>
00012 #include <opentrep/basic/StructAbstract.hpp>
00013 // Xapian
00014 #include <xapian.h>
00015
00016 namespace OPENTREP {
00017
00022   struct ScoreBoard : public StructAbstract {
00023   public:
00024     // ///////////////// Type definitions /////////////////
00028     typedef std::map<ScoreType::EN_ScoreType, Score_T> ScoreMap_T;
00029
00030
00031   public:
00032     // ////////////////// Getters ////////////////
00036     const TravelQuery_T& getQueryString() const {
00037       return _queryString;
00038     }
00039
00043     const ScoreMap_T& getScoreMap() const {
00044      return _scoreMap;
00045     }
00046
```

```
00051     Score_T getScore (const ScoreType&) const;
00052
00056     Score_T getCombinedWeight() const {
00057       return getScore (ScoreType::COMBINATION);
00058     }
00059
00060
00061   public:
00062     // ////////////////////// Setters ///////////////////
00066     void setQueryString (const TravelQuery_T& iQueryString) {
00067       _queryString = iQueryString;
00068     }
00069
00098     void setScore (const ScoreType&, const Score_T&);
00099
00103     void setCombinedWeight (const Score_T& iScore) {
00104       setScore (ScoreType::COMBINATION, iScore);
00105     }
00106
00112     Percentage_T calculateCombinedWeight();
00113
00114
00115   public:
00116     // ////////////////// Business methods /////////////////
00117
00118
00119   public:
00120     // /////////// Display support methods /////////
00126     void toStream (std::ostream&) const;
00127
00133     void fromStream (std::istream&);
00134
00140     std::string describeKey() const;
00141
00147     std::string describe() const;
00148
00149
00150   public:
00151     // ///////////////// Constructors and Destructors //////////////
00155     ScoreBoard (const TravelQuery_T&, const ScoreType&, const
      Score_T&);
00156
00160     ScoreBoard (const TravelQuery_T&);
00161
00165     ScoreBoard (const ScoreBoard&);
00166
00170     ~ScoreBoard();
00171
00172
00173   private:
00174     // ////////////////// Attributes //////////////////
00178     TravelQuery_T _queryString;
00179
00183     ScoreMap_T _scoreMap;
00184   };
00185
00186 }
00187 #endif // __OPENTREP_BOM_SCOREBOARD_HPP
```

24.165 opentrep/bom/StringPartition.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <set>
#include <opentrep/basic/Utilities.hpp>
#include <opentrep/bom/Filter.hpp>
#include <opentrep/bom/StringPartition.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.166 StringPartition.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <set>
00008 // OpenTrep
00009 #include <opentrep/basic/Utilities.hpp>
00010 #include <opentrep/bom/Filter.hpp>
00011 #include <opentrep/bom/StringPartition.hpp>
00012 #include <opentrep/service/Logger.hpp>
00013
00014 namespace OPENTREP {
00015
00016   // //////////////////////////////////////////////////////////////////////
00017   StringPartition::StringPartition (const std::string& iString)
00018     : _initialString (iString) {
00019     init (iString);
00020   }
00021
00022   // //////////////////////////////////////////////////////////////////////
00023   StringPartition::~StringPartition() {
00024   }
00025
00026   // //////////////////////////////////////////////////////////////////////
00027   void StringPartition::push_back (const StringSet& iStringSet) {
00028     if (iStringSet.empty() == false) {
00029       _partition.push_back (iStringSet);
00030     }
00031   }
00032
00033   // //////////////////////////////////////////////////////////////////////
00034   size_t StringPartition::size() const {
00035     return _partition.size();
00036   }
00037
00038   // //////////////////////////////////////////////////////////////////////
00039   bool StringPartition::empty() const {
00040     return _partition.empty();
00041   }
00042
00043   // //////////////////////////////////////////////////////////////////////
00044   void StringPartition::clear() {
00045     _partition.clear();
00046   }
00047
00048   // //////////////////////////////////////////////////////////////////////
00049   std::string StringPartition::describeKey() const {
00050     std::ostringstream oStr;
00051     oStr << "";
00052     return oStr.str();
00053   }
00054
00055   // //////////////////////////////////////////////////////////////////////
00056   std::string StringPartition::describe() const {
00057     std::ostringstream oStr;
00058     oStr << describeKey();
00059
00060     //
00061     oStr << "{";
00062
00063     short idx_sublist = 0;
00064     for (StringPartition_T::const_iterator itSet = _partition.begin();
00065          itSet != _partition.end(); ++itSet, ++idx_sublist) {
00066       //
00067       if (idx_sublist != 0) {
00068         oStr << ", ";
00069       }
00070
00071       //
00072       const StringSet& lStringSet = *itSet;
00073
00074       //
00075       oStr << lStringSet;
00076     }
00077
00078     //
00079     oStr << " }";
00080
00081     return oStr.str();
00082   }
00083
00084   // //////////////////////////////////////////////////////////////////////
00085   void StringPartition::toStream (std::ostream& ioOut) const {
```

```
00086     ioOut << describe();
00087   }
00088
00089   // ////////////////////////////////////////////////////////////////////
00090   void StringPartition::fromStream (std::istream& ioIn) {
00091   }
00092
00093   // ////////////////////////////////////////////////////////////////////
00094   void StringPartition::init (const std::string& iPhrase) {
00095     // 0. Initialisation
00096     // 0.1. Initialisation of the tokenizer
00097     WordList_T lWordList;
00098     tokeniseStringIntoWordList (iPhrase, lWordList);
00099     const short nbOfWords = lWordList.size();
00100
00101     // 0.2. Re-create the initial phrase, without any (potential) seperator
00102     const std::string lPhrase = createStringFromWordList (lWordList);
00103
00104     // 0.3. Create the list with a single sub-list, itself containing only
00105     //      the given input string.
00106     StringSet oStringSet (lPhrase);
00107
00108     // 0.4. If the string contains no more than one word, the job is finished.
00109     if (nbOfWords <= 1) {
00110       _partition.push_back (oStringSet);
00111       return;
00112     }
00113
00114     // 1. Iteration on all the words of the given string, from 1 to nbOfWords-1
00115     for (short idx_word = 1; idx_word != nbOfWords; ++idx_word) {
00116       // 1.1. Create a sub-string copy of the first idx_word
00117       const std::string& lLeftHandString = createStringFromWordList (lWordList,
00118                                                                       idx_word);
00119
00120       // DEBUG
00121       // std::cout << "[" << lPhrase << ", " << idx_word
00122       //           << "] Left-hand string: '" << lLeftHandString << "'"
00123       //           << std::endl;
00124
00125       // 1.2. Create another sub-string with the remaining of the string
00126       const std::string& lRightHandString = createStringFromWordList (lWordList,
00127                                                                        idx_word,
00128                                                                        false);
00129
00130       // DEBUG
00131       // std::cout << "[" << lPhrase << ", " << idx_word
00132       //           << "] Right-hand string: '" << lRightHandString << "'"
00133       //           << std::endl;
00134
00135       // 1.3. Recurse
00136       // 1.3.1. Call the Partition algorithm on the right-hand side string
00137       StringPartition lStringPartition (lRightHandString);
00138
00139       // 1.3.2. Extract the sub-lists of strings, and add them to the current
00140       //        sub-lists.
00141       const StringPartition_T& lStringRHSPartition= lStringPartition._partition;
00142       for (StringPartition_T::const_iterator itSet= lStringRHSPartition.begin();
00143            itSet != lStringRHSPartition.end(); ++itSet) {
00144         const StringSet& lRHSStringSet = *itSet;
00145
00146         // Create the sub-list which will accommodate:
00147         //  - The left-hand-side sub-string.
00148         //  - The the sub-list having been generated by the recursion on the
00149         //    the right-hand-side sub-string.
00150         StringSet lNewStringSet;
00151
00152         // Add the string to the list
00153         lNewStringSet.push_back (lLeftHandString);
00154
00155         //
00156         lNewStringSet.push_back (lRHSStringSet);
00157
00158         //
00159         _partition.push_back (lNewStringSet);
00160       }
00161     }
00162
00163
00164     // 2.
00165     // 2.1. Add the sub-list with the full string (the one given as input) to
00166     // the back of the list.
00167     _partition.push_back (oStringSet);
00168   }
00169
00170   // ////////////////////////////////////////////////////////////////////
00171   StringSet StringPartition::calculateUniqueCombinations
      () const {
```

```
00172     StringSet oStringSet;
00173
00174     // Set of unique strings
00175     WordSet_T lStringList;
00176
00177     // Browse all the word combinations. Then, for every word combination,
00178     // add it if not already in the list (STD set) of strings.
00179     for (StringPartition_T::const_iterator itSet = _partition.begin();
00180          itSet != _partition.end(); ++itSet) {
00181       const StringSet& itStringSet = *itSet;
00182
00183       const StringSet::StringSet_T& lStringSet = itStringSet.
      _set;
00184       for (StringSet::StringSet_T::const_iterator itString = lStringSet.begin();
00185            itString != lStringSet.end(); ++itString) {
00186         const std::string& lWordCombination = *itString;
00187
00188         // Check whether that word combination has already been stored once.
00189         WordSet_T::const_iterator itExistingString = lStringList.find (lWordCombination);
00190         if (itExistingString == lStringList.end()) {
00191           // If not, add it to the dedicated list (STD set).
00192           lStringList.insert (lWordCombination);
00193         }
00194       }
00195     }
00196
00197     // Convert the STD set into a StringSet structure
00198     for (WordSet_T::const_iterator itString = lStringList.begin();
00199          itString != lStringList.end(); ++itString) {
00200       const std::string& lWordCombination = *itString;
00201       oStringSet.push_back (lWordCombination);
00202     }
00203
00204     //
00205     return oStringSet;
00206   }
00207
00208 }
```

24.167 opentrep/bom/StringPartition.hpp File Reference

```
#include <string>
#include <list>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/basic/StructAbstract.hpp>
#include <opentrep/bom/StringSet.hpp>
```

Classes

- struct OPENTREP::StringPartition

Namespaces

- OPENTREP

Typedefs

- typedef std::list
  < StringPartition > OPENTREP::StringPartitionList_T

24.168 StringPartition.hpp

```
00001 #ifndef __OPENTREP_BOM_STRINGPARTITION_HPP
00002 #define __OPENTREP_BOM_STRINGPARTITION_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
```

```
00007 // STL
00008 #include <string>
00009 #include <list>
00010 // OpenTrep
00011 #include <opentrep/OPENTREP_Types.hpp>
00012 #include <opentrep/basic/StructAbstract.hpp>
00013 #include <opentrep/bom/StringSet.hpp>
00014
00015 namespace OPENTREP {
00016
00035   struct StringPartition : public StructAbstract {
00036     // ////////////////// Type definitions ///////////////////
00040     typedef std::list<StringSet> StringPartition_T;
00041
00042   public:
00049     void push_back (const StringSet& iStringSet);
00050
00054     size_t size() const;
00055
00059     bool empty() const;
00060
00064     void clear();
00065
00070     const std::string& getInitialString() const {
00071       return _initialString;
00072     }
00073
00090     StringSet calculateUniqueCombinations() const;
00091
00092
00093   private:
00102     void init (const std::string& iStringToBePartitioned);
00103
00104
00105   public:
00106     // /////////// Display support methods /////////
00112     void toStream (std::ostream& ioOut) const;
00113
00119     void fromStream (std::istream& ioIn);
00120
00125     std::string describeKey() const;
00126
00130     std::string describe() const;
00131
00132
00133   public:
00134     // ////////////////// Constructors and Destructors //////////////
00140     StringPartition (const std::string& iStringToBePartitioned);
00141
00145     ~StringPartition();
00146
00147
00148   public:
00149     // ////////////////// Attributes ///////////////
00153     std::string _initialString;
00154
00158     StringPartition_T _partition;
00159   };
00160
00161
00162   // ////////////////// Type definitions ///////////////////
00166   typedef std::list<StringPartition> StringPartitionList_T;
00167
00168 }
00169 #endif // __OPENTREP_BOM_STRINGPARTITION_HPP
```

24.169 opentrep/bom/StringSet.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <opentrep/bom/StringSet.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.170 StringSet.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // OpenTrep
00008 #include <opentrep/bom/StringSet.hpp>
00009 #include <opentrep/service/Logger.hpp>
00010
00011 namespace OPENTREP {
00012
00013   // //////////////////////////////////////////////////////////////////////
00014   StringSet::StringSet() {
00015   }
00016
00017   // //////////////////////////////////////////////////////////////////////
00018   StringSet::StringSet (const StringSet& iStringSet) {
00019     _set = iStringSet._set;
00020   }
00021
00022   // //////////////////////////////////////////////////////////////////////
00023   StringSet::StringSet (const std::string& iString) {
00024     if (iString.empty() == false) {
00025       _set.push_back (iString);
00026     }
00027   }
00028
00029   // //////////////////////////////////////////////////////////////////////
00030   StringSet::~StringSet () {
00031   }
00032
00033   // //////////////////////////////////////////////////////////////////////
00034   size_t StringSet::size() const {
00035     return _set.size();
00036   }
00037
00038   // //////////////////////////////////////////////////////////////////////
00039   bool StringSet::empty() const {
00040     return _set.empty();
00041   }
00042
00043   // //////////////////////////////////////////////////////////////////////
00044   void StringSet::clear() {
00045     _set.clear();
00046   }
00047
00048   // //////////////////////////////////////////////////////////////////////
00049   void StringSet::push_back (const std::string& iString) {
00050     _set.push_back (iString);
00051   }
00052
00053   // //////////////////////////////////////////////////////////////////////
00054   void StringSet::push_back (const StringSet& iStringSet) {
00055     const StringSet_T& lStringSet = iStringSet._set;
00056     for (StringSet_T::const_iterator itString = lStringSet.begin();
00057          itString != lStringSet.end(); ++itString) {
00058       const std::string& lString = *itString;
00059       push_back (lString);
00060     }
00061   }
00062
00063   // //////////////////////////////////////////////////////////////////////
00064   std::string StringSet::getFirstString() const {
00065     std::string oString ("");
00066
00067     // When the underlying set is empty, then the first string is empty
00068     if (_set.empty() == true) {
00069       return oString;
00070     }
00071
00072     // Extract the first string of the underlying set
00073     StringSet_T::const_iterator itString = _set.begin();
00074     oString = *itString;
00075
00076     //
00077     return oString;
00078   }
00079
00080   // //////////////////////////////////////////////////////////////////////
00081   std::string StringSet::describeKey() const {
00082     std::ostringstream oStr;
00083     oStr << "";
00084     return oStr.str();
00085   }
```

```
00086
00087   // ////////////////////////////////////////////////////////////////////
00088   std::string StringSet::describe() const {
00089     std::ostringstream oStr;
00090     oStr << describeKey();
00091
00092     //
00093     oStr << "  {";
00094
00095     short idx_string = 0;
00096     for (StringSet_T::const_iterator itString = _set.begin();
00097          itString != _set.end(); ++itString, ++idx_string) {
00098       //
00099       if (idx_string != 0) {
00100         oStr << ", ";
00101       }
00102
00103       //
00104       const std::string& lString = *itString;
00105
00106       //
00107       oStr << "\"" << lString << "\"";
00108     }
00109
00110     //
00111     oStr << "}";
00112
00113     return oStr.str();
00114   }
00115
00116   // ////////////////////////////////////////////////////////////////////
00117   void StringSet::toStream (std::ostream& ioOut) const {
00118     ioOut << describe();
00119   }
00120
00121   // ////////////////////////////////////////////////////////////////////
00122   void StringSet::fromStream (std::istream& ioIn) {
00123   }
00124
00125 }
```

24.171 opentrep/bom/StringSet.hpp File Reference

```
#include <string>
#include <list>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/basic/StructAbstract.hpp>
```

Classes

- struct OPENTREP::StringSet

  Class holding a set of strings, e.g., {"rio", "de", "janeiro"}.

Namespaces

- OPENTREP

24.172 StringSet.hpp

```
00001 #ifndef __OPENTREP_BOM_STRINGSET_HPP
00002 #define __OPENTREP_BOM_STRINGSET_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <list>
00010 // OpenTrep
00011 #include <opentrep/OPENTREP_Types.hpp>
00012 #include <opentrep/basic/StructAbstract.hpp>
```

```
00013
00014 namespace OPENTREP {
00015
00019   struct StringSet : public StructAbstract {
00020   public:
00021     // ///////////////// Type definitions /////////////////
00025     typedef std::list<std::string> StringSet_T;
00026
00027   public:
00034     void push_back (const std::string&);
00035
00039     void push_back (const StringSet&);
00040
00044     size_t size() const;
00045
00049     bool empty() const;
00050
00054     void clear();
00055
00059     std::string getFirstString() const;
00060
00061
00062   public:
00063     // /////////// Display support methods /////////
00069     void toStream (std::ostream&) const;
00070
00076     void fromStream (std::istream&);
00077
00082     std::string describeKey() const;
00083
00087     std::string describe() const;
00088
00089
00090   public:
00091     // ///////////////// Constructors and Destructors /////////////
00095     StringSet();
00096
00100     StringSet (const StringSet&);
00101
00105     StringSet (const std::string&);
00106
00110     ~StringSet();
00111
00112
00113   public:
00114     // ///////////////// Attributes ///////////////
00118     StringSet_T _set;
00119   };
00120
00121 }
00122 #endif // __OPENTREP_BOM_STRINGSET_HPP
```

24.173 opentrep/bom/WordCombinationHolder.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <set>
#include <opentrep/basic/Utilities.hpp>
#include <opentrep/bom/Filter.hpp>
#include <opentrep/bom/StringPartition.hpp>
#include <opentrep/bom/WordCombinationHolder.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.174 WordCombinationHolder.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
```

```
00005 #include <cassert>
00006 #include <sstream>
00007 #include <set>
00008 // OpenTrep
00009 #include <opentrep/basic/Utilities.hpp>
00010 #include <opentrep/bom/Filter.hpp>
00011 #include <opentrep/bom/StringPartition.hpp>
00012 #include <opentrep/bom/WordCombinationHolder.hpp>
00013 #include <opentrep/service/Logger.hpp>
00014
00015 namespace OPENTREP {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   WordCombinationHolder::WordCombinationHolder (const
      std::string& iString) {
00019     init (iString);
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   WordCombinationHolder::~WordCombinationHolder () {
00024   }
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   void WordCombinationHolder::push_back (const std::string& iString) {
00028     _list.push_back (iString);
00029   }
00030
00031   // ////////////////////////////////////////////////////////////////////
00032   size_t WordCombinationHolder::size() const {
00033     return _list.size();
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   void WordCombinationHolder::clear() {
00038     _list.clear();
00039   }
00040
00041   // ////////////////////////////////////////////////////////////////////
00042   std::string WordCombinationHolder::describeKey() const {
00043     std::ostringstream oStr;
00044     oStr << "";
00045     return oStr.str();
00046   }
00047
00048   // ////////////////////////////////////////////////////////////////////
00049   std::string WordCombinationHolder::describe() const {
00050     std::ostringstream oStr;
00051     oStr << describeKey();
00052
00053     //
00054     oStr << "{";
00055
00056     short idx_sublist = 0;
00057     for (StringList_T::const_iterator itWordCombination = _list.begin();
00058          itWordCombination != _list.end(); ++itWordCombination, ++idx_sublist) {
00059       //
00060       if (idx_sublist != 0) {
00061         oStr << ", ";
00062       }
00063
00064       //
00065       const StringSet& lStringSet = *itWordCombination;
00066
00067       //
00068       oStr << lStringSet;
00069     }
00070
00071     //
00072     oStr << " }";
00073
00074     return oStr.str();
00075   }
00076
00077   // ////////////////////////////////////////////////////////////////////
00078   void WordCombinationHolder::toStream (std::ostream& ioOut) const {
00079     ioOut << describe();
00080   }
00081
00082   // ////////////////////////////////////////////////////////////////////
00083   void WordCombinationHolder::fromStream (std::istream& ioIn) {
00084   }
00085
00086   // ////////////////////////////////////////////////////////////////////
00087   void WordCombinationHolder::init (const std::string& iPhrase) {
00088     // Set of unique strings
00089     typedef std::set<std::string> StringSet_T;
00090     StringSet_T lStringSet;
```

```
00091
00092     // 1. Derive all the partitions of the initial (full) string
00093     const StringPartition lStringPartitionHolder (iPhrase);
00094     const StringPartition::StringPartition_T& lStringPartition =
00095       lStringPartitionHolder._partition;
00096
00097     // 2.1. For every word combination, add it if not already in the
00098     //      list (STL set) of strings.
00099     for (StringPartition::StringPartition_T::const_iterator itSet =
00100            lStringPartition.begin(); itSet != lStringPartition.end(); ++itSet) {
00101       const StringSet& itStringList = *itSet;
00102
00103       const StringList_T& lStringList = itStringList._set;
00104       for (StringList_T::const_iterator itWordCombination = lStringList.begin();
00105            itWordCombination != lStringList.end(); ++itWordCombination) {
00106         const std::string& lWordCombination = *itWordCombination;
00107
00108         // Check whether the (remaining) word combination should be filtered out
00109         //const bool isToBeAdded= Filter::shouldKeep (iPhrase, lWordCombination);
00110         const bool isToBeAdded = true;
00111         if (isToBeAdded == true) {
00112           lStringSet.insert (lWordCombination);
00113         }
00114       }
00115     }
00116
00117     // 2.2. Convert the STL set into a STL list
00118     for (StringSet_T::const_iterator itWordCombination = lStringSet.begin();
00119          itWordCombination != lStringSet.end(); ++itWordCombination) {
00120       const std::string& lWordCombination = *itWordCombination;
00121
00122       // Add that word combination in the list for indexation by Xapian.
00123       // Note that if that word combination is already present in the list,
00124       // it will not be added a second time (thanks to the STL list design).
00125       _list.push_back (lWordCombination);
00126     }
00127
00128     // 3. Add the word combinations, made by removing all the possible groups
00129     //    of continuous words inbetween the two extreme words (from left- and
00130     //    right-hand sides).
00131     // 3.0. Initialisation of the list of words, made of all the words of the
00132     //      given string.
00133     WordList_T lWordList;
00134     tokeniseStringIntoWordList (iPhrase, lWordList);
00135     const short nbOfWords = lWordList.size();
00136
00137     // 3.1. If the string contains no more than two words, the job is finished.
00138     if (nbOfWords <= 2) {
00139       return;
00140     }
00141
00142     // 3.2. Iteration on the number of words to remove in the middle of the
00143     //      string, from 1 to (nbOfWords - 2)
00144     for (short mdl_string_len = 1; mdl_string_len != nbOfWords-1;
00145          ++mdl_string_len) {
00146
00147       // 3.2. Iteration on all the middle words of the given string,
00148       //      from 1 to (nbOfWords - mdl_string_len)
00149       for (short idx_word=1; idx_word != nbOfWords-mdl_string_len; ++idx_word) {
00150         // 3.2.1. Copy the first idx_word word(s)
00151         const std::string& lLeftHandString =
00152           createStringFromWordList (lWordList, idx_word);
00153
00154         // 3.2.2. Copy the last (nbOfWords - (idx_word + mdl_string_len)) words
00155         const std::string& lRightHandString =
00156           createStringFromWordList (lWordList,
00157                                     idx_word + mdl_string_len,
00158                                     false);
00159
00160         // 3.2.3. Concatenate both sub-strings
00161         std::ostringstream lConcatenatedStr;
00162         lConcatenatedStr << lLeftHandString << " " << lRightHandString;
00163         const std::string& lConcatenatedString = lConcatenatedStr.str();
00164
00165         // 3.2.4. Add the concatenated string into the list, if not filtered out
00166         // const bool isToBeAdded =
00167         //   Filter::shouldKeep (iPhrase, lConcatenatedString);
00168         const bool isToBeAdded = true;
00169         if (isToBeAdded == true) {
00170           _list.push_back (lConcatenatedString);
00171         }
00172       }
00173     }
00174   }
00175
00176 }
```

24.175 opentrep/bom/WordCombinationHolder.hpp File Reference

```
#include <string>
#include <list>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/basic/StructAbstract.hpp>
```

Classes

- struct OPENTREP::WordCombinationHolder

Namespaces

- OPENTREP

24.176 WordCombinationHolder.hpp

```
00001 #ifndef __OPENTREP_BOM_WORDCOMBINATIONHOLDER_HPP
00002 #define __OPENTREP_BOM_WORDCOMBINATIONHOLDER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <list>
00010 // OpenTrep
00011 #include <opentrep/OPENTREP_Types.hpp>
00012 #include <opentrep/basic/StructAbstract.hpp>
00013
00014 namespace OPENTREP {
00015
00051   struct WordCombinationHolder : public StructAbstract {
00052     // ///////////////// Type definitions ////////////////////
00056     typedef std::list<std::string> StringList_T;
00057
00058   public:
00062     void push_back (const std::string&);
00063
00067     size_t size() const;
00068
00072     void clear();
00073
00074
00075   private:
00083     void init (const std::string& iString);
00084
00085
00086   public:
00087     // ////////// Display support methods /////////
00093     void toStream (std::ostream&) const;
00094
00100     void fromStream (std::istream&);
00101
00106     std::string describeKey() const;
00107
00111     std::string describe() const;
00112
00113
00114   public:
00115     // ///////////////// Constructors and Destructors /////////////
00119     WordCombinationHolder (const std::string&);
00120
00124     ~WordCombinationHolder();
00125
00126
00127   public:
00128     // ///////////////// Attributes ///////////////
00132     StringList_T _list;
00133   };
00134
00135 }
00136 #endif // __OPENTREP_BOM_WORDCOMBINATIONHOLDER_HPP
```

24.177 opentrep/bom/WordHolder.cpp File Reference

```
#include <cassert>
#include <boost/tokenizer.hpp>
#include <opentrep/bom/WordHolder.hpp>
```

Namespaces

- OPENTREP

Functions

- static const boost::char_separator< char > OPENTREP::AllSepatorList (" .,;:|+-*/_=!@#$%‘~^&(){}[]?’<>\"")
- static const boost::char_separator< char > OPENTREP::DocSepatorList (" ,-%")
- void OPENTREP::baseTokeniseStringIntoWordList (const std::string &iPhrase, WordList_T &ioWordList, const boost::char_separator< char > &iSepatorList)

24.178 WordHolder.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // C
00005 #include <cassert>
00006 // Boost
00007 #include <boost/tokenizer.hpp>
00008 // OpenTREP
00009 #include <opentrep/bom/WordHolder.hpp>
00010
00011 namespace OPENTREP {
00012
00013   // Define the separators
00014   static const boost::char_separator<char> AllSepatorList (" .,;:|+-*/_=!@#$%'~^&(){}[]?'<>\"
      ");
00015   static const boost::char_separator<char> DocSepatorList (" ,-%");
00016
00017   // //////////////////////////////////////////////////////////////////////
00018   void baseTokeniseStringIntoWordList (const std::string& iPhrase,
00019                                        WordList_T& ioWordList,
00020                                        const boost::char_separator<char>& iSepatorList) {
00021     // Empty the word list
00022     ioWordList.clear();
00023
00024     // Boost Tokeniser
00025     typedef boost::tokenizer<boost::char_separator<char> > Tokeniser_T;
00026
00027     // Initialise the phrase to be tokenised
00028     Tokeniser_T lTokens (iPhrase, iSepatorList);
00029     for (Tokeniser_T::const_iterator tok_iter = lTokens.begin();
00030          tok_iter != lTokens.end(); ++tok_iter) {
00031       const std::string& lTerm = *tok_iter;
00032       ioWordList.push_back (lTerm);
00033     }
00034   }
00035
00036   // //////////////////////////////////////////////////////////////////////
00037   void WordHolder::tokeniseStringIntoWordList (const std::string&
      iPhrase,
00038                                                WordList_T& ioWordList) {
00039     OPENTREP::baseTokeniseStringIntoWordList (iPhrase, ioWordList,
00040                                               AllSepatorList);
00041   }
00042
00043   // //////////////////////////////////////////////////////////////////////
00044   void WordHolder::tokeniseDocIntoWordList (const std::string& iPhrase,
00045                                             WordList_T& ioWordList) {
00046     OPENTREP::baseTokeniseStringIntoWordList (iPhrase, ioWordList,
00047                                               DocSepatorList);
00048   }
00049
```

```
00050   // ////////////////////////////////////////////////////////////////////
00051   std::string WordHolder::
00052   createStringFromWordList (const WordList_T& iWordList) {
00053     std::ostringstream oStr;
00054
00055     unsigned short idx = iWordList.size();
00056     for (WordList_T::const_iterator itWord = iWordList.begin();
00057          itWord != iWordList.end(); ++itWord, --idx) {
00058       const std::string& lWord = *itWord;
00059       oStr << lWord;
00060       if (idx > 1) {
00061         oStr << " ";
00062       }
00063     }
00064
00065     return oStr.str();
00066   }
00067
00068 }
```

24.179 opentrep/bom/WordHolder.hpp File Reference

```
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/bom/BomAbstract.hpp>
```

Classes

- class OPENTREP::WordHolder

  Class wrapping utility functions to transform back and forth strings from and into a list of words.

Namespaces

- OPENTREP

24.180 WordHolder.hpp

```
00001 #ifndef __OPENTREP_BOM_WORDHOLDER_HPP
00002 #define __OPENTREP_BOM_WORDHOLDER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // OpenTREP
00008 #include <opentrep/OPENTREP_Types.hpp>
00009 #include <opentrep/bom/BomAbstract.hpp>
00010
00011 namespace OPENTREP {
00012
00017   class WordHolder : public BomAbstract {
00018     friend class FacWordHolder;
00019   public:
00020
00021     // ///////////////// Business Methods /////////////////
00027     static void tokeniseStringIntoWordList (const
      TravelQuery_T&, WordList_T&);
00028
00034     static void tokeniseDocIntoWordList (const
      TravelQuery_T&, WordList_T&);
00035
00039     static std::string createStringFromWordList (const
      WordList_T& iWordList);
00040
00041   private:
00042     // ///////////////// Constructors and Destructors /////////////
00046     WordHolder();
00050     WordHolder (const WordHolder&);
00054     ~WordHolder();
00055
00056   private:
00057     // ///////////////// Attributes /////////////////
00058   };
00059
```

```
00060 }
00061 #endif // __OPENTREP_BOM_WORDHOLDER_HPP
```

24.181 opentrep/bom/World.cpp File Reference

```
#include <assert.h>
#include <sstream>
#include <opentrep/bom/Place.hpp>
#include <opentrep/bom/World.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.182 World.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // C
00005 #include <assert.h>
00006 // STL
00007 #include <sstream>
00008 // OpenTrep
00009 #include <opentrep/bom/Place.hpp>
00010 #include <opentrep/bom/World.hpp>
00011 #include <opentrep/service/Logger.hpp>
00012
00013 namespace OPENTREP {
00014
00015   // //////////////////////////////////////////////////////////////////////
00016   World::World () {
00017   }
00018
00019   // //////////////////////////////////////////////////////////////////////
00020   World::World (const World& iWorld) {
00021   }
00022
00023   // //////////////////////////////////////////////////////////////////////
00024   World::~World () {
00025   }
00026
00027   // //////////////////////////////////////////////////////////////////////
00028   std::string World::describeShortKey() const {
00029     std::ostringstream oStr;
00030     return oStr.str();
00031   }
00032
00033   // //////////////////////////////////////////////////////////////////////
00034   std::string World::describeKey() const {
00035     return describeShortKey();
00036   }
00037
00038   // //////////////////////////////////////////////////////////////////////
00039   std::string World::toString() const {
00040     return describeKey();
00041   }
00042
00043   // //////////////////////////////////////////////////////////////////////
00044   void World::toStream (std::ostream& ioOut) const {
00045     ioOut << toString();
00046   }
00047
00048   // //////////////////////////////////////////////////////////////////////
00049   void World::fromStream (std::istream& ioIn) {
00050   }
00051
00052   // //////////////////////////////////////////////////////////////////////
00053   std::string World::shortDisplay() const {
00054     std::ostringstream oStr;
00055     oStr << describeKey();
00056     return oStr.str();
00057   }
```

```
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   std::string World::display() const {
00061     std::ostringstream oStr;
00062
00063     oStr << shortDisplay();
00064
00065     for (PlaceOrderedList_T::const_iterator itPlace = _placeOrderedList.begin();
00066          itPlace != _placeOrderedList.end(); ++itPlace) {
00067       const Place* lPlace_ptr = *itPlace;
00068       assert (lPlace_ptr != NULL);
00069
00070       oStr << lPlace_ptr->display ();
00071     }
00072
00073     return oStr.str();
00074   }
00075
00076   // ////////////////////////////////////////////////////////////////////
00077   GenericBom_T World::getGenericBom (const
      XapianDocID_T& iDocID) const {
00078     BomType::EN_BomType oBomType = BomType::LAST_VALUE;
00079     BomAbstract* oBom_ptr = NULL;
00080
00081     GenericBomList_T::const_iterator itBom = _genericBomList.find (iDocID);
00082     if (itBom != _genericBomList.end()) {
00083       const GenericBom_T& lGenericBom = itBom->second;
00084       oBomType = lGenericBom.first;
00085       oBom_ptr = lGenericBom.second;
00086     }
00087
00088     return GenericBom_T (oBomType, oBom_ptr);
00089   }
00090
00091 }
```

24.183 opentrep/bom/World.hpp File Reference

```
#include <ostream>
#include <istream>
#include <string>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/bom/BomAbstract.hpp>
#include <opentrep/bom/GenericBom.hpp>
#include <opentrep/bom/PlaceList.hpp>
```

Classes

- class OPENTREP::World

Namespaces

- OPENTREP

Typedefs

- typedef std::size_t OPENTREP::WorldID_T

24.184 World.hpp

```
00001 #ifndef __OPENTREP_BOM_WORLD_HPP
00002 #define __OPENTREP_BOM_WORLD_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
```

```
00008 #include <ostream>
00009 #include <istream>
00010 #include <string>
00011 // OpenTrep
00012 #include <opentrep/OPENTREP_Types.hpp>
00013 #include <opentrep/bom/BomAbstract.hpp>
00014 #include <opentrep/bom/GenericBom.hpp>
00015 #include <opentrep/bom/PlaceList.hpp>
00016
00017 namespace OPENTREP {
00018
00020   class World : public BomAbstract {
00021     friend class FacWorld;
00022     friend class DbaWorld;
00023   public:
00024     // ////////////// Getters //////////////
00032     const PlaceOrderedList_T& getSimplePlaceList () const {
00033       return _placeOrderedList;
00034     }
00035
00036     // ////////////// Setters //////////////
00037
00038
00039     // ///////// Display methods ////////
00042     void toStream (std::ostream&) const;
00043
00046     void fromStream (std::istream&);
00047
00049     std::string toString() const;
00050
00053     std::string describeKey() const;
00054
00057     std::string describeShortKey() const;
00058
00060     std::string display() const;
00061
00063     std::string shortDisplay() const;
00064
00066     GenericBom_T getGenericBom (const XapianDocID_T& iDocID) const;
00067
00068   private:
00070     World ();
00071     World (const World&);
00072
00074     virtual ~World();
00075
00076   private:
00078     GenericBomList_T _genericBomList;
00079
00083     // PlaceDirectList_T _placeList;
00084
00088     PlaceOrderedList_T _placeOrderedList;
00089   };
00090
00091   // ////////////// Type definitions ////////////////////
00092   typedef std::size_t WorldID_T;
00093
00094 }
00095 #endif // __OPENTREP_BOM_WORLD_HPP
```

24.185 opentrep/command/DBManager.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <soci/soci.h>
#include <soci/sqlite3/soci-sqlite3.h>
#include <opentrep/bom/World.hpp>
#include <opentrep/bom/Place.hpp>
#include <opentrep/factory/FacPlace.hpp>
#include <opentrep/dbadaptor/DbaPlace.hpp>
#include <opentrep/command/DBManager.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.186 DBManager.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // SOCI
00008 #include <soci/soci.h>
00009 #include <soci/sqlite3/soci-sqlite3.h>
00010 // OpenTrep
00011 #include <opentrep/bom/World.hpp>
00012 #include <opentrep/bom/Place.hpp>
00013 #include <opentrep/factory/FacPlace.hpp>
00014 #include <opentrep/dbadaptor/DbaPlace.hpp>
00015 #include <opentrep/command/DBManager.hpp>
00016 #include <opentrep/service/Logger.hpp>
00017
00018 namespace OPENTREP {
00019
00020   // //////////////////////////////////////////////////////////////////////
00021   void DBManager::
00022   prepareSelectAllStatement (soci::session& ioSociSession,
00023                              soci::statement& ioSelectStatement,
00024                              Place& ioPlace) {
00025
00026     try {
00027
00028       // Instanciate a SQL statement (no request is performed at that stage)
00061       ioSelectStatement =
00062         (ioSociSession.prepare
00063          << "select dts.iata_code as iata_code, "
00064          << "  dts.location_type as location_type, "
00065          << "  dts.geoname_id as geoname_id, "
00066          << "  icao_code, faa_code, "
00067          << "  envelope_id, dts.name as utf_name, asciiname, "
00068          << "  latitude, longitude, fclass, fcode, "
00069          << "  page_rank, "
00070          << "  date(date_from) as date_from, date(date_until) as date_until, "
00071          << "  comment, "
00072          << "  country_code, cc2, country_name, continent_name, "
00073          << "  admin1_code, admin1_UTF8_name, admin1_ASCII_name, "
00074          << "  admin2_code, admin2_UTF8_name, admin2_ASCII_name, "
00075          << "  admin3_code, admin4_code, "
00076          << "  population, elevation, gtopo30, "
00077          << "  time_zone, gmt_offset, dst_offset, raw_offset, "
00078          << "  date(moddate) as moddate, "
00079          << "  state_code, wiki_link, "
00080          << "  alt.lang_code as alt_lang_code, alt.name as alt_name, "
00081          << "  alt.specifiers as alt_spec, "
00082          << "  city_iata_code, city_location_type, city_geoname_id, "
00083          << "  city_UTF8_name, city_ASCII_name "
00084          << "from ori_por_public_details dts, ori_por_public_alt_names alt, "
00085          << "     ori_por_public_served_cities cty "
00086          << "where dts.iata_code = alt.iata_code "
00087          << "  and dts.location_type = alt.location_type "
00088          << "  and dts.geoname_id = alt.geoname_id "
00089          << "  and dts.iata_code = cty.iata_code "
00090          << "  and dts.location_type = cty.location_type "
00091          << "  and dts.geoname_id = cty.geoname_id "
00092          << "order by dts.iata_code, dts.location_type, dts.geoname_id",
00093          soci::into (ioPlace));
00094
00095       // Execute the SQL query
00096       ioSelectStatement.execute();
00097
00098     } catch (std::exception const& lException) {
00099       std::ostringstream errorStr;
00100       errorStr << "Error in the 'select * from ori_por_public_xxx' SQL request: "
00101                << lException.what();
00102       OPENTREP_LOG_ERROR (errorStr.str());
00103       throw SQLDatabaseException (errorStr.str());
00104     }
00105   }
00106
00107   // //////////////////////////////////////////////////////////////////////
00108   void DBManager::
00109   prepareSelectFromCodeStatement (soci::session& ioSociSession,
```

```
00110                                 soci::statement& ioSelectStatement,
00111                                 Place& ioPlace) {
00112
00113     try {
00114
00115       // Instanciate a SQL statement (no request is performed at that stage)
00137       ioSelectStatement =
00138         (ioSociSession.prepare
00139          << "select rpd.iata_code, xapian_docid, icao_code, "
00140          << "is_geonames, geonameid, "
00141          << "latitude, longitude, fclass, fcode, "
00142          << "country_code, cc2, admin1, admin2, admin3, admin4, "
00143          << "population, elevation, gtopo30, "
00144          << "timezone, gmt_offset, dst_offset, raw_offset, moddate, "
00145          << "is_airport, is_commercial, "
00146          << "city_code, state_code, region_code, location_type, wiki_link, "
00147          << "language_code, ascii_name, utf_name, "
00148          << "alternate_name1, alternate_name2, alternate_name3, "
00149          << "alternate_name4, alternate_name5, alternate_name6, "
00150          << "alternate_name7, alternate_name8, alternate_name9, "
00151          << "alternate_name10, "
00152          << "page_rank, por_type "
00153          << "from place_names as pn, place_details as rpd "
00154          << "left join airport_pageranked pr on pr.iata_code = rpd.iata_code "
00155          << "where rpd.iata_code = pn.iata_code "
00156          << "order by rpd.iata_code, icao_code, geonameid", soci::into(ioPlace));
00157
00158       // Execute the SQL query
00159       ioSelectStatement.execute();
00160
00161     } catch (std::exception const& lException) {
00162       std::ostringstream errorStr;
00163       errorStr << "Error in the 'select * from place_details' SQL request: "
00164                << lException.what();
00165       OPENTREP_LOG_ERROR (errorStr.str());
00166       throw SQLDatabaseException (errorStr.str());
00167     }
00168   }
00169
00170   // ////////////////////////////////////////////////////////////////////
00171   void DBManager::
00172   prepareSelectFromCoordStatement (soci::session& ioSociSession,
00173                                    soci::statement& ioSelectStatement,
00174                                    const double& iLatitude,
00175                                    const double& iLongitude) {
00176
00177     try {
00178
00179       // Instanciate a SQL statement (no request is performed at that stage)
00203       Place& lPlace = FacPlace::instance().create();
00204       const double K_ERROR = 2.0;
00205       const double lLowerBoundLatitude = iLatitude - K_ERROR;
00206       const double lUpperBoundLatitude = iLatitude + K_ERROR;
00207       const double lLowerBoundLongitude = iLongitude - K_ERROR;
00208       const double lUpperBoundLongitude = iLongitude + K_ERROR;
00209
00210       ioSelectStatement =
00211         (ioSociSession.prepare
00212          << "select rpd.iata_code, xapian_docid, icao_code, "
00213          << "is_geonames, geonameid, "
00214          << "latitude, longitude, fclass, fcode, "
00215          << "country_code, cc2, admin1, admin2, admin3, admin4, "
00216          << "population, elevation, gtopo30, "
00217          << "timezone, gmt_offset, dst_offset, raw_offset, moddate, "
00218          << "is_airport, is_commercial, "
00219          << "city_code, state_code, region_code, location_type, wiki_link, "
00220          << "language_code, ascii_name, utf_name, "
00221          << "alternate_name1, alternate_name2, alternate_name3, "
00222          << "alternate_name4, alternate_name5, alternate_name6, "
00223          << "alternate_name7, alternate_name8, alternate_name9, "
00224          << "alternate_name10, "
00225          << "page_rank, por_type "
00226          << "from place_names as pn, place_details as rpd "
00227          << "left join airport_pageranked pr on pr.iata_code = rpd.iata_code "
00228          << "where latitude >= :lower_latitude "
00229          << "  and latitude <= :upper_latitude "
00230          << "  and longitude >= :lower_longitude "
00231          << "  and longitude <= :upper_longitude "
00232          << "  and rpd.iata_code = pn.iata_code",
00233          soci::into (lPlace), soci::use (lLowerBoundLatitude),
00234          soci::use (lUpperBoundLatitude), soci::use (lLowerBoundLongitude),
00235          soci::use (lUpperBoundLongitude));
00236
00237       // Execute the SQL query
00238       ioSelectStatement.execute();
00239
00240     } catch (std::exception const& lException) {
```

```
00241       std::ostringstream errorStr;
00242       errorStr << "Error in the 'select * from place_details' SQL request: "
00243                << lException.what();
00244       OPENTREP_LOG_ERROR (errorStr.str());
00245       throw SQLDatabaseException (errorStr.str());
00246     }
00247   }
00248
00249   // ////////////////////////////////////////////////////////////////////
00250   void DBManager::
00251   prepareSelectOnPlaceCodeStatement (soci::session& ioSociSession,
00252                                      soci::statement& ioSelectStatement,
00253                                      const std::string& iIataCode,
00254                                      const std::string& iIcaoCode,
00255                                      const GeonamesID_T& iGeonamesID,
00256                                      Place& ioPlace) {
00257
00258     try {
00259
00260       // Instanciate a SQL statement (no request is performed at that stage)
00284       ioSelectStatement =
00285         (ioSociSession.prepare
00286          << "select rpd.iata_code, xapian_docid, icao_code, "
00287          << "is_geonames, geonameid, "
00288          << "latitude, longitude, fclass, fcode, "
00289          << "country_code, cc2, admin1, admin2, admin3, admin4, "
00290          << "population, elevation, gtopo30, "
00291          << "timezone, gmt_offset, dst_offset, raw_offset, moddate, "
00292          << "is_airport, is_commercial, "
00293          << "city_code, state_code, region_code, location_type, wiki_link, "
00294          << "language_code, ascii_name, utf_name, "
00295          << "alternate_name1, alternate_name2, alternate_name3, "
00296          << "alternate_name4, alternate_name5, alternate_name6, "
00297          << "alternate_name7, alternate_name8, alternate_name9, "
00298          << "alternate_name10, "
00299          << "page_rank, por_type "
00300          << "from place_names as pn, place_details as rpd "
00301          << "left join airport_pageranked pr on pr.iata_code = rpd.iata_code "
00302          << "where rpd.iata_code = :place_iata_code "
00303          << "  and rpd.icao_code = :place_icao_code "
00304          << "  and rpd.geonameid = :place_geonameid "
00305          << "  and rpd.iata_code = pn.iata_code",
00306          soci::into (ioPlace), soci::use (iIataCode), soci::use (iIcaoCode),
00307          soci::use (iGeonamesID));
00308
00309       // Execute the SQL query
00310       ioSelectStatement.execute();
00311
00312     } catch (std::exception const& lException) {
00313       std::ostringstream errorStr;
00314       errorStr << "Error in the 'select * from place_details' SQL request: "
00315                << lException.what();
00316       OPENTREP_LOG_ERROR (errorStr.str());
00317       throw SQLDatabaseException (errorStr.str());
00318     }
00319   }
00320
00321   // ////////////////////////////////////////////////////////////////////
00322   void DBManager::
00323   prepareSelectOnDocIDStatement (soci::session& ioSociSession,
00324                                  soci::statement& ioSelectStatement,
00325                                  const XapianDocID_T& iDocID,
00326                                  Place& ioPlace) {
00327
00328     try {
00329
00330       // Instanciate a SQL statement (no request is performed at that stage)
00353       ioSelectStatement =
00354         (ioSociSession.prepare
00355          << "select rpd.iata_code, xapian_docid, icao_code, "
00356          << "is_geonames, geonameid, "
00357          << "latitude, longitude, fclass, fcode, "
00358          << "country_code, cc2, admin1, admin2, admin3, admin4, "
00359          << "population, elevation, gtopo30, "
00360          << "timezone, gmt_offset, dst_offset, raw_offset, moddate, "
00361          << "is_airport, is_commercial, "
00362          << "city_code, state_code, region_code, location_type, wiki_link, "
00363          << "language_code, ascii_name, utf_name, "
00364          << "alternate_name1, alternate_name2, alternate_name3, "
00365          << "alternate_name4, alternate_name5, alternate_name6, "
00366          << "alternate_name7, alternate_name8, alternate_name9, "
00367          << "alternate_name10, "
00368          << "page_rank, por_type "
00369          << "from place_names as pn, place_details as rpd "
00370          << "left join airport_pageranked pr on pr.iata_code = rpd.iata_code "
00371          << "where rpd.xapian_docid = :xapian_docid "
00372          << "  and rpd.iata_code = pn.iata_code "
```

```
00373         << "  and rpd.iata_code = pr.iata_code",
00374         soci::into (ioPlace), soci::use (iDocID));
00375
00376       // Execute the SQL query
00377       ioSelectStatement.execute();
00378
00379     } catch (std::exception const& lException) {
00380       std::ostringstream errorStr;
00381       errorStr << "Error: " << lException.what();
00382       OPENTREP_LOG_ERROR (errorStr.str());
00383       throw SQLDatabaseException (errorStr.str());
00384     }
00385   }
00386
00387   // ////////////////////////////////////////////////////////////////////
00388   bool DBManager::iterateOnStatement (soci::statement& ioStatement,
00389                                       Place& ioPlace) {
00390     bool hasStillData = false;
00391
00392     try {
00393
00394       // Retrieve the next row of Place object
00395       hasStillData = ioStatement.fetch();
00396
00397     } catch (std::exception const& lException) {
00398       std::ostringstream errorStr;
00399       errorStr << "Error when iterating on the SQL fetch: " << lException.what();
00400       errorStr << ". The current place key is: " << ioPlace.describeKey()
00401                << " (it may be mis-leading, though, if the key could not be retrieved).";
00402       OPENTREP_LOG_ERROR (errorStr.str());
00403       throw SQLDatabaseException (errorStr.str());
00404     }
00405
00406     return hasStillData;
00407   }
00408
00409   // ////////////////////////////////////////////////////////////////////
00410   void DBManager::updatePlaceInDB (soci::session& ioSociSession,
00411                                    const Place& iPlace) {
00412
00413     try {
00414
00415       // Begin a transaction on the database
00416       ioSociSession.begin();
00417
00418       // Instanciate a SQL statement (no request is performed at that stage)
00419       XapianDocID_T lDocID;
00420       std::string lIataCode;
00421       soci::statement lUpdateStatement =
00422         (ioSociSession.prepare
00423          << "update place_details "
00424          << "set xapian_docid = :xapian_docid "
00425          << "where iata_code = :iata_code",
00426          soci::use (lDocID), soci::use (lIataCode));
00427
00428       // Execute the SQL query
00429       lDocID = iPlace.getDocID();
00430       lIataCode = iPlace.getIataCode();
00431       lUpdateStatement.execute (true);
00432
00433       // Commit the transaction on the database
00434       ioSociSession.commit();
00435
00436       // Debug
00437       // OPENTREP_LOG_DEBUG ("[" << lDocID << "] " << iPlace);
00438
00439     } catch (std::exception const& lException) {
00440       std::ostringstream errorStr;
00441       errorStr << "Error when updating " << iPlace.toString() << ": "
00442                << lException.what();
00443       OPENTREP_LOG_ERROR (errorStr.str());
00444       throw SQLDatabaseException (errorStr.str());
00445     }
00446   }
00447
00448   // ////////////////////////////////////////////////////////////////////
00449   bool DBManager::retrievePlace (soci::session& ioSociSession,
00450                                  const LocationKey& iPlaceKey, Place& ioPlace) {
00451     bool oHasRetrievedPlace = false;
00452
00453     try {
00454
00455       // Prepare the SQL request corresponding to the select statement
00456       soci::statement lSelectStatement (ioSociSession);
00457       const std::string& lIataCode = iPlaceKey.getIataCode();
00458       const IATAType& lIataType = iPlaceKey.getIataType();
00459       const std::string& lIataTypeStr = lIataType.getTypeAsString();
```

```
00460       const GeonamesID_T& lGeonamesID = iPlaceKey.getGeonamesID();
00461       DBManager::prepareSelectOnPlaceCodeStatement (ioSociSession,
00462                                                     lSelectStatement,
00463                                                     lIataCode, lIataTypeStr,
00464                                                     lGeonamesID, ioPlace);
00469       bool hasStillData = true;
00470       while (hasStillData == true) {
00471         hasStillData = iterateOnStatement (lSelectStatement, ioPlace);
00472
00473         // It is enough to have (at least) one database retrieved row
00474         if (hasStillData == true) {
00475           oHasRetrievedPlace = true;
00476         }
00477
00478         // Debug
00479         OPENTREP_LOG_DEBUG ("[" << iPlaceKey << "] " << ioPlace);
00480       }
00481
00482     } catch (std::exception const& lException) {
00483       std::ostringstream errorStr;
00484       errorStr << "Error when trying to retrieve " << iPlaceKey
00485                << " from the SQLite3 database: " << lException.what();
00486       OPENTREP_LOG_ERROR (errorStr.str());
00487       throw SQLDatabaseException (errorStr.str());
00488     }
00489
00490     return oHasRetrievedPlace;
00491   }
00492
00493   // ////////////////////////////////////////////////////////////////////
00494   bool DBManager::retrieveClosestPlaces (soci::session& ioSociSession,
00495                                          const double& iLatitude,
00496                                          const double& iLongitude,
00497                                          PlaceOrderedList_T& ioPlaceList) {
00498     bool oHasRetrievedPlace = false;
00499
00500     try {
00501
00502
00503     } catch (std::exception const& lException) {
00504       std::ostringstream errorStr;
00505       errorStr << "Error when trying to retrieve the closest places for "
00506                << iLatitude << " / " << iLongitude << ": " << lException.what();
00507       OPENTREP_LOG_ERROR (errorStr.str());
00508       throw SQLDatabaseException (errorStr.str());
00509     }
00510
00511     return oHasRetrievedPlace;
00512   }
00513
00514   // ////////////////////////////////////////////////////////////////////
00515   NbOfDBEntries_T DBManager::
00516   buildSQLDB (const PORFilePath_T& iPORFilepath,
00517               const SQLiteDBFilePath_T& iSQLiteDBFilePath) {
00518     NbOfDBEntries_T oNbOfEntries = 0;
00519
00520     try {
00521
00522       // Open the connection to the database
00523       soci::session* lSociSession_ptr = new soci::session();
00524       assert (lSociSession_ptr != NULL);
00525       soci::session& lSociSession = *lSociSession_ptr;
00526       lSociSession.open (soci::sqlite3, iSQLiteDBFilePath);
00527
00528       // Prepare the SQL request corresponding to the select statement
00529       soci::statement lSelectStatement (lSociSession);
00530       Place& lPlace = FacPlace::instance().create();
00531       DBManager::prepareSelectAllStatement (lSociSession,
    lSelectStatement,
00532                                             lPlace);
00537       bool hasStillData = true;
00538       while (hasStillData == true) {
00539         hasStillData = iterateOnStatement (lSelectStatement, lPlace);
00540
00541         // It is enough to have (at least) one database retrieved row
00542         if (hasStillData == true) {
00543           ++oNbOfEntries;
00544
00545           // Debug
00546           OPENTREP_LOG_DEBUG ("[" << oNbOfEntries << "][" << lPlace.
    getKey()
00547                               << "] " << lPlace);
00548         }
00549       }
00550
00551     } catch (std::exception const& lException) {
00552       std::ostringstream errorStr;
```

```
00553       errorStr << "Error when trying to retrieve " << oNbOfEntries
00554                << "-th row from the SQLite3 database: " << lException.what();
00555       OPENTREP_LOG_ERROR (errorStr.str());
00556       throw SQLDatabaseException (errorStr.str());
00557     }
00558
00559     return oNbOfEntries;
00560   }
00561
00562 }
```

24.187 opentrep/command/DBManager.hpp File Reference

```
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/LocationKey.hpp>
#include <opentrep/bom/PlaceList.hpp>
```

Classes

- class OPENTREP::DBManager

  Class building the Business Object Model (BOM) from data retrieved from the database.

Namespaces

- soci
- OPENTREP

24.188 DBManager.hpp

```
00001 #ifndef __OPENTREP_CMD_DBMANAGER_HPP
00002 #define __OPENTREP_CMD_DBMANAGER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/OPENTREP_Types.hpp>
00009 #include <opentrep/LocationKey.hpp>
00010 #include <opentrep/bom/PlaceList.hpp>
00011
00012 // Forward declarations
00013 namespace soci {
00014   class session;
00015   class statement;
00016 }
00017
00018 namespace OPENTREP {
00019
00020   // Forward declarations
00021   struct PlaceKey;
00022
00023
00028   class DBManager {
00029   public:
00039     static NbOfDBEntries_T buildSQLDB (const
      PORFilePath_T&,
00040                                        const SQLiteDBFilePath_T&);
00041
00049     static void updatePlaceInDB (soci::session&, const Place&);
00050
00062     static bool retrievePlace (soci::session&, const LocationKey&,
      Place&);
00063
00077     static bool retrieveClosestPlaces (soci::session&, const double& iLatitude,
00078                                        const double& iLongitude,
00079                                        PlaceOrderedList_T&);
00080
00081
00082   public:
00091     static void prepareSelectAllStatement (soci::session&, soci::statement&,
00092                                            Place&);
```

```
00101     static void prepareSelectFromCodeStatement (soci::session&,
00102                                                 soci::statement&, Place&);
00103
00112     static void prepareSelectFromCoordStatement (soci::session&,
00113                                                  soci::statement&,
00114                                                  const double& iLatitude,
00115                                                  const double& iLongitude);
00116
00126     static bool iterateOnStatement (soci::statement&, Place&);
00127
00128
00129   private:
00141     static void prepareSelectOnPlaceCodeStatement(soci::session&,
00142                                                   soci::statement&,
00143                                                   const std::string& iPlaceCode,
00144                                                   const std::string& iIcaoCode,
00145                                                   const GeonamesID_T&,
      Place&);
00146
00156     static void prepareSelectOnDocIDStatement (soci::session&, soci::statement&,
00157                                                const XapianDocID_T&,
      Place&);
00158
00159
00160   private:
00164     DBManager() {}
00168     DBManager (const DBManager&) {}
00172     ~DBManager() {}
00173   };
00174
00175 }
00176 #endif // __OPENTREP_CMD_DBMANAGER_HPP
```

24.189 opentrep/command/DBSessionManager.cpp File Reference

```
#include <cassert>
#include <string>
#include <sstream>
#include <soci/soci.h>
#include <soci/sqlite3/soci-sqlite3.h>
#include <opentrep/DBParams.hpp>
#include <opentrep/command/DBSessionManager.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.190 DBSessionManager.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <sstream>
00008 // SOCI
00009 #include <soci/soci.h>
00010 #include <soci/sqlite3/soci-sqlite3.h>
00011 // OpenTrep
00012 #include <opentrep/DBParams.hpp>
00013 #include <opentrep/command/DBSessionManager.hpp>
00014 #include <opentrep/service/Logger.hpp>
00015
00016 namespace OPENTREP {
00017
00018   // //////////////////////////////////////////////////////////////////////
00019   DBSessionManager::DBSessionManager() : _dbSession (NULL) {
00020   }
00021
00022   // //////////////////////////////////////////////////////////////////////
00023   DBSessionManager::DBSessionManager (const DBParams& iDBParams)
```

```
00024     : _dbSession (NULL) {
00025     init (iDBParams);
00026   }
00027
00028   // ////////////////////////////////////////////////////////////////////
00029   DBSessionManager::~DBSessionManager() {
00030     delete _dbSession; _dbSession = NULL;
00031   }
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   soci::session& DBSessionManager::getDBSessionRef() const {
00035     assert (_dbSession != NULL);
00036     return *_dbSession;
00037   }
00038
00039   // ////////////////////////////////////////////////////////////////////
00040   void DBSessionManager::init (const DBParams& iDBParams) {
00041
00042     // Check that the parameters for the SQL database are not empty
00043     if (iDBParams.checkSQLite() == false) {
00044       std::ostringstream errorStr;
00045       errorStr << "At least one of the parameters for the SQL "
00046                << "database is empty: " << iDBParams;
00047       OPENTREP_LOG_ERROR (errorStr.str());
00048       throw XapianTravelDatabaseEmptyException (errorStr.str());
00049     }
00050
00051     // Instanciate a (SOCI) database session: nothing is performed at
00052     // that stage
00053     _dbSession = new soci::session();
00054
00055     try {
00056
00057       // Open the connection to the database
00058       _dbSession->open (soci::sqlite3, iDBParams.toSQLiteConnectionString());
00059
00060     } catch (std::exception const& lException) {
00061       std::ostringstream errorStr;
00062       errorStr << "Error while opening a connection to database: "
00063                << lException.what() << std::endl;
00064       errorStr << "Database parameters used: " << iDBParams.toString();
00065       OPENTREP_LOG_ERROR (errorStr.str());
00066       throw SQLDatabaseConnectionImpossibleException (errorStr.str());
00067     }
00068   }
00069
00070 }
```

24.191 opentrep/command/DBSessionManager.hpp File Reference

Classes

- class OPENTREP::DBSessionManager

  Class handling the SOCI session.

Namespaces

- soci
- OPENTREP

24.192 DBSessionManager.hpp

```
00001 #ifndef __OPENTREP_CMD_DBSESSIONMANAGER_HPP
00002 #define __OPENTREP_CMD_DBSESSIONMANAGER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007
00008 // Forward declarations
00009 namespace soci {
00010   class session;
00011 }
00012
00013 namespace OPENTREP {
```

```
00014
00015   // Forward declarations
00016   struct DBParams;
00017
00021   class DBSessionManager {
00022     friend class OPENTREP_ServiceContext;
00023   private:
00024     // ////////////////// Getters ////////////////////
00028     soci::session* getDBSession() const {
00029       return _dbSession;
00030     }
00031
00035     soci::session& getDBSessionRef() const;
00036
00037
00038   private:
00039     // ////////////////// Constructors and Destructors ////////////////////
00043     DBSessionManager (const DBParams&);
00044
00048     DBSessionManager();
00049
00053     DBSessionManager (const DBSessionManager&);
00054
00058     ~DBSessionManager();
00059
00060   private:
00064     void init (const DBParams&);
00065
00066
00067   private:
00068     // ////////////////////// Attributes ////////////////////
00072     soci::session* _dbSession;
00073   };
00074
00075 }
00076 #endif // __OPENTREP_CMD_DBSESSIONMANAGER_HPP
```

24.193 opentrep/command/IndexBuilder.cpp File Reference

```
#include <cassert>
#include <string>
#include <vector>
#include <exception>
#include <boost/filesystem.hpp>
#include <boost/filesystem/fstream.hpp>
#include <boost/tokenizer.hpp>
#include <opentrep/basic/OTransliterator.hpp>
#include <opentrep/basic/Utilities.hpp>
#include <opentrep/bom/StringPartition.hpp>
#include <opentrep/bom/WordCombinationHolder.hpp>
#include <opentrep/bom/World.hpp>
#include <opentrep/bom/Place.hpp>
#include <opentrep/bom/PORParserHelper.hpp>
#include <opentrep/factory/FacPlace.hpp>
#include <opentrep/command/IndexBuilder.hpp>
#include <opentrep/service/Logger.hpp>
#include <xapian.h>
```

Namespaces

- OPENTREP

Functions

- void OPENTREP::addToXapian (const Place &iPlace, Xapian::Document &ioDocument, Xapian::WritableDatabase &ioDatabase)

24.194 IndexBuilder.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <vector>
00008 #include <exception>
00009 // Boost
00010 #include <boost/filesystem.hpp>
00011 #include <boost/filesystem/fstream.hpp>
00012 #include <boost/tokenizer.hpp>
00013 // OpenTrep
00014 #include <opentrep/basic/OTransliterator.hpp>
00015 #include <opentrep/basic/Utilities.hpp>
00016 #include <opentrep/bom/StringPartition.hpp>
00017 #include <opentrep/bom/WordCombinationHolder.hpp>
00018 #include <opentrep/bom/World.hpp>
00019 #include <opentrep/bom/Place.hpp>
00020 #include <opentrep/bom/PORParserHelper.hpp>
00021 #include <opentrep/factory/FacPlace.hpp>
00022 #include <opentrep/command/IndexBuilder.hpp>
00023 #include <opentrep/service/Logger.hpp>
00024 // Xapian
00025 #include <xapian.h>
00026
00027 namespace OPENTREP {
00028
00029   // //////////////////////////////////////////////////////////////////////
00030   void addToXapian (const Place& iPlace, Xapian::Document& ioDocument,
00031                     Xapian::WritableDatabase& ioDatabase) {
00038     Xapian::TermGenerator lTermGenerator;
00039     lTermGenerator.set_database (ioDatabase);
00040     lTermGenerator.set_document (ioDocument);
00041
00042     // DEBUG
00043     // OPENTREP_LOG_DEBUG ("Indexing for " << iPlace.describeKey());
00044
00045     const Place::TermSetMap_T& lTermSetMap = iPlace.
      getTermSetMap();
00046     for (Place::TermSetMap_T::const_iterator itStringSet = lTermSetMap.begin();
00047          itStringSet != lTermSetMap.end(); ++itStringSet) {
00048       // Retrieve the weight
00049       const Weight_T& lWeight = itStringSet->first;
00050       const Xapian::termcount lWDFInc =
00051         static_cast<const Xapian::termcount> (lWeight);
00052
00053       // Retrieve the set of strings for that weight
00054       const Place::StringSet_T& lTermSet = itStringSet->second;
00055       for (Place::StringSet_T::const_iterator itString = lTermSet.begin();
00056            itString != lTermSet.end(); ++itString) {
00057         const std::string& lString = *itString;
00058         lTermGenerator.index_text (lString, lWDFInc);
00059
00060         // DEBUG
00061         //OPENTREP_LOG_DEBUG("[" << lWeight << "/" << lWDFInc << "] "<< lString);
00062       }
00063     }
00064
00065     // Spelling terms
00066     const Place::StringSet_T& lSpellingSet = iPlace.
      getSpellingSet();
00067     for (Place::StringSet_T::const_iterator itTerm = lSpellingSet.begin();
00068          itTerm != lSpellingSet.end(); ++itTerm) {
00069       const std::string& lTerm = *itTerm;
00070       ioDatabase.add_spelling (lTerm);
00071     }
00072
00073     // DEBUG
00074     OPENTREP_LOG_DEBUG ("Added terms for '" << iPlace.
      describeKey()
00075                         << "': " << iPlace.describeSets()
00076                         << " into " << ioDocument.get_description());
00077   }
00078
00079   // //////////////////////////////////////////////////////////////////////
00080   void IndexBuilder::addDocumentToIndex(Xapian::WritableDatabase& ioDatabase,
00081                                         Place& ioPlace,
00082                                         const OTransliterator& iTransliterator) {
00083
00084     // Create an empty Xapian document
00085     Xapian::Document lDocument;
00086
00087     // Retrieve the raw data string, to be stored as is within
00088     // the Xapian document
```

```
00089     const RawDataString_T& lRawDataString = ioPlace.getRawDataString();
00090
00091     // The Xapian document data is indeed the same as the one of the
00092     // ORI-maintained list of POR (points of reference), allowing the search
00093     // process to use exactly the same parser as the indexation process
00094     lDocument.set_data (lRawDataString);
00095
00096     // Build the (STL) sets of terms to be added to the Xapian index and
00097     // spelling dictionary
00098     ioPlace.buildIndexSets (iTransliterator);
00099
00100    // Add the (STL) sets of terms to the Xapian index and spelling dictionary
00101    addToXapian (ioPlace, lDocument, ioDatabase);
00102
00103    // Add the document to the database
00104    const Xapian::docid& lDocID = ioDatabase.add_document (lDocument);
00105
00106    // Assign back the newly generated Xapian document ID to the
00107    // Place object
00108    ioPlace.setDocID (lDocID);
00109  }
00110
00111  // ////////////////////////////////////////////////////////////////////
00112  NbOfDBEntries_T IndexBuilder::
00113  buildSearchIndex (const PORFilePath_T& iPORFilePath,
00114                    const TravelDBFilePath_T& iTravelDBFilePath,
00115                    const OTransliterator& iTransliterator) {
00116    NbOfDBEntries_T oNbOfEntries = 0;
00117
00118    // Check that the directory for the Xapian database (index) exists and,
00119    // if not, create it.
00120    // DEBUG
00121    OPENTREP_LOG_DEBUG ("The Xapian database ('" << iTravelDBFilePath
00122                        << "') will be cleared");
00123    boost::filesystem::path lTravelDBFilePath (iTravelDBFilePath.begin(),
00124                                               iTravelDBFilePath.end());
00125    boost::filesystem::remove_all (lTravelDBFilePath);
00126    boost::filesystem::create_directories (lTravelDBFilePath);
00127
00128    // Check whether the just created directory exists and is a directory.
00129    if (!(boost::filesystem::exists (lTravelDBFilePath)
00130          && boost::filesystem::is_directory (lTravelDBFilePath))) {
00131      std::ostringstream oStr;
00132      oStr << "The file-path to the Xapian database/index ('"
00133           << iPORFilePath << "') does not exist or is not a directory.";
00134      OPENTREP_LOG_ERROR (oStr.str());
00135      throw FileNotFoundException (oStr.str());
00136    }
00137
00138    // Create the Xapian database (index). As the directory has been fully
00139    // cleaned, deleted and re-created, that Xapian database (index) is empty.
00140    Xapian::WritableDatabase lDatabase (iTravelDBFilePath, Xapian::DB_CREATE);
00141
00142    // DEBUG
00143    OPENTREP_LOG_DEBUG ("The Xapian database ('" << iTravelDBFilePath
00144                        << "') has been checked and open");
00145
00154    lDatabase.begin_transaction();
00155
00156    // DEBUG
00157    OPENTREP_LOG_DEBUG ("A transaction has begun on the Xapian database ('"
00158                        << iTravelDBFilePath << "')");
00159
00164    // DEBUG
00165    OPENTREP_LOG_DEBUG ("Parsing por input file: " << iPORFilePath);
00166
00167    // Check whether the file to be parsed exists and is readable.
00168    boost::filesystem::path lPORFilePath (iPORFilePath.begin(),
00169                                          iPORFilePath.end());
00170    if (!(boost::filesystem::exists (lPORFilePath)
00171          && boost::filesystem::is_regular_file (lPORFilePath))) {
00172      OPENTREP_LOG_ERROR ("The POR file " << iPORFilePath
00173                          << " does not exist or cannot be open." << std::endl);
00174
00175      throw FileNotFoundException ("The POR file " + iPORFilePath
00176                                   + " does not exist or cannot be read");
00177    }
00178
00179    // Open the file to be parsed
00180    boost::filesystem::ifstream fileToBeParsed (lPORFilePath);
00181    Place& lPlace = FacPlace::instance().create();
00182    std::string itReadLine;
00183    while (std::getline (fileToBeParsed, itReadLine)) {
00184      // Initialise the parser
00185      PORStringParser lStringParser (itReadLine);
00186
00187      // Parse the string
```

```
00188       const Location& lLocation = lStringParser.generateLocation();
00189       //const LocationKey& lLocationKey = lLocation.getKey();
00190
00191       // DEBUG
00192       //OPENTREP_LOG_DEBUG ("[BEF-ADD] " << lLocationKey);
00193
00194       // When the line/string was relevant, create a BOM instance from
00195       // the Location structure.
00196       if (!(lLocation.getCommonName() == "NotAvailable")) {
00197         // Fill the Place object with the Location structure.
00198         lPlace.setLocation (lLocation);
00199
00200         // Add the document, associated to the Place object, to the Xapian index
00201         IndexBuilder::addDocumentToIndex (lDatabase, lPlace, iTransliterator);
00202
00203         // DEBUG
00204         /*
00205         OPENTREP_LOG_DEBUG ("[AFT-ADD] " << lLocationKey
00206                             << ", Place: " << lPlace);
00207         */
00208
00209         // Iteration
00210         ++oNbOfEntries;
00211
00212         // DEBUG
00213         OPENTREP_LOG_DEBUG ("[" << oNbOfEntries << "] " << lPlace);
00214
00215         // Reset for next turn
00216         lPlace.resetMatrix();
00217         lPlace.resetIndexSets();
00218       }
00219     }
00220
00221     // Commit the pending modifications on the Xapian database (index)
00222     lDatabase.commit_transaction();
00223
00224     // DEBUG
00225     OPENTREP_LOG_DEBUG ("Xapian has indexed " << oNbOfEntries << " entries.");
00226
00234     lDatabase.close();
00235
00236     return oNbOfEntries;
00237   }
00238
00239 }
```

24.195 opentrep/command/IndexBuilder.hpp File Reference

```
#include <opentrep/OPENTREP_Types.hpp>
```

Classes

- class OPENTREP::IndexBuilder

  Command wrapping the travel request process.

Namespaces

- Xapian
- OPENTREP

24.196 IndexBuilder.hpp

```
00001 #ifndef __OPENTREP_CMD_INDEXBUILDER_HPP
00002 #define __OPENTREP_CMD_INDEXBUILDER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/OPENTREP_Types.hpp>
00009
00010 // Forward declarations
```

```
00011 namespace Xapian {
00012   class WritableDatabase;
00013 }
00014
00015 namespace OPENTREP {
00016
00017   // Forward declarations
00018   class Place;
00019   struct OTransliterator;
00020
00024   class IndexBuilder {
00025     friend class OPENTREP_Service;
00026   private:
00027
00035     static void addDocumentToIndex (Xapian::WritableDatabase&, Place&,
00036                                     const OTransliterator&);
00037
00045     static NbOfDBEntries_T buildSearchIndex (const PORFilePath_T&,
00046                                              const TravelDBFilePath_T&,
00047                                              const OTransliterator&);
00048
00049   private:
00053     IndexBuilder() {}
00054
00058     IndexBuilder(const IndexBuilder&) {}
00059
00063     ~IndexBuilder() {}
00064   };
00065
00066 }
00067 #endif // __OPENTREP_CMD_INDEXBUILDER_HPP
```

24.197 opentrep/command/PORParser.cpp File Reference

```
#include <cassert>
#include <string>
#include <opentrep/basic/BasFileMgr.hpp>
#include <opentrep/bom/PORParserHelper.hpp>
#include <opentrep/command/PORParser.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.198 PORParser.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 // OpenTREP
00008 #include <opentrep/basic/BasFileMgr.hpp>
00009 #include <opentrep/bom/PORParserHelper.hpp>
00010 #include <opentrep/command/PORParser.hpp>
00011 #include <opentrep/service/Logger.hpp>
00012
00013 namespace OPENTREP {
00014
00015   // //////////////////////////////////////////////////////////////////////
00016   void PORParser::PORGeneration (const PORFilePath_T& iPORFilename) {
00017
00018     // Check that the file path given as input corresponds to an actual file
00019     const bool doesExistAndIsReadable =
00020       BasFileMgr::doesExistAndIsReadable (iPORFilename);
00021     if (doesExistAndIsReadable == false) {
00022       OPENTREP_LOG_ERROR ("The POR input file, '" << iPORFilename
00023                           << "', can not be retrieved on the file-system");
00024       throw FileNotFoundException ("The POR input file '" + iPORFilename
00025                                    + "' does not exist or can not be read");
00026     }
00027
```

```
00028     // Initialise the POR file parser.
00029     PORFileParser lPorFileParser (iPORFilename);
00030
00031     // Parse the CSV-formatted POR input file and generate the corresponding
00032     // POR structures.
00033     lPorFileParser.generateLocations();
00034   }
00035
00036 }
```

24.199 opentrep/command/PORParser.hpp File Reference

```
#include <opentrep/OPENTREP_Types.hpp>
```

Classes

- class OPENTREP::PORParser

Namespaces

- OPENTREP

24.200 PORParser.hpp

```
00001 #ifndef __OPENTREP_CMD_PORPARSER_HPP
00002 #define __OPENTREP_CMD_PORPARSER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // OpenTREP
00008 #include <opentrep/OPENTREP_Types.hpp>
00009
00010 namespace OPENTREP {
00011
00015   class PORParser {
00016   public:
00023     static void PORGeneration (const PORFilePath_T&);
00024   };
00025 }
00026 #endif // __OPENTREP_CMD_PORPARSER_HPP
```

24.201 opentrep/command/RequestInterpreter.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <string>
#include <vector>
#include <exception>
#include <boost/filesystem.hpp>
#include <opentrep/bom/Filter.hpp>
#include <opentrep/bom/WordHolder.hpp>
#include <opentrep/bom/Place.hpp>
#include <opentrep/bom/ResultCombination.hpp>
#include <opentrep/bom/ResultHolder.hpp>
#include <opentrep/bom/Result.hpp>
#include <opentrep/bom/PlaceHolder.hpp>
#include <opentrep/bom/QuerySlices.hpp>
#include <opentrep/bom/StringPartition.hpp>
#include <opentrep/factory/FacPlaceHolder.hpp>
#include <opentrep/factory/FacPlace.hpp>
#include <opentrep/factory/FacResultCombination.hpp>
#include <opentrep/factory/FacResultHolder.hpp>
#include <opentrep/factory/FacResult.hpp>
#include <opentrep/command/RequestInterpreter.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

Functions

- void OPENTREP::addUnmatchedWord (const TravelQuery_T &iQueryString, WordList_T &ioWordList, WordSet_T &ioWordSet)
- void OPENTREP::createPlaces (const ResultCombination &iResultCombination, PlaceHolder &ioPlaceHolder)
- void OPENTREP::searchString (const StringPartition &iStringPartition, const Xapian::Database &iDatabase, ResultCombination &ioResultCombination, WordList_T &ioWordList)
- void OPENTREP::chooseBestMatchingResultHolder (ResultCombination &ioResultCombination)

24.202 RequestInterpreter.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <string>
00008 #include <vector>
00009 #include <exception>
00010 // Boost
00011 #include <boost/filesystem.hpp>
00012 // OpenTrep
00013 #include <opentrep/bom/Filter.hpp>
00014 #include <opentrep/bom/WordHolder.hpp>
00015 #include <opentrep/bom/Place.hpp>
00016 #include <opentrep/bom/ResultCombination.hpp>
00017 #include <opentrep/bom/ResultHolder.hpp>
00018 #include <opentrep/bom/Result.hpp>
00019 #include <opentrep/bom/PlaceHolder.hpp>
00020 #include <opentrep/bom/QuerySlices.hpp>
00021 #include <opentrep/bom/StringPartition.hpp>
```

```
00022 #include <opentrep/factory/FacPlaceHolder.hpp>
00023 #include <opentrep/factory/FacPlace.hpp>
00024 #include <opentrep/factory/FacResultCombination.hpp>
00025 #include <opentrep/factory/FacResultHolder.hpp>
00026 #include <opentrep/factory/FacResult.hpp>
00027 #include <opentrep/command/RequestInterpreter.hpp>
00028 #include <opentrep/service/Logger.hpp>
00029
00030 namespace OPENTREP {
00031
00043   // ////////////////////////////////////////////////////////////////////
00044   void addUnmatchedWord (const TravelQuery_T& iQueryString,
00045                          WordList_T& ioWordList, WordSet_T& ioWordSet) {
00046     // Token-ise the given string
00047     WordList_T lQueryStringWordList;
00048     WordHolder::tokeniseStringIntoWordList (iQueryString,
00049                                             lQueryStringWordList);
00050     if (lQueryStringWordList.size() == 1) {
00051       // Add the unmatched/unknown word, only when that latter has not
00052       // already been stored, and when it is not black-listed.
00053       const bool shouldBeKept = Filter::shouldKeep ("", iQueryString);
00054       //const bool shouldBeKept = true;
00055
00056       WordSet_T::const_iterator itWord = ioWordSet.find (iQueryString);
00057       if (shouldBeKept == true && itWord == ioWordSet.end()) {
00058         ioWordSet.insert (iQueryString);
00059         ioWordList.push_back (iQueryString);
00060       }
00061     }
00062   }
00063
00064   // ////////////////////////////////////////////////////////////////////
00065   void createPlaces (const ResultCombination& iResultCombination,
00066                      PlaceHolder& ioPlaceHolder) {
00067
00068     // Retrieve the best matching ResultHolder object.
00069     const ResultHolder& lResultHolder =
00070       iResultCombination.getBestMatchingResultHolder();
00071
00072     // Browse the list of result objects
00073     const ResultList_T& lResultList = lResultHolder.getResultList();
00074     for (ResultList_T::const_iterator itResult = lResultList.begin();
00075          itResult != lResultList.end(); ++itResult) {
00076       // Retrieve the result object
00077       const Result* lResult_ptr = *itResult;
00078       assert (lResult_ptr != NULL);
00079
00084       const bool hasFullTextMatched = lResult_ptr->hasFullTextMatched();
00085       if (hasFullTextMatched == false) {
00086         continue;
00087       }
00088       assert (hasFullTextMatched == true);
00089
00090       // Retrieve the Xapian document data (string)
00091       const std::string& lDocDataStr = lResult_ptr->getBestDocData();
00092       const RawDataString_T& lDocData = RawDataString_T (lDocDataStr);
00093
00094       // Parse the POR details and create the corresponding Location structure
00095       const Location& lLocation = Result::retrieveLocation (lDocData);
00096
00097       // Instanciate an empty place object, which will be filled from the
00098       // rows retrieved from the database.
00099       Place& lPlace = FacPlace::instance().create (lLocation);
00100
00101       // Insert the Place object within the PlaceHolder object
00102       FacPlaceHolder::initLinkWithPlace (ioPlaceHolder, lPlace);
00103
00104       // Fill the place with the remaining of the Result details.
00105       lResult_ptr->fillPlace (lPlace);
00106
00107       // DEBUG
00108       OPENTREP_LOG_DEBUG ("Retrieved Document: " << lPlace.
      toString());
00109     }
00110   }
00111
00124   // ////////////////////////////////////////////////////////////////////
00125   void searchString (const StringPartition& iStringPartition,
00126                      const Xapian::Database& iDatabase,
00127                      ResultCombination& ioResultCombination,
00128                      WordList_T& ioWordList) {
00129
00130     // Catch any thrown Xapian::Error exceptions
00131     try {
00132
00133       // Set of unknown words (just to eliminate the duplicates)
00134       WordSet_T lWordSet;
```

```
00135
00136       // Browse the partitions
00137       for (StringPartition::StringPartition_T::const_iterator itSet =
00138             iStringPartition._partition.begin();
00139           itSet != iStringPartition._partition.end(); ++itSet) {
00140         const StringSet& lStringSet = *itSet;
00141
00142         // DEBUG
00143         OPENTREP_LOG_DEBUG ("  ==========");
00144         OPENTREP_LOG_DEBUG ("  String set: " << lStringSet);
00145
00146         // Create a ResultHolder object.
00147         ResultHolder& lResultHolder =
00148           FacResultHolder::instance().create (lStringSet.
      describe(), iDatabase);
00149
00150         // Add the ResultHolder object to the dedicated list.
00151         FacResultCombination::initLinkWithResultHolder (
      ioResultCombination,
00152                                                         lResultHolder);
00153
00154         // Browse through all the word combinations of the partition
00155         for (StringSet::StringSet_T::const_iterator itString =
00156                lStringSet._set.begin();
00157              itString != lStringSet._set.end(); ++itString) {
00158           //
00159           const std::string lQueryString (*itString);
00160
00161           // DEBUG
00162           OPENTREP_LOG_DEBUG ("    --------");
00163           OPENTREP_LOG_DEBUG ("    Query string: '" << lQueryString << "'");
00164
00165           // Create an empty Result object
00166           Result& lResult = FacResult::instance().
      create (lQueryString,
00167                                                   iDatabase);
00168
00169           // Add the Result object to the dedicated list.
00170           FacResultHolder::initLinkWithResult (lResultHolder, lResult);
00171
00172           // Perform the Xapian-based full-text match: the set of
00173           // matching documents is filled.
00174           const std::string& lMatchedString =
00175             lResult.fullTextMatch (iDatabase, lQueryString);
00176
00177           // When a single-word string is unmatched/unknown by/from Xapian,
00178           // add it to the dedicated list (i.e., ioWordList).
00179           if (lMatchedString.empty() == true) {
00180             OPENTREP::addUnmatchedWord (lQueryString, ioWordList, lWordSet);
00181           }
00182         }
00183
00184         // DEBUG
00185         OPENTREP_LOG_DEBUG (std::endl
00186                             << "=========================================="
00187                             << std::endl << "Result holder: "
00188                             << lResultHolder.toString() << std::endl
00189                             << "=========================================="
00190                             << std::endl << std::endl);
00191       }
00192
00193       // DEBUG
00194       OPENTREP_LOG_DEBUG ("********************");
00195
00196     } catch (const Xapian::Error& error) {
00197       // Error
00198       OPENTREP_LOG_ERROR ("Exception: "  << error.get_msg());
00199       throw XapianException (error.get_msg());
00200     }
00201   }
00202
00219   // ////////////////////////////////////////////////////////////////////
00220   void chooseBestMatchingResultHolder (
      ResultCombination& ioResultCombination) {
00221
00222     // Calculate the weights for the full-text matches
00223     const bool doesBestMatchingResultHolderExist =
00224       ioResultCombination.chooseBestMatchingResultHolder();
00225
00226     if (doesBestMatchingResultHolderExist == true) {
00227       const ResultHolder& lBestMatchingResultHolder =
00228         ioResultCombination.getBestMatchingResultHolder();
00229
00230       // DEBUG
00231       const StringSet& lCorrectedStringSet =
00232         ioResultCombination.getCorrectedStringSet();
00233       OPENTREP_LOG_DEBUG ("The best matching string partition for '"
```

```
00234                           << ioResultCombination.describeShortKey() << "' is "
00235                           << lBestMatchingResultHolder.describeShortKey()
00236                           << ", and has got a weight of "
00237                           << ioResultCombination.getBestMatchingWeight()
00238                           << "%. The corrected string set is: "
00239                           << lCorrectedStringSet);
00240
00241     } else {
00242       // DEBUG
00243       OPENTREP_LOG_DEBUG ("There is no match for '"
00244                           << ioResultCombination.describeShortKey() << "'");
00245     }
00246   }
00247
00248   // ////////////////////////////////////////////////////////////////////
00249   NbOfMatches_T RequestInterpreter::
00250   interpretTravelRequest (const TravelDBFilePath_T& iTravelDBFilePath,
00251                           const TravelQuery_T& iTravelQuery,
00252                           LocationList_T& ioLocationList,
00253                           WordList_T& ioWordList) {
00254     NbOfMatches_T oNbOfMatches = 0;
00255
00256     // Sanity check
00257     assert (iTravelQuery.empty() == false);
00258
00259     // Create a PlaceHolder object, to collect the matching Place objects
00260     PlaceHolder& lPlaceHolder = FacPlaceHolder::instance().
    create();
00261
00262     // Check whether the file-path to the Xapian database/index exists
00263     // and is a directory.
00264     boost::filesystem::path lTravelDBFilePath (iTravelDBFilePath.begin(),
00265                                                iTravelDBFilePath.end());
00266     if (!(boost::filesystem::exists (lTravelDBFilePath)
00267           && boost::filesystem::is_directory (lTravelDBFilePath))) {
00268       std::ostringstream oStr;
00269       oStr << "The file-path to the Xapian database/index ('"
00270            << iTravelDBFilePath << "') does not exist or is not a directory.";
00271       OPENTREP_LOG_ERROR (oStr.str());
00272       throw FileNotFoundException (oStr.str());
00273     }
00274
00275     // Open the Xapian database
00276     Xapian::Database lXapianDatabase (iTravelDBFilePath);
00277
00278     // DEBUG
00279     OPENTREP_LOG_DEBUG (std::endl
00280                         << "=========================================");
00281
00282     // First, cut the travel query in slices and calculate all the partitions
00283     // for each of those query slices
00284     QuerySlices lQuerySlices (lXapianDatabase, iTravelQuery);
00285
00286     // DEBUG
00287     OPENTREP_LOG_DEBUG ("+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+");
00288     OPENTREP_LOG_DEBUG ("Travel query: `" << iTravelQuery << "'");
00289     OPENTREP_LOG_DEBUG ("Query slices: `" << lQuerySlices << "'");
00290
00291     // Browse the travel query slices
00292     const StringPartitionList_T& lStringPartitionList =
00293       lQuerySlices.getStringPartitionList();
00294     for (StringPartitionList_T::const_iterator itSlice =
00295            lStringPartitionList.begin();
00296          itSlice != lStringPartitionList.end(); ++itSlice) {
00297       StringPartition lStringPartition = *itSlice;
00298       const std::string& lTravelQuerySlice = lStringPartition.getInitialString();
00299
00305       ResultCombination& lResultCombination =
00306         FacResultCombination::instance().create (lTravelQuerySlice);
00307
00308       // DEBUG
00309       // DEBUG
00310       OPENTREP_LOG_DEBUG ("++++++++++++++++++++");
00311       OPENTREP_LOG_DEBUG ("Travel query slice: `" << lTravelQuerySlice << "'");
00312       OPENTREP_LOG_DEBUG ("Partitions: " << lStringPartition);
00313
00318       OPENTREP::searchString (lTravelQuerySlice, lXapianDatabase,
00319                               lResultCombination, ioWordList);
00320
00324       lResultCombination.displayXapianPercentages();
00325
00329       lResultCombination.calculateEnvelopeWeights();
00330
00335       lResultCombination.calculateCodeMatches();
00336
00340       lResultCombination.calculatePageRanks();
00341
```

```
00345       lResultCombination.calculateHeuristicWeights();
00346
00350       lResultCombination.calculateCombinedWeights();
00351
00355       OPENTREP::chooseBestMatchingResultHolder (lResultCombination)
      ;
00356
00362       createPlaces (lResultCombination, lPlaceHolder);
00363
00364       // DEBUG
00365       OPENTREP_LOG_DEBUG (std::endl
00366                           << "========================================"
00367                           << std::endl << "Summary:" << std::endl
00368                           << lPlaceHolder.toShortString() << std::endl
00369                           << "========================================"
00370                           << std::endl);
00371     }
00372
00378     lPlaceHolder.createLocations (ioLocationList);
00379     oNbOfMatches = ioLocationList.size();
00380
00381     return oNbOfMatches;
00382   }
00383
00384 }
```

24.203 opentrep/command/RequestInterpreter.hpp File Reference

```
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/Location.hpp>
```

Classes

- class OPENTREP::RequestInterpreter

  Command wrapping the travel request process.

Namespaces

- OPENTREP

24.204 RequestInterpreter.hpp

```
00001 #ifndef __OPENTREP_CMD_REQUESTINTERPRETER_HPP
00002 #define __OPENTREP_CMD_REQUESTINTERPRETER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/OPENTREP_Types.hpp>
00009 #include <opentrep/Location.hpp>
00010
00011 namespace OPENTREP {
00012
00016   class RequestInterpreter {
00017     friend class OPENTREP_Service;
00018   private:
00031     static NbOfMatches_T interpretTravelRequest (const
      TravelDBFilePath_T&,
00032                                                  const TravelQuery_T&,
00033                                                  LocationList_T&,
      WordList_T&);
00034
00035   private:
00039     RequestInterpreter() {}
00040     RequestInterpreter (const RequestInterpreter&) {}
00044     ~RequestInterpreter() {}
00045   };
00046
00047 }
00048 #endif // __OPENTREP_CMD_REQUESTINTERPRETER_HPP
```

24.205 opentrep/command/XapianIndexManager.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <string>
#include <vector>
#include <exception>
#include <boost/filesystem.hpp>
#include <boost/random/random_device.hpp>
#include <boost/random/uniform_int_distribution.hpp>
#include <xapian.h>
#include <opentrep/Location.hpp>
#include <opentrep/bom/Result.hpp>
#include <opentrep/command/XapianIndexManager.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

Functions

- boost::filesystem::path OPENTREP::checkTravelDBFilePath (const TravelDBFilePath_T &iTravelDBFilePath)

24.206 XapianIndexManager.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <string>
00008 #include <vector>
00009 #include <exception>
00010 // Boost
00011 #include <boost/filesystem.hpp>
00012 #include <boost/random/random_device.hpp>
00013 #include <boost/random/uniform_int_distribution.hpp>
00014 // Xapian
00015 #include <xapian.h>
00016 // OpenTrep
00017 #include <opentrep/Location.hpp>
00018 #include <opentrep/bom/Result.hpp>
00019 #include <opentrep/command/XapianIndexManager.hpp>
00020 #include <opentrep/service/Logger.hpp>
00021
00022 namespace OPENTREP {
00023
00024   // //////////////////////////////////////////////////////////////////
00025   boost::filesystem::path
00026   checkTravelDBFilePath (const TravelDBFilePath_T& iTravelDBFilePath
      ) {
00027     boost::filesystem::path oTravelDBFilePath (iTravelDBFilePath.begin(),
00028                                                iTravelDBFilePath.end());
00029     if (!(boost::filesystem::exists (oTravelDBFilePath)
00030           && boost::filesystem::is_directory (oTravelDBFilePath))) {
00031       std::ostringstream oStr;
00032       oStr << "The file-path to the Xapian database/index ('"
00033            << iTravelDBFilePath << "') does not exist or is not a directory.";
00034       OPENTREP_LOG_ERROR (oStr.str());
00035       throw FileNotFoundException (oStr.str());
00036     }
00037
00038     return oTravelDBFilePath;
00039   }
00040
00041   // //////////////////////////////////////////////////////////////////
00042   NbOfDBEntries_T XapianIndexManager::
00043   getSize (const TravelDBFilePath_T& iTravelDBFilePath) {
```

```
00044     NbOfDBEntries_T oNbOfDBEntries = 0;
00045
00046     // Check whether the file-path to the Xapian database/index exists
00047     // and is a directory.
00048     checkTravelDBFilePath (iTravelDBFilePath);
00049
00050     // Open the Xapian database
00051     Xapian::Database lXapianDatabase (iTravelDBFilePath);
00052
00053     // Retrieve the actual number of documents indexed by the Xapian database
00054     const Xapian::doccount& lDocCount = lXapianDatabase.get_doccount();
00055
00056     //
00057     oNbOfDBEntries = static_cast<const NbOfDBEntries_T> (lDocCount);
00058
00059     return oNbOfDBEntries;
00060   }
00061
00062   // ////////////////////////////////////////////////////////////////////
00063   NbOfMatches_T XapianIndexManager::
00064   drawRandomLocations (const TravelDBFilePath_T& iTravelDBFilePath,
00065                        const NbOfMatches_T& iNbOfDraws,
00066                        LocationList_T& ioLocationList) {
00067     NbOfMatches_T oNbOfMatches = 0;
00068
00069     // Check whether the file-path to the Xapian database/index exists
00070     // and is a directory.
00071     checkTravelDBFilePath (iTravelDBFilePath);
00072
00073     // Open the Xapian database
00074     Xapian::Database lXapianDatabase (iTravelDBFilePath);
00075
00076     // Retrieve the number of documents indexed by the database
00077     const NbOfDBEntries_T& lTotalNbOfDocs = getSize (iTravelDBFilePath);
00078
00079     // No need to go further when the Xapian database (index) is empty
00080     if (lTotalNbOfDocs == 0) {
00081       //
00082       OPENTREP_LOG_NOTIFICATION ("The Xapian database is empty");
00083       return oNbOfMatches;
00084     }
00085
00086     // random_device is used as a source of entropy, since the generated
00087     // locations are expected not to be reproducible.
00088     boost::random::random_device lRandomDevice;
00089     boost::random::uniform_int_distribution<> uniformDistrib (1, lTotalNbOfDocs);
00090
00091     // Randomly generate document IDs. If they the corresponding documents
00092     // do not exist in the Xapian database, generate another one.
00093     for (NbOfMatches_T idx = 1; idx <= iNbOfDraws; ++idx) {
00094       unsigned int lRandomNbInt = uniformDistrib (lRandomDevice);
00095       Xapian::docid lDocID = static_cast<Xapian::docid> (lRandomNbInt);
00096
00097       // Retrieve the document from the Xapian database/index
00098       Xapian::Document::Internal* lDocPtr =
00099         lXapianDatabase.get_document_lazily (lDocID);
00100
00101       unsigned short currentNbOfIterations = 0;
00102       while (lDocPtr == NULL && currentNbOfIterations <= 100) {
00103         // DEBUG
00104         OPENTREP_LOG_DEBUG ("[" << idx << "] The " << lDocID
00105                             << " document ID does not exist in the Xapian "
00106                             << "database. Another ID will be generated.");
00107
00108         // Re-draw another random document ID
00109         lRandomNbInt = uniformDistrib (lRandomDevice);
00110         lDocID = static_cast<Xapian::docid> (lRandomNbInt);
00111
00112         // Retrieve the document from the Xapian database/index
00113         lDocPtr = lXapianDatabase.get_document_lazily (lDocID);
00114       }
00115
00116       // Bad luck: no document ID can be generated so that it corresponds to
00117       // an actual document in the Xapian database/index
00118       if (lDocPtr == NULL) {
00119         //
00120         OPENTREP_LOG_NOTIFICATION ("[" << idx << "] No document ID can be "
00121                                    << "generated so that it corresponds to "
00122                                    << "a document in the Xapian database.");
00123
00124       } else {
00125         // Retrieve the actual document.
00126         const Xapian::Document lDoc (lDocPtr);
00127         const std::string& lDocDataStr = lDoc.get_data();
00128         const RawDataString_T& lDocData = RawDataString_T (lDocDataStr);
00129
00130         // Parse the POR details and create the corresponding Location structure
```

```
00131         const Location& lLocation = Result::retrieveLocation (lDocData);
00132
00133         // Add the Location structure to the dedicated list
00134         ioLocationList.push_back (lLocation);
00135       }
00136     }
00137
00138     // Consistency check
00139     oNbOfMatches = ioLocationList.size();
00140     if (oNbOfMatches != iNbOfDraws) {
00141         //
00142         OPENTREP_LOG_NOTIFICATION (iNbOfDraws << " random draws were expected, "
00143                                    << "but " << oNbOfMatches
00144                                    << " have been generated.");
00145     }
00146
00147     //
00148     return oNbOfMatches;
00149   }
00150
00151 }
```

24.207 opentrep/command/XapianIndexManager.hpp File Reference

```
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/LocationList.hpp>
```

Classes

- class OPENTREP::XapianIndexManager

  Command wrapping utilities for the management of the Xapian (database) index.

Namespaces

- OPENTREP

24.208 XapianIndexManager.hpp

```
00001 #ifndef __OPENTREP_CMD_XAPIANINDEXMANAGER_HPP
00002 #define __OPENTREP_CMD_XAPIANINDEXMANAGER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/OPENTREP_Types.hpp>
00009 #include <opentrep/LocationList.hpp>
00010
00011 namespace OPENTREP {
00012
00017   class XapianIndexManager {
00018     friend class OPENTREP_Service;
00019   private:
00027     static NbOfDBEntries_T getSize (const TravelDBFilePath_T&);
00028
00037     static NbOfMatches_T drawRandomLocations (const
      TravelDBFilePath_T&,
00038                                               const NbOfMatches_T& iNbOfDraws,
00039                                               LocationList_T&);
00040
00041   private:
00045     XapianIndexManager() {}
00046     XapianIndexManager (const XapianIndexManager&) {}
00050     ~XapianIndexManager() {}
00051   };
00052
00053 }
00054 #endif // __OPENTREP_CMD_XAPIANINDEXMANAGER_HPP
```

24.209 opentrep/dbadaptor/DbaAbstract.cpp File Reference

```
#include <opentrep/dbadaptor/DbaAbstract.hpp>
```

Namespaces

- OPENTREP

24.210 DbaAbstract.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // OPENTREP
00005 #include <opentrep/dbadaptor/DbaAbstract.hpp>
00006
00007 namespace OPENTREP {
00008
00009 }
```

24.211 opentrep/dbadaptor/DbaAbstract.hpp File Reference

```
#include <iostream>
#include <sstream>
```

Classes

- class OPENTREP::DbaAbstract

Namespaces

- OPENTREP

Functions

- template<class charT , class traits >
  std::basic_ostream< charT,
  traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const OPENTREP::DbaAbstract &iDba)
- template<class charT , class traits >
  std::basic_istream< charT,
  traits > & operator>> (std::basic_istream< charT, traits > &ioIn, OPENTREP::DbaAbstract &ioDba)

24.211.1 Function Documentation

24.211.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut,* const OPENTREP::DbaAbstract & *iDba*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 42 of file DbaAbstract.hpp.

24.211.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, OPENTREP::DbaAbstract & *ioDba*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 70 of file DbaAbstract.hpp.

References OPENTREP::DbaAbstract::fromStream().

24.212 DbaAbstract.hpp

```
00001 #ifndef __OPENTREP_COM_DBA_DBAABSTRACT_HPP
00002 #define __OPENTREP_COM_DBA_DBAABSTRACT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iostream>
00009 #include <sstream>
00010
00011 namespace OPENTREP {
00012
00014   class DbaAbstract {
00015   public:
00016
00018     virtual ~DbaAbstract() {}
00019
00022     virtual void toStream (std::ostream& ioOut) const {}
00023
00026     virtual void fromStream (std::istream& ioIn) {}
00027
00028   protected:
00030     DbaAbstract() {}
00031   };
00032 }
00033
00039 template <class charT, class traits>
00040 inline
00041 std::basic_ostream<charT, traits>&
00042 operator<< (std::basic_ostream<charT, traits>& ioOut,
00043             const OPENTREP::DbaAbstract& iDba) {
00049   std::basic_ostringstream<charT,traits> ostr;
00050   ostr.copyfmt (ioOut);
00051   ostr.width (0);
00052
00053   // Fill string stream
00054   iDba.toStream (ostr);
00055
00056   // Print string stream
00057   ioOut << ostr.str();
00058
00059   return ioOut;
00060 }
00061
00067 template <class charT, class traits>
00068 inline
00069 std::basic_istream<charT, traits>&
00070 operator>> (std::basic_istream<charT, traits>& ioIn,
00071             OPENTREP::DbaAbstract& ioDba) {
00072   // Fill Dba object with input stream
00073   ioDba.fromStream (ioIn);
00074   return ioIn;
00075 }
00076
00077 #endif // __OPENTREP_COM_DBA_DBAABSTRACT_HPP
```

24.213 opentrep/dbadaptor/DbaPlace.cpp File Reference

```
#include <exception>
```

```
#include <string>
#include <sstream>
#include <ctime>
#include <opentrep/basic/BasConst_General.hpp>
#include <opentrep/bom/Place.hpp>
#include <opentrep/dbadaptor/DbaPlace.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- soci
- OPENTREP

24.214 DbaPlace.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <exception>
00006 #include <string>
00007 #include <sstream>
00008 #include <ctime>
00009 // OpenTrep
00010 #include <opentrep/basic/BasConst_General.hpp>
00011 #include <opentrep/bom/Place.hpp>
00012 #include <opentrep/dbadaptor/DbaPlace.hpp>
00013 #include <opentrep/service/Logger.hpp>
00014
00015 namespace soci {
00016
00017   // //////////////////////////////////////////////////////////////////////
00018   void type_conversion<OPENTREP::Place>::
00019   from_base (values const& iPlaceValues, indicator /* ind */,
00020              OPENTREP::Place& ioPlace) {
00021     /*
00022          select dts.iata_code, dts.location_type, dts.geoname_id,
00023          icao_code, faa_code,
00024          envelope_id, dts.name as utf_name, asciiname, latitude, longitude,
00025          fclass, fcode, page_rank,
00026          date(date_from) as date_from, date(date_until) as date_until,
00027          comment,
00028          country_code, cc2, country_name, continent_name,
00029          admin1_code, admin1_UTF8_name, admin1_ASCII_name,
00030          admin2_code, admin2_UTF8_name, admin2_ASCII_name,
00031          admin3_code, admin4_code,
00032          population, elevation, gtopo30,
00033          time_zone, gmt_offset, dst_offset, raw_offset,
00034          date(moddate) as moddate,
00035          state_code, wiki_link,
00036          alt.lang_code as alt_lang_code, alt.name as alt_name,
00037          alt.specifiers as alt_spec,
00038          city_iata_code, city_location_type, city_geoname_id,
00039          city_UTF8_name, city_ASCII_name
00040          from ori_por_public_details dts, ori_por_public_alt_names alt,
00041               ori_por_public_served_cities cty
00042          where dts.iata_code = alt.iata_code
00043            and dts.location_type = alt.location_type
00044            and dts.geoname_id = alt.geoname_id
00045            and dts.iata_code = cty.iata_code
00046            and dts.location_type = cty.location_type
00047            and dts.geoname_id = cty.geoname_id
00048          order by dts.iata_code, dts.location_type, dts.geoname_id
00049          ;
00050     */
00051
00052     /*
00053      * Retrieve the place key (IATA code, location type, Geonames ID).
00054      * If ioPlace has already that key, then only the alternate names need
00055      * to be added.
00056      * Otherwise, all the fields of the Place need to be set.
00057      */
00058     // First, retrieve the key from the database fields
00059     //   IATA code
00060     const std::string lIataCodeStr =
00061       iPlaceValues.get<std::string> ("iata_code", "");
```

```
00062     const OPENTREP::IATACode_T lIataCode (lIataCodeStr);
00063     //   Location type ('C' for city only, 'A' for airport only,
00064     //   'CA' for airport/city)
00065     const std::string lIataTypeStr =
00066       iPlaceValues.get<std::string> ("location_type", "");
00067     const OPENTREP::IATAType lIataType (lIataTypeStr);
00068     //   Geonames ID
00069     const OPENTREP::GeonamesID_T lGeonameID =
00070       iPlaceValues.get<int> ("geoname_id", 0);
00071     //
00072     const OPENTREP::LocationKey lNewPlaceKey (lIataCode, lIataType, lGeonameID);
00073
00074     // Then, retrieve the key from the previous Place object
00075     const OPENTREP::LocationKey lPlaceKey = ioPlace.getKey();
00076
00077     // Compare the keys
00078     if (lNewPlaceKey != lPlaceKey) {
00079       // The place keys are different: the current place retrieved from
00080       // the database is not the same as the one retrieved previously
00081
00082       // Re-set the (STL) sets of terms for the Xapian index, spelling
00083       // dictionary, etc.
00084       ioPlace.resetIndexSets();
00085
00086       // Reset the list of names of the given Place object
00087       ioPlace.resetMatrix();
00088
00089       // The IATA code will be set to the default value (empty string)
00090       // when the column is null
00091       ioPlace.setIataCode (lIataCode);
00092
00093       // The ICAO code will be set to the default value (empty string)
00094       // when the column is null
00095       ioPlace.setIataType (lIataTypeStr);
00096
00097       // Geonames ID
00098       ioPlace.setGeonamesID (lGeonameID);
00099
00100      // Envelope ID
00101      const OPENTREP::EnvelopeID_T lEnvelopeID =
00102        iPlaceValues.get<int> ("envelope_id");
00103      ioPlace.setEnvelopeID (lEnvelopeID);
00104
00105      // Beginning date of the validity range (empty when indefinitely valid)
00106      const std::string lDateFromStr =
00107        iPlaceValues.get<std::string> ("date_from", OPENTREP::DEFAULT_DATE_STR);
00108      const OPENTREP::Date_T lDateFrom (boost::gregorian::from_string (lDateFromStr));
00109      ioPlace.setDateFrom (lDateFrom);
00110
00111      // End date of the validity range (empty when indefinitely valid)
00112      const std::string lDateUntilStr =
00113        iPlaceValues.get<std::string> ("date_until", OPENTREP::DEFAULT_DATE_STR);
00114      const OPENTREP::Date_T lDateUntil (boost::gregorian::from_string (lDateUntilStr));
00115      ioPlace.setDateEnd (lDateUntil);
00116
00117      // The comment will be set to the default value (empty string)
00118      // when the column is null
00119      ioPlace.setComment (iPlaceValues.get<std::string> ("comment", ""));
00120
00121      // The UTF8 name will be set to the default value (empty string)
00122      // when the column is null
00123      ioPlace.setCommonName (iPlaceValues.get<std::string> ("utf_name", ""));
00124
00125      // The ASCII name will be set to the default value (empty string)
00126      // when the column is null
00127      ioPlace.setAsciiName (iPlaceValues.get<std::string> ("asciiname", ""));
00128
00129      // The FAA code will be set to the default value (empty string)
00130      // when the column is null
00131      ioPlace.setFaaCode (iPlaceValues.get<std::string> ("faa_code", ""));
00132
00133      // Feature class
00134      const std::string lFeatClass = iPlaceValues.get<std::string>("fclass", "");
00135      ioPlace.setFeatureClass (lFeatClass);
00136      // Feature code
00137      const std::string lFeatCode = iPlaceValues.get<std::string> ("fcode", "");
00138      ioPlace.setFeatureCode (lFeatCode);
00139
00140      // The state code will be set to the default value (empty string)
00141      // when the column is null
00142      ioPlace.setStateCode (iPlaceValues.get<std::string> ("state_code", ""));
00143      ioPlace.setCountryCode (iPlaceValues.get<std::string> ("country_code"));
00144      ioPlace.setCountryName (iPlaceValues.get<std::string> ("country_name"));
00145      ioPlace.setAltCountryCode (iPlaceValues.get<std::string> ("cc2", ""));
00146      ioPlace.setContinentName (iPlaceValues.get<std::string>("continent_name"));
00147      ioPlace.setLatitude (iPlaceValues.get<double> ("latitude"));
00148      ioPlace.setLongitude (iPlaceValues.get<double> ("longitude"));
```

```
00149       ioPlace.setPageRank (iPlaceValues.get<double> ("page_rank",
00150
    OPENTREP::K_DEFAULT_PAGE_RANK));
00151
00152       // The administrative level 1 details will be set to the default value
00153       // (empty string) when the column is null
00154       ioPlace.setAdmin1Code (iPlaceValues.get<std::string> ("admin1_code", ""));
00155       ioPlace.setAdmin1UtfName (iPlaceValues.get<std::string> ("admin1_UTF8_name", ""));
00156       ioPlace.setAdmin1AsciiName (iPlaceValues.get<std::string> ("admin1_ASCII_name", "")
    );
00157
00158       // The administrative level 1 details will be set to the default value
00159       // (empty string) when the column is null
00160       ioPlace.setAdmin2Code (iPlaceValues.get<std::string> ("admin2_code", ""));
00161       ioPlace.setAdmin2UtfName (iPlaceValues.get<std::string> ("admin2_UTF8_name", ""));
00162       ioPlace.setAdmin2AsciiName (iPlaceValues.get<std::string> ("admin2_ASCII_name", "")
    );
00163
00164       // The administrative level 3 code will be set to the default value
00165       // (empty string) when the column is null
00166       ioPlace.setAdmin3Code (iPlaceValues.get<std::string> ("admin3_code", ""));
00167
00168       // The administrative level 4 code will be set to the default value
00169       // (empty string) when the column is null
00170       ioPlace.setAdmin4Code (iPlaceValues.get<std::string> ("admin4_code", ""));
00171
00172       // Population
00173       const OPENTREP::Population_T lPopulation =
00174         iPlaceValues.get<int> ("population");
00175       ioPlace.setPopulation (lPopulation);
00176
00177       // Elevation
00178       const OPENTREP::Elevation_T lElevation =
00179         iPlaceValues.get<int> ("elevation");
00180       ioPlace.setElevation (lElevation);
00181
00182       // GTopo30
00183       const OPENTREP::GTopo30_T lGTopo30 = iPlaceValues.get<int> ("gtopo30");
00184       ioPlace.setGTopo30 (lGTopo30);
00185
00186       // The time-zone code will be set to the default value (empty string)
00187       // when the column is null
00188       ioPlace.setTimeZone (iPlaceValues.get<std::string> ("time_zone", ""));
00189
00190       // The time off-sets will be set to the default value (empty string)
00191       // when the column is null
00192       const OPENTREP::GMTOffset_T lGMTOffset =
00193         iPlaceValues.get<int> ("gmt_offset", 0.0);
00194       ioPlace.setGMTOffset (lGMTOffset);
00195       const OPENTREP::DSTOffset_T lDSTOffset =
00196         iPlaceValues.get<int> ("dst_offset", 0.0);
00197       ioPlace.setDSTOffset (lDSTOffset);
00198       const OPENTREP::RawOffset_T lRawOffset =
00199         iPlaceValues.get<int> ("raw_offset", 0.0);
00200       ioPlace.setRawOffset (lRawOffset);
00201
00202       // Modification date
00203       const std::string lModDateStr =
00204         iPlaceValues.get<std::string> ("moddate", OPENTREP::DEFAULT_DATE_STR);
00205       const OPENTREP::Date_T lModDate (boost::gregorian::from_string (lModDateStr));
00206       ioPlace.setModificationDate (lModDate);
00207
00208       // The Wikipedia link will be set to the default value (empty string)
00209       // when the column is null
00210       ioPlace.setWikiLink (iPlaceValues.get<std::string> ("wiki_link", ""));
00211     }
00212
00213     // Alternate name
00214     const std::string& lLanguageString =
00215       iPlaceValues.get<std::string> ("alt_lang_code", "");
00216     /*
00217     if (lLanguageString.empty() == true) {
00218       std::ostringstream errorStr;
00219       errorStr << "The language field is empty for the place "
00220                << ioPlace.getIataCode();
00221       OPENTREP_LOG_ERROR (errorStr.str());
00222       throw OPENTREP::LanguageCodeNotDefinedInNameTableException(errorStr.str());
00223     }
00224     */
00225
00226     // TODO: check that that alternate name does not already exist
00227     // Convert the language code string (e.g., "en") into
00228     // the corresponding enumeration
00229     //const OPENTREP::Language::EN_Language lLanguageCode =
00230     // OPENTREP::Language::getCode (lLanguageString);
00231     const OPENTREP::LanguageCode_T lLanguageStr (lLanguageString);
00232
```

```
00233     // Add the (UTF8) name only when the language code is standard English
00234     //if (lLanguageCode == OPENTREP::Language::en) {
00235     // UTF-8 name (which may be null/empty)
00236     const std::string& lUtfName =
00237       iPlaceValues.get<std::string> ("alt_name", "");
00238     ioPlace.addName (lLanguageStr, lUtfName);
00239     //}
00240
00241     // TODO: check that that city code/name does not already exist, and add it
00242     // The city code will be set to the default value (empty string)
00243     // when the column is null
00244     ioPlace.setCityCode (iPlaceValues.get<std::string> ("city_iata_code", ""));
00245     // The city UTF8 name will be set to the default value (empty string)
00246     // when the column is null
00247     ioPlace.setCityUtfName (iPlaceValues.get<std::string>("city_UTF8_name", ""));
00248     // The city ASCII name will be set to the default value (empty string)
00249     // when the column is null
00250     ioPlace.setCityAsciiName (iPlaceValues.get<std::string> ("city_ASCII_name", ""));
00251
00252
00253     // DEBUG
00254     /*
00255     OPENTREP_LOG_DEBUG ("[Old:" << lPlaceKey << "][New:" << lNewPlaceKey << "] Lang: "
00256                         << lLanguageString << " => "
00257                         << OPENTREP::Language::getLongLabel (lLanguageCode)
00258                         << " -- Details: " << ioPlace);
00259     */
00260   }
00261
00262   // ////////////////////////////////////////////////////////////////////
00263   void type_conversion<OPENTREP::Place>::
00264   to_base (const OPENTREP::Place& iPlace, values& ioPlaceValues,
00265            indicator& ioIndicator) {
00266     // IATA code
00267     const OPENTREP::IATACode_T& lIataCode = iPlace.
      getIataCode();
00268     const std::string lIataCodeStr (lIataCode);
00269     const indicator lIataCodeIndicator = lIataCodeStr.empty() ? i_null : i_ok;
00270
00271     // ICAO code
00272     const OPENTREP::ICAOCode_T& lIcaoCode = iPlace.
      getIcaoCode();
00273     const std::string lIcaoCodeStr (lIcaoCode);
00274     const indicator lIcaoCodeIndicator = lIcaoCodeStr.empty() ? i_null : i_ok;
00275
00276     // FAA code
00277     const OPENTREP::FAACode_T& lFaaCode = iPlace.getFaaCode();
00278     const std::string lFaaCodeStr (lFaaCode);
00279     const indicator lFaaCodeIndicator = lFaaCodeStr.empty() ? i_null : i_ok;
00280
00281     // City IATA code
00282     const OPENTREP::IATACode_T& lCityCode = iPlace.
      getCityCode();
00283     const std::string lCityCodeStr (lCityCode);
00284     const indicator lCityCodeIndicator = lCityCodeStr.empty() ? i_null : i_ok;
00285
00286     // State code
00287     const OPENTREP::StateCode_T& lStateCode = iPlace.
      getStateCode();
00288     const std::string lStateCodeStr (lStateCode);
00289     const indicator lStateCodeIndicator = lStateCodeStr.empty() ? i_null : i_ok;
00290
00291     // Country code
00292     const OPENTREP::CountryCode_T& lCountryCode = iPlace.
      getCountryCode();
00293     const std::string lCountryCodeStr (lCountryCode);
00294
00295     // Country name
00296     const OPENTREP::CountryName_T& lCountryName = iPlace.
      getCountryName();
00297     const std::string lCountryNameStr (lCountryName);
00298
00299     // Continent name
00300     const OPENTREP::ContinentName_T& lContinentName = iPlace.
      getContinentName();
00301     const std::string lContinentNameStr (lContinentName);
00302
00303     // Time-zone
00304     const OPENTREP::TimeZone_T& lTimeZone = iPlace.
      getTimeZone();
00305     const std::string lTimeZoneStr (lTimeZone);
00306
00307     // Wikipedia link
00308     const OPENTREP::WikiLink_T& lWikiLink = iPlace.
      getWikiLink();
00309     const std::string lWikiLinkStr (lWikiLink);
00310
```

```
00311     //
00312     ioPlaceValues.set ("iata_code", lIataCodeStr, lIataCodeIndicator);
00313     ioPlaceValues.set ("icao_code", lIcaoCodeStr, lIcaoCodeIndicator);
00314     ioPlaceValues.set ("geoname_id", iPlace.getGeonamesID());
00315     ioPlaceValues.set ("faa_code", lFaaCodeStr, lFaaCodeIndicator);
00316     ioPlaceValues.set ("city_code", lCityCodeStr, lCityCodeIndicator);
00317     ioPlaceValues.set ("state_code", lStateCodeStr, lStateCodeIndicator);
00318     ioPlaceValues.set ("country_code", lCountryCodeStr);
00319     ioPlaceValues.set ("country_name", lCountryNameStr);
00320     ioPlaceValues.set ("continent_name", lContinentNameStr);
00321     ioPlaceValues.set ("time_zone", lTimeZoneStr);
00322     ioPlaceValues.set ("latitude", iPlace.getLatitude());
00323     ioPlaceValues.set ("longitude", iPlace.getLongitude());
00324     ioPlaceValues.set ("page_rank", iPlace.getPageRank());
00325     ioPlaceValues.set ("wiki_link", lWikiLinkStr);
00326     ioIndicator = i_ok;
00327   }
00328
00329 }
00330
00331 namespace OPENTREP {
00332
00333 }
```

24.215 opentrep/dbadaptor/DbaPlace.hpp File Reference

```
#include <soci/soci.h>
```

Classes

- struct soci::type_conversion< OPENTREP::Place >

  Specify how the Place class can be converted to (resp. from) values stored into (resp. retrieved from) database, using the SOCI framework.

Namespaces

- OPENTREP
- soci

24.216 DbaPlace.hpp

```
00001 #ifndef __OPENTREP_DBA_DBAPLACE_HPP
00002 #define __OPENTREP_DBA_DBAPLACE_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // SOCI
00008 #if defined(SOCI_HEADERS_BURIED)
00009 #include <soci/core/soci.h>
00010 #else // SOCI_HEADERS_BURIED
00011 #include <soci/soci.h>
00012 #endif // SOCI_HEADERS_BURIED
00013
00014 // Forward declarations
00015 namespace OPENTREP {
00016   class Place;
00017 }
00018
00019 namespace soci {
00020
00026   template <>
00027   struct type_conversion<OPENTREP::Place> {
00028
00029     typedef values base_type;
00030
00034     static void from_base (values const& iPlaceValues, indicator,
00035                            OPENTREP::Place&);
00036
00037
00041     static void to_base (const OPENTREP::Place&, values& ioPlaceValues,
00042                          indicator&);
```

```
00043   };
00044 }
00045 #endif // __OPENTREP_DBA_DBAPLACE_HPP
```

24.217 opentrep/DBParams.hpp File Reference

```
#include <sstream>
#include <string>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/OPENTREP_Abstract.hpp>
```

Classes

- struct OPENTREP::DBParams

  Structure modelling a database parameters.

Namespaces

- OPENTREP

Typedefs

- typedef std::list< std::string > OPENTREP::DBParamsNameList_T

24.218 DBParams.hpp

```
00001 #ifndef __OPENTREP_DBPARAMS_HPP
00002 #define __OPENTREP_DBPARAMS_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <sstream>
00009 #include <string>
00010 // OpenTrep
00011 #include <opentrep/OPENTREP_Types.hpp>
00012 #include <opentrep/OPENTREP_Abstract.hpp>
00013
00014 namespace OPENTREP {
00015
00019   typedef std::list<std::string> DBParamsNameList_T;
00020
00021
00025   struct DBParams : public OPENTREP_Abstract {
00026   public:
00027     // ////////////////////// Getters ///////////////////////
00031     std::string getUser() const {
00032       return _user;
00033     }
00034
00038     std::string getPassword() const {
00039       return _passwd;
00040     }
00041
00045     std::string getHost() const {
00046       return _host;
00047     }
00048
00052     std::string getPort() const {
00053       return _port;
00054     }
00055
00059     std::string getDBName() const {
00060       return _dbname;
00061     }
00062
00063
```

```
00064   public:
00065     // ////////////////////// Setters ////////////////////////
00069     void setUser (const std::string& iUser) {
00070       _user = iUser;
00071     }
00072
00076     void setPassword (const std::string& iPasswd) {
00077       _passwd = iPasswd;
00078     }
00079
00083     void setHost (const std::string& iHost) {
00084       _host = iHost;
00085     }
00086
00090     void setPort (const std::string& iPort) {
00091       _port = iPort;
00092     }
00093
00097     void setDBName (const std::string& iDBName) {
00098       _dbname = iDBName;
00099     }
00100
00101
00102   public:
00103     // ////////////////////// Busines methods ////////////////////
00107     bool checkSQLite () const {
00108       if (_dbname.empty() == true) {
00109         return false;
00110       }
00111       return true;
00112     }
00113
00117     bool checkMySQL () const {
00118       if (_user.empty() == true || _passwd.empty() == true
00119           || _host.empty() == true || _port.empty()
00120           || _dbname.empty() == true) {
00121         return false;
00122       }
00123       return true;
00124     }
00125
00126   public:
00127     // ////////////////////// Display methods ////////////////////////
00133     void toStream (std::ostream& ioOut) const {
00134       ioOut << toString();
00135     }
00136
00141     void fromStream (std::istream&) {
00142     }
00143
00147     std::string toShortString() const {
00148       std::ostringstream oStr;
00149       oStr << _dbname << "." << _user << "@" << _host << ":" << _port;
00150       return oStr.str();
00151     }
00152
00156     std::string toString() const {
00157       std::ostringstream oStr;
00158       oStr << _dbname << "." << _user << "@" << _host << ":" << _port;
00159       return oStr.str();
00160     }
00161
00165     std::string toMySQLConnectionString() const {
00166       std::ostringstream oStr;
00167       oStr << "db=" << _dbname << " user=" << _user << " password=" << _passwd
00168            << " port=" << _port << " host=" << _host;
00169       return oStr.str();
00170     }
00171
00175     std::string toSQLiteConnectionString() const {
00176       std::ostringstream oStr;
00177       oStr << "db=" << _dbname;
00178       return oStr.str();
00179     }
00180
00181
00182   public:
00183     // ////////////////// Constructors and Destructors ////////////////////
00187     DBParams (const std::string& iDBUser, const std::string& iDBPasswd,
00188               const std::string& iDBHost, const std::string& iDBPort,
00189               const std::string& iDBName)
00190       : _user (iDBUser), _passwd (iDBPasswd), _host (iDBHost), _port (iDBPort),
00191         _dbname (iDBName) {
00192     }
00193     DBParams (const std::string& iDBName) : _dbname (iDBName) {
00194     }
00195
```

```
00199     // DBParams();
00203     // DBParams (const DBParams&);
00204
00208     virtual ~DBParams() {}
00209
00210
00211   private:
00212     // ////////////////////////// Attributes ////////////////////////
00216     std::string _user;
00220     std::string _passwd;
00224     std::string _host;
00228     std::string _port;
00232     std::string _dbname;
00233   };
00234
00235 }
00236 #endif // __OPENTREP_DBPARAMS_HPP
```

24.219 opentrep/DistanceErrorRule.hpp File Reference

```
#include <istream>
#include <ostream>
#include <sstream>
#include <string>
#include <map>
#include <boost/array.hpp>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/OPENTREP_Abstract.hpp>
```

Classes

- struct OPENTREP::DistanceErrorRule

Namespaces

- OPENTREP

Variables

- static const DistanceErrorScaleArray_T OPENTREP::K_DEFAULT_ERROR_SCALE

24.220 DistanceErrorRule.hpp

```
00001 #ifndef __OPENTREP_DISTANCEERRORRULE_HPP
00002 #define __OPENTREP_DISTANCEERRORRULE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <istream>
00009 #include <ostream>
00010 #include <sstream>
00011 #include <string>
00012 #include <map>
00013 // Boost Array
00014 #include <boost/array.hpp>
00015 // OpenTrep
00016 #include <opentrep/OPENTREP_Types.hpp>
00017 #include <opentrep/OPENTREP_Abstract.hpp>
00018
00019 namespace OPENTREP {
00020
00021   // ////////////////////////////////////////////////////////////////////
00035   static const DistanceErrorScaleArray_T
```

```
      K_DEFAULT_ERROR_SCALE =
00036     { {3, 6, 9, 14, 19} };
00037
00038
00039   // ////////////////////////////////////////////////////////////////
00043   struct DistanceErrorRule : public OPENTREP_Abstract {
00044   public:
00045     // ///////// Getters ////////
00049     const DistanceErrorScale_T& getDistanceErrorScale() const {
00050       return _scale;
00051     }
00052
00053
00054     // ///////// Setters //////////
00055
00056
00057   public:
00058     // ///////// Business methods ////////
00063     NbOfErrors_T getAllowedDistanceError (const
      NbOfLetters_T& iNbOfLetters) const {
00064       NbOfErrors_T oNbOfErrors_T = 0;
00065
00066       DistanceErrorScale_T::const_iterator itError =
00067         _scale.lower_bound (iNbOfLetters);
00068
00069       if (itError != _scale.end()) {
00070         oNbOfErrors_T = itError->second;
00071
00072       } else {
00073         oNbOfErrors_T = iNbOfLetters / 4;
00074       }
00075
00076       return oNbOfErrors_T;
00077     }
00078
00079
00080   public:
00081     // ///////// Display methods ////////
00087     void toStream (std::ostream& ioOut) const {
00088       ioOut << toString();
00089     }
00090
00096     void fromStream (std::istream&) {
00097     }
00098
00102     std::string toShortString() const {
00103       std::ostringstream oStr;
00104       NbOfLetters_T idx = 0;
00105       for (DistanceErrorScale_T::const_iterator itError = _scale.begin();
00106            itError != _scale.end(); ++itError, ++idx) {
00107         if (idx != 0) {
00108           oStr << ", ";
00109         }
00110         oStr << itError->second << ": " << itError->first;
00111       }
00112       return oStr.str();
00113     }
00114
00118     std::string toString() const {
00119       std::ostringstream oStr;
00120       oStr << toShortString();
00121       return oStr.str();
00122     }
00123
00124
00125   public:
00129     DistanceErrorRule (const DistanceErrorScale_T& iScale)
00130       : _scale (iScale) {
00131     }
00132
00136     DistanceErrorRule (const NbOfErrors_T iSize,
00137                        const NbOfLetters_T iScaleArray[]) {
00138       for (NbOfErrors_T idx = 0; idx != iSize; ++idx) {
00139         _scale.insert (DistanceErrorScale_T::value_type (iScaleArray[idx],
00140                                                          idx));
00141       }
00142     }
00143
00147     DistanceErrorRule (const DistanceErrorScaleArray_T&
      iScaleArray) {
00148       for (NbOfErrors_T idx = 0; idx != iScaleArray.size(); ++idx) {
00149         _scale.insert (DistanceErrorScale_T::value_type (iScaleArray[idx],
00150                                                          idx));
00151       }
00152     }
00153
00154   private:
```

```
00158     DistanceErrorRule ();
00159
00163     DistanceErrorRule (const DistanceErrorRule&);
00164
00168     virtual ~DistanceErrorRule() {}
00169
00170
00171   private:
00172     // /////// Attributes /////////
00176     DistanceErrorScale_T _scale;
00177   };
00178
00179 }
00180 #endif // __OPENTREP_DISTANCEERRORRULE_HPP
```

24.221 opentrep/factory/FacBomAbstract.cpp File Reference

```
#include <assert.h>
#include <sstream>
#include <boost/functional/hash/hash.hpp>
#include <opentrep/bom/BomAbstract.hpp>
#include <opentrep/factory/FacBomAbstract.hpp>
```

Namespaces

- OPENTREP

24.222 FacBomAbstract.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // C
00005 #include <assert.h>
00006 // STL
00007 #include <sstream>
00008 // Boost (STL Extension)
00009 #include <boost/functional/hash/hash.hpp>
00010 // OpenTrep
00011 #include <opentrep/bom/BomAbstract.hpp>
00012 #include <opentrep/factory/FacBomAbstract.hpp>
00013
00014 namespace OPENTREP {
00015
00016   // //////////////////////////////////////////////////////////////////
00017   FacBomAbstract::~FacBomAbstract() {
00018     clean ();
00019   }
00020
00021   // //////////////////////////////////////////////////////////////////
00022   void FacBomAbstract::clean() {
00023     for (BomPool_T::iterator itBom = _pool.begin();
00024          itBom != _pool.end(); itBom++) {
00025       BomAbstract* currentBom_ptr = *itBom;
00026       assert (currentBom_ptr != NULL);
00027
00028       delete (currentBom_ptr); currentBom_ptr = NULL;
00029     }
00030
00031     // Empty the pool of Factories
00032     _pool.clear();
00033   }
00034
00035   // //////////////////////////////////////////////////////////////////
00036   std::size_t FacBomAbstract::getID (const BomAbstract* iBomAbstract_ptr) {
00037     const void* lPtr = iBomAbstract_ptr;
00038     boost::hash<const void*> ptr_hash;
00039     const std::size_t lID = ptr_hash (lPtr);
00040     return lID;
00041   }
00042
00043   // //////////////////////////////////////////////////////////////////
00044   std::size_t FacBomAbstract::getID (const BomAbstract& iBomAbstract) {
00045     return getID (&iBomAbstract);
```

```
00046   }
00047
00048   // ////////////////////////////////////////////////////////////////////
00049   std::string FacBomAbstract::getIDString(const
      BomAbstract* iBomAbstract_ptr) {
00050     const std::size_t lID = getID (iBomAbstract_ptr);
00051     std::ostringstream oStr;
00052     oStr << lID;
00053     return oStr.str();
00054   }
00055
00056   // ////////////////////////////////////////////////////////////////////
00057   std::string FacBomAbstract::getIDString (const
      BomAbstract& iBomAbstract) {
00058     return getIDString (&iBomAbstract);
00059   }
00060
00061 }
```

24.223 opentrep/factory/FacBomAbstract.hpp File Reference

```
#include <string>
#include <vector>
```

Classes

- class OPENTREP::FacBomAbstract

Namespaces

- OPENTREP

24.224 FacBomAbstract.hpp

```
00001 #ifndef __OPENTREP_FAC_FACBOMABSTRACT_HPP
00002 #define __OPENTREP_FAC_FACBOMABSTRACT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010
00011 namespace OPENTREP {
00012
00013   // Forward declarations
00014   class BomAbstract;
00015
00017   class FacBomAbstract {
00018     friend class FacSupervisor;
00019   public:
00020
00022     typedef std::vector<BomAbstract*> BomPool_T;
00023
00025     static std::size_t getID (const BomAbstract*);
00026
00028     static std::size_t getID (const BomAbstract&);
00029
00032     static std::string getIDString (const BomAbstract*);
00033
00036     static std::string getIDString (const BomAbstract&);
00037
00038   protected:
00041     FacBomAbstract() {}
00042     FacBomAbstract(const FacBomAbstract&) {}
00043
00045     virtual ~FacBomAbstract();
00046
00047   private:
00049     void clean();
00050
00051   protected:
```

```
00053     BomPool_T _pool;
00054   };
00055 }
00056 #endif // __OPENTREP_FAC_FACBOMABSTRACT_HPP
```

24.225 opentrep/factory/FacOpenTrepServiceContext.cpp File Reference

```
#include <cassert>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/factory/FacSupervisor.hpp>
#include <opentrep/factory/FacOpenTrepServiceContext.hpp>
#include <opentrep/service/OPENTREP_ServiceContext.hpp>
```

Namespaces

- OPENTREP

24.226 FacOpenTrepServiceContext.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // OpenTrep
00007 #include <opentrep/OPENTREP_Types.hpp>
00008 #include <opentrep/factory/FacSupervisor.hpp>
00009 #include <opentrep/factory/FacOpenTrepServiceContext.hpp>
00010 #include <opentrep/service/OPENTREP_ServiceContext.hpp>
00011
00012 namespace OPENTREP {
00013
00014   FacOpenTrepServiceContext* FacOpenTrepServiceContext::_instance = NULL;
00015
00016   // ////////////////////////////////////////////////////////////////////
00017   FacOpenTrepServiceContext::~FacOpenTrepServiceContext
      () {
00018     _instance = NULL;
00019   }
00020
00021   // ////////////////////////////////////////////////////////////////////
00022   FacOpenTrepServiceContext&
      FacOpenTrepServiceContext::instance() {
00023
00024     if (_instance == NULL) {
00025       _instance = new FacOpenTrepServiceContext();
00026       assert (_instance != NULL);
00027
00028       FacSupervisor::instance().registerServiceFactory (
      _instance);
00029     }
00030     return *_instance;
00031   }
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   OPENTREP_ServiceContext& FacOpenTrepServiceContext::
00035   create (const TravelDBFilePath_T& iTravelDBFilePath,
00036           const SQLiteDBFilePath_T& iSQLiteDBFilePath) {
00037     OPENTREP_ServiceContext* aOPENTREP_ServiceContext_ptr = NULL;
00038
00039     aOPENTREP_ServiceContext_ptr =
00040       new OPENTREP_ServiceContext (iTravelDBFilePath, iSQLiteDBFilePath);
00041     assert (aOPENTREP_ServiceContext_ptr != NULL);
00042
00043     // The new object is added to the Bom pool
00044     _pool.push_back (aOPENTREP_ServiceContext_ptr);
00045
00046     return *aOPENTREP_ServiceContext_ptr;
00047   }
00048
00049   // ////////////////////////////////////////////////////////////////////
00050   OPENTREP_ServiceContext& FacOpenTrepServiceContext::
00051   create (const PORFilePath_T& iPORFilePath,
00052           const TravelDBFilePath_T& iTravelDBFilePath,
```

```
00053           const SQLiteDBFilePath_T& iSQLiteDBFilePath) {
00054     OPENTREP_ServiceContext* aOPENTREP_ServiceContext_ptr = NULL;
00055
00056     aOPENTREP_ServiceContext_ptr =
00057       new OPENTREP_ServiceContext (iPORFilePath, iTravelDBFilePath,
      iSQLiteDBFilePath);
00058     assert (aOPENTREP_ServiceContext_ptr != NULL);
00059
00060     // The new object is added to the Bom pool
00061     _pool.push_back (aOPENTREP_ServiceContext_ptr);
00062
00063     return *aOPENTREP_ServiceContext_ptr;
00064   }
00065
00066 }
```

24.227 opentrep/factory/FacOpenTrepServiceContext.hpp File Reference

```
#include <opentrep/factory/FacServiceAbstract.hpp>
```

Classes

- class OPENTREP::FacOpenTrepServiceContext

  Factory for Bucket.

Namespaces

- OPENTREP

24.228 FacOpenTrepServiceContext.hpp

```
00001 #ifndef __OPENTREP_FAC_FACOPENTREPSERVICECONTEXT_HPP
00002 #define __OPENTREP_FAC_FACOPENTREPSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/factory/FacServiceAbstract.hpp>
00009
00010 namespace OPENTREP {
00011
00012   // Forward declarations.
00013   class OPENTREP_ServiceContext;
00014
00015
00019   class FacOpenTrepServiceContext : public
      FacServiceAbstract {
00020   public:
00021
00029     static FacOpenTrepServiceContext& instance();
00030
00037     ~FacOpenTrepServiceContext();
00038
00048     OPENTREP_ServiceContext& create (const
      TravelDBFilePath_T&, const SQLiteDBFilePath_T&);
00049
00060     OPENTREP_ServiceContext& create (const
      PORFilePath_T&,
00061                                      const TravelDBFilePath_T&, const
      SQLiteDBFilePath_T&);
00062
00063
00064   protected:
00071     FacOpenTrepServiceContext() {}
00072
00073   private:
00077     static FacOpenTrepServiceContext* _instance;
00078   };
00079
00080 }
00081 #endif // __OPENTREP_FAC_FACOPENTREPSERVICECONTEXT_HPP
```

24.229 opentrep/factory/FacPlace.cpp File Reference

```
#include <cassert>
#include <opentrep/bom/Place.hpp>
#include <opentrep/factory/FacSupervisor.hpp>
#include <opentrep/factory/FacPlace.hpp>
```

Namespaces

- OPENTREP

24.230 FacPlace.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // OpenTrep
00007 #include <opentrep/bom/Place.hpp>
00008 #include <opentrep/factory/FacSupervisor.hpp>
00009 #include <opentrep/factory/FacPlace.hpp>
00010
00011 namespace OPENTREP {
00012
00013   FacPlace* FacPlace::_instance = NULL;
00014
00015   // //////////////////////////////////////////////////////////////////////
00016   FacPlace::FacPlace() {
00017   }
00018
00019   // //////////////////////////////////////////////////////////////////////
00020   FacPlace::FacPlace (const FacPlace&) {
00021   }
00022
00023   // //////////////////////////////////////////////////////////////////////
00024   FacPlace::~FacPlace() {
00025     _instance = NULL;
00026   }
00027
00028   // //////////////////////////////////////////////////////////////////////
00029   FacPlace& FacPlace::instance() {
00030
00031     if (_instance == NULL) {
00032       _instance = new FacPlace();
00033       assert (_instance != NULL);
00034
00035       FacSupervisor::instance().registerBomFactory (_instance);
00036     }
00037     return *_instance;
00038   }
00039
00040   // //////////////////////////////////////////////////////////////////////
00041   Place& FacPlace::create() {
00042     Place* oPlace_ptr = NULL;
00043
00044     oPlace_ptr = new Place();
00045     assert (oPlace_ptr != NULL);
00046
00047     // The new object is added to the Bom pool
00048     _pool.push_back (oPlace_ptr);
00049
00050     return *oPlace_ptr;
00051   }
00052
00053   // //////////////////////////////////////////////////////////////////////
00054   Place& FacPlace::create (const LocationKey& iLocationKey) {
00055     Place* oPlace_ptr = NULL;
00056
00057     oPlace_ptr = new Place (iLocationKey);
00058     assert (oPlace_ptr != NULL);
00059
00060     // The new object is added to the Bom pool
00061     _pool.push_back (oPlace_ptr);
00062
00063     return *oPlace_ptr;
00064   }
```

```
00065
00066   // ////////////////////////////////////////////////////////////////////
00067   Place& FacPlace::create (const Location& iLocation) {
00068     Place* oPlace_ptr = NULL;
00069
00070     oPlace_ptr = new Place (iLocation);
00071     assert (oPlace_ptr != NULL);
00072
00073     // The new object is added to the Bom pool
00074     _pool.push_back (oPlace_ptr);
00075
00076     return *oPlace_ptr;
00077   }
00078
00079   // ////////////////////////////////////////////////////////////////////
00080   Place& FacPlace::clone (const Place& iPlace) {
00081     Place* oPlace_ptr = NULL;
00082
00083     oPlace_ptr = new Place (iPlace);
00084     assert (oPlace_ptr != NULL);
00085
00086     // The new object is added to the Bom pool
00087     _pool.push_back (oPlace_ptr);
00088
00089     return *oPlace_ptr;
00090   }
00091
00092   // ////////////////////////////////////////////////////////////////////
00093   void FacPlace::initLinkWithExtraPlace (Place& ioMainPlace,
00094                                          Place& ioExtraPlace) {
00095       // Link the main Place to the extra Place, and vice versa
00096       ioExtraPlace._mainPlace = &ioMainPlace;
00097
00098       // Add the extra Place to the main Place internal map (of extra
00099       // Place objects)
00100       ioMainPlace._extraPlaceList.push_back (&ioExtraPlace);
00101     }
00102
00103   // ////////////////////////////////////////////////////////////////////
00104   void FacPlace::initLinkWithAlternatePlace (
      Place& ioMainPlace,
00105                                              Place& ioAlternatePlace) {
00106       // Link the main Place to the alternate Place, and vice versa
00107       ioAlternatePlace._mainPlace = &ioMainPlace;
00108
00109       // Add the alternate Place to the main Place internal map (of
00110       // alternate Place objects)
00111       ioMainPlace._alternatePlaceList.push_back (&ioAlternatePlace);
00112     }
00113
00114 }
```

24.231 opentrep/factory/FacPlace.hpp File Reference

```
#include <opentrep/factory/FacBomAbstract.hpp>
```

Classes

- class OPENTREP::FacPlace

Namespaces

- OPENTREP

24.232 FacPlace.hpp

```
00001 #ifndef __OPENTREP_FAC_FACPLACE_HPP
00002 #define __OPENTREP_FAC_FACPLACE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // OpenTrep
```

```
00008 #include <opentrep/factory/FacBomAbstract.hpp>
00009
00010 namespace OPENTREP {
00011
00012   // Forward declarations.
00013   class Place;
00014   struct LocationKey;
00015   struct Location;
00016
00020   class FacPlace : public FacBomAbstract {
00021   public:
00022
00030     static FacPlace& instance();
00031
00038     virtual ~FacPlace();
00039
00047     Place& create();
00048
00058     Place& create (const LocationKey&);
00059
00069     Place& create (const Location&);
00070
00077     Place& clone (const Place&);
00078
00087     static void initLinkWithExtraPlace (Place&, Place&);
00088
00096     static void initLinkWithAlternatePlace (Place&,
      Place&);
00097
00098   private:
00105     FacPlace ();
00106     FacPlace (const FacPlace&);
00107
00108   private:
00112     static FacPlace* _instance;
00113   };
00114 }
00115 #endif // __OPENTREP_FAC_FACPLACE_HPP
```

24.233 opentrep/factory/FacPlaceHolder.cpp File Reference

```
#include <cassert>
#include <opentrep/bom/PlaceHolder.hpp>
#include <opentrep/bom/Place.hpp>
#include <opentrep/factory/FacSupervisor.hpp>
#include <opentrep/factory/FacPlaceHolder.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.234 FacPlaceHolder.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // C
00005 #include <cassert>
00006 // OPENTREP
00007 #include <opentrep/bom/PlaceHolder.hpp>
00008 #include <opentrep/bom/Place.hpp>
00009 #include <opentrep/factory/FacSupervisor.hpp>
00010 #include <opentrep/factory/FacPlaceHolder.hpp>
00011 #include <opentrep/service/Logger.hpp>
00012
00013 namespace OPENTREP {
00014
00015   FacPlaceHolder* FacPlaceHolder::_instance = NULL;
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   FacPlaceHolder::FacPlaceHolder () {
00019   }
00020
```

```
00021   // ////////////////////////////////////////////////////////////////////
00022   FacPlaceHolder::FacPlaceHolder (const FacPlaceHolder&) {
00023   }
00024
00025   // ////////////////////////////////////////////////////////////////////
00026   FacPlaceHolder::~FacPlaceHolder () {
00027     _instance = NULL;
00028   }
00029
00030   // ////////////////////////////////////////////////////////////////////
00031   FacPlaceHolder& FacPlaceHolder::instance () {
00032
00033     if (_instance == NULL) {
00034       _instance = new FacPlaceHolder();
00035       assert (_instance != NULL);
00036
00037       FacSupervisor::instance().registerBomFactory (_instance);
00038     }
00039     return *_instance;
00040   }
00041
00042   // ////////////////////////////////////////////////////////////////////
00043   PlaceHolder& FacPlaceHolder::create () {
00044     PlaceHolder* oPlaceHolder_ptr = NULL;
00045
00046     oPlaceHolder_ptr = new PlaceHolder ();
00047     assert (oPlaceHolder_ptr != NULL);
00048
00049     // The new object is added to the Bom pool
00050     _pool.push_back (oPlaceHolder_ptr);
00051
00052     return *oPlaceHolder_ptr;
00053   }
00054
00055   // ////////////////////////////////////////////////////////////////////
00056   void FacPlaceHolder::initLinkWithPlace (
      PlaceHolder& ioPlaceHolder,
00057                                           Place& ioPlace) {
00058       // Link the PlaceHolder to the Place, and vice versa
00059       ioPlace._placeHolder = &ioPlaceHolder;
00060
00061       // Add the Place to the PlaceHolder internal map (of Place objects)
00062       // const bool insertSucceeded =
00063       ioPlaceHolder._placeList.
00064         insert (PlaceList_T::value_type (ioPlace.describeShortKey(), &ioPlace));
00065
00066 //       if (insertSucceeded == false) {
00067 //         OPENTREP_LOG_ERROR ("Insertion failed for "
00068 //                             << ioPlaceHolder.describeKey()
00069 //                             << " and " << ioPlace.describeShortKey());
00070 //         assert (insertSucceeded == true);
00071 //       }
00072
00073       // Add the Place to the PlaceHolder internal list (of Place objects)
00074       ioPlaceHolder._placeOrderedList.push_back (&ioPlace);
00075     }
00076
00077 }
```

24.235 opentrep/factory/FacPlaceHolder.hpp File Reference

```
#include <opentrep/factory/FacBomAbstract.hpp>
```

Classes

- class OPENTREP::FacPlaceHolder

Namespaces

- OPENTREP

24.236 FacPlaceHolder.hpp

```
00001 #ifndef __OPENTREP_FAC_FACPLACEHOLDER_HPP
```

```
00002 #define __OPENTREP_FAC_FACPLACEHOLDER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // OPENTREP
00008 #include <opentrep/factory/FacBomAbstract.hpp>
00009
00010 namespace OPENTREP {
00011
00013   class PlaceHolder;
00014   class Place;
00015
00017   class FacPlaceHolder : public FacBomAbstract {
00018   public:
00019
00023     static FacPlaceHolder& instance();
00024
00028     virtual ~FacPlaceHolder();
00029
00033     PlaceHolder& create ();
00034
00040     static void initLinkWithPlace (PlaceHolder&,
      Place&);
00041
00042
00043   private:
00047     FacPlaceHolder ();
00048     FacPlaceHolder (const FacPlaceHolder&);
00049
00050   private:
00052     static FacPlaceHolder* _instance;
00053
00054   };
00055 }
00056 #endif // __OPENTREP_FAC_FACPLACEHOLDER_HPP
```

24.237 opentrep/factory/FacResult.cpp File Reference

```
#include <cassert>
#include <opentrep/bom/Result.hpp>
#include <opentrep/factory/FacSupervisor.hpp>
#include <opentrep/factory/FacResult.hpp>
```

Namespaces

- OPENTREP

24.238 FacResult.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // C
00005 #include <cassert>
00006 // OpenTrep
00007 #include <opentrep/bom/Result.hpp>
00008 #include <opentrep/factory/FacSupervisor.hpp>
00009 #include <opentrep/factory/FacResult.hpp>
00010
00011 namespace OPENTREP {
00012
00013   FacResult* FacResult::_instance = NULL;
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   FacResult::FacResult () {
00017   }
00018
00019   // ////////////////////////////////////////////////////////////////////
00020   FacResult::FacResult (const FacResult&) {
00021   }
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   FacResult::~FacResult () {
00025     _instance = NULL;
```

```
00026   }
00027
00028   // ////////////////////////////////////////////////////////////////////
00029   FacResult& FacResult::instance () {
00030
00031     if (_instance == NULL) {
00032       _instance = new FacResult();
00033       assert (_instance != NULL);
00034
00035       FacSupervisor::instance().registerBomFactory (_instance);
00036     }
00037     return *_instance;
00038   }
00039
00040   // ////////////////////////////////////////////////////////////////////
00041   Result& FacResult::create (const TravelQuery_T& iQueryString,
00042                              const Xapian::Database& iXapianDatabase) {
00043     Result* oResult_ptr = NULL;
00044
00045     oResult_ptr = new Result (iQueryString, iXapianDatabase);
00046     assert (oResult_ptr != NULL);
00047
00048     // The new object is added to the Bom pool
00049     _pool.push_back (oResult_ptr);
00050
00051     return *oResult_ptr;
00052   }
00053
00054 }
```

24.239 opentrep/factory/FacResult.hpp File Reference

```
#include <opentrep/factory/FacBomAbstract.hpp>
```

Classes

- class OPENTREP::FacResult

Namespaces

- OPENTREP

24.240 FacResult.hpp

```
00001 #ifndef __OPENTREP_FAC_FACRESULT_HPP
00002 #define __OPENTREP_FAC_FACRESULT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/factory/FacBomAbstract.hpp>
00009
00010 namespace OPENTREP {
00011
00012   // Forward declarations.
00013   class Result;
00014
00018   class FacResult : public FacBomAbstract {
00019   public:
00020
00028     static FacResult& instance();
00029
00036     virtual ~FacResult();
00037
00045     Result& create (const TravelQuery_T&, const Xapian::Database&);
00046
00047
00048   private:
00055     FacResult ();
00056     FacResult (const FacResult&);
00057
00058   private:
```

```
00062     static FacResult* _instance;
00063   };
00064 }
00065 #endif // __OPENTREP_FAC_FACRESULT_HPP
```

24.241 opentrep/factory/FacResultCombination.cpp File Reference

```
#include <cassert>
#include <opentrep/bom/ResultCombination.hpp>
#include <opentrep/bom/ResultHolder.hpp>
#include <opentrep/factory/FacSupervisor.hpp>
#include <opentrep/factory/FacResultCombination.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.242 FacResultCombination.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // C
00005 #include <cassert>
00006 // OpenTrep
00007 #include <opentrep/bom/ResultCombination.hpp>
00008 #include <opentrep/bom/ResultHolder.hpp>
00009 #include <opentrep/factory/FacSupervisor.hpp>
00010 #include <opentrep/factory/FacResultCombination.hpp>
00011 #include <opentrep/service/Logger.hpp>
00012
00013 namespace OPENTREP {
00014
00015   FacResultCombination* FacResultCombination::_instance = NULL;
00016
00017   // //////////////////////////////////////////////////////////////////
00018   FacResultCombination::FacResultCombination() {
00019   }
00020
00021   // //////////////////////////////////////////////////////////////////
00022   FacResultCombination::FacResultCombination (const FacResultCombination&) {
00023   }
00024
00025   // //////////////////////////////////////////////////////////////////
00026   FacResultCombination::~FacResultCombination() {
00027     _instance = NULL;
00028   }
00029
00030   // //////////////////////////////////////////////////////////////////
00031   FacResultCombination& FacResultCombination::instance()
    {
00032
00033     if (_instance == NULL) {
00034       _instance = new FacResultCombination();
00035       assert (_instance != NULL);
00036
00037       FacSupervisor::instance().registerBomFactory (_instance);
00038     }
00039     return *_instance;
00040   }
00041
00042   // //////////////////////////////////////////////////////////////////
00043   ResultCombination& FacResultCombination::
00044   create (const TravelQuery_T& iQueryString) {
00045     ResultCombination* oResultCombination_ptr = NULL;
00046
00047     oResultCombination_ptr = new ResultCombination (iQueryString);
00048     assert (oResultCombination_ptr != NULL);
00049
00050     // The new object is added to the Bom pool
00051     _pool.push_back (oResultCombination_ptr);
00052
00053     return *oResultCombination_ptr;
```

```
00054   }
00055
00056   // ////////////////////////////////////////////////////////////////////
00057   void FacResultCombination::
00058   initLinkWithResultHolder (ResultCombination&
      ioResultCombination,
00059                             ResultHolder& ioResultHolder) {
00060     // Link the ResultCombination to the ResultHolder, and vice versa
00061     ioResultHolder._resultCombination = &ioResultCombination;
00062
00063     // Add the Result to the ResultCombination internal list (of
00064     // ResultHolder objects)
00065     ioResultCombination._resultHolderList.push_back (&ioResultHolder);
00066   }
00067
00068 }
```

24.243 opentrep/factory/FacResultCombination.hpp File Reference

```
#include <opentrep/factory/FacBomAbstract.hpp>
#include <opentrep/OPENTREP_Types.hpp>
```

Classes

- class OPENTREP::FacResultCombination

Namespaces

- Xapian
- OPENTREP

24.244 FacResultCombination.hpp

```
00001 #ifndef __OPENTREP_FAC_FACRESULTCOMBINATION_HPP
00002 #define __OPENTREP_FAC_FACRESULTCOMBINATION_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // OpenTREP
00008 #include <opentrep/factory/FacBomAbstract.hpp>
00009 #include <opentrep/OPENTREP_Types.hpp>
00010
00011 // Forward declarations
00012 namespace Xapian {
00013   class Database;
00014 }
00015
00016 namespace OPENTREP {
00017
00018   // Forward declarations.
00019   class ResultCombination;
00020   class Result;
00021
00025   class FacResultCombination : public FacBomAbstract {
00026   public:
00027
00035     static FacResultCombination& instance();
00036
00043     virtual ~FacResultCombination();
00044
00052     ResultCombination& create (const TravelQuery_T& iQueryString);
00053
00062     static void initLinkWithResultHolder (
      ResultCombination&, ResultHolder&);
00063
00064
00065   private:
00072     FacResultCombination ();
00073     FacResultCombination (const FacResultCombination&);
00074
00075   private:
```

```
00079     static FacResultCombination* _instance;
00080   };
00081 }
00082 #endif // __OPENTREP_FAC_FACRESULTCOMBINATION_HPP
```

24.245 opentrep/factory/FacResultHolder.cpp File Reference

```
#include <cassert>
#include <opentrep/bom/ResultHolder.hpp>
#include <opentrep/bom/Result.hpp>
#include <opentrep/factory/FacSupervisor.hpp>
#include <opentrep/factory/FacResultHolder.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.246 FacResultHolder.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // C
00005 #include <cassert>
00006 // OPENTREP
00007 #include <opentrep/bom/ResultHolder.hpp>
00008 #include <opentrep/bom/Result.hpp>
00009 #include <opentrep/factory/FacSupervisor.hpp>
00010 #include <opentrep/factory/FacResultHolder.hpp>
00011 #include <opentrep/service/Logger.hpp>
00012
00013 namespace OPENTREP {
00014
00015   FacResultHolder* FacResultHolder::_instance = NULL;
00016
00017   // //////////////////////////////////////////////////////////////////
00018   FacResultHolder::FacResultHolder () {
00019   }
00020
00021   // //////////////////////////////////////////////////////////////////
00022   FacResultHolder::FacResultHolder (const FacResultHolder&) {
00023   }
00024
00025   // //////////////////////////////////////////////////////////////////
00026   FacResultHolder::~FacResultHolder () {
00027     _instance = NULL;
00028   }
00029
00030   // //////////////////////////////////////////////////////////////////
00031   FacResultHolder& FacResultHolder::instance () {
00032
00033     if (_instance == NULL) {
00034       _instance = new FacResultHolder();
00035       assert (_instance != NULL);
00036
00037       FacSupervisor::instance().registerBomFactory (_instance);
00038     }
00039     return *_instance;
00040   }
00041
00042   // //////////////////////////////////////////////////////////////////
00043   ResultHolder& FacResultHolder::create (const
      TravelQuery_T& iQueryString,
00044                                          const Xapian::Database& iDatabase) {
00045     ResultHolder* oResultHolder_ptr = NULL;
00046
00047     oResultHolder_ptr = new ResultHolder (iQueryString, iDatabase);
00048     assert (oResultHolder_ptr != NULL);
00049
00050     // The new object is added to the Bom pool
00051     _pool.push_back (oResultHolder_ptr);
00052
00053     return *oResultHolder_ptr;
```

```
00054   }
00055
00056   // ////////////////////////////////////////////////////////////////////
00057   void FacResultHolder::initLinkWithResult (
      ResultHolder& ioResultHolder,
00058                                             Result& ioResult) {
00059       // Link the ResultHolder to the Result, and vice versa
00060       ioResult._resultHolder = &ioResultHolder;
00061
00062       // Add the Result to the ResultHolder internal list (of Result objects)
00063       ioResultHolder._resultList.push_back (&ioResult);
00064     }
00065
00066 }
```

24.247 opentrep/factory/FacResultHolder.hpp File Reference

```
#include <opentrep/factory/FacBomAbstract.hpp>
#include <opentrep/OPENTREP_Types.hpp>
```

Classes

- class OPENTREP::FacResultHolder

Namespaces

- Xapian
- OPENTREP

24.248 FacResultHolder.hpp

```
00001 #ifndef __OPENTREP_FAC_FACRESULTHOLDER_HPP
00002 #define __OPENTREP_FAC_FACRESULTHOLDER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // OpenTREP
00008 #include <opentrep/factory/FacBomAbstract.hpp>
00009 #include <opentrep/OPENTREP_Types.hpp>
00010
00011 // Forward declarations
00012 namespace Xapian {
00013   class Database;
00014 }
00015
00016 namespace OPENTREP {
00017
00019   class ResultHolder;
00020   class Result;
00021
00023   class FacResultHolder : public FacBomAbstract {
00024   public:
00025
00029     static FacResultHolder& instance();
00030
00034     virtual ~FacResultHolder();
00035
00039     ResultHolder& create (const TravelQuery_T& iQueryString,
00040                           const Xapian::Database& iDatabase);
00041
00047     static void initLinkWithResult (ResultHolder&,
      Result&);
00048
00049
00050   private:
00054     FacResultHolder ();
00055     FacResultHolder (const FacResultHolder&);
00056
00057   private:
00059     static FacResultHolder* _instance;
00060
```

```
00061   };
00062 }
00063 #endif // __OPENTREP_FAC_FACRESULTHOLDER_HPP
```

24.249 opentrep/factory/FacServiceAbstract.cpp File Reference

```
#include <assert.h>
#include <opentrep/service/ServiceAbstract.hpp>
#include <opentrep/factory/FacServiceAbstract.hpp>
```

Namespaces

- OPENTREP

24.250 FacServiceAbstract.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // C
00005 #include <assert.h>
00006 // OPENTREP
00007 #include <opentrep/service/ServiceAbstract.hpp>
00008 #include <opentrep/factory/FacServiceAbstract.hpp>
00009
00010 namespace OPENTREP {
00011
00012   // //////////////////////////////////////////////////////////////////
00013   FacServiceAbstract::~FacServiceAbstract() {
00014     clean ();
00015   }
00016
00017   // //////////////////////////////////////////////////////////////////
00018   void FacServiceAbstract::clean() {
00019     for (ServicePool_T::iterator itService = _pool.begin();
00020          itService != _pool.end(); itService++) {
00021       ServiceAbstract* currentService_ptr = *itService;
00022       assert (currentService_ptr != NULL);
00023
00024       delete (currentService_ptr); currentService_ptr = NULL;
00025     }
00026
00027     // Empty the pool of Service Factories
00028     _pool.clear();
00029   }
00030
00031 }
```

24.251 opentrep/factory/FacServiceAbstract.hpp File Reference

```
#include <vector>
```

Classes

- class OPENTREP::FacServiceAbstract

Namespaces

- OPENTREP

24.252 FacServiceAbstract.hpp

```
00001 #ifndef __OPENTREP_FAC_FACSERVICEABSTRACT_HPP
00002 #define __OPENTREP_FAC_FACSERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <vector>
00009
00010 namespace OPENTREP {
00011
00012   // Forward declarations
00013   class ServiceAbstract;
00014
00016   class FacServiceAbstract {
00017   public:
00018
00020     typedef std::vector<ServiceAbstract*> ServicePool_T;
00021
00023     virtual ~FacServiceAbstract();
00024
00026     void clean();
00027
00028   protected:
00031     FacServiceAbstract() {}
00032
00034     ServicePool_T _pool;
00035   };
00036
00037 }
00038 #endif // __OPENTREP_FAC_FACSERVICEABSTRACT_HPP
```

24.253 opentrep/factory/FacSupervisor.cpp File Reference

```
#include <assert.h>
#include <opentrep/factory/FacBomAbstract.hpp>
#include <opentrep/factory/FacServiceAbstract.hpp>
#include <opentrep/factory/FacSupervisor.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.254 FacSupervisor.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // C
00005 #include <assert.h>
00006 // OPENTREP
00007 #include <opentrep/factory/FacBomAbstract.hpp>
00008 #include <opentrep/factory/FacServiceAbstract.hpp>
00009 #include <opentrep/factory/FacSupervisor.hpp>
00010 #include <opentrep/service/Logger.hpp>
00011
00012 namespace OPENTREP {
00013
00014   FacSupervisor* FacSupervisor::_instance = NULL;
00015
00016   // //////////////////////////////////////////////////////////////////
00017   FacSupervisor::FacSupervisor () :
00018     _logger (NULL) {
00019   }
00020
00021   // //////////////////////////////////////////////////////////////////
00022   FacSupervisor& FacSupervisor::instance() {
00023     if (_instance == NULL) {
00024       _instance = new FacSupervisor();
00025     }
00026
```

```
00027     return *_instance;
00028   }
00029
00030   // ////////////////////////////////////////////////////////////////////
00031   void FacSupervisor::
00032   registerBomFactory (FacBomAbstract* ioFacBomAbstract_ptr) {
00033     _bomPool.push_back (ioFacBomAbstract_ptr);
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   void FacSupervisor::
00038   registerServiceFactory (FacServiceAbstract*
      ioFacServiceAbstract_ptr) {
00039     _svcPool.push_back (ioFacServiceAbstract_ptr);
00040   }
00041
00042   // ////////////////////////////////////////////////////////////////////
00043   void FacSupervisor::registerLoggerService (
      Logger* ioLogger_ptr) {
00044     _logger = ioLogger_ptr;
00045   }
00046
00047   // ////////////////////////////////////////////////////////////////////
00048   FacSupervisor::~FacSupervisor() {
00049     cleanBomLayer();
00050     cleanServiceLayer();
00051     cleanLoggerService();
00052   }
00053
00054   // ////////////////////////////////////////////////////////////////////
00055   void FacSupervisor::cleanBomLayer() {
00056     for (BomFactoryPool_T::const_iterator itFactory = _bomPool.begin();
00057          itFactory != _bomPool.end(); itFactory++) {
00058       const FacBomAbstract* currentFactory_ptr = *itFactory;
00059       assert (currentFactory_ptr != NULL);
00060
00061       delete (currentFactory_ptr); currentFactory_ptr = NULL;
00062     }
00063
00064     // Empty the pool of Bom Factories
00065     _bomPool.clear();
00066   }
00067
00068   // ////////////////////////////////////////////////////////////////////
00069   void FacSupervisor::cleanServiceLayer() {
00070     for (ServiceFactoryPool_T::const_iterator itFactory = _svcPool.begin();
00071          itFactory != _svcPool.end(); itFactory++) {
00072       const FacServiceAbstract* currentFactory_ptr = *itFactory;
00073       assert (currentFactory_ptr != NULL);
00074
00075       delete (currentFactory_ptr); currentFactory_ptr = NULL;
00076     }
00077
00078     // Empty the pool of Service Factories
00079     _svcPool.clear();
00080   }
00081
00082   // ////////////////////////////////////////////////////////////////////
00083   void FacSupervisor::cleanLoggerService() {
00084     delete _logger; _logger = NULL;
00085   }
00086
00087   // ////////////////////////////////////////////////////////////////////
00088   void FacSupervisor::cleanFactory () {
00089       if (_instance != NULL) {
00090               _instance->cleanBomLayer();
00091               _instance->cleanServiceLayer();
00092               _instance->cleanLoggerService();
00093       }
00094     delete (_instance); _instance = NULL;
00095   }
00096
00097 }
```

24.255 opentrep/factory/FacSupervisor.hpp File Reference

```
#include <vector>
```

Classes

- class OPENTREP::FacSupervisor

Namespaces

- OPENTREP

24.256 FacSupervisor.hpp

```
00001 #ifndef __OPENTREP_FAC_FACSUPERVISOR_HPP
00002 #define __OPENTREP_FAC_FACSUPERVISOR_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <vector>
00009
00010 namespace OPENTREP {
00011
00012   // Forward declarations
00013   class FacBomAbstract;
00014   class FacServiceAbstract;
00015   class Logger;
00016
00018   class FacSupervisor {
00019   public:
00020
00022     typedef std::vector<FacBomAbstract*> BomFactoryPool_T;
00023       typedef std::vector<FacServiceAbstract*> ServiceFactoryPool_T;
00024
00028     static FacSupervisor& instance();
00029
00034     void registerBomFactory (FacBomAbstract*);
00035
00040     void registerServiceFactory (FacServiceAbstract*);
00041
00050     void registerLoggerService (Logger*);
00051
00055     void cleanBomLayer();
00056
00060     void cleanServiceLayer();
00061
00063     void cleanLoggerService();
00064
00067     static void cleanFactory ();
00068
00072     ~FacSupervisor();
00073
00074
00075   protected:
00079     FacSupervisor ();
00080     FacSupervisor (const FacSupervisor&) {}
00081
00082
00083   private:
00085     static FacSupervisor* _instance;
00086
00088     Logger* _logger;
00089
00091     BomFactoryPool_T _bomPool;
00092
00094     ServiceFactoryPool_T _svcPool;
00095   };
00096 }
00097 #endif // __OPENTREP_FAC_FACSUPERVISOR_HPP
```

24.257 opentrep/factory/FacWorld.cpp File Reference

```
#include <assert.h>
```

```
#include <opentrep/bom/GenericBom.hpp>
#include <opentrep/bom/World.hpp>
#include <opentrep/bom/Place.hpp>
#include <opentrep/bom/PlaceList.hpp>
#include <opentrep/factory/FacSupervisor.hpp>
#include <opentrep/factory/FacBomAbstract.hpp>
#include <opentrep/factory/FacWorld.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.258 FacWorld.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // C
00005 #include <assert.h>
00006 // OPENTREP
00007 #include <opentrep/bom/GenericBom.hpp>
00008 #include <opentrep/bom/World.hpp>
00009 #include <opentrep/bom/Place.hpp>
00010 #include <opentrep/bom/PlaceList.hpp>
00011 #include <opentrep/factory/FacSupervisor.hpp>
00012 #include <opentrep/factory/FacBomAbstract.hpp>
00013 #include <opentrep/factory/FacWorld.hpp>
00014 #include <opentrep/service/Logger.hpp>
00015
00016 namespace OPENTREP {
00017
00018   FacWorld* FacWorld::_instance = NULL;
00019
00020   // //////////////////////////////////////////////////////////////////
00021   FacWorld::~FacWorld () {
00022     _instance = NULL;
00023   }
00024
00025   // //////////////////////////////////////////////////////////////////
00026   FacWorld& FacWorld::instance () {
00027
00028     if (_instance == NULL) {
00029       _instance = new FacWorld();
00030       assert (_instance != NULL);
00031
00032       FacSupervisor::instance().registerBomFactory (_instance);
00033     }
00034     return *_instance;
00035   }
00036
00037   // //////////////////////////////////////////////////////////////////
00038   World& FacWorld::create () {
00039     World* oWorld_ptr = NULL;
00040
00041     oWorld_ptr = new World ();
00042     assert (oWorld_ptr != NULL);
00043
00044     // The new object is added to the Bom pool
00045     _pool.push_back (oWorld_ptr);
00046
00047     return *oWorld_ptr;
00048   }
00049
00050   // //////////////////////////////////////////////////////////////////
00051   void FacWorld::linkPlaceToWorld (const bool iNeedsGeneric,
00052                                    World& ioWorld, Place& ioPlace) {
00053     // Link the World to the Place, and vice versa
00054     ioPlace._world = &ioWorld;
00055
00056     // Retrieve the Xapian document ID of the Place object
00057     const XapianDocID_T& lDocID = ioPlace.getDocID();
00058
00059     // Build a generic BOM object
00060     const GenericBom_T lGenericBom (BomType::PLACE, &ioPlace);
00061
```

```
00062     if (iNeedsGeneric == true) {
00063       // Add the Place pointer to the dedicated list within the World object
00064       const bool insertSucceeded = ioWorld._genericBomList.
00065         insert (GenericBomList_T::value_type (lDocID, lGenericBom)).second;
00066       if (insertSucceeded == false) {
00067         OPENTREP_LOG_ERROR ("Insertion failed for " << ioWorld.
    describeKey()
00068                             << " and " << ioPlace.describeShortKey());
00069         assert (insertSucceeded == true);
00070       }
00071
00072       // Add the Place pointer to the dedicated list within the World object
00073       /*
00074         const bool insertSucceeded2 = ioWorld._placeDirectList.
00075         insert (PlaceDirectList_T::value_type (lPlaceID, &ioPlace)).second;
00076         if (insertSucceeded2 == false) {
00077         OPENTREP_LOG_ERROR ("Insertion failed for " << ioWorld.describeKey()
00078         << " and " << ioPlace.describeShortKey());
00079         assert (insertSucceeded2 == true);
00080         }
00081       */
00082
00083     }
00084
00085     // Add the Place pointer to the dedicated list within the World object
00086     ioWorld._placeOrderedList.push_back (&ioPlace);
00087   }
00088
00089   // ////////////////////////////////////////////////////////////////////
00090   World& FacWorld::clone (const World& iWorld) {
00091     World* oWorld_ptr = NULL;
00092
00093     oWorld_ptr = new World (iWorld);
00094     assert (oWorld_ptr != NULL);
00095
00096     // The new object is added to the Bom pool
00097     _pool.push_back (oWorld_ptr);
00098
00099     return *oWorld_ptr;
00100   }
00101
00102 }
```

24.259 opentrep/factory/FacWorld.hpp File Reference

```
#include <opentrep/factory/FacBomAbstract.hpp>
```

Classes

- class OPENTREP::FacWorld

Namespaces

- OPENTREP

24.260 FacWorld.hpp

```
00001 #ifndef __OPENTREP_FAC_FACWORLD_HPP
00002 #define __OPENTREP_FAC_FACWORLD_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // OpenTrep
00008 #include <opentrep/factory/FacBomAbstract.hpp>
00009
00010 namespace OPENTREP {
00011
00013   class World;
00014   class Place;
00015
00017   class FacWorld : public FacBomAbstract {
00018   public:
```

```
00019
00023     static FacWorld& instance();
00024
00028     virtual ~FacWorld();
00029
00033     World& create ();
00034
00037     static void linkPlaceToWorld (const bool iNeedsGeneric,
      World&, Place&);
00038
00042     World& clone (const World&);
00043
00044   private:
00048     FacWorld () {}
00049     FacWorld (const FacWorld&) {}
00050
00051   private:
00053     static FacWorld* _instance;
00054
00055   };
00056 }
00057 #endif // __OPENTREP_FAC_FACWORLD_HPP
```

24.261 opentrep/IATAType.hpp File Reference

```
#include <string>
#include <opentrep/OPENTREP_Types.hpp>
```

Classes

- struct OPENTREP::IATAType

  Enumeration of output types.

Namespaces

- treppb
- OPENTREP

24.262 IATAType.hpp

```
00001 #ifndef __OPENTREP_IATATYPE_HPP
00002 #define __OPENTREP_IATATYPE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // OpenTrep
00010 #include <opentrep/OPENTREP_Types.hpp>
00011
00012 // Forward declarations
00013 namespace treppb {
00014   class PlaceType;
00015 }
00016
00017 namespace OPENTREP {
00018
00022   struct IATAType {
00023   public:
00024     typedef enum {
00025       CTY_AIRP = 0,
00026       CTY_HPT,
00027       CTY_RSTN,
00028       CTY_BSTN,
00029       CTY_FERRY,
00030       CITY,
00031       AIRP,
00032       HPT,
00033       RSTN,
00034       BSTN,
```

```
00035       FERRY,
00036       OFF,
00037       LAST_VALUE
00038     } EN_IATAType;
00039
00043     static const std::string& getLabel (const EN_IATAType&);
00044
00048     static EN_IATAType getType (const char);
00049
00053     static char getTypeLabel (const EN_IATAType&);
00054
00058     static std::string getTypeLabelAsString (const
      EN_IATAType&);
00059
00063     static treppb::PlaceType getTypeLabelAsPB (const EN_IATAType&);
00064
00068     static std::string describeLabels();
00069
00073     EN_IATAType getType() const;
00074
00078     char getTypeAsChar() const;
00079
00083     std::string getTypeAsString() const;
00084
00088     treppb::PlaceType getTypeAsPB() const;
00089
00093     const std::string describe() const;
00094
00095   public:
00099     bool operator== (const EN_IATAType&) const;
00100     bool operator== (const IATAType&) const;
00101
00102   public:
00106     IATAType (const EN_IATAType&);
00110     IATAType (const char iType);
00114     IATAType (const std::string& iType);
00118     IATAType (const IATAType&);
00119
00120   private:
00124     IATAType();
00125
00126
00127   private:
00131     static const std::string _labels[LAST_VALUE];
00135     static const char _typeLabels[LAST_VALUE];
00136
00137   private:
00138     // ///////// Attributes /////////
00142     EN_IATAType _type;
00143   };
00144
00145 }
00146 #endif // __OPENTREP_IATATYPE_HPP
```

24.263 opentrep/Language.hpp File Reference

```
#include <string>
```

Classes

- struct OPENTREP::Language

Namespaces

- OPENTREP

24.264 Language.hpp

```
00001 #ifndef __OPENTREP_LANGUAGE_HPP
00002 #define __OPENTREP_LANGUAGE_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
```

```
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace OPENTREP {
00011
00018   struct Language {
00019   public:
00025     typedef enum {
00026       lg_com = 0,
00027       ar, ar_DZ, ar_MA, ar_SA, ar_TN,
00028       de, de_DE,
00029       en, en_GB, en_US,
00030       es, es_AR, es_ES, es_MX,
00031       fa, fa_IR,
00032       fr, fr_BE, fr_CA, fr_CH, fr_FR,
00033       he, he_IL,
00034       ko, ko_KR,
00035       pl, pl_PL,
00036       pt, pt_BR, pt_PT,
00037       sv, sv_SE,
00038       ru, ru_RU,
00039       uk, uk_UA,
00040       zh, zh_CN, zh_HK, zh_TW,
00041       LAST_VALUE
00042     } EN_Language;
00043
00047     static const std::string& getLongLabel (const EN_Language&);
00048
00052     static const std::string& getShortLabel (const EN_Language&);
00053
00057     static EN_Language getCode (const std::string&);
00058
00059   private:
00064     static const std::string _longLabels[LAST_VALUE+1];
00065
00069     static const std::string _shortLabels[LAST_VALUE+1];
00070   };
00071
00072 }
00073 #endif // __OPENTREP_LANGUAGE_HPP
```

24.265 opentrep/Location.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/OPENTREP_ParserHelperTypes.hpp>
#include <opentrep/OPENTREP_Abstract.hpp>
#include <opentrep/Language.hpp>
#include <opentrep/LocationKey.hpp>
#include <opentrep/NameMatrix.hpp>
#include <opentrep/LocationList.hpp>
```

Classes

- struct OPENTREP::Location

  Structure modelling a (geographical) location.

Namespaces

- OPENTREP

24.266 Location.hpp

```
00001 #ifndef __OPENTREP_LOCATION_HPP
00002 #define __OPENTREP_LOCATION_HPP
00003
```

```
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // OpenTrep
00011 #include <opentrep/OPENTREP_Types.hpp>
00012 #include <opentrep/OPENTREP_ParserHelperTypes.hpp>
00013 #include <opentrep/OPENTREP_Abstract.hpp>
00014 #include <opentrep/Language.hpp>
00015 #include <opentrep/LocationKey.hpp>
00016 #include <opentrep/NameMatrix.hpp>
00017 #include <opentrep/LocationList.hpp>
00018
00019 namespace OPENTREP {
00020
00024   struct Location : public OPENTREP_Abstract {
00025   public:
00026     // ///////// Getters ////////
00030     const LocationKey& getKey() const {
00031       return _key;
00032     }
00033
00037     const IATACode_T& getIataCode() const {
00038       return _key.getIataCode();
00039     }
00040
00044     const IATAType& getIataType() const {
00045       return _key.getIataType();
00046     }
00047
00051     const GeonamesID_T& getGeonamesID() const {
00052       return _key.getGeonamesID();
00053     }
00054
00058     const IsGeonames_T& isGeonames() const {
00059       return _key.isGeonames();
00060     }
00061
00065     const ICAOCode_T& getIcaoCode() const {
00066       return _icaoCode;
00067     }
00068
00072     const FAACode_T& getFaaCode() const {
00073       return _faaCode;
00074     }
00075
00080     const CommonName_T& getCommonName() const {
00081       return _commonName;
00082     }
00083
00087     const ASCIIName_T& getAsciiName() const {
00088       return _asciiName;
00089     }
00090
00094     const AltNameShortListString_T&
      getAltNameShortListString() const {
00095       return _altNameShortListString;
00096     }
00097
00101     const TvlPORListString_T& getTvlPORListString() const {
00102       return _tvlPORListString;
00103     }
00104
00108     const EnvelopeID_T& getEnvelopeID() const {
00109       return _envelopeID;
00110     }
00111
00115     const Date_T& getDateFrom() const {
00116       return _dateFrom;
00117     }
00118
00122     const Date_T& getDateEnd() const {
00123       return _dateEnd;
00124     }
00125
00129     const Comment_T& getComment() const {
00130       return _comment;
00131     }
00132
00136     const CityCode_T& getCityCode() const {
00137       return _cityCode;
00138     }
00139
00143     const CityUTFName_T& getCityUtfName() const {
00144       return _cityUtfName;
```

```
00145     }
00146
00150     const CityASCIIName_T& getCityAsciiName() const {
00151       return _cityAsciiName;
00152     }
00153
00157     const GeonamesID_T& getCityGeonamesID() const {
00158       return _cityGeonamesID;
00159     }
00160
00164     const StateCode_T& getStateCode() const {
00165       return _stateCode;
00166     }
00167
00171     const CountryCode_T& getCountryCode() const {
00172       return _countryCode;
00173     }
00174
00178     const AltCountryCode_T& getAltCountryCode() const {
00179       return _altCountryCode;
00180     }
00181
00185     const CountryName_T& getCountryName() const {
00186       return _countryName;
00187     }
00188
00192     const ContinentCode_T& getContinentCode() const {
00193       return _continentCode;
00194     }
00195
00199     const ContinentName_T& getContinentName() const {
00200       return _continentName;
00201     }
00202
00206     const TimeZone_T& getTimeZone() const {
00207       return _timeZone;
00208     }
00209
00213     const GMTOffset_T& getGMTOffset() const {
00214       return _gmtOffset;
00215     }
00216
00220     const DSTOffset_T& getDSTOffset() const {
00221       return _dstOffset;
00222     }
00223
00227     const RawOffset_T& getRawOffset() const {
00228       return _rawOffset;
00229     }
00230
00234     const Latitude_T& getLatitude() const {
00235       return _latitude;
00236     }
00237
00241     const Longitude_T& getLongitude() const {
00242       return _longitude;
00243     }
00244
00248     const FeatureClass_T& getFeatureClass() const {
00249       return _featClass;
00250     }
00251
00255     const FeatureCode_T& getFeatureCode() const {
00256       return _featCode;
00257     }
00258
00262     const Admin1Code_T& getAdmin1Code() const {
00263       return _admin1Code;
00264     }
00265
00269     const Admin1UTFName_T& getAdmin1UtfName() const {
00270       return _admin1UtfName;
00271     }
00272
00276     const Admin1ASCIIName_T& getAdmin1AsciiName() const {
00277       return _admin1AsciiName;
00278     }
00279
00283     const Admin2Code_T& getAdmin2Code() const {
00284       return _admin2Code;
00285     }
00286
00290     const Admin2UTFName_T& getAdmin2UtfName() const {
00291       return _admin2UtfName;
00292     }
00293
00297     const Admin2ASCIIName_T& getAdmin2AsciiName() const {
```

```
00298       return _admin2AsciiName;
00299     }
00300
00304     const Admin3Code_T& getAdmin3Code() const {
00305       return _admin3Code;
00306     }
00307
00311     const Admin4Code_T& getAdmin4Code() const {
00312       return _admin4Code;
00313     }
00314
00318     const Population_T& getPopulation() const {
00319       return _population;
00320     }
00321
00325     const Elevation_T& getElevation() const {
00326       return _elevation;
00327     }
00328
00332     const GTopo30_T& getGTopo30() const {
00333       return _gTopo30;
00334     }
00335
00339     const PageRank_T& getPageRank() const {
00340       return _pageRank;
00341     }
00342
00346     const Date_T& getModificationDate() const {
00347       return _modificationDate;
00348     }
00349
00353     const WikiLink_T& getWikiLink() const {
00354       return _wikiLink;
00355     }
00356
00360     const NameMatrix& getNameMatrix() const {
00361       return _nameMatrix;
00362     }
00363
00373     bool getNameList (const LanguageCode_T& iLanguageCode,
00374                       NameList_T& ioNameList) const {
00375       return _nameMatrix.getNameList (iLanguageCode, ioNameList);
00376     }
00377
00381     const std::string& getOriginalKeywords() const {
00382       return _originalKeywords;
00383     }
00384
00388     const std::string& getCorrectedKeywords() const {
00389       return _correctedKeywords;
00390     }
00391
00395     const MatchingPercentage_T& getPercentage() const {
00396       return _percentage;
00397     }
00398
00402     const NbOfErrors_T& getEditDistance() const {
00403       return _editDistance;
00404     }
00405
00410     const NbOfErrors_T& getAllowableEditDistance () const {
00411       return _allowableEditDistance;
00412     }
00413
00417     const LocationList_T& getExtraLocationList() const {
00418       return _extraLocationList;
00419     }
00420
00424     const LocationList_T& getAlternateLocationList() const {
00425       return _alternateLocationList;
00426     }
00427
00431     const RawDataString_T& getRawDataString() const {
00432       return _rawDataString;
00433     }
00434
00435
00436   public:
00437     // ////////////////////// Setters //////////////////////
00441     void setKey (const LocationKey& iKey) {
00442       _key = iKey;
00443     }
00444
00448     void setIataCode (const std::string& iIataCode) {
00449       _key.setIataCode (iIataCode);
00450     }
00451
```

```
00455     void setIataType (const IATAType& iIATAType) {
00456       _key.setIataType (iIATAType);
00457     }
00458
00462     void setGeonamesID (const GeonamesID_T& iGeonamesID) {
00463       _key.setGeonamesID (iGeonamesID);
00464     }
00465
00469     void setIcaoCode (const std::string& iIcaoCode) {
00470       _icaoCode = ICAOCode_T (iIcaoCode);
00471     }
00472
00476     void setFaaCode (const std::string& iFaaCode) {
00477       _faaCode = FAACode_T (iFaaCode);
00478     }
00479
00484     void setCommonName (const std::string& iName) {
00485       _commonName = CommonName_T (iName);
00486     }
00487
00491     void setAsciiName (const std::string& iName) {
00492       _asciiName = ASCIIName_T (iName);
00493     }
00494
00498     void setAltNameShortListString (const std::string& iNameListString) {
00499       _altNameShortListString = AltNameShortListString_T (iNameListString);
00500     }
00501
00505     void setTvlPORListString (const std::string& iPORListString) {
00506       _tvlPORListString = TvlPORListString_T (iPORListString);
00507     }
00508
00512     void setEnvelopeID (const EnvelopeID_T& iEnvelopeID) {
00513       _envelopeID = iEnvelopeID;
00514     }
00515
00519     void setDateFrom (const Date_T& iDate) {
00520       _dateFrom = iDate;
00521     }
00522
00526     void setDateEnd (const Date_T& iDate) {
00527       _dateEnd = iDate;
00528     }
00529
00533     void setComment (const std::string& iComment) {
00534       _comment = Comment_T (iComment);
00535     }
00536
00540     void setCityCode (const std::string& iCityCode) {
00541       _cityCode = CityCode_T (iCityCode);
00542     }
00543
00547     void setCityUtfName (const std::string& iCityUtfName) {
00548       _cityUtfName = CityUTFName_T (iCityUtfName);
00549     }
00550
00554     void setCityAsciiName (const std::string& iCityAsciiName) {
00555       _cityAsciiName = CityASCIIName_T (iCityAsciiName);
00556     }
00557
00561     void setCityGeonamesID (const GeonamesID_T& iGeonamesID) {
00562       _cityGeonamesID = iGeonamesID;
00563     }
00564
00568     void setCountryCode (const std::string& iCountryCode) {
00569       _countryCode = CountryCode_T (iCountryCode);
00570     }
00571
00575     void setAltCountryCode (const std::string& iCountryCode) {
00576       _altCountryCode = AltCountryCode_T (iCountryCode);
00577     }
00578
00582     void setCountryName (const std::string& iCountryName) {
00583       _countryName = CountryName_T (iCountryName);
00584     }
00585
00589     void setStateCode (const std::string& iStateCode) {
00590       _stateCode = StateCode_T (iStateCode);
00591     }
00592
00596     void setContinentCode (const std::string& iContinentCode) {
00597       _continentCode = ContinentCode_T (iContinentCode);
00598     }
00599
00603     void setContinentName (const std::string& iContinentName) {
00604       _continentName = ContinentName_T (iContinentName);
00605     }
```

```
00606
00610     void setTimeZone (const std::string& iTimeZone) {
00611       _timeZone = TimeZone_T (iTimeZone);
00612     }
00613
00617     void setGMTOffset (const GMTOffset_T& iOffset) {
00618       _gmtOffset = iOffset;
00619     }
00620
00624     void setDSTOffset (const DSTOffset_T& iOffset) {
00625       _dstOffset = iOffset;
00626     }
00627
00631     void setRawOffset (const RawOffset_T& iOffset) {
00632       _rawOffset = iOffset;
00633     }
00634
00638     void setLatitude (const Latitude_T& iLatitude) {
00639       _latitude = iLatitude;
00640     }
00641
00645     void setLongitude (const Longitude_T& iLongitude) {
00646       _longitude = iLongitude;
00647     }
00648
00652     void setFeatureClass (const std::string& iFeatClass) {
00653       _featClass = FeatureClass_T (iFeatClass);
00654     }
00655
00659     void setFeatureCode (const std::string& iFeatCode) {
00660       _featCode = FeatureCode_T (iFeatCode);
00661     }
00662
00666     void setAdmin1Code (const std::string& iAdminCode) {
00667       _admin1Code = Admin1Code_T (iAdminCode);
00668     }
00669
00673     void setAdmin1UtfName (const std::string& iAdminName) {
00674       _admin1UtfName = Admin1UTFName_T (iAdminName);
00675     }
00676
00680     void setAdmin1AsciiName (const std::string& iAdminName) {
00681       _admin1AsciiName = Admin1ASCIIName_T (iAdminName);
00682     }
00683
00687     void setAdmin2Code (const std::string& iAdminCode) {
00688       _admin2Code = Admin2Code_T (iAdminCode);
00689     }
00690
00694     void setAdmin2UtfName (const std::string& iAdminName) {
00695       _admin2UtfName = Admin2UTFName_T (iAdminName);
00696     }
00697
00701     void setAdmin2AsciiName (const std::string& iAdminName) {
00702       _admin2AsciiName = Admin2ASCIIName_T (iAdminName);
00703     }
00704
00708     void setAdmin3Code (const std::string& iAdminCode) {
00709       _admin3Code = Admin3Code_T (iAdminCode);
00710     }
00711
00715     void setAdmin4Code (const std::string& iAdminCode) {
00716       _admin4Code = Admin4Code_T (iAdminCode);
00717     }
00718
00722     void setPopulation (const Population_T& iPopulation) {
00723       _population = iPopulation;
00724     }
00725
00729     void setElevation (const Elevation_T& iElevation) {
00730       _elevation = iElevation;
00731     }
00732
00736     void setGTopo30 (const GTopo30_T& iGTopo30) {
00737       _gTopo30 = iGTopo30;
00738     }
00739
00743     void setPageRank (const PageRank_T& iPageRank) {
00744       _pageRank = iPageRank;
00745     }
00746
00750     void setModificationDate (const Date_T& iModDate) {
00751       _modificationDate = iModDate;
00752     }
00753
00757     void setWikiLink (const std::string& iWikiLink) {
00758       _wikiLink = WikiLink_T (iWikiLink);
```

```
00759     }
00760
00767     void addName (const LanguageCode_T& iLanguageCode,
00768                   const std::string& iName) {
00769       _nameMatrix.addName (iLanguageCode, iName);
00770     }
00771
00775     void resetMatrix() {
00776       _nameMatrix.reset();
00777     }
00778
00782     void setOriginalKeywords (const std::string& iOriginalKeywords) {
00783       _originalKeywords = iOriginalKeywords;
00784     }
00785
00789     void setCorrectedKeywords (const std::string& iCorrectedKeywords) {
00790       _correctedKeywords = iCorrectedKeywords;
00791     }
00792
00796     void setPercentage (const MatchingPercentage_T& iPercentage) {
00797       _percentage = iPercentage;
00798     }
00799
00803     void setEditDistance (const NbOfErrors_T& iEditDistance) {
00804       _editDistance = iEditDistance;
00805     }
00806
00811     void setAllowableEditDistance (const NbOfErrors_T&
    iAllowableEditDistance) {
00812       _allowableEditDistance = iAllowableEditDistance;
00813     }
00814
00818     void addExtraLocation (const Location& iExtraLocation) {
00819       _extraLocationList.push_back (iExtraLocation);
00820     }
00821
00825     void addAlternateLocation (const Location& iAlternateLocation) {
00826       _alternateLocationList.push_back (iAlternateLocation);
00827     }
00828
00832     void setRawDataString (const std::string& iRawDataString) {
00833       _rawDataString = RawDataString_T (iRawDataString);
00834     }
00835
00836
00837   public:
00838     // ///////// Parsing support methods ////////
00842     Date_T calculateDate() const;
00843
00848     void consolidateAltNameShortListString();
00849
00853     void consolidateTvlPORListString();
00854
00870     static FeatureNameList_T getFeatureList (const
    FeatureCode_T&);
00871
00872
00873   public:
00874     // ///////// Display methods ////////
00880     void toStream (std::ostream&) const;
00881
00887     void fromStream (std::istream&);
00888
00894     std::string toString() const;
00895
00899     std::string describeKey() const;
00900
00904     std::string describeShortKey() const;
00905
00912     std::string toBasicString() const;
00913
00920     std::string toShortString() const;
00921
00928     std::string toSingleLocationString() const;
00929
00930
00931   public:
00935     Location (const IATACode_T&, const IATAType&, const
    GeonamesID_T&,
00936               const ICAOCode_T&, const FAACode_T&,
00937               const CommonName_T&, const ASCIIName_T&,
00938               const EnvelopeID_T&,
00939               const Date_T& iDateFrom, const Date_T& iDateEnd, const
    Comment_T&,
00940               const CityCode_T&, const CityUTFName_T&, const
    CityASCIIName_T&,
00941               const GeonamesID_T&,
```

```
00942               const StateCode_T&, const CountryCode_T&, const
      AltCountryCode_T&,
00943               const CountryName_T&, const ContinentName_T&,
00944               const Latitude_T&, const Longitude_T&,
00945               const FeatureClass_T&, const FeatureCode_T&,
00946               const Admin1Code_T&, const Admin1UTFName_T&,
00947               const Admin1ASCIIName_T&,
00948               const Admin2Code_T&, const Admin2UTFName_T&,
00949               const Admin2ASCIIName_T&,
00950               const Admin3Code_T&, const Admin4Code_T&,
00951               const Population_T&, const Elevation_T&, const
      GTopo30_T&,
00952               const TimeZone_T&,
00953               const GMTOffset_T&, const DSTOffset_T&, const
      RawOffset_T&,
00954               const Date_T& iModDate, const TvlPORListString_T&,
00955               const WikiLink_T&, const PageRank_T&,
00956               const std::string& iOriginalKeywords,
00957               const std::string& iCorrectedKeywords,
00958               const MatchingPercentage_T& iPercentage,
00959               const NbOfErrors_T& iEditDistance,
00960               const NbOfErrors_T& iAllowableEditDistance,
00961               const RawDataString_T&);
00962
00966     Location();
00967
00971     Location (const Location&);
00972
00976     ~Location();
00977
00978
00979   private:
00980     // /////// Attributes /////////
00984     LocationKey _key;
00985
00989     ICAOCode_T _icaoCode;
00990
00994     FAACode_T _faaCode;
00995
00999     CommonName_T _commonName;
01000
01004     ASCIIName_T _asciiName;
01005
01011     AltNameShortListString_T _altNameShortListString;
01012
01017     TvlPORListString_T _tvlPORListString;
01018
01027     EnvelopeID_T _envelopeID;
01028
01032     Date_T _dateFrom;
01033
01037     Date_T _dateEnd;
01038
01042     Comment_T _comment;
01043
01049     CityCode_T _cityCode;
01050
01056     CityUTFName_T _cityUtfName;
01057
01064     CityASCIIName_T _cityAsciiName;
01065
01071     GeonamesID_T _cityGeonamesID;
01072
01076     StateCode_T _stateCode;
01077
01081     CountryCode_T _countryCode;
01082
01086     AltCountryCode_T _altCountryCode;
01087
01091     CountryName_T _countryName;
01092
01096     ContinentCode_T _continentCode;
01097
01101     ContinentName_T _continentName;
01102
01106     Latitude_T _latitude;
01107
01111     Longitude_T _longitude;
01112
01116     FeatureClass_T _featClass;
01117
01121     FeatureCode_T _featCode;
01122
01126     Admin1Code_T _admin1Code;
01127
01131     Admin1UTFName_T _admin1UtfName;
01132
```

```
01136     Admin1ASCIIName_T _admin1AsciiName;
01137
01141     Admin2Code_T _admin2Code;
01142
01146     Admin2UTFName_T _admin2UtfName;
01147
01151     Admin2ASCIIName_T _admin2AsciiName;
01152
01156     Admin3Code_T _admin3Code;
01157
01161     Admin4Code_T _admin4Code;
01162
01166     Population_T _population;
01167
01171     Elevation_T _elevation;
01172
01176     GTopo30_T _gTopo30;
01177
01181     TimeZone_T _timeZone;
01182
01186     GMTOffset_T _gmtOffset;
01187
01191     DSTOffset_T _dstOffset;
01192
01196     RawOffset_T _rawOffset;
01197
01201     Date_T _modificationDate;
01202
01206     WikiLink_T _wikiLink;
01207
01218     PageRank_T _pageRank;
01219
01223     NameMatrix _nameMatrix;
01224
01225   private:
01226     // ////////////// Full-text matching process support attributes //////////
01230     std::string _originalKeywords;
01231
01235     std::string _correctedKeywords;
01236
01240     MatchingPercentage_T _percentage;
01241
01245     NbOfErrors_T _editDistance;
01246
01251     NbOfErrors_T _allowableEditDistance;
01252
01256     LocationList_T _extraLocationList;
01257
01261     LocationList_T _alternateLocationList;
01262
01267     RawDataString_T _rawDataString;
01268
01269
01270   public:
01271     // ///////// Parsing support temporary attributes ////////
01275     year_t _itYear;
01276     month_t _itMonth;
01277     day_t _itDay;
01278
01282     Language::EN_Language _itLanguageCodeNum;
01283     LanguageCode_T _itLanguageCode;
01284     std::string _itLangCode2Char;
01285     std::string _itLangCodeExt;
01286     std::string _itLangCodeHist;
01287     AltNameShortList_T _itAltNameShortList;
01288     TvlPORList_T _itTvlPORList;
01289   };
01290
01291 }
01292 #endif // __OPENTREP_LOCATION_HPP
```

24.267 opentrep/LocationKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/OPENTREP_Abstract.hpp>
#include <opentrep/IATAType.hpp>
```

Classes

- struct OPENTREP::LocationKey

  Class modelling the primary key of a location/POR (point of reference).

Namespaces

- OPENTREP

24.268 LocationKey.hpp

```
00001 #ifndef __OPENTREP_LOCATIONKEY_HPP
00002 #define __OPENTREP_LOCATIONKEY_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // OpenTrep
00011 #include <opentrep/OPENTREP_Types.hpp>
00012 #include <opentrep/OPENTREP_Abstract.hpp>
00013 #include <opentrep/IATAType.hpp>
00014
00015 namespace OPENTREP {
00016
00021   struct LocationKey : public OPENTREP_Abstract {
00022   public:
00023     // ///////////////// Getters ///////////////
00027     const IATACode_T& getIataCode() const {
00028       return _iataCode;
00029     }
00030
00034     const IATAType& getIataType() const {
00035       return _iataType;
00036     }
00037
00041     const GeonamesID_T& getGeonamesID() const {
00042       return _geonamesID;
00043     }
00044
00048     const IsGeonames_T& isGeonames() const {
00049       return _isGeonames;
00050     }
00051
00052
00053   public:
00054     // ////////////////// Setters /////////////////
00058     void setIataCode (const std::string& iIataCode) {
00059       _iataCode = IATACode_T (iIataCode);
00060     }
00061
00065     void setIataType (const IATAType& iIATAType) {
00066       _iataType = iIATAType;
00067     }
00068
00072     void setGeonamesID (const GeonamesID_T& iGeonamesID) {
00073       _geonamesID = iGeonamesID;
00074     }
00075
00079     void setIsGeonames (const IsGeonames_T& isGeonames) {
00080       _isGeonames = isGeonames;
00081     }
00082
00083
00084   public:
00085     // ////////////// Display methods //////////////
00091     void toStream (std::ostream&) const;
00092
00098     void fromStream (std::istream&);
00099
00103     std::string toString() const;
00104
00109     std::string describe() const;
00110
00111
00112   public:
00113     // ////////////// Constructors and destructors //////////////
00117     LocationKey (const IATACode_T&, const IATAType&, const
```

```
      GeonamesID_T&);
00118
00122     LocationKey (const LocationKey&);
00123
00127     ~LocationKey();
00128
00132     bool operator== (const LocationKey&) const;
00133
00137     bool operator!= (const LocationKey& iLocationKey) const {
00138       return !(iLocationKey == *this);
00139     }
00140
00141   private:
00145     LocationKey();
00146
00147
00148   private:
00149     // ////////////////////// Attributes ////////////////////////
00153     IATACode_T _iataCode;
00154
00158     IATAType _iataType;
00159
00163     GeonamesID_T _geonamesID;
00164
00168     IsGeonames_T _isGeonames;
00169   };
00170
00171 }
00172 #endif // __OPENTREP_LOCATIONKEY_HPP
```

24.269 opentrep/LocationList.hpp File Reference

```
#include <list>
```

Namespaces

- OPENTREP

Typedefs

- typedef std::list< Location > OPENTREP::LocationList_T

24.270 LocationList.hpp

```
00001 #ifndef __OPENTREP_LOCATIONLIST_HPP
00002 #define __OPENTREP_LOCATIONLIST_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <list>
00009
00010 namespace OPENTREP {
00011
00012   // Forward declaration
00013   struct Location;
00014
00018   typedef std::list<Location> LocationList_T;
00019
00020 }
00021 #endif // __OPENTREP_LOCATIONLIST_HPP
00022
```

24.271 opentrep/NameMatrix.hpp File Reference

```
#include <iosfwd>
```

```
#include <string>
#include <list>
#include <map>
#include <opentrep/OPENTREP_Abstract.hpp>
#include <opentrep/Names.hpp>
```

Classes

- struct OPENTREP::NameMatrix

Namespaces

- OPENTREP

24.272 NameMatrix.hpp

```
00001 #ifndef __OPENTREP_NAMEMATRIX_HPP
00002 #define __OPENTREP_NAMEMATRIX_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <list>
00011 #include <map>
00012 // OpenTrep
00013 #include <opentrep/OPENTREP_Abstract.hpp>
00014 #include <opentrep/Names.hpp>
00015
00016 namespace OPENTREP {
00017
00022   struct NameMatrix : public OPENTREP_Abstract {
00023   public:
00024     // //// Constructors / Destructors ///////
00028     NameMatrix();
00032     NameMatrix (const NameMatrix&);
00036     ~NameMatrix();
00037
00038
00039   public:
00040     // ////////// Getters ///////////////
00044     const NameMatrix_T& getNameMatrix() const {
00045       return _nameMatrix;
00046     }
00047
00057     bool getNameList (const LanguageCode_T&, NameList_T&) const;
00058
00059
00060   public:
00061     // ////////// Setters ///////////////
00068     void addName (const LanguageCode_T&, const std::string& iName);
00069
00073     void reset();
00074
00075
00076   public:
00077     // ////////// Display support methods /////////
00083     void toStream (std::ostream&) const;
00084
00090     void fromStream (std::istream&);
00091
00095     std::string toString() const;
00096
00102     std::string describeKey() const;
00103
00109     std::string describe() const;
00110
00111
00112   private:
00113     // ////////// Attributes ///////////////
00117     NameMatrix_T _nameMatrix;
00118   };
00119
```

```
00120 }
00121 #endif // __OPENTREP_BOM_NAMEMATRIX_HPP
```

24.273 opentrep/Names.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <list>
#include <map>
#include <opentrep/OPENTREP_Abstract.hpp>
#include <opentrep/OPENTREP_Types.hpp>
```

Classes

- struct OPENTREP::Names

Namespaces

- OPENTREP

Typedefs

- typedef std::list< std::string > OPENTREP::NameList_T
- typedef std::map
  < LanguageCode_T, Names > OPENTREP::NameMatrix_T

24.274 Names.hpp

```
00001 #ifndef __OPENTREP_NAMES_HPP
00002 #define __OPENTREP_NAMES_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <list>
00011 #include <map>
00012 // OpenTrep
00013 #include <opentrep/OPENTREP_Abstract.hpp>
00014 #include <opentrep/OPENTREP_Types.hpp>
00015
00016 namespace OPENTREP {
00017
00018
00019   // ////////////////// Type definitions ////////////////////
00020   typedef std::list<std::string> NameList_T;
00021
00026   struct Names : public OPENTREP_Abstract {
00027   public:
00028     // //// Constructors / Destructors ///////
00032     Names (const LanguageCode_T&);
00036     Names (const Names&);
00040     ~Names();
00041   private:
00045     Names();
00046
00047
00048   public:
00049     // /////////// Getters ///////////////
00053     LanguageCode_T getLanguageCode() const {
00054       return _languageCode;
00055     }
00056
00060     const NameList_T& getNameList() const {
00061       return _nameList;
```

```
00062     }
00063
00069     std::string getFirstName() const;
00070
00071
00072   public:
00073     // ////////// Setters ///////////////
00077     void setLanguageCode (const LanguageCode_T& iLanguageCode) {
00078       _languageCode = iLanguageCode;
00079     }
00080
00084     void addName (const std::string& iName);
00085
00089     void resetList();
00090
00091
00092   public:
00093     // ////////// Display support methods ////////
00099     void toStream (std::ostream& ioOut) const;
00100
00106     void fromStream (std::istream& ioIn);
00107
00111     std::string toString() const;
00112
00118     std::string describeKey() const;
00119
00125     std::string describe() const;
00126
00127
00128   private:
00129     // ////////// Attributes //////////////
00133     LanguageCode_T _languageCode;
00134
00138     NameList_T _nameList;
00139   };
00140
00141   // ////////////// Type definitions ////////////
00149   typedef std::map<LanguageCode_T, Names> NameMatrix_T;
00150
00151 }
00152 #endif // __OPENTREP_NAMES_HPP
```

24.275 opentrep/OPENTREP_Abstract.hpp File Reference

```
#include <iosfwd>
#include <string>
```

Classes

- struct OPENTREP::OPENTREP_Abstract

Namespaces

- OPENTREP

Functions

- template<class charT , class traits >
  std::basic_ostream< charT,
  traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const OPENTREP::OPENTREP_-Abstract &iStructure)
- template<class charT , class traits >
  std::basic_istream< charT,
  traits > & operator>> (std::basic_istream< charT, traits > &ioIn, OPENTREP::OPENTREP_Abstract &io-Stucture)

24.275.1 Function Documentation

24.275.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut,* const OPENTREP::OPENTREP_Abstract & *iStructure*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 61 of file OPENTREP_Abstract.hpp.

24.275.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn,* OPENTREP::OPENTREP_Abstract & *ioStucture*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 89 of file OPENTREP_Abstract.hpp.

References OPENTREP::OPENTREP_Abstract::fromStream().

24.276 OPENTREP_Abstract.hpp

```
00001 #ifndef __OPENTREP_OPENTREP_ABSTRACT_HPP
00002 #define __OPENTREP_OPENTREP_ABSTRACT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010
00011 namespace OPENTREP {
00012
00016   struct OPENTREP_Abstract {
00017   public:
00018     // ////////// Display support methods /////////
00024     virtual void toStream (std::ostream& ioOut) const = 0;
00025
00031     virtual void fromStream (std::istream& ioIn) = 0;
00032
00036     virtual std::string toString() const = 0;
00037
00038
00039   protected:
00043     OPENTREP_Abstract () {}
00044     OPENTREP_Abstract (const OPENTREP_Abstract&) {}
00045
00049     virtual ~OPENTREP_Abstract() {}
00050   };
00051 }
00052
00058 template <class charT, class traits>
00059 inline
00060 std::basic_ostream<charT, traits>&
00061 operator<< (std::basic_ostream<charT, traits>& ioOut,
00062             const OPENTREP::OPENTREP_Abstract& iStructure) {
00068   std::basic_ostringstream<charT,traits> ostr;
00069   ostr.copyfmt (ioOut);
00070   ostr.width (0);
00071
00072   // Fill string stream
00073   iStructure.toStream (ostr);
00074
00075   // Print string stream
00076   ioOut << ostr.str();
00077
00078   return ioOut;
00079 }
00080
00086 template <class charT, class traits>
00087 inline
00088 std::basic_istream<charT, traits>&
00089 operator>> (std::basic_istream<charT, traits>& ioIn,
00090             OPENTREP::OPENTREP_Abstract& ioStucture) {
00091   // Fill Bom object with input stream
00092   ioStucture.fromStream (ioIn);
00093   return ioIn;
```

```
00094 }
00095
00096 #endif // __OPENTREP_OPENTREP_ABSTRACT_HPP
```

24.277 opentrep/OPENTREP_exceptions.hpp File Reference

```
#include <exception>
```

Classes

- class OPENTREP::RootException
  Root of the OpenTREP exceptions.
- class OPENTREP::NonInitialisedServiceException
- class OPENTREP::UnicodeException
- class OPENTREP::UnicodeTransliteratorCreationException
- class OPENTREP::MultipleRowsForASingleDocIDException
- class OPENTREP::LanguageCodeNotDefinedInNameTableException
- class OPENTREP::MemoryAllocationException
- class OPENTREP::ObjectNotFoundException
- class OPENTREP::FileException
- class OPENTREP::FileNotFoundException
- class OPENTREP::ParserException
- class OPENTREP::PorFileParsingException
- class OPENTREP::PRFileParsingException
- class OPENTREP::CodeConversionException
- class OPENTREP::CodeDuplicationException
- class OPENTREP::XapianException
- class OPENTREP::DocumentNotFoundException
- class OPENTREP::XapianDatabaseFailureException
- class OPENTREP::XapianTravelDatabaseEmptyException
- class OPENTREP::XapianTravelDatabaseWrongPathnameException
- class OPENTREP::NoBestMatchingResultHolderException
- class OPENTREP::SQLDatabaseException
- class OPENTREP::SQLDatabaseConnectionImpossibleException
- class OPENTREP::BuildIndexException
- class OPENTREP::InterpreterUseCaseException
- class OPENTREP::InterpreteTravelRequestException
- class OPENTREP::TravelRequestEmptyException

Namespaces

- OPENTREP

24.278 OPENTREP_exceptions.hpp

```
00001 #ifndef __OPENTREP_OPENTREP_EXCEPTIONS_HPP
00002 #define __OPENTREP_OPENTREP_EXCEPTIONS_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <exception>
00009
00010 namespace OPENTREP {
00011
00012   // ///////// Exceptions ////////////
```

```
00020   class RootException : public std::exception {
00021   public:
00025     RootException (const std::string& iWhat) : _what (iWhat) {}
00029     RootException() : _what ("No further details") {}
00030
00034     virtual ~RootException() throw() {}
00035
00039     const char* what() const throw() {
00040       return _what.c_str();
00041     }
00042
00043   protected:
00047     std::string _what;
00048   };
00049
00053   class NonInitialisedServiceException : public
      RootException {
00054   public:
00058     NonInitialisedServiceException (const std::string& iWhat)
00059       : RootException (iWhat) {}
00060   };
00061
00065   class UnicodeException : public RootException {
00066   public:
00070     UnicodeException (const std::string& iWhat) : RootException (iWhat) {}
00071   };
00072
00076   class UnicodeTransliteratorCreationException : public
      UnicodeException {
00077   public:
00081     UnicodeTransliteratorCreationException (const std::string& iWhat)
00082       : UnicodeException (iWhat) {}
00083   };
00084
00088   class MultipleRowsForASingleDocIDException : public
      RootException {
00089   public:
00093     MultipleRowsForASingleDocIDException (const std::string& iWhat)
00094       : RootException (iWhat) {}
00095   };
00096
00100   class LanguageCodeNotDefinedInNameTableException : public
      RootException {
00101   public:
00105     LanguageCodeNotDefinedInNameTableException (const std::string
      & iWhat)
00106       : RootException (iWhat) {}
00107   };
00108
00112   class MemoryAllocationException : public
      RootException {
00113   public:
00117     MemoryAllocationException (const std::string& iWhat) :
      RootException (iWhat) {}
00118   };
00119
00123   class ObjectNotFoundException : public RootException {
00124   public:
00128     ObjectNotFoundException (const std::string& iWhat) :
      RootException (iWhat) {}
00129   };
00130
00134   class FileException : public RootException {
00135   public:
00139     FileException (const std::string& iWhat) : RootException (iWhat) {}
00140   };
00141
00145   class FileNotFoundException : public RootException {
00146   public:
00150     FileNotFoundException (const std::string& iWhat) :
      RootException (iWhat) {}
00151   };
00152
00156   class ParserException : public RootException {
00157   public:
00161     ParserException (const std::string& iWhat) : RootException (iWhat) {}
00162   };
00163
00167   class PorFileParsingException : public ParserException {
00168   public:
00172     PorFileParsingException (const std::string& iWhat)
00173       : ParserException (iWhat) {}
00174   };
00175
00179   class PRFileParsingException : public ParserException {
00180   public:
00184     PRFileParsingException (const std::string& iWhat)
```

```
00185       : ParserException (iWhat) {}
00186   };
00187
00191   class CodeConversionException : public ParserException {
00192   public:
00196     CodeConversionException (const std::string& iWhat)
00197       : ParserException (iWhat) {}
00198   };
00199
00203   class CodeDuplicationException : public
      ParserException {
00204   public:
00208     CodeDuplicationException (const std::string& iWhat)
00209       : ParserException (iWhat) {}
00210   };
00211
00215   class XapianException : public RootException {
00216   public:
00220     XapianException (const std::string& iWhat) : RootException (iWhat) {}
00221   };
00222
00226   class DocumentNotFoundException : public
      XapianException {
00227   public:
00231     DocumentNotFoundException (const std::string& iWhat)
00232       : XapianException (iWhat) {}
00233   };
00234
00238   class XapianDatabaseFailureException : public
      XapianException {
00239   public:
00243     XapianDatabaseFailureException (const std::string& iWhat)
00244       : XapianException (iWhat) {}
00245   };
00246
00250   class XapianTravelDatabaseEmptyException : public
      XapianException {
00251   public:
00255     XapianTravelDatabaseEmptyException (const std::string& iWhat)
00256       : XapianException (iWhat) {}
00257   };
00258
00262   class XapianTravelDatabaseWrongPathnameException : public
      XapianException {
00263   public:
00267     XapianTravelDatabaseWrongPathnameException (const std::string
      & iWhat)
00268       : XapianException (iWhat) {}
00269   };
00270
00274   class NoBestMatchingResultHolderException : public
      RootException {
00275   public:
00279     NoBestMatchingResultHolderException (const std::string& iWhat)
00280       : RootException (iWhat) {}
00281   };
00282
00286   class SQLDatabaseException : public RootException {
00287   public:
00291     SQLDatabaseException (const std::string& iWhat) :
      RootException (iWhat) {}
00292   };
00293
00297   class SQLDatabaseConnectionImpossibleException : public
      SQLDatabaseException {
00298   public:
00302     SQLDatabaseConnectionImpossibleException (const std::string&
      iWhat)
00303       : SQLDatabaseException (iWhat) {}
00304   };
00305
00309   class BuildIndexException : public RootException {
00310   public:
00314     BuildIndexException (const std::string& iWhat) :
      RootException (iWhat) {}
00315   };
00316
00320   class InterpreterUseCaseException : public
      RootException {
00321   public:
00325     InterpreterUseCaseException (const std::string& iWhat)
00326       : RootException (iWhat) {}
00327   };
00328
00332   class InterpreteTravelRequestException : public
      InterpreterUseCaseException {
00333   public:
```

```
00337     InterpreteTravelRequestException (const std::string& iWhat)
00338       : InterpreterUseCaseException (iWhat) {}
00339   };
00340
00344   class TravelRequestEmptyException : public
      InterpreterUseCaseException {
00345   public:
00349     TravelRequestEmptyException (const std::string& iWhat)
00350       : InterpreterUseCaseException (iWhat) {}
00351   };
00352
00353 }
00354 #endif // __OPENTREP_OPENTREP_EXCEPTIONS_HPP
```

24.279 opentrep/OPENTREP_log.hpp File Reference

Namespaces

- OPENTREP
- OPENTREP::LOG

Enumerations

- enum OPENTREP::LOG::EN_LogLevel {
  OPENTREP::LOG::CRITICAL = 0, OPENTREP::LOG::ERROR, OPENTREP::LOG::NOTIFICATION, OPENTREP::LOG::WARNING,
  OPENTREP::LOG::DEBUG, OPENTREP::LOG::VERBOSE, OPENTREP::LOG::LAST_VALUE }

24.280 OPENTREP_log.hpp

```
00001 #ifndef __OPENTREP_OPENTREP_LOG_HPP
00002 #define __OPENTREP_OPENTREP_LOG_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007
00008 namespace OPENTREP {
00009
00010   // ///////////////// Log /////////////
00014   namespace LOG {
00015     typedef enum {
00016       CRITICAL = 0,
00017       ERROR,
00018       NOTIFICATION,
00019       WARNING,
00020       DEBUG,
00021       VERBOSE,
00022       LAST_VALUE
00023     } EN_LogLevel;
00024   }
00025
00026 }
00027 #endif // __OPENTREP_OPENTREP_LOG_HPP
```

24.281 opentrep/OPENTREP_ParserHelperTypes.hpp File Reference

```
#include <string>
#include <sstream>
#include <opentrep/OPENTREP_exceptions.hpp>
```

Classes

- struct OPENTREP::date_time_element< MIN, MAX >

Namespaces

- OPENTREP

Typedefs

- typedef date_time_element< 0, 23 > OPENTREP::hour_t
- typedef date_time_element< 0, 59 > OPENTREP::minute_t
- typedef date_time_element< 0, 59 > OPENTREP::second_t
- typedef date_time_element< 1900, 2100 > OPENTREP::year_t
- typedef date_time_element< 1, 12 > OPENTREP::month_t
- typedef date_time_element< 1, 31 > OPENTREP::day_t

Functions

- template<int MIN, int MAX>
  date_time_element< MIN, MAX > OPENTREP::operator* (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)
- template<int MIN, int MAX>
  date_time_element< MIN, MAX > OPENTREP::operator+ (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)

24.282 OPENTREP_ParserHelperTypes.hpp

```
00001 #ifndef __OPENTREP_PARSERHELPERTYPES_HPP
00002 #define __OPENTREP_PARSERHELPERTYPES_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <sstream>
00010 // OpenTREP
00011 #include <opentrep/OPENTREP_exceptions.hpp>
00012
00013 namespace OPENTREP {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   //
00017   // Parser structure helper
00018   //
00019   // ////////////////////////////////////////////////////////////////////
00021   template <int MIN = 0, int MAX = 0>
00022   struct date_time_element {
00023     unsigned int _value;
00024
00025     // ////////// Constructors ////////////
00027     date_time_element () { }
00029     date_time_element (const date_time_element& t) :
      _value (t._value) { }
00031     date_time_element (int i) : _value (i) { }
00033     void check () const {
00034       if (_value < MIN || _value > MAX) {
00035         std::ostringstream oMessage;
00036         oMessage << "The value: " << _value << " is out of range ("
00037                  << MIN << ", " << MAX << ")";
00038         throw OPENTREP::ParserException (oMessage.str());
00039       }
00040     }
00041   };
00042
00044   template <int MIN, int MAX>
00045   inline date_time_element<MIN,
00046                            MAX> operator*(const
      date_time_element<MIN, MAX>& o1,
00047                                           const date_time_element<MIN, MAX>& o2)
      {
00048     return date_time_element<MIN, MAX> (o1._value * o2.
      _value);
00049   }
00050
```

```
00052   template <int MIN, int MAX>
00053   inline date_time_element<MIN,
00054                            MAX> operator+(const
      date_time_element<MIN, MAX>& o1,
00055                                           const date_time_element<MIN, MAX>& o2)
      {
00056     return date_time_element<MIN, MAX> (o1._value + o2.
      _value);
00057   }
00058
00060   typedef date_time_element<0, 23> hour_t;
00061   typedef date_time_element<0, 59> minute_t;
00062   typedef date_time_element<0, 59> second_t;
00063   typedef date_time_element<1900, 2100> year_t;
00064   typedef date_time_element<1, 12> month_t;
00065   typedef date_time_element<1, 31> day_t;
00066
00067 }
00068 #endif // __OPENTREP_PARSERHELPERTYPES_HPP
```

24.283 opentrep/OPENTREP_Service.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/LocationList.hpp>
#include <opentrep/DistanceErrorRule.hpp>
```

Classes

- class OPENTREP::OPENTREP_Service

  Interface for the OPENTREP Services.

Namespaces

- OPENTREP

24.284 OPENTREP_Service.hpp

```
00001 #ifndef __OPENTREP_SVC_OPENTREP_SERVICE_HPP
00002 #define __OPENTREP_SVC_OPENTREP_SERVICE_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // OpenTREP
00011 #include <opentrep/OPENTREP_Types.hpp>
00012 #include <opentrep/LocationList.hpp>
00013 #include <opentrep/DistanceErrorRule.hpp>
00014
00015 namespace OPENTREP {
00016
00017   // Forward declaration
00018   class OPENTREP_ServiceContext;
00019
00020
00024   class OPENTREP_Service {
00025   public:
00026     // ////////// Business Methods /////////////
00032     NbOfDBEntries_T getIndexSize();
00033
00043     NbOfMatches_T drawRandomLocations (const
      NbOfMatches_T& iNbOfDraws,
00044                                        LocationList_T&);
00045
00052     NbOfDBEntries_T buildSQLDB();
00053
```

```
00061     NbOfDBEntries_T buildSearchIndex();
00062
00074     NbOfMatches_T interpretTravelRequest (const std::string&
      iTravelQuery,
00075                                           LocationList_T&,
      WordList_T&);
00076
00077
00089     typedef std::pair<const TravelDBFilePath_T,
00090                       const SQLiteDBFilePath_T>
      DBFilePathPair_T;
00091     typedef std::pair<const PORFilePath_T, const DBFilePathPair_T> FilePathSet_T;
00092     FilePathSet_T getFilePaths() const;
00093
00094
00095   public:
00096     // ////////// Constructors and destructors /////////
00104     OPENTREP_Service (std::ostream& ioLogStream, const
      TravelDBFilePath_T&,
00105                       const SQLiteDBFilePath_T&);
00106
00116     OPENTREP_Service (std::ostream& ioLogStream, const
      PORFilePath_T&,
00117                       const TravelDBFilePath_T&, const
      SQLiteDBFilePath_T&);
00118
00122     ~OPENTREP_Service();
00123
00124
00125   private:
00126     // /////// Construction and Destruction helper methods ///////
00130     OPENTREP_Service();
00134     OPENTREP_Service (const OPENTREP_Service&);
00135
00143     void init (std::ostream& ioLogStream, const TravelDBFilePath_T&,
00144                const SQLiteDBFilePath_T&);
00145
00155     void init (std::ostream& ioLogStream, const PORFilePath_T&,
00156                const TravelDBFilePath_T&, const
      SQLiteDBFilePath_T&);
00157
00161     void finalise();
00162
00163
00164   private:
00165     // ///////// Service Context /////////
00169     OPENTREP_ServiceContext* _opentrepServiceContext;
00170   };
00171 }
00172 #endif // __OPENTREP_SVC_OPENTREP_SERVICE_HPP
```

24.285 opentrep/OPENTREP_Types.hpp File Reference

```
#include <exception>
#include <string>
#include <list>
#include <map>
#include <set>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/array.hpp>
#include <opentrep/OPENTREP_exceptions.hpp>
#include <opentrep/OPENTREP_log.hpp>
```

Classes

- struct OPENTREP::FilePath_T
- struct OPENTREP::PORFilePath_T
- struct OPENTREP::TravelDBFilePath_T
- struct OPENTREP::SQLiteDBFilePath_T
- struct OPENTREP::RawDataString_T
- struct OPENTREP::IATACode_T

- struct OPENTREP::ICAOCode_T
- struct OPENTREP::FAACode_T
- struct OPENTREP::LocationName_T
- struct OPENTREP::CommonName_T
- struct OPENTREP::ASCIIName_T
- struct OPENTREP::AltNameShortListString_T
- struct OPENTREP::TvlPORListString_T
- struct OPENTREP::LanguageCode_T
- struct OPENTREP::Comment_T
- struct OPENTREP::CityCode_T
- struct OPENTREP::CityUTFName_T
- struct OPENTREP::CityASCIIName_T
- struct OPENTREP::StateCode_T
- struct OPENTREP::CountryCode_T
- struct OPENTREP::AltCountryCode_T
- struct OPENTREP::CountryName_T
- struct OPENTREP::ContinentCode_T
- struct OPENTREP::ContinentName_T
- struct OPENTREP::Admin1Code_T
- struct OPENTREP::Admin1UTFName_T
- struct OPENTREP::Admin1ASCIIName_T
- struct OPENTREP::Admin2Code_T
- struct OPENTREP::Admin2UTFName_T
- struct OPENTREP::Admin2ASCIIName_T
- struct OPENTREP::Admin3Code_T
- struct OPENTREP::Admin4Code_T
- struct OPENTREP::FeatureClass_T
- struct OPENTREP::FeatureCode_T
- struct OPENTREP::FeatureName_T
- struct OPENTREP::WikiLink_T
- struct OPENTREP::TimeZone_T

Namespaces

- OPENTREP

Typedefs

- typedef int OPENTREP::GeonamesID_T
- typedef int OPENTREP::EnvelopeID_T
- typedef std::list < AltNameShortListString_T > OPENTREP::AltNameShortList_T
- typedef std::list < TvlPORListString_T > OPENTREP::TvlPORList_T
- typedef std::list< FeatureName_T > OPENTREP::FeatureNameList_T
- typedef double OPENTREP::GeoCoord_T
- typedef GeoCoord_T OPENTREP::Latitude_T
- typedef GeoCoord_T OPENTREP::Longitude_T
- typedef boost::gregorian::date OPENTREP::Date_T
- typedef unsigned int OPENTREP::Population_T
- typedef int OPENTREP::Elevation_T
- typedef int OPENTREP::GTopo30_T
- typedef double OPENTREP::PageRank_T
- typedef int OPENTREP::XapianDocID_T

- typedef float OPENTREP::GMTOffset_T
- typedef float OPENTREP::DSTOffset_T
- typedef float OPENTREP::RawOffset_T
- typedef bool OPENTREP::IsGeonames_T
- typedef bool OPENTREP::IsAirport_T
- typedef bool OPENTREP::IsCommercial_T
- typedef unsigned short OPENTREP::Weight_T
- typedef std::string OPENTREP::TravelQuery_T
- typedef double OPENTREP::MatchingPercentage_T
- typedef double OPENTREP::Percentage_T
- typedef double OPENTREP::Score_T
- typedef unsigned int OPENTREP::NbOfDBEntries_T
- typedef std::string OPENTREP::Word_T
- typedef std::list< Word_T > OPENTREP::WordList_T
- typedef std::set< std::string > OPENTREP::WordSet_T
- typedef std::set< std::string > OPENTREP::BlackList_T
- typedef unsigned int OPENTREP::NbOfLetters_T
- typedef unsigned short OPENTREP::NbOfWords_T
- typedef unsigned short OPENTREP::NbOfMatches_T
- typedef unsigned short OPENTREP::NbOfErrors_T
- typedef std::map
  < NbOfLetters_T, NbOfErrors_T > OPENTREP::DistanceErrorScale_T
- typedef boost::array
  < NbOfLetters_T, 5 > OPENTREP::DistanceErrorScaleArray_T

24.286 OPENTREP_Types.hpp

```
00001 #ifndef __OPENTREP_OPENTREP_TYPES_HPP
00002 #define __OPENTREP_OPENTREP_TYPES_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <exception>
00009 #include <string>
00010 #include <list>
00011 #include <map>
00012 #include <set>
00013 // Boost Date-Time
00014 #include <boost/date_time/gregorian/gregorian.hpp>
00015 #include <boost/date_time/posix_time/posix_time.hpp>
00016 // Boost Array
00017 #include <boost/array.hpp>
00018 // OpenTrep
00019 #include <opentrep/OPENTREP_exceptions.hpp>
00020 #include <opentrep/OPENTREP_log.hpp>
00021
00022 namespace OPENTREP {
00023
00024   // //////// Type definitions /////////
00028   struct FilePath_T : public std::string {
00029   public:
00030     explicit FilePath_T (const std::string& iValue) : std::string (iValue) {
00031     }
00032   };
00033
00037   struct PORFilePath_T : public FilePath_T {
00038   public:
00039     explicit PORFilePath_T (const std::string& iValue) : FilePath_T (iValue) { }
00040   };
00041
00046   struct TravelDBFilePath_T : public std::string {
00047   public:
00048     explicit TravelDBFilePath_T (const std::string& iValue)
00049       : std::string (iValue) { }
00050   };
00051
00056   struct SQLiteDBFilePath_T : public std::string {
00057   public:
```

```
00058     explicit SQLiteDBFilePath_T (const std::string& iValue)
00059       : std::string (iValue) { }
00060   };
00061
00066   struct RawDataString_T : public std::string {
00067   public:
00068     explicit RawDataString_T (const std::string& iValue)
00069       : std::string (iValue) { }
00070   };
00071
00075   struct IATACode_T : public std::string {
00076   public:
00077     explicit IATACode_T (const std::string& iValue) : std::string (iValue) { }
00078   };
00079
00083   struct ICAOCode_T : public std::string {
00084   public:
00085     explicit ICAOCode_T (const std::string& iValue) : std::string (iValue) { }
00086   };
00087
00091   typedef int GeonamesID_T;
00092
00096   typedef int EnvelopeID_T;
00097
00101   struct FAACode_T : public std::string {
00102   public:
00103     explicit FAACode_T (const std::string& iValue) : std::string (iValue) { }
00104   };
00105
00109   struct LocationName_T : public std::string {
00110   public:
00111     explicit LocationName_T (const std::string& iValue) : std::string (iValue) {
00112     }
00113   };
00114
00119   struct CommonName_T : public LocationName_T {
00120   public:
00121     explicit CommonName_T (const std::string& iValue) :
      LocationName_T (iValue) {
00122     }
00123   };
00124
00128   struct ASCIIName_T : public LocationName_T {
00129   public:
00130     explicit ASCIIName_T (const std::string& iValue) : LocationName_T (iValue) {
00131     }
00132   };
00133
00139   struct AltNameShortListString_T : public
      LocationName_T {
00140   public:
00141     explicit AltNameShortListString_T (const std::string& iValue)
00142       : LocationName_T (iValue) {
00143     }
00144   };
00145   typedef std::list<AltNameShortListString_T> AltNameShortList_T;
00146
00151   struct TvlPORListString_T : public IATACode_T {
00152   public:
00153     explicit TvlPORListString_T (const std::string& iValue)
00154       : IATACode_T (iValue) {
00155     }
00156   };
00157   typedef std::list<TvlPORListString_T> TvlPORList_T;
00158
00163   struct LanguageCode_T : public std::string {
00164   public:
00165     explicit LanguageCode_T (const std::string& iValue) : std::string (iValue) {
00166     }
00167   };
00168
00172   struct Comment_T : public std::string {
00173   public:
00174     explicit Comment_T (const std::string& iValue) : std::string (iValue) { }
00175   };
00176
00180   struct CityCode_T : public IATACode_T {
00181   public:
00182     explicit CityCode_T (const std::string& iValue)
00183       : IATACode_T (std::string (iValue)) {
00184     }
00185   };
00186
00190   struct CityUTFName_T : public std::string {
00191   public:
00192     explicit CityUTFName_T (const std::string& iValue) : std::string (iValue) { }
00193   };
```

```
00194
00199   struct CityASCIIName_T : public std::string {
00200   public:
00201     explicit CityASCIIName_T (const std::string& iValue)
00202       : std::string (iValue) { }
00203   };
00204
00208   struct StateCode_T : public std::string {
00209   public:
00210     explicit StateCode_T (const std::string& iValue) : std::string (iValue) { }
00211   };
00212
00216   struct CountryCode_T : public std::string {
00217   public:
00218     explicit CountryCode_T (const std::string& iValue) : std::string (iValue) { }
00219   };
00220
00224   struct AltCountryCode_T : public std::string {
00225   public:
00226     explicit AltCountryCode_T (const std::string& iValue)
00227       : std::string (iValue) { }
00228   };
00229
00233   struct CountryName_T : public std::string {
00234   public:
00235     explicit CountryName_T (const std::string& iValue) : std::string (iValue) { }
00236   };
00237
00238
00242   struct ContinentCode_T : public std::string {
00243   public:
00244     explicit ContinentCode_T (const std::string& iValue) : std::string (iValue) {
00245     }
00246   };
00247
00251   struct ContinentName_T : public std::string {
00252   public:
00253     explicit ContinentName_T (const std::string& iValue) : std::string (iValue) {
00254     }
00255   };
00256
00260   struct Admin1Code_T : public std::string {
00261   public:
00262     explicit Admin1Code_T (const std::string& iValue) : std::string (iValue) { }
00263   };
00264
00268   struct Admin1UTFName_T : public std::string {
00269   public:
00270     explicit Admin1UTFName_T (const std::string& iValue)
00271       : std::string (iValue) { }
00272   };
00273
00277   struct Admin1ASCIIName_T : public std::string {
00278   public:
00279     explicit Admin1ASCIIName_T (const std::string& iValue)
00280       : std::string (iValue) { }
00281   };
00282
00286   struct Admin2Code_T : public std::string {
00287   public:
00288     explicit Admin2Code_T (const std::string& iValue) : std::string (iValue) { }
00289   };
00290
00294   struct Admin2UTFName_T : public std::string {
00295   public:
00296     explicit Admin2UTFName_T (const std::string& iValue)
00297       : std::string (iValue) { }
00298   };
00299
00303   struct Admin2ASCIIName_T : public std::string {
00304   public:
00305     explicit Admin2ASCIIName_T (const std::string& iValue)
00306       : std::string (iValue) { }
00307   };
00308
00312   struct Admin3Code_T : public std::string {
00313   public:
00314     explicit Admin3Code_T (const std::string& iValue) : std::string (iValue) { }
00315   };
00316
00320   struct Admin4Code_T : public std::string {
00321   public:
00322     explicit Admin4Code_T (const std::string& iValue) : std::string (iValue) { }
00323   };
00324
00328   struct FeatureClass_T : public std::string {
00329   public:
```

```
00330     explicit FeatureClass_T (const std::string& iValue) : std::string (iValue) {
00331     }
00332   };
00333
00337   struct FeatureCode_T : public std::string {
00338   public:
00339     explicit FeatureCode_T (const std::string& iValue) : std::string (iValue) { }
00340   };
00341
00345   struct FeatureName_T : public std::string {
00346   public:
00347     explicit FeatureName_T (const std::string& iValue) : std::string (iValue) { }
00348   };
00349   typedef std::list<FeatureName_T> FeatureNameList_T;
00350
00354   typedef double GeoCoord_T;
00355   typedef GeoCoord_T Latitude_T;
00356   typedef GeoCoord_T Longitude_T;
00357
00361   struct WikiLink_T : public std::string {
00362   public:
00363     explicit WikiLink_T (const std::string& iValue) : std::string (iValue) { }
00364   };
00365
00369   typedef boost::gregorian::date Date_T;
00370
00374   typedef unsigned int Population_T;
00375
00379   typedef int Elevation_T;
00380
00384   typedef int GTopo30_T;
00385
00389   struct TimeZone_T : public std::string {
00390   public:
00391     explicit TimeZone_T (const std::string& iValue) : std::string (iValue) { }
00392   };
00393
00397   typedef double PageRank_T;
00398
00402   typedef int XapianDocID_T;
00403
00407   typedef float GMTOffset_T;
00408
00412   typedef float DSTOffset_T;
00413
00417   typedef float RawOffset_T;
00418
00422   typedef bool IsGeonames_T;
00423
00427   typedef bool IsAirport_T;
00428
00432   typedef bool IsCommercial_T;
00433
00434
00438   typedef unsigned short Weight_T;
00439
00443   typedef std::string TravelQuery_T;
00444
00448   typedef double MatchingPercentage_T;
00449
00453   typedef double Percentage_T;
00454
00458   typedef double Score_T;
00459
00463   typedef unsigned int NbOfDBEntries_T;
00464
00468   typedef std::string Word_T;
00469
00473   typedef std::list<Word_T> WordList_T;
00474
00478   typedef std::set<std::string> WordSet_T;
00479
00483   typedef std::set<std::string> BlackList_T;
00484
00488   typedef unsigned int NbOfLetters_T;
00489
00493   typedef unsigned short NbOfWords_T;
00494
00498   typedef unsigned short NbOfMatches_T;
00499
00503   typedef unsigned short NbOfErrors_T;
00504
00508   typedef std::map<NbOfLetters_T, NbOfErrors_T> DistanceErrorScale_T;
00509
00513   typedef boost::array<NbOfLetters_T, 5> DistanceErrorScaleArray_T;
00514 }
00515 #endif // __OPENTREP_OPENTREP_TYPES_HPP
```

24.287 opentrep/OutputFormat.hpp File Reference

```
#include <string>
#include <opentrep/OPENTREP_Types.hpp>
```

Classes

- struct OPENTREP::OutputFormat

  Enumeration of output formats.

Namespaces

- OPENTREP

24.288 OutputFormat.hpp

```
00001 #ifndef __OPENTREP_OUTPUTFORMAT_HPP
00002 #define __OPENTREP_OUTPUTFORMAT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // OpenTrep
00010 #include <opentrep/OPENTREP_Types.hpp>
00011
00012 namespace OPENTREP {
00013
00017   struct OutputFormat {
00018   public:
00019     typedef enum {
00020       SHORT = 0,
00021       FULL,
00022       JSON,
00023       PROTOBUF,
00024       LAST_VALUE
00025     } EN_OutputFormat;
00026
00030     static const std::string& getLabel (const EN_OutputFormat&);
00031
00036     static EN_OutputFormat getFormat (const char);
00037
00041     static char getFormatLabel (const EN_OutputFormat&);
00042
00046     static std::string getFormatLabelAsString (const
      EN_OutputFormat&);
00047
00051     static std::string describeLabels();
00052
00056     EN_OutputFormat getFormat() const;
00057
00061     char getFormatAsChar() const;
00062
00066     std::string getFormatAsString() const;
00067
00072     const std::string describe() const;
00073
00074   public:
00078     bool operator== (const EN_OutputFormat&) const;
00079
00080   public:
00084     OutputFormat (const EN_OutputFormat&);
00088     OutputFormat (const char iFormat);
00092     OutputFormat (const std::string& iFormat);
00096     OutputFormat (const OutputFormat&);
00097
00098   private:
00102     OutputFormat();
00103
00104
00105   private:
00109     static const std::string _labels[LAST_VALUE];
00113     static const char _formatLabels[LAST_VALUE];
```

```
00114
00115   private:
00116     // ///////// Attributes /////////
00120     EN_OutputFormat _format;
00121   };
00122
00123 }
00124 #endif // __OPENTREP_OUTPUTFORMAT_HPP
```

24.289 opentrep/python/pyopentrep.cpp File Reference

```
#include <boost/python.hpp>
#include <cassert>
#include <stdexcept>
#include <fstream>
#include <sstream>
#include <string>
#include <list>
#include <vector>
#include <boost/filesystem.hpp>
#include <opentrep/OPENTREP_Service.hpp>
#include <opentrep/OutputFormat.hpp>
#include <opentrep/Location.hpp>
#include <opentrep/bom/BomJSONExport.hpp>
#include <opentrep/bom/LocationExchange.hpp>
```

Classes

- struct OPENTREP::OpenTrepSearcher

  API wrapper around the OpenTREP C++ API, so that Python scripts can use it seamlessly.

Namespaces

- OPENTREP

Functions

- BOOST_PYTHON_MODULE (libpyopentrep)

24.289.1 Function Documentation

24.289.1.1 BOOST_PYTHON_MODULE (libpyopentrep)

Definition at line 706 of file pyopentrep.cpp.

References OPENTREP::OpenTrepSearcher::finalize(), OPENTREP::OpenTrepSearcher::generate(), OPENTREP::OpenTrepSearcher::getPaths(), OPENTREP::OpenTrepSearcher::index(), OPENTREP::OpenTrepSearcher::init(), and OPENTREP::OpenTrepSearcher::search().

24.290 pyopentrep.cpp

```
00001 // Python
00002 #include <boost/python.hpp>
00003 // STL
00004 #include <cassert>
00005 #include <stdexcept>
00006 #include <fstream>
00007 #include <sstream>
```

```
00008 #include <string>
00009 #include <list>
00010 #include <vector>
00011 // Boost Python
00012 #include <boost/filesystem.hpp>
00013 // OpenTREP
00014 #include <opentrep/OPENTREP_Service.hpp>
00015 #include <opentrep/OutputFormat.hpp>
00016 #include <opentrep/Location.hpp>
00017 #include <opentrep/bom/BomJSONExport.hpp>
00018 #include <opentrep/bom/LocationExchange.hpp>
00019
00020 namespace OPENTREP {
00021
00026   struct OpenTrepSearcher {
00027   public:
00028
00036     std::string getPaths() {
00037       return getPathsImpl();
00038     }
00039
00043     std::string index() {
00044       return indexImpl();
00045     }
00046
00050     std::string generate (const std::string& iOutputFormatString,
00051                           const NbOfMatches_T& iNbOfDraws) {
00052       const OutputFormat lOutputFormat (iOutputFormatString);
00053       const OutputFormat::EN_OutputFormat& lOutputFormatEnum =
00054         lOutputFormat.getFormat();
00055       return generateImpl (iNbOfDraws, lOutputFormatEnum);
00056     }
00057
00061     std::string search (const std::string& iOutputFormatString,
00062                         const std::string& iTravelQuery) {
00063       const OutputFormat lOutputFormat (iOutputFormatString);
00064       const OutputFormat::EN_OutputFormat& lOutputFormatEnum =
00065         lOutputFormat.getFormat();
00066       return searchImpl (iTravelQuery, lOutputFormatEnum);
00067     }
00068
00069   private:
00073     std::string getPathsImpl() {
00074       std::ostringstream oPythonLogStr;
00075
00076       // Sanity check
00077       if (_logOutputStream == NULL) {
00078         oPythonLogStr << "The log filepath is not valid." << std::endl;
00079         return oPythonLogStr.str();
00080       }
00081       assert (_logOutputStream != NULL);
00082
00083       try {
00084
00085         // DEBUG
00086         *_logOutputStream << "Get the file-path details" << std::endl;
00087
00088         if (_opentrepService == NULL) {
00089           oPythonLogStr << "The OpenTREP service has not been initialised, "
00090                         << "i.e., the init() method has not been called "
00091                         << "correctly on the OpenTrepSearcher object. Please "
00092                         << "check that all the parameters are not empty and "
00093                         << "point to actual files.";
00094           *_logOutputStream << oPythonLogStr.str();
00095           return oPythonLogStr.str();
00096         }
00097         assert (_opentrepService != NULL);
00098
00099         // Retrieve the underlying file-path details
00100         const OPENTREP_Service::FilePathSet_T& lFilePathSet =
00101           _opentrepService->getFilePaths();
00102         const PORFilePath_T& lPORFilePath = lFilePathSet.first;
00103         const OPENTREP_Service::DBFilePathPair_T& lDBFilePathPair =
00104           lFilePathSet.second;
00105         const TravelDBFilePath_T& lTravelDBFilePath = lDBFilePathPair.first;
00106         const SQLiteDBFilePath_T& lSQLiteDBFilePath = lDBFilePathPair.second;
00107
00108         // Dump the results into the output string
00109         oPythonLogStr << lPORFilePath << ";" << lTravelDBFilePath
00110                       << ";" << lSQLiteDBFilePath;
00111
00112         // DEBUG
00113         *_logOutputStream << "ORI-maintained list of POR: '" << lPORFilePath
00114                           << "'" << std::endl;
00115         *_logOutputStream << "Xapian travel database/index: '"
00116                           << lTravelDBFilePath << "'" << std::endl;
00117         *_logOutputStream << "SQLite3 database: '"
```

```
00118                           << lSQLiteDBFilePath << "'" << std::endl;
00119
00120       } catch (const RootException& eOpenTrepError) {
00121         *_logOutputStream << "OpenTrep error: "  << eOpenTrepError.what()
00122                           << std::endl;
00123
00124       } catch (const std::exception& eStdError) {
00125         *_logOutputStream << "Error: "  << eStdError.what() << std::endl;
00126
00127       } catch (...) {
00128         *_logOutputStream << "Unknown error" << std::endl;
00129       }
00130
00131       //
00132       return oPythonLogStr.str();
00133     }
00134
00138     std::string indexImpl() {
00139       std::ostringstream oPythonLogStr;
00140
00141       // Sanity check
00142       if (_logOutputStream == NULL) {
00143         oPythonLogStr << "The log filepath is not valid." << std::endl;
00144         return oPythonLogStr.str();
00145       }
00146       assert (_logOutputStream != NULL);
00147
00148       try {
00149
00150         // DEBUG
00151         *_logOutputStream << "Indexation by Xapian" << std::endl;
00152
00153         if (_opentrepService == NULL) {
00154           oPythonLogStr << "The OpenTREP service has not been initialised, "
00155                         << "i.e., the init() method has not been called "
00156                         << "correctly on the OpenTrepSearcher object. Please "
00157                         << "check that all the parameters are not empty and "
00158                         << "point to actual files.";
00159           *_logOutputStream << oPythonLogStr.str();
00160           return oPythonLogStr.str();
00161         }
00162         assert (_opentrepService != NULL);
00163
00164         // Retrieve the underlying file-path details
00165         const OPENTREP_Service::FilePathSet_T& lFilePathSet =
00166           _opentrepService->getFilePaths();
00167         const PORFilePath_T& lPORFilePath = lFilePathSet.first;
00168         const OPENTREP_Service::DBFilePathPair_T& lDBFilePathPair =
00169           lFilePathSet.second;
00170         const TravelDBFilePath_T& lTravelDBFilePath = lDBFilePathPair.first;
00171         const SQLiteDBFilePath_T& lSQLiteDBFilePath = lDBFilePathPair.second;
00172
00173
00174         // DEBUG
00175         *_logOutputStream << "ORI-maintained list of POR: '" << lPORFilePath
00176                           << "'" << std::endl;
00177         *_logOutputStream << "Xapian travel database/index: '"
00178                           << lTravelDBFilePath << "'" << std::endl;
00179         *_logOutputStream << "SQLite3 database: '"
00180                           << lSQLiteDBFilePath << "'" << std::endl;
00181
00182         // Launch the indexation by Xapian of the ORI-maintained list of POR
00183         const NbOfDBEntries_T lNbOfEntries= _opentrepService->
     buildSearchIndex();
00184
00185         // Dump the results into the output string
00186         oPythonLogStr << lNbOfEntries;
00187
00188         // DEBUG
00189         *_logOutputStream << "Xapian indexation yielded " << lNbOfEntries
00190                           << " POR (points of reference) entries." << std::endl;
00191
00192       } catch (const RootException& eOpenTrepError) {
00193         *_logOutputStream << "OpenTrep error: "  << eOpenTrepError.what()
00194                           << std::endl;
00195
00196       } catch (const std::exception& eStdError) {
00197         *_logOutputStream << "Error: "  << eStdError.what() << std::endl;
00198
00199       } catch (...) {
00200         *_logOutputStream << "Unknown error" << std::endl;
00201       }
00202
00203       //
00204       return oPythonLogStr.str();
00205     }
00206
```

```
00210    std::string searchImpl(const std::string& iTravelQuery,
00211                           const OutputFormat::EN_OutputFormat& iOutputFormat)
     {
00212      std::ostringstream oNoDetailedStr;
00213      std::ostringstream oDetailedStr;
00214      std::ostringstream oJSONStr;
00215      std::ostringstream oProtobufStr;
00216
00217      // Sanity check
00218      if (_logOutputStream == NULL) {
00219        oNoDetailedStr << "The log filepath is not valid." << std::endl;
00220        return oNoDetailedStr.str();
00221      }
00222      assert (_logOutputStream != NULL);
00223
00224      try {
00225
00226        // DEBUG
00227        *_logOutputStream << "Travel query (’" << iTravelQuery << "’"
00228                          << "’) search" << std::endl;
00229
00230        if (_opentrepService == NULL) {
00231          oNoDetailedStr << "The OpenTREP service has not been initialised, "
00232                         << "i.e., the init() method has not been called "
00233                         << "correctly on the OpenTrepSearcher object. Please "
00234                         << "check that all the parameters are not empty and "
00235                         << "point to actual files.";
00236          *_logOutputStream << oNoDetailedStr.str();
00237          return oNoDetailedStr.str();
00238        }
00239        assert (_opentrepService != NULL);
00240
00241        // Retrieve the underlying file-path details
00242        const OPENTREP_Service::FilePathSet_T& lFilePathSet =
00243          _opentrepService->getFilePaths();
00244        const PORFilePath_T& lPORFilePath = lFilePathSet.first;
00245        const OPENTREP_Service::DBFilePathPair_T& lDBFilePathPair =
00246          lFilePathSet.second;
00247        const TravelDBFilePath_T& lTravelDBFilePath = lDBFilePathPair.first;
00248        const SQLiteDBFilePath_T& lSQLiteDBFilePath = lDBFilePathPair.second;
00249
00250        // DEBUG
00251        *_logOutputStream << "Xapian travel database/index: ’"
00252                          << lTravelDBFilePath
00253                          << "’ - SQLite3 database: ’"
00254                          << lSQLiteDBFilePath
00255                          << "’ - ORI-maintained list of POR: ’"
00256                          << lPORFilePath << "’" << std::endl;
00257
00258        // Query the Xapian database (index)
00259        WordList_T lNonMatchedWordList;
00260        LocationList_T lLocationList;
00261        const NbOfMatches_T nbOfMatches =
00262          _opentrepService->interpretTravelRequest (iTravelQuery, lLocationList,
00263                                                    lNonMatchedWordList);
00264
00265        // DEBUG
00266        *_logOutputStream << "Python search for ’" << iTravelQuery << "’ gave "
00267                          << nbOfMatches << " matches." << std::endl;
00268
00269        if (nbOfMatches != 0) {
00270          NbOfMatches_T idx = 0;
00271
00272          for(LocationList_T::const_iterator itLocation = lLocationList.begin();
00273              itLocation != lLocationList.end(); ++itLocation, ++idx) {
00274            const Location& lLocation = *itLocation;
00275
00276            if (idx != 0) {
00277              oNoDetailedStr << ",";
00278            }
00279
00280            //
00281            oNoDetailedStr << lLocation.getIataCode();
00282            oDetailedStr << idx+1 << ". " << lLocation.toSingleLocationString()
00283                         << std::endl;
00284
00285            // List of extra matching locations (those with the same
00286            // matching weight/percentage)
00287            const LocationList_T& lExtraLocationList =
00288              lLocation.getExtraLocationList();
00289            if (lExtraLocationList.empty() == false) {
00290              oDetailedStr << "  Extra matches: " << std::endl;
00291
00292              NbOfMatches_T idxExtra = 0;
00293              for (LocationList_T::const_iterator itLoc =
00294                     lExtraLocationList.begin();
00295                   itLoc != lExtraLocationList.end(); ++itLoc, ++idxExtra) {
```

```
00296                 oNoDetailedStr << ":";
00297                 oDetailedStr << "    " << idx+1 << "." << idxExtra+1 << ". ";
00298
00299                 const Location& lExtraLocation = *itLoc;
00300                 oNoDetailedStr << lExtraLocation.getIataCode();
00301                 oDetailedStr << lExtraLocation << std::endl;
00302               }
00303             }
00304
00305             // The matching weight/percentage is the same for the main
00306             // and the extra matching locations
00307             oNoDetailedStr << "/" << lLocation.getPercentage();
00308
00309             // List of alternate matching locations (those with a lower
00310             // matching weight/percentage)
00311             const LocationList_T& lAlternateLocationList =
00312               lLocation.getAlternateLocationList();
00313             if (lAlternateLocationList.empty() == false) {
00314               oDetailedStr << "  Alternate matches: " << std::endl;
00315
00316               NbOfMatches_T idxAlter = 0;
00317               for (LocationList_T::const_iterator itLoc =
00318                      lAlternateLocationList.begin();
00319                    itLoc != lAlternateLocationList.end(); ++itLoc, ++idxAlter) {
00320                 oNoDetailedStr << "-";
00321                 oDetailedStr << "    " << idx+1 << "." << idxAlter+1 << ". ";
00322
00323                 const Location& lAlternateLocation = *itLoc;
00324                 oNoDetailedStr << lAlternateLocation.getIataCode()
00325                                << "/" << lAlternateLocation.getPercentage();
00326                 oDetailedStr << lAlternateLocation << std::endl;
00327               }
00328             }
00329           }
00330         }
00331
00332         if (lNonMatchedWordList.empty() == false) {
00333           oNoDetailedStr << ";";
00334           oDetailedStr << "Not recognised words:" << std::endl;
00335           NbOfMatches_T idx = 0;
00336           for (WordList_T::const_iterator itWord = lNonMatchedWordList.begin();
00337                itWord != lNonMatchedWordList.end(); ++itWord, ++idx) {
00338             const Word_T& lWord = *itWord;
00339             if (idx != 0) {
00340               oNoDetailedStr << ",";
00341               oDetailedStr << idx+1 << "." << std::endl;
00342             }
00343             oNoDetailedStr << lWord;
00344             oDetailedStr << lWord;
00345           }
00346         }
00347
00348         // DEBUG
00349         *_logOutputStream << "Python search for '" << iTravelQuery
00350                           << "' yielded:" << std::endl;
00351
00352         // Export the list of Location objects into a JSON-formatted string
00353         BomJSONExport::jsonExportLocationList (oJSONStr, lLocationList
     );
00354
00355         // Export the list of Location objects into a Protobuf-formatted string
00356         LocationExchange::exportLocationList (oProtobufStr,
     lLocationList,
00357                                               lNonMatchedWordList);
00358
00359         // DEBUG
00360         *_logOutputStream << "Short version: "
00361                           << oNoDetailedStr.str() << std::endl;
00362         *_logOutputStream << "Long version: "
00363                           << oDetailedStr.str() << std::endl;
00364         *_logOutputStream << "JSON version: "
00365                           << oJSONStr.str() << std::endl;
00366         *_logOutputStream << "Protobuf version: "
00367                           << oProtobufStr.str() << std::endl;
00368
00369       } catch (const RootException& eOpenTrepError) {
00370         *_logOutputStream << "OpenTrep error: "  << eOpenTrepError.what()
00371                           << std::endl;
00372
00373       } catch (const std::exception& eStdError) {
00374         *_logOutputStream << "Error: "  << eStdError.what() << std::endl;
00375
00376       } catch (...) {
00377         *_logOutputStream << "Unknown error" << std::endl;
00378       }
00379
00380       // Return the string corresponding to the request (either with
```

```
00381       // or without details).
00382       switch (iOutputFormat) {
00383       case OutputFormat::SHORT: {
00384         return oNoDetailedStr.str();
00385       }
00386
00387       case OutputFormat::FULL: {
00388         return oDetailedStr.str();
00389       }
00390
00391       case OutputFormat::JSON: {
00392         return oJSONStr.str();
00393       }
00394
00395       case OutputFormat::PROTOBUF: {
00396         return oProtobufStr.str();
00397       }
00398
00399       default: {
00400         // If the output format is not known, an exception is thrown by
00401         // the call to the OutputFormat() constructor above.
00402         assert (false);
00403       }
00404       }
00405     }
00406
00410     std::string generateImpl(const NbOfMatches_T& iNbOfDraws,
00411                              const OutputFormat::EN_OutputFormat&
      iOutputFormat){
00412       std::ostringstream oNoDetailedStr;
00413       std::ostringstream oDetailedStr;
00414       std::ostringstream oJSONStr;
00415       std::ostringstream oProtobufStr;
00416
00417       // Sanity check
00418       if (_logOutputStream == NULL) {
00419         oNoDetailedStr << "The log filepath is not valid." << std::endl;
00420         return oNoDetailedStr.str();
00421       }
00422       assert (_logOutputStream != NULL);
00423
00424       try {
00425
00426         // DEBUG
00427         *_logOutputStream << "Number of random draws: " << iNbOfDraws
00428                           << std::endl;
00429
00430         if (_opentrepService == NULL) {
00431           oNoDetailedStr << "The OpenTREP service has not been initialised, "
00432                          << "i.e., the init() method has not been called "
00433                          << "correctly on the OpenTrepSearcher object. Please "
00434                          << "check that all the parameters are not empty and "
00435                          << "point to actual files.";
00436           *_logOutputStream << oNoDetailedStr.str();
00437           return oNoDetailedStr.str();
00438         }
00439         assert (_opentrepService != NULL);
00440
00441         // Retrieve the underlying file-path details
00442         const OPENTREP_Service::FilePathSet_T& lFilePathSet =
00443           _opentrepService->getFilePaths();
00444         const PORFilePath_T& lPORFilePath = lFilePathSet.first;
00445         const OPENTREP_Service::DBFilePathPair_T& lDBFilePathPair =
00446           lFilePathSet.second;
00447         const TravelDBFilePath_T& lTravelDBFilePath = lDBFilePathPair.first;
00448         const SQLiteDBFilePath_T& lSQLiteDBFilePath = lDBFilePathPair.second;
00449
00450         // DEBUG
00451         *_logOutputStream << "Xapian travel database/index: '"
00452                           << lTravelDBFilePath
00453                           << "' - SQLite3 database: '"
00454                           << lSQLiteDBFilePath
00455                           << "' - ORI-maintained list of POR: '"
00456                           << lPORFilePath << "'" << std::endl;
00457
00458         // Query the Xapian database (index)
00459         LocationList_T lLocationList;
00460         const NbOfMatches_T nbOfMatches =
00461           _opentrepService->drawRandomLocations (iNbOfDraws, lLocationList);
00462
00463         // DEBUG
00464         *_logOutputStream << "Python generation of " << iNbOfDraws << " gave "
00465                           << nbOfMatches << " documents." << std::endl;
00466
00467         if (nbOfMatches != 0) {
00468           NbOfMatches_T idx = 0;
00469
```

```
00470           for(LocationList_T::const_iterator itLocation = lLocationList.begin();
00471               itLocation != lLocationList.end(); ++itLocation, ++idx) {
00472             const Location& lLocation = *itLocation;
00473
00474             if (idx != 0) {
00475               oNoDetailedStr << ",";
00476             }
00477
00478             //
00479             oNoDetailedStr << lLocation.getIataCode();
00480             oDetailedStr << idx+1 << ". " << lLocation.toSingleLocationString()
00481                          << std::endl;
00482
00483             // List of extra matching locations (those with the same
00484             // matching weight/percentage)
00485             const LocationList_T& lExtraLocationList =
00486               lLocation.getExtraLocationList();
00487             if (lExtraLocationList.empty() == false) {
00488               oDetailedStr << "  Extra matches: " << std::endl;
00489
00490               NbOfMatches_T idxExtra = 0;
00491               for (LocationList_T::const_iterator itLoc =
00492                      lExtraLocationList.begin();
00493                    itLoc != lExtraLocationList.end(); ++itLoc, ++idxExtra) {
00494                 oNoDetailedStr << ":";
00495                 oDetailedStr << "    " << idx+1 << "." << idxExtra+1 << ". ";
00496
00497                 const Location& lExtraLocation = *itLoc;
00498                 oNoDetailedStr << lExtraLocation.getIataCode();
00499                 oDetailedStr << lExtraLocation << std::endl;
00500               }
00501             }
00502
00503             // The matching weight/percentage is the same for the main
00504             // and the extra matching locations
00505             oNoDetailedStr << "/" << lLocation.getPercentage();
00506
00507             // List of alternate matching locations (those with a lower
00508             // matching weight/percentage)
00509             const LocationList_T& lAlternateLocationList =
00510               lLocation.getAlternateLocationList();
00511             if (lAlternateLocationList.empty() == false) {
00512               oDetailedStr << "  Alternate matches: " << std::endl;
00513
00514               NbOfMatches_T idxAlter = 0;
00515               for (LocationList_T::const_iterator itLoc =
00516                      lAlternateLocationList.begin();
00517                    itLoc != lAlternateLocationList.end(); ++itLoc, ++idxAlter) {
00518                 oNoDetailedStr << "-";
00519                 oDetailedStr << "    " << idx+1 << "." << idxAlter+1 << ". ";
00520
00521                 const Location& lAlternateLocation = *itLoc;
00522                 oNoDetailedStr << lAlternateLocation.getIataCode()
00523                                << "/" << lAlternateLocation.getPercentage();
00524                 oDetailedStr << lAlternateLocation << std::endl;
00525               }
00526             }
00527           }
00528         }
00529
00530         // DEBUG
00531         *_logOutputStream << "Python generation of " << iNbOfDraws
00532                           << " yielded:" << std::endl;
00533
00534         // Export the list of Location objects into a JSON-formatted string
00535         BomJSONExport::jsonExportLocationList (oJSONStr, lLocationList
      );
00536
00537         // Export the list of Location objects into a Protobuf-formatted string
00538         WordList_T lNonMatchedWordList;
00539         LocationExchange::exportLocationList (oProtobufStr,
      lLocationList,
00540                                               lNonMatchedWordList);
00541
00542         // DEBUG
00543         *_logOutputStream << "Short version: "
00544                           << oNoDetailedStr.str() << std::endl;
00545         *_logOutputStream << "Long version: "
00546                           << oDetailedStr.str() << std::endl;
00547         *_logOutputStream << "JSON version: "
00548                           << oJSONStr.str() << std::endl;
00549         *_logOutputStream << "Protobuf version: "
00550                           << oProtobufStr.str() << std::endl;
00551
00552       } catch (const RootException& eOpenTrepError) {
00553         *_logOutputStream << "OpenTrep error: "  << eOpenTrepError.what()
00554                           << std::endl;
```

```
00555
00556       } catch (const std::exception& eStdError) {
00557         *_logOutputStream << "Error: "  << eStdError.what() << std::endl;
00558
00559       } catch (...) {
00560         *_logOutputStream << "Unknown error" << std::endl;
00561       }
00562
00563       // Return the string corresponding to the request (either with
00564       // or without details).
00565       switch (iOutputFormat) {
00566       case OutputFormat::SHORT: {
00567         return oNoDetailedStr.str();
00568       }
00569
00570       case OutputFormat::FULL: {
00571         return oDetailedStr.str();
00572       }
00573
00574       case OutputFormat::JSON: {
00575         return oJSONStr.str();
00576       }
00577
00578       case OutputFormat::PROTOBUF: {
00579         return oProtobufStr.str();
00580       }
00581
00582       default: {
00583         // If the output format is not known, an exception is thrown by
00584         // the call to the OutputFormat() constructor above.
00585         assert (false);
00586       }
00587       }
00588     }
00589
00590   public:
00594     OpenTrepSearcher() : _opentrepService (NULL), _logOutputStream (NULL) {
00595     }
00596
00600     OpenTrepSearcher (const OpenTrepSearcher& iOpenTrepSearcher)
00601       : _opentrepService (iOpenTrepSearcher._opentrepService),
00602         _logOutputStream (iOpenTrepSearcher._logOutputStream) {
00603     }
00604
00608     ~OpenTrepSearcher() {
00609       _opentrepService = NULL;
00610       _logOutputStream = NULL;
00611     }
00612
00616     bool init (const std::string& iTravelDBFilePath,
00617                const std::string& iSQLiteDBFilePath,
00618                const std::string& iLogFilePath) {
00619       bool isEverythingOK = true;
00620
00621       try {
00622
00623         // Check that the file-path exist and are accessible
00624         boost::filesystem::path lXapianFilePath (iTravelDBFilePath.begin(),
00625                                                  iTravelDBFilePath.end());
00626         if (!(boost::filesystem::exists (lXapianFilePath)
00627               && boost::filesystem::is_directory (lXapianFilePath))) {
00628           isEverythingOK = false;
00629           return isEverythingOK;
00630         }
00631
00632         // Set the log parameters
00633         _logOutputStream = new std::ofstream;
00634         assert (_logOutputStream != NULL);
00635
00636         // Open and clean the log outputfile
00637         _logOutputStream->open (iLogFilePath.c_str());
00638         _logOutputStream->clear();
00639
00640         // DEBUG
00641         *_logOutputStream << "Python wrapper initialisation" << std::endl;
00642
00643         // Initialise the context
00644         const OPENTREP::TravelDBFilePath_T lTravelDBFilePath (iTravelDBFilePath
      );
00645         const OPENTREP::SQLiteDBFilePath_T lSQLiteDBFilePath (iSQLiteDBFilePath
      );
00646
00647         _opentrepService = new OPENTREP_Service (*_logOutputStream,
00648                                                  lTravelDBFilePath,
00649                                                  lSQLiteDBFilePath);
00650
00651         // DEBUG
```

```
00652         *_logOutputStream << "Python wrapper initialised" << std::endl;
00653
00654       } catch (const RootException& eOpenTrepError) {
00655         *_logOutputStream << "OpenTrep error: "  << eOpenTrepError.what()
00656                           << std::endl;
00657
00658       } catch (const std::exception& eStdError) {
00659         *_logOutputStream << "Error: "  << eStdError.what() << std::endl;
00660
00661       } catch (...) {
00662         *_logOutputStream << "Unknown error" << std::endl;
00663       }
00664
00665       return isEverythingOK;
00666     }
00667
00671     bool finalize () {
00672       bool isEverythingOK = true;
00673
00674       try {
00675
00676         // Finalize the context
00677         if (_opentrepService != NULL) {
00678           delete _opentrepService; _opentrepService = NULL;
00679         }
00680
00681         // Close the output stream
00682         if (_logOutputStream != NULL) {
00683           // DEBUG
00684           *_logOutputStream << "Python wrapper finalization" << std::endl;
00685           _logOutputStream->close();
00686           delete _logOutputStream; _logOutputStream = NULL;
00687         }
00688
00689       } catch (...) {
00690       }
00691
00692       return isEverythingOK;
00693     }
00694
00695   private:
00699     OPENTREP_Service* _opentrepService;
00700     std::ofstream* _logOutputStream;
00701   };
00702
00703 }
00704
00705 // ////////////////////////////////////////////////////////////////
00706 BOOST_PYTHON_MODULE(libpyopentrep) {
00707   boost::python::class_<OPENTREP::OpenTrepSearcher> ("OpenTrepSearcher")
00708     .def ("index", &OPENTREP::OpenTrepSearcher::index)
00709     .def ("search", &OPENTREP::OpenTrepSearcher::search)
00710     .def ("generate", &OPENTREP::OpenTrepSearcher::generate)
00711     .def ("getPaths", &OPENTREP::OpenTrepSearcher::getPaths)
00712     .def ("init", &OPENTREP::OpenTrepSearcher::init)
00713     .def ("finalize", &OPENTREP::OpenTrepSearcher::finalize);
00714 }
```

24.291 opentrep/service/Logger.cpp File Reference

```
#include <assert.h>
#include <iostream>
#include <opentrep/factory/FacSupervisor.hpp>
#include <opentrep/service/Logger.hpp>
```

Namespaces

- OPENTREP

24.292 Logger.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // C
```

```
00005 #include <assert.h>
00006 // STL
00007 #include <iostream>
00008 // Opentrep Logger
00009 #include <opentrep/factory/FacSupervisor.hpp>
00010 #include <opentrep/service/Logger.hpp>
00011
00012 namespace OPENTREP {
00013
00014     Logger* Logger::_instance = NULL;
00015
00016     // //////////////////////////////////////////////////////////////////
00017     Logger::Logger () : _logStream (&std::cout) {
00018       assert (false);
00019     }
00020
00021     // //////////////////////////////////////////////////////////////////
00022     Logger::Logger (const Logger&) : _logStream (&std::cout) {
00023       assert (false);
00024     }
00025
00026     // //////////////////////////////////////////////////////////////////
00027     Logger::Logger (const LOG::EN_LogLevel iLevel, std::ostream& ioLogStream)
00028       : _level (iLevel), _logStream (&ioLogStream) {
00029     }
00030
00031     // //////////////////////////////////////////////////////////////////
00032     Logger::~Logger () {
00033       _logStream = NULL;
00034     }
00035
00036     // //////////////////////////////////////////////////////////////////
00037     LOG::EN_LogLevel Logger::getLogLevel() {
00038       return _level;
00039     }
00040
00041     // //////////////////////////////////////////////////////////////////
00042     std::ostream& Logger::getLogStream() {
00043       assert (_logStream != NULL);
00044       return *_logStream;
00045     }
00046
00047     // //////////////////////////////////////////////////////////////////
00048     void Logger::setLogParameters (const LOG::EN_LogLevel iLogLevel,
00049                                    std::ostream& ioLogStream) {
00050       _level = iLogLevel;
00051       _logStream = &ioLogStream;
00052     }
00053
00054     // //////////////////////////////////////////////////////////////////
00055     Logger& Logger::instance() {
00056       if (_instance == NULL) {
00057         _instance = new Logger (LOG::DEBUG, std::cout);
00058
00059         assert (_instance != NULL);
00060
00061         FacSupervisor::instance().registerLoggerService (_instance);
00062       }
00063       return *_instance;
00064     }
00065
00066 }
```

24.293 opentrep/service/Logger.hpp File Reference

```
#include <assert.h>
#include <sstream>
#include <string>
#include <opentrep/OPENTREP_Types.hpp>
```

Classes

- class OPENTREP::Logger

Namespaces

- OPENTREP

Macros

- #define OPENTREP_LOG_CORE(iLevel, iToBeLogged)
- #define OPENTREP_LOG_CRITICAL(iToBeLogged) OPENTREP_LOG_CORE (OPENTREP::LOG::CRITICAL, iToBeLogged)
- #define OPENTREP_LOG_ERROR(iToBeLogged) OPENTREP_LOG_CORE (OPENTREP::LOG::ERROR, iToBeLogged)
- #define OPENTREP_LOG_NOTIFICATION(iToBeLogged) OPENTREP_LOG_CORE (OPENTREP::LOG::NOTIFICATION, iToBeLogged)
- #define OPENTREP_LOG_WARNING(iToBeLogged) OPENTREP_LOG_CORE (OPENTREP::LOG::WARNING, iToBeLogged)
- #define OPENTREP_LOG_DEBUG(iToBeLogged) OPENTREP_LOG_CORE (OPENTREP::LOG::DEBUG, iToBeLogged)
- #define OPENTREP_LOG_VERBOSE(iToBeLogged) OPENTREP_LOG_CORE (OPENTREP::LOG::VERBOSE, iToBeLogged)

24.293.1 Macro Definition Documentation

24.293.1.1 #define OPENTREP_LOG_CORE(*iLevel, iToBeLogged*)

Value:

```
{ std::ostringstream ostr; ostr << iToBeLogged; \
    OPENTREP::Logger::instance().log (iLevel, __LINE__, __FILE__, ostr.str
      ()); }
```

Definition at line 16 of file Logger.hpp.

24.293.1.2 #define OPENTREP_LOG_CRITICAL(*iToBeLogged*) OPENTREP_LOG_CORE (OPENTREP::LOG::CRITICAL, iToBeLogged)

Definition at line 20 of file Logger.hpp.

24.293.1.3 #define OPENTREP_LOG_ERROR(*iToBeLogged*) OPENTREP_LOG_CORE (OPENTREP::LOG::ERROR, iToBeLogged)

Definition at line 23 of file Logger.hpp.

Referenced by OPENTREP::NameMatrix::addName(), OPENTREP::DBManager::buildSQLDB(), OPENTREP::check(), OPENTREP::checkTravelDBFilePath(), OPENTREP::Place::completeLocation(), OPENTREP::doesMatch(), OPENTREP::Result::fullTextMatch(), OPENTREP::PORStringParser::generateLocation(), OPENTREP::PORFileParser::generateLocations(), OPENTREP::ResultCombination::getBestMatchingResultHolder(), OPENTREP::Result::getDocumentPair(), OPENTREP::OPENTREP_Service::interpretTravelRequest(), OPENTREP::ServiceUtilities::isDirectory(), OPENTREP::DBManager::iterateOnStatement(), OPENTREP::FacWorld::linkPlaceToWorld(), OPENTREP::PORParser::PORGeneration(), OPENTREP::DBManager::prepareSelectAllStatement(), OPENTREP::DBManager::prepareSelectFromCodeStatement(), OPENTREP::DBManager::prepareSelectFromCoordStatement(), OPENTREP::DBManager::retrieveClosestPlaces(), OPENTREP::DBManager::retrievePlace(), OPENTREP::searchString(), OPENTREP::ScoreBoard::setScore(), OPENTREP::Result::setScoreOnDocMap(), and OPENTREP::DBManager::updatePlaceInDB().

24.293.1.4 #define OPENTREP_LOG_NOTIFICATION(*iToBeLogged*) OPENTREP_LOG_CORE (OPENTREP::LOG::NOTIFICATION, iToBeLogged)

Definition at line 26 of file Logger.hpp.

Referenced by OPENTREP::Result::calculateCodeMatches(), OPENTREP::Result::calculateEnvelopeWeights(), OPENTREP::Result::calculatePageRanks(), and OPENTREP::Result::displayXapianPercentages().

24.293.1.5 #define OPENTREP_LOG_WARNING(*iToBeLogged*) OPENTREP_LOG_CORE (OPENTREP::LOG::WARNING, iToBeLogged)

Definition at line 29 of file Logger.hpp.

24.293.1.6 #define OPENTREP_LOG_DEBUG(*iToBeLogged*) OPENTREP_LOG_CORE (OPENTREP::LOG::DEBUG, iToBeLogged)

Definition at line 32 of file Logger.hpp.

Referenced by OPENTREP::addToXapian(), OPENTREP::OPENTREP_Service::buildSearchIndex(), OPENTREP::DBManager::buildSQLDB(), OPENTREP::OPENTREP_Service::buildSQLDB(), OPENTREP::ResultHolder::calculateCombinedWeights(), OPENTREP::Result::calculateCombinedWeights(), OPENTREP::ResultCombination::chooseBestMatchingResultHolder(), OPENTREP::chooseBestMatchingResultHolder(), OPENTREP::createPlaces(), OPENTREP::OPENTREP_Service::drawRandomLocations(), OPENTREP::Result::fillPlace(), OPENTREP::Result::fullTextMatch(), OPENTREP::PORFileParser::generateLocations(), OPENTREP::OPENTREP_Service::getIndexSize(), OPENTREP::OPENTREP_Service::interpretTravelRequest(), OPENTREP::DBManager::retrievePlace(), and OPENTREP::searchString().

24.293.1.7 #define OPENTREP_LOG_VERBOSE(*iToBeLogged*) OPENTREP_LOG_CORE (OPENTREP::LOG::VERBOSE, iToBeLogged)

Definition at line 35 of file Logger.hpp.

24.294 Logger.hpp

```
00001 #ifndef __OPENTREP_SVC_LOGGER_HPP
00002 #define __OPENTREP_SVC_LOGGER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // C
00008 #include <assert.h>
00009 // STL
00010 #include <sstream>
00011 #include <string>
00012 // OPENTREP
00013 #include <opentrep/OPENTREP_Types.hpp>
00014
00015 // ///////////////// LOG MACROS /////////////////
00016 #define OPENTREP_LOG_CORE(iLevel, iToBeLogged) \
00017   { std::ostringstream ostr; ostr << iToBeLogged; \
00018     OPENTREP::Logger::instance().log (iLevel, __LINE__, __FILE__, ostr.str()); }
00019
00020 #define OPENTREP_LOG_CRITICAL(iToBeLogged) \
00021   OPENTREP_LOG_CORE (OPENTREP::LOG::CRITICAL, iToBeLogged)
00022
00023 #define OPENTREP_LOG_ERROR(iToBeLogged) \
00024   OPENTREP_LOG_CORE (OPENTREP::LOG::ERROR, iToBeLogged)
00025
00026 #define OPENTREP_LOG_NOTIFICATION(iToBeLogged) \
00027   OPENTREP_LOG_CORE (OPENTREP::LOG::NOTIFICATION, iToBeLogged)
00028
00029 #define OPENTREP_LOG_WARNING(iToBeLogged) \
00030   OPENTREP_LOG_CORE (OPENTREP::LOG::WARNING, iToBeLogged)
00031
00032 #define OPENTREP_LOG_DEBUG(iToBeLogged) \
00033   OPENTREP_LOG_CORE (OPENTREP::LOG::DEBUG, iToBeLogged)
00034
00035 #define OPENTREP_LOG_VERBOSE(iToBeLogged) \
00036   OPENTREP_LOG_CORE (OPENTREP::LOG::VERBOSE, iToBeLogged)
00037 // ////////// (END OF) LOG MACROS //////////
00038
00039
00040 namespace OPENTREP {
00041
00045   class Logger {
00046     // Friend classes
00047     friend class FacSupervisor;
00048   public:
```

```
00049
00051     template <typename T>
00052     void log (const LOG::EN_LogLevel iLevel, const int iLineNumber,
00053               const std::string& iFileName, const T& iToBeLogged) {
00054       if (iLevel <= _level) {
00055         assert (_logStream != NULL);
00056         *_logStream << iFileName << ":" << iLineNumber
00057                     << ": " << iToBeLogged << std::endl;
00058       }
00059     }
00060
00062     LOG::EN_LogLevel getLogLevel();
00063
00065     std::ostream& getLogStream();
00066
00068     void setLogParameters (const LOG::EN_LogLevel iLogLevel,
00069                            std::ostream& ioLogStream);
00070
00072     static Logger& instance();
00073
00074   private:
00077     Logger ();
00078     Logger (const Logger&);
00079     Logger (const LOG::EN_LogLevel iLevel, std::ostream& ioLogStream);
00081     ~Logger ();
00082
00083   private:
00085     LOG::EN_LogLevel _level;
00086
00088     std::ostream* _logStream;
00089
00091     static Logger* _instance;
00092   };
00093
00094 }
00095 #endif // __OPENTREP_SVC_LOGGER_HPP
```

24.295 opentrep/service/OPENTREP_Service.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/ptime.hpp>
#include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
#include <opentrep/basic/BasChronometer.hpp>
#include <opentrep/factory/FacWorld.hpp>
#include <opentrep/command/DBManager.hpp>
#include <opentrep/command/IndexBuilder.hpp>
#include <opentrep/command/XapianIndexManager.hpp>
#include <opentrep/command/RequestInterpreter.hpp>
#include <opentrep/factory/FacOpenTrepServiceContext.hpp>
#include <opentrep/service/OPENTREP_ServiceContext.hpp>
#include <opentrep/service/ServiceUtilities.hpp>
#include <opentrep/service/Logger.hpp>
#include <opentrep/OPENTREP_Service.hpp>
```

Namespaces

- OPENTREP

Functions

- void OPENTREP::logInit (const LOG::EN_LogLevel iLogLevel, std::ostream &ioLogOutputFile)

24.296 OPENTREP_Service.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <ostream>
00007 // Boost
00008 #include <boost/date_time/gregorian/gregorian.hpp>
00009 #include <boost/date_time/posix_time/ptime.hpp>
00010 // OpenTrep
00011 #include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
00012 #include <opentrep/basic/BasChronometer.hpp>
00013 #include <opentrep/factory/FacWorld.hpp>
00014 #include <opentrep/command/DBManager.hpp>
00015 #include <opentrep/command/IndexBuilder.hpp>
00016 #include <opentrep/command/XapianIndexManager.hpp>
00017 #include <opentrep/command/RequestInterpreter.hpp>
00018 #include <opentrep/factory/FacOpenTrepServiceContext.hpp>
00019 #include <opentrep/service/OPENTREP_ServiceContext.hpp>
00020 #include <opentrep/service/ServiceUtilities.hpp>
00021 #include <opentrep/service/Logger.hpp>
00022 #include <opentrep/OPENTREP_Service.hpp>
00023
00024 namespace OPENTREP {
00025
00026   // //////////////////////////////////////////////////////////////////////
00027   OPENTREP_Service::
00028   OPENTREP_Service (std::ostream& ioLogStream, const PORFilePath_T& iPORFilepath,
00029                     const TravelDBFilePath_T& iTravelDBFilePath,
00030                     const SQLiteDBFilePath_T& iSQLiteDBFilePath)
00031     : _opentrepServiceContext (NULL) {
00032     init (ioLogStream, iPORFilepath, iTravelDBFilePath, iSQLiteDBFilePath);
00033   }
00034
00035   // //////////////////////////////////////////////////////////////////////
00036   OPENTREP_Service::
00037   OPENTREP_Service (std::ostream& ioLogStream,
00038                     const TravelDBFilePath_T& iTravelDBFilePath,
00039                     const SQLiteDBFilePath_T& iSQLiteDBFilePath)
00040     : _opentrepServiceContext (NULL) {
00041     init (ioLogStream, iTravelDBFilePath, iSQLiteDBFilePath);
00042   }
00043
00044   // //////////////////////////////////////////////////////////////////////
00045   OPENTREP_Service::OPENTREP_Service() : _opentrepServiceContext (NULL) {
00046     assert (false);
00047   }
00048
00049   // //////////////////////////////////////////////////////////////////////
00050   OPENTREP_Service::OPENTREP_Service (const OPENTREP_Service& iService) {
00051     assert (false);
00052   }
00053
00054   // //////////////////////////////////////////////////////////////////////
00055   OPENTREP_Service::~OPENTREP_Service() {
00056     // Delete/Clean all the objects from memory
00057     finalise();
00058   }
00059
00060   // //////////////////////////////////////////////////////////////////////
00061   void logInit (const LOG::EN_LogLevel iLogLevel,
00062                 std::ostream& ioLogOutputFile) {
00063     Logger::instance().setLogParameters (iLogLevel, ioLogOutputFile);
00064   }
00065
00066   // //////////////////////////////////////////////////////////////////////
00067   void OPENTREP_Service::init (std::ostream& ioLogStream,
00068                                const TravelDBFilePath_T& iTravelDBFilePath,
00069                                const SQLiteDBFilePath_T& iSQLiteDBFilePath) {
00070     // Set the log file
00071     logInit (LOG::DEBUG, ioLogStream);
00072
00073     // Initialise the context
00074     OPENTREP_ServiceContext& lOPENTREP_ServiceContext =
00075       FacOpenTrepServiceContext::instance().
      create (iTravelDBFilePath, iSQLiteDBFilePath);
00076     _opentrepServiceContext = &lOPENTREP_ServiceContext;
00077
00078     // Instanciate an empty World object
00079     World& lWorld = FacWorld::instance().create();
00080     lOPENTREP_ServiceContext.setWorld (lWorld);
00081   }
00082
00083   // //////////////////////////////////////////////////////////////////////
00084   void OPENTREP_Service::init (std::ostream& ioLogStream,
```

```
00085                                const PORFilePath_T& iPORFilepath,
00086                                const TravelDBFilePath_T& iTravelDBFilePath,
00087                                const SQLiteDBFilePath_T& iSQLiteDBFilePath) {
00088     // Set the log file
00089     logInit (LOG::DEBUG, ioLogStream);
00090
00091     // Initialise the context
00092     OPENTREP_ServiceContext& lOPENTREP_ServiceContext =
00093       FacOpenTrepServiceContext::instance().
      create (iPORFilepath,
00094                                                 iTravelDBFilePath, iSQLiteDBFilePath);
00095     _opentrepServiceContext = &lOPENTREP_ServiceContext;
00096
00097     // Instanciate an empty World object
00098     World& lWorld = FacWorld::instance().create();
00099     lOPENTREP_ServiceContext.setWorld (lWorld);
00100   }
00101
00102   // ////////////////////////////////////////////////////////////////////
00103   void OPENTREP_Service::finalise() {
00104   }
00105
00106   // ////////////////////////////////////////////////////////////////////
00107   OPENTREP_Service::FilePathSet_T
      OPENTREP_Service::getFilePaths() const {
00108     if (_opentrepServiceContext == NULL) {
00109       throw NonInitialisedServiceException ("The OpenTREP service has not
       been"
00110                                             " initialised");
00111     }
00112     assert (_opentrepServiceContext != NULL);
00113     OPENTREP_ServiceContext& lOPENTREP_ServiceContext = *_opentrepServiceContext;
00114
00115     // Retrieve the file-path of the POR (points of reference) file
00116     const PORFilePath_T& lPORFilePath = lOPENTREP_ServiceContext.
      getPORFilePath();
00117
00118     // Retrieve the Xapian database name (directorty of the index)
00119     const TravelDBFilePath_T& lTravelDBFilePath =
00120       lOPENTREP_ServiceContext.getTravelDBFilePath();
00121
00122     // Retrieve the SQLite3 database file-path
00123     const SQLiteDBFilePath_T& lSQLiteDBFilePath =
00124       lOPENTREP_ServiceContext.getSQLiteDBFilePath();
00125
00126     const DBFilePathPair_T lDBFilePathPair (lTravelDBFilePath, lSQLiteDBFilePath);
00127     FilePathSet_T oFilePathSet (lPORFilePath, lDBFilePathPair);
00128     return oFilePathSet;
00129   }
00130
00131   // ////////////////////////////////////////////////////////////////////
00132   NbOfDBEntries_T OPENTREP_Service::getIndexSize() {
00133     NbOfDBEntries_T oNbOfEntries = 0;
00134
00135     if (_opentrepServiceContext == NULL) {
00136       throw NonInitialisedServiceException ("The OpenTREP service has not
       been"
00137                                             " initialised");
00138     }
00139     assert (_opentrepServiceContext != NULL);
00140     OPENTREP_ServiceContext& lOPENTREP_ServiceContext = *_opentrepServiceContext;
00141
00142     // Retrieve the Xapian database name (directorty of the index)
00143     const TravelDBFilePath_T& lTravelDBFilePath =
00144       lOPENTREP_ServiceContext.getTravelDBFilePath();
00145
00146     // Delegate the query execution to the dedicated command
00147     BasChronometer lIndexSizeChronometer; lIndexSizeChronometer.
      start();
00148     oNbOfEntries = XapianIndexManager::getSize (lTravelDBFilePath);
00149     const double lIndexSizeMeasure = lIndexSizeChronometer.elapsed();
00150
00151     // DEBUG
00152     OPENTREP_LOG_DEBUG ("Size retrieval of the Xapian database (index): "
00153                         << lIndexSizeMeasure << " - "
00154                         << lOPENTREP_ServiceContext.display());
00155
00156     return oNbOfEntries;
00157   }
00158
00159   // ////////////////////////////////////////////////////////////////////
00160   NbOfMatches_T OPENTREP_Service::
00161   drawRandomLocations (const NbOfMatches_T& iNbOfDraws,
00162                        LocationList_T& ioLocationList) {
00163     NbOfMatches_T oNbOfMatches = 0;
00164
00165     if (_opentrepServiceContext == NULL) {
```

```
00166       throw NonInitialisedServiceException ("The OpenTREP service has not
      been"
00167                                             " initialised");
00168     }
00169     assert (_opentrepServiceContext != NULL);
00170     OPENTREP_ServiceContext& lOPENTREP_ServiceContext= *_opentrepServiceContext;
00171
00172     // Retrieve the Xapian database name (directorty of the index)
00173     const TravelDBFilePath_T& lTravelDBFilePath =
00174       lOPENTREP_ServiceContext.getTravelDBFilePath();
00175
00176     // Delegate the query execution to the dedicated command
00177     BasChronometer lRandomGetChronometer; lRandomGetChronometer.
      start();
00178     oNbOfMatches = XapianIndexManager::drawRandomLocations (lTravelDBFilePath,
00179                                                             iNbOfDraws,
00180                                                             ioLocationList);
00181     const double lRandomGetMeasure = lRandomGetChronometer.elapsed();
00182
00183     // DEBUG
00184     OPENTREP_LOG_DEBUG ("Random retrieval of locations (index): "
00185                         << lRandomGetMeasure << " - "
00186                         << lOPENTREP_ServiceContext.display());
00187
00188     return oNbOfMatches;
00189   }
00190
00191   // ////////////////////////////////////////////////////////////////////
00192   NbOfDBEntries_T OPENTREP_Service::buildSQLDB() {
00193     NbOfDBEntries_T oNbOfEntries = 0;
00194
00195     if (_opentrepServiceContext == NULL) {
00196       throw NonInitialisedServiceException ("The OpenTREP service has not
      been"
00197                                             " initialised");
00198     }
00199     assert (_opentrepServiceContext != NULL);
00200     OPENTREP_ServiceContext& lOPENTREP_ServiceContext = *_opentrepServiceContext;
00201
00202     // Retrieve the file-path of the POR (points of reference) file
00203     const PORFilePath_T& lPORFilePath= lOPENTREP_ServiceContext.
      getPORFilePath();
00204
00205     // Retrieve the SQLite3 database file-path
00206     const SQLiteDBFilePath_T& lSQLiteDBFilePath =
00207       lOPENTREP_ServiceContext.getSQLiteDBFilePath();
00208
00209     // Delegate the database creation to the dedicated command
00210     BasChronometer lDBCreationChronometer;
00211     lDBCreationChronometer.start();
00212     oNbOfEntries = DBManager::buildSQLDB (lPORFilePath, lSQLiteDBFilePath);
00213     const double lDBCreationMeasure = lDBCreationChronometer.elapsed();
00214
00215     // DEBUG
00216     OPENTREP_LOG_DEBUG ("Created the SQLite3 database: " << lDBCreationMeasure
00217                         << " - " << lOPENTREP_ServiceContext.display());
00218
00219     return oNbOfEntries;
00220   }
00221
00222   // ////////////////////////////////////////////////////////////////////
00223   NbOfDBEntries_T OPENTREP_Service::buildSearchIndex() {
00224     NbOfDBEntries_T oNbOfEntries = 0;
00225
00226     if (_opentrepServiceContext == NULL) {
00227       throw NonInitialisedServiceException ("The OpenTREP service has not
      been"
00228                                             " initialised");
00229     }
00230     assert (_opentrepServiceContext != NULL);
00231     OPENTREP_ServiceContext& lOPENTREP_ServiceContext = *_opentrepServiceContext;
00232
00233     // Retrieve the file-path of the POR (points of reference) file
00234     const PORFilePath_T& lPORFilePath= lOPENTREP_ServiceContext.
      getPORFilePath();
00235
00236     // Retrieve the Xapian database name (directorty of the index)
00237     const TravelDBFilePath_T& lTravelDBFilePath =
00238       lOPENTREP_ServiceContext.getTravelDBFilePath();
00239
00240     // Retrieve the Unicode transliterator
00241     const OTransliterator& lTransliterator =
00242       lOPENTREP_ServiceContext.getTransliterator();
00243
00244     // Delegate the index building to the dedicated command
00245     BasChronometer lBuildSearchIndexChronometer;
00246     lBuildSearchIndexChronometer.start();
```

```
00247     oNbOfEntries = IndexBuilder::buildSearchIndex (lPORFilePath,
00248                                                    lTravelDBFilePath,
00249                                                    lTransliterator);
00250     const double lBuildSearchIndexMeasure =
00251       lBuildSearchIndexChronometer.elapsed();
00252
00253     // DEBUG
00254     OPENTREP_LOG_DEBUG ("Built Xapian database (index): "
00255                         << lBuildSearchIndexMeasure << " - "
00256                         << lOPENTREP_ServiceContext.display());
00257
00258     return oNbOfEntries;
00259   }
00260
00261   // ////////////////////////////////////////////////////////////////////
00262   NbOfMatches_T OPENTREP_Service::
00263   interpretTravelRequest (const std::string& iTravelQuery,
00264                           LocationList_T& ioLocationList,
00265                           WordList_T& ioWordList) {
00266     NbOfMatches_T nbOfMatches = 0;
00267
00268     if (_opentrepServiceContext == NULL) {
00269       throw NonInitialisedServiceException ("The OpenTREP service has not
    been"
00270                                             " initialised");
00271     }
00272     assert (_opentrepServiceContext != NULL);
00273     OPENTREP_ServiceContext& lOPENTREP_ServiceContext= *_opentrepServiceContext;
00274
00275     // Get the date-time for the present time
00276     boost::posix_time::ptime lNowDateTime =
00277       boost::posix_time::second_clock::local_time();
00278     // boost::gregorian::date lNowDate = lNowDateTime.date();
00279
00280     // DEBUG
00281     OPENTREP_LOG_DEBUG (std::endl
00282                         << "================================================="
00283                         << std::endl
00284                         << lNowDateTime << " - Match query '" << iTravelQuery
00285                         << "' on Xapian database (index)");
00286
00287     // Check that the travel request is not empty
00288     if (iTravelQuery.empty() == true) {
00289       std::ostringstream errorStr;
00290       errorStr << "The travel request is empty.";
00291       OPENTREP_LOG_ERROR (errorStr.str());
00292       throw TravelRequestEmptyException (errorStr.str());
00293     }
00294
00295     // Retrieve the Xapian database name (directorty of the index)
00296     const TravelDBFilePath_T& lTravelDBFilePath =
00297       lOPENTREP_ServiceContext.getTravelDBFilePath();
00298
00299     // Delegate the query execution to the dedicated command
00300     BasChronometer lRequestInterpreterChronometer;
00301     lRequestInterpreterChronometer.start();
00302     nbOfMatches = RequestInterpreter::interpretTravelRequest (lTravelDBFilePath,
00303                                                               iTravelQuery,
00304                                                               ioLocationList,
00305                                                               ioWordList);
00306     const double lRequestInterpreterMeasure =
00307       lRequestInterpreterChronometer.elapsed();
00308
00309     // DEBUG
00310     OPENTREP_LOG_DEBUG ("Match query on Xapian database (index): "
00311                         << lRequestInterpreterMeasure << " - "
00312                         << lOPENTREP_ServiceContext.display());
00313
00314     return nbOfMatches;
00315   }
00316
00317 }
```

24.297 opentrep/service/OPENTREP_ServiceContext.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <sstream>
#include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
#include <opentrep/bom/World.hpp>
#include <opentrep/service/OPENTREP_ServiceContext.hpp>
```

Namespaces

- OPENTREP

24.298 OPENTREP_ServiceContext.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <ostream>
00007 #include <sstream>
00008 // OpenTrep
00009 #include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
00010 #include <opentrep/bom/World.hpp>
00011 #include <opentrep/service/OPENTREP_ServiceContext.hpp>
00012
00013 namespace OPENTREP {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   OPENTREP_ServiceContext::OPENTREP_ServiceContext()
00017     : _world (NULL),
00018       _porFilePath (DEFAULT_OPENTREP_POR_FILEPATH),
00019       _travelDBFilePath (DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH),
00020       _sqliteDBFilePath (DEFAULT_OPENTREP_SQLITE_DB_FILEPATH) {
00021     assert (false);
00022   }
00023
00024   // ////////////////////////////////////////////////////////////////////
00025   OPENTREP_ServiceContext::
00026   OPENTREP_ServiceContext (const TravelDBFilePath_T& iTravelDBFilePath,
00027                            const SQLiteDBFilePath_T& iSQLiteDBFilePath)
00028     : _world (NULL),
00029       _porFilePath (DEFAULT_OPENTREP_POR_FILEPATH),
00030       _travelDBFilePath (iTravelDBFilePath),
00031       _sqliteDBFilePath (iSQLiteDBFilePath) {
00032   }
00033
00034   // ////////////////////////////////////////////////////////////////////
00035   OPENTREP_ServiceContext::
00036   OPENTREP_ServiceContext (const PORFilePath_T& iPORFilePath,
00037                            const TravelDBFilePath_T& iTravelDBFilePath,
00038                            const SQLiteDBFilePath_T& iSQLiteDBFilePath)
00039     : _world (NULL), _porFilePath (iPORFilePath),
00040       _travelDBFilePath (iTravelDBFilePath),
00041       _sqliteDBFilePath (iSQLiteDBFilePath) {
00042   }
00043
00044   // ////////////////////////////////////////////////////////////////////
00045   OPENTREP_ServiceContext::~OPENTREP_ServiceContext() {
00046   }
00047
00048   // ////////////////////////////////////////////////////////////////////
00049   World& OPENTREP_ServiceContext::getWorldHandler() const {
00050     assert (_world != NULL);
00051     return *_world;
00052   }
00053
00054   // ////////////////////////////////////////////////////////////////////
00055   const std::string OPENTREP_ServiceContext::shortDisplay() const {
00056     std::ostringstream oStr;
00057     oStr << "OPENTREP_ServiceContext: "
00058          << "file-path of the POR file: " << _porFilePath
00059          << "; Xapian database (directory of the index): " << _travelDBFilePath
00060          << "; SQLite3 database file-path: " << _sqliteDBFilePath
00061          << std::endl;
00062     return oStr.str();
00063   }
00064
00065   // ////////////////////////////////////////////////////////////////////
00066   const std::string OPENTREP_ServiceContext::display() const {
00067     std::ostringstream oStr;
00068     oStr << shortDisplay();
00069     if (_world != NULL) {
00070       oStr << _world->display();
00071     }
00072     return oStr.str();
00073   }
```

```
00074
00075 }
```

24.299 opentrep/service/OPENTREP_ServiceContext.hpp File Reference

```
#include <string>
#include <opentrep/OPENTREP_Types.hpp>
#include <opentrep/basic/OTransliterator.hpp>
#include <opentrep/service/ServiceAbstract.hpp>
```

Classes

- class OPENTREP::OPENTREP_ServiceContext

  Class holding the context of the OpenTrep services.

Namespaces

- soci
- OPENTREP

24.300 OPENTREP_ServiceContext.hpp

```
00001 #ifndef __OPENTREP_SVC_OPENTREPSERVICECONTEXT_HPP
00002 #define __OPENTREP_SVC_OPENTREPSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // OpenTrep
00010 #include <opentrep/OPENTREP_Types.hpp>
00011 #include <opentrep/basic/OTransliterator.hpp>
00012 #include <opentrep/service/ServiceAbstract.hpp>
00013
00014 // Forward declarations
00015 namespace soci {
00016   class session;
00017 }
00018
00019 namespace OPENTREP {
00020
00021   // Forward declarations
00022   class World;
00023
00027   class OPENTREP_ServiceContext : public ServiceAbstract {
00028     friend class FacOpenTrepServiceContext;
00029   public:
00030     // ////////////////// Getters ///////////////////////
00034     World* getWorld() const {
00035       return _world;
00036     }
00037
00041     World& getWorldHandler() const;
00042
00046     const PORFilePath_T& getPORFilePath() const {
00047       return _porFilePath;
00048     }
00049
00053     const TravelDBFilePath_T& getTravelDBFilePath() const {
00054       return _travelDBFilePath;
00055     }
00056
00060     const SQLiteDBFilePath_T& getSQLiteDBFilePath() const {
00061       return _sqliteDBFilePath;
00062     }
00063
00067     const OTransliterator& getTransliterator() const {
00068       return _transliterator;
00069     }
```

```
00070
00071   public:
00072     // ////////////////// Setters //////////////////////
00076     void setWorld (World& ioWorld) {
00077       _world = &ioWorld;
00078     }
00079
00083     void setPORFilePath (const std::string& iPORFilePath) {
00084       _porFilePath = PORFilePath_T (iPORFilePath);
00085     }
00086
00090     void setTravelDBFilePath (const std::string& iTravelDBFilePath) {
00091       _travelDBFilePath = TravelDBFilePath_T (iTravelDBFilePath);
00092     }
00093
00097     void setSQLiteDBFilePath (const std::string& iSQLiteDBFilePath) {
00098       _sqliteDBFilePath = SQLiteDBFilePath_T (iSQLiteDBFilePath);
00099     }
00100
00104     void setTransliterator (const OTransliterator& iTransliterator) {
00105       _transliterator = iTransliterator;
00106     }
00107
00108
00109   public:
00110     // ///////// Display Methods //////////
00114     const std::string shortDisplay() const;
00115
00119     const std::string display() const;
00120
00121
00122   private:
00123     // /////// Construction / initialisation ////////
00127     OPENTREP_ServiceContext (const TravelDBFilePath_T&, const
      SQLiteDBFilePath_T&);
00128
00132     OPENTREP_ServiceContext (const PORFilePath_T&, const
      TravelDBFilePath_T&,
00133                              const SQLiteDBFilePath_T&);
00134
00138     OPENTREP_ServiceContext();
00139
00143     OPENTREP_ServiceContext (const
      OPENTREP_ServiceContext&);
00144
00148     ~OPENTREP_ServiceContext();
00149
00150
00151   private:
00152     // /////////////// Attributes ///////////////
00156     World* _world;
00157
00161     PORFilePath_T _porFilePath;
00162
00166     TravelDBFilePath_T _travelDBFilePath;
00167
00176     SQLiteDBFilePath_T _sqliteDBFilePath;
00177
00181     OTransliterator _transliterator;
00182   };
00183
00184 }
00185 #endif // __OPENTREP_SVC_OPENTREPSERVICECONTEXT_HPP
```

24.301 opentrep/service/ServiceAbstract.cpp File Reference

```
#include <opentrep/service/ServiceAbstract.hpp>
```

Namespaces

- OPENTREP

24.302 ServiceAbstract.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
```

```
00003 // ////////////////////////////////////////////////////////////////////
00004 // OPENTREP
00005 #include <opentrep/service/ServiceAbstract.hpp>
00006
00007 namespace OPENTREP {
00008
00009 }
```

24.303 opentrep/service/ServiceAbstract.hpp File Reference

```
#include <iostream>
#include <sstream>
```

Classes

- class OPENTREP::ServiceAbstract

Namespaces

- OPENTREP

Functions

- template<class charT , class traits >
  std::basic_ostream< charT,
  traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const OPENTREP::ServiceAbstract &iService)
- template<class charT , class traits >
  std::basic_istream< charT,
  traits > & operator>> (std::basic_istream< charT, traits > &ioIn, OPENTREP::ServiceAbstract &ioService)

24.303.1 Function Documentation

24.303.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut,* const OPENTREP::ServiceAbstract & *iService*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 42 of file ServiceAbstract.hpp.

24.303.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn,* OPENTREP::ServiceAbstract & *ioService*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 70 of file ServiceAbstract.hpp.

References OPENTREP::ServiceAbstract::fromStream().

24.304 ServiceAbstract.hpp

```
00001 #ifndef __OPENTREP_SVC_SERVICEABSTRACT_HPP
00002 #define __OPENTREP_SVC_SERVICEABSTRACT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
```

```
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iostream>
00009 #include <sstream>
00010
00011 namespace OPENTREP {
00012
00014   class ServiceAbstract {
00015   public:
00016
00018     virtual ~ServiceAbstract() {}
00019
00022     virtual void toStream (std::ostream& ioOut) const {}
00023
00026     virtual void fromStream (std::istream& ioIn) {}
00027
00028   protected:
00030     ServiceAbstract() {}
00031   };
00032 }
00033
00039 template <class charT, class traits>
00040 inline
00041 std::basic_ostream<charT, traits>&
00042 operator<< (std::basic_ostream<charT, traits>& ioOut,
00043             const OPENTREP::ServiceAbstract& iService) {
00049   std::basic_ostringstream<charT,traits> ostr;
00050   ostr.copyfmt (ioOut);
00051   ostr.width (0);
00052
00053   // Fill string stream
00054   iService.toStream (ostr);
00055
00056   // Print string stream
00057   ioOut << ostr.str();
00058
00059   return ioOut;
00060 }
00061
00067 template <class charT, class traits>
00068 inline
00069 std::basic_istream<charT, traits>&
00070 operator>> (std::basic_istream<charT, traits>& ioIn,
00071             OPENTREP::ServiceAbstract& ioService) {
00072   // Fill Service object with input stream
00073   ioService.fromStream (ioIn);
00074   return ioIn;
00075 }
00076
00077 #endif // __OPENTREP_SVC_SERVICEABSTRACT_HPP
```

24.305 opentrep/service/ServiceUtilities.cpp File Reference

```
#include <cassert>
#include <boost/filesystem.hpp>
#include <opentrep/service/Logger.hpp>
#include <opentrep/service/ServiceUtilities.hpp>
```

Namespaces

- OPENTREP

24.306 ServiceUtilities.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost::Filesystem
00007 #include <boost/filesystem.hpp>
00008 // OpenTREP
00009 #include <opentrep/service/Logger.hpp>
00010 #include <opentrep/service/ServiceUtilities.hpp>
00011
```

```
00012 namespace OPENTREP {
00013
00014   // ////////////////////////////////////////////////////////////////////
00015   bool ServiceUtilities::isDirectory (const std::string& iPath) {
00016     bool oIsDirectory = false;
00017
00018     if (boost::filesystem::exists (iPath) == true) {
00019
00020       if (boost::filesystem::is_directory (iPath) == true) {
00021         oIsDirectory = true;
00022
00023       } else {
00024         OPENTREP_LOG_ERROR (iPath << " exists, but is not a directory");
00025       }
00026
00027     } else {
00028       OPENTREP_LOG_ERROR (iPath << " does not exist");
00029     }
00030
00031     return oIsDirectory;
00032   }
00033
00034 }
```

24.307 opentrep/service/ServiceUtilities.hpp File Reference

```
#include <string>
```

Classes

- class OPENTREP::ServiceUtilities

Namespaces

- OPENTREP

24.308 ServiceUtilities.hpp

```
00001 #ifndef __OPENTREP_SVC_SERVICEUTILITIES_HPP
00002 #define __OPENTREP_SVC_SERVICEUTILITIES_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace OPENTREP {
00011
00013   class ServiceUtilities {
00014   public:
00018     static bool isDirectory (const std::string& iPath);
00019   };
00020
00021 }
00022 #endif // __OPENTREP_SVC_SERVICEUTILITIES_HPP
```

24.309 test/opentrep/IndexBuildingTestSuite.cpp File Reference

24.310 IndexBuildingTestSuite.cpp

```
00001
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <iostream>
00011 #include <sstream>
```

```
00012 #include <fstream>
00013 #include <string>
00014 // Boost Unit Test Framework (UTF)
00015 #define BOOST_TEST_DYN_LINK
00016 #define BOOST_TEST_MAIN
00017 #define BOOST_TEST_MODULE IndexBuildingTestSuite
00018 #include <boost/test/unit_test.hpp>
00019 // OpenTrep
00020 #include <opentrep/OPENTREP_Service.hpp>
00021 #include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
00022 #include <opentrep/Location.hpp>
00023 #include <opentrep/config/opentrep-paths.hpp>
00024
00025 namespace boost_utf = boost::unit_test;
00026
00027 // (Boost) Unit Test XML Report
00028 std::ofstream utfReportStream ("IndexBuildingTestSuite_utfresults.xml");
00029
00033 struct UnitTestConfig {
00035   UnitTestConfig() {
00036     boost_utf::unit_test_log.set_stream (utfReportStream);
00037     boost_utf::unit_test_log.set_format (boost_utf::XML);
00038     boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00039     //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
00040   }
00041
00043   ~UnitTestConfig() {
00044   }
00045 };
00046
00047
00048 // ////////////// Constants for the tests ////////////////
00052 const std::string K_POR_FILEPATH (OPENTREP_POR_DATA_DIR
00053                                   "/test_ori_por_public.csv");
00054
00058 const std::string X_XAPIAN_DB_FP ("/tmp/opentrep/test_traveldb");
00059
00063 const std::string X_SQLITE_DB_FP ("/tmp/opentrep/test_traveldb/ori_por_public.csv");
00064
00065
00066 // ///////////////// Main: Unit Test Suite //////////////
00067
00068 // Set the UTF configuration (re-direct the output to a specific file)
00069 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00070
00071 // Start the test suite
00072 BOOST_AUTO_TEST_SUITE (master_test_suite)
00073
00074
00077 BOOST_AUTO_TEST_CASE (opentrep_simple_index) {
00078
00079   // Output log File
00080   std::string lLogFilename ("IndexBuildingTestSuite.log");
00081
00082   // Set the log parameters
00083   std::ofstream logOutputFile;
00084   // Open and clean the log outputfile
00085   logOutputFile.open (lLogFilename.c_str());
00086   logOutputFile.clear();
00087
00088   // Initialise the context
00089   const OPENTREP::PORFilePath_T lPORFilePath (K_POR_FILEPATH);
00090   const OPENTREP::TravelDBFilePath_T lTravelDBFilePath (
      X_XAPIAN_DB_FP);
00091   const OPENTREP::SQLiteDBFilePath_T lSQLiteDBFilePath (
      X_SQLITE_DB_FP);
00092   OPENTREP::OPENTREP_Service opentrepService  (logOutputFile, lPORFilePath,
00093                                                lTravelDBFilePath,
00094                                                lSQLiteDBFilePath);
00095
00096   // Query the Xapian database (index)
00097   OPENTREP::WordList_T lNonMatchedWordList;
00098   OPENTREP::LocationList_T lLocationList;
00099   // Launch the indexation
00100   const OPENTREP::NbOfDBEntries_T nbOfEntries =
00101     opentrepService.buildSearchIndex();
00102
00103   BOOST_CHECK_MESSAGE (nbOfEntries == 9,
00104                        "The Xapian index ('" << lTravelDBFilePath
00105                        << "') contains " << nbOfEntries
00106                        << " entries, where as 9 are expected.");
00107
00108   // Close the Log outputFile
00109   logOutputFile.close();
00110 }
00111
00112 // End the test suite
```

```
00113 BOOST_AUTO_TEST_SUITE_END()
00114
00115
```

24.311 test/opentrep/PartitionTestSuite.cpp File Reference

```
#include <sstream>
#include <fstream>
#include <string>
#include <list>
#include <boost/test/unit_test.hpp>
#include <opentrep/bom/StringPartition.hpp>
```

Classes

- struct UnitTestConfig

Macros

- #define BOOST_TEST_DYN_LINK
- #define BOOST_TEST_MAIN
- #define BOOST_TEST_MODULE PartitionTestSuite

Functions

- std::ofstream utfReportStream ("PartitionTestSuite_utfresults.xml")
- BOOST_GLOBAL_FIXTURE (UnitTestConfig)
- BOOST_AUTO_TEST_CASE (partition_small_string)

24.311.1 Macro Definition Documentation

24.311.1.1 #define BOOST_TEST_DYN_LINK

Definition at line 15 of file PartitionTestSuite.cpp.

24.311.1.2 #define BOOST_TEST_MAIN

Definition at line 16 of file PartitionTestSuite.cpp.

24.311.1.3 #define BOOST_TEST_MODULE PartitionTestSuite

Definition at line 17 of file PartitionTestSuite.cpp.

24.311.2 Function Documentation

24.311.2.1 std::ofstream utfReportStream ("PartitionTestSuite_utfresults.xml")

Referenced by UnitTestConfig::UnitTestConfig().

24.311.2.2 BOOST_GLOBAL_FIXTURE (UnitTestConfig)

24.311.2.3 BOOST_AUTO_TEST_CASE (partition_small_string)

Test the partitioning algorithm with a small string

Definition at line 56 of file PartitionTestSuite.cpp.

References OPENTREP::StringPartition::size().

24.312 PartitionTestSuite.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 //
00003 // String partition algorithm
00004 //
00005 // Author: Denis Arnaud
00006 // Date:   April 2012
00007 //
00008 // //////////////////////////////////////////////////////////////////////
00009 // STL
00010 #include <sstream>
00011 #include <fstream>
00012 #include <string>
00013 #include <list>
00014 // Boost Unit Test Framework (UTF)
00015 #define BOOST_TEST_DYN_LINK
00016 #define BOOST_TEST_MAIN
00017 #define BOOST_TEST_MODULE PartitionTestSuite
00018 #include <boost/test/unit_test.hpp>
00019 // OpenTrep
00020 #include <opentrep/bom/StringPartition.hpp>
00021
00022 namespace boost_utf = boost::unit_test;
00023
00024 // (Boost) Unit Test XML Report
00025 std::ofstream utfReportStream ("PartitionTestSuite_utfresults.xml");
00026
00030 struct UnitTestConfig {
00032   UnitTestConfig() {
00033     boost_utf::unit_test_log.set_stream (utfReportStream);
00034     boost_utf::unit_test_log.set_format (boost_utf::XML);
00035     boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00036     //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
00037   }
00038
00040   ~UnitTestConfig() {
00041   }
00042 };
00043
00044
00045 // ///////////////// Main: Unit Test Suite //////////////
00046
00047 // Set the UTF configuration (re-direct the output to a specific file)
00048 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00049
00050 // Start the test suite
00051 BOOST_AUTO_TEST_SUITE (master_test_suite)
00052
00053
00056 BOOST_AUTO_TEST_CASE (partition_small_string) {
00057
00058   // Output log File
00059   std::string lLogFilename ("PartitionTestSuite.log");
00060
00061   // Set the log parameters
00062   std::ofstream logOutputFile;
00063   // Open and clean the log outputfile
00064   logOutputFile.open (lLogFilename.c_str());
00065   logOutputFile.clear();
00066
00067   const std::string lLax1Str = "los angeles";
00068   const std::string lLax2Str = "lso angeles";
00069   const std::string lRio1Str = "rio de janeiro";
00070   const std::string lRio2Str = "rio de janero";
00071   const std::string lRek1Str = "reikjavik";
00072   const std::string lRek2Str = "rekyavik";
00073   const std::string lSfoRio1Str = "san francisco rio de janeiro";
00074   const std::string lSfoRio2Str = "san francicso rio de janero";
00075   const std::string lSfoRio3Str = "sna francicso rio de janero";
00076   const std::string lChelseaStr = "chelsea municipal airport";
00077
00078   //
00079   OPENTREP::StringPartition lStringPartition (lSfoRio3Str);
00080   logOutputFile << lStringPartition << std::endl;
00081
00082   BOOST_CHECK_MESSAGE (lStringPartition.size() == 16,
00083                        "The partition, for '" << lRio2Str
00084                        << "', should contain a single list."
00085                        << " However, its size is " << lStringPartition.size()
```

```
00086                       << ".");
00087
00088   //
00089   OPENTREP::StringPartition lAnotherStringPartition (lChelseaStr);
00090   logOutputFile << lAnotherStringPartition << std::endl;
00091
00092   // Close the Log outputFile
00093   logOutputFile.close();
00094 }
00095
00096 // End the test suite
00097 BOOST_AUTO_TEST_SUITE_END()
00098
```

24.313 test/opentrep/SearchingTestSuite.cpp File Reference

24.314 SearchingTestSuite.cpp

```
00001
00005 // //////////////////////////////////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <sstream>
00010 #include <fstream>
00011 #include <string>
00012 // Boost Unit Test Framework (UTF)
00013 #define BOOST_TEST_DYN_LINK
00014 #define BOOST_TEST_MAIN
00015 #define BOOST_TEST_MODULE SearchingTestSuite
00016 #include <boost/test/unit_test.hpp>
00017 // OpenTrep
00018 #include <opentrep/OPENTREP_Service.hpp>
00019 #include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
00020 #include <opentrep/Location.hpp>
00021
00022 namespace boost_utf = boost::unit_test;
00023
00024 // (Boost) Unit Test XML Report
00025 std::ofstream utfReportStream ("SearchingTestSuite_utfresults.xml");
00026
00030 struct UnitTestConfig {
00032   UnitTestConfig() {
00033     boost_utf::unit_test_log.set_stream (utfReportStream);
00034     boost_utf::unit_test_log.set_format (boost_utf::XML);
00035     boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00036     //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
00037   }
00038
00040   ~UnitTestConfig() {
00041   }
00042 };
00043
00044
00045 // ////////////// Constants for the tests ///////////////
00049 const std::string X_XAPIAN_DB_FP ("/tmp/opentrep/test_traveldb");
00050
00054 const std::string X_SQLITE_DB_FP ("/tmp/opentrep/test_traveldb/ori_por_public.csv");
00055
00056
00057 // ///////////////// Main: Unit Test Suite //////////////
00058
00059 // Set the UTF configuration (re-direct the output to a specific file)
00060 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00061
00062 // Start the test suite
00063 BOOST_AUTO_TEST_SUITE (master_test_suite)
00064
00065
00068 BOOST_AUTO_TEST_CASE (opentrep_simple_search) {
00069
00070   // Output log File
00071   std::string lLogFilename ("SearchingTestSuite.log");
00072
00073   // Travel query
00074   std::string lTravelQuery ("nce");
00075
00076   // Set the log parameters
00077   std::ofstream logOutputFile;
00078   // Open and clean the log outputfile
00079   logOutputFile.open (lLogFilename.c_str());
00080   logOutputFile.clear();
00081
```

```
00082   // Initialise the context
00083   const OPENTREP::TravelDBFilePath_T lTravelDBFilePath (
      X_XAPIAN_DB_FP);
00084   const OPENTREP::SQLiteDBFilePath_T lSQLiteDBFilePath (
      X_SQLITE_DB_FP);
00085   OPENTREP::OPENTREP_Service opentrepService (logOutputFile, lTravelDBFilePath,
00086                                               lSQLiteDBFilePath);
00087
00088   // Query the Xapian database (index)
00089   OPENTREP::WordList_T lNonMatchedWordList;
00090   OPENTREP::LocationList_T lLocationList;
00091   const OPENTREP::NbOfMatches_T nbOfMatches =
00092     opentrepService.interpretTravelRequest (lTravelQuery, lLocationList,
00093                                             lNonMatchedWordList);
00094   BOOST_CHECK_MESSAGE (nbOfMatches == 1,
00095                        "The travel query ('" << lTravelQuery
00096                        << "') matches with " << nbOfMatches
00097                        << " key-words, whereas 1 is expected.");
00098
00099   // Close the Log outputFile
00100   logOutputFile.close();
00101 }
00102
00103 // End the test suite
00104 BOOST_AUTO_TEST_SUITE_END()
00105
00106
```

24.315 test/opentrep/SliceTestSuite.cpp File Reference

```
#include <sstream>
#include <fstream>
#include <string>
#include <list>
#include <boost/filesystem.hpp>
#include <boost/test/unit_test.hpp>
#include <opentrep/bom/QuerySlices.hpp>
#include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
#include <opentrep/OPENTREP_Service.hpp>
```

Classes

- struct UnitTestConfig

Macros

- #define BOOST_TEST_DYN_LINK
- #define BOOST_TEST_MAIN
- #define BOOST_TEST_MODULE SliceTestSuite

Functions

- std::ofstream utfReportStream ("SliceTestSuite_utfresults.xml")
- const std::string X_XAPIAN_DB_FP (OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH)
- const std::string X_SQLITE_DB_FP (OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH)
- BOOST_GLOBAL_FIXTURE (UnitTestConfig)
- BOOST_AUTO_TEST_CASE (slice_small_string)

24.315.1 Macro Definition Documentation

24.315.1.1 #define BOOST_TEST_DYN_LINK

Definition at line 17 of file SliceTestSuite.cpp.

24.315.1.2 #define BOOST_TEST_MAIN

Definition at line 18 of file SliceTestSuite.cpp.

24.315.1.3 #define BOOST_TEST_MODULE SliceTestSuite

Definition at line 19 of file SliceTestSuite.cpp.

24.315.2 Function Documentation

24.315.2.1 std::ofstream utfReportStream ("SliceTestSuite_utfresults.xml")

Referenced by UnitTestConfig::UnitTestConfig().

24.315.2.2 const std::string X_XAPIAN_DB_FP (OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH)

Xapian database/index file-path (directory containing the index).

Referenced by BOOST_AUTO_TEST_CASE().

24.315.2.3 const std::string X_SQLITE_DB_FP (OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH)

SQlite3 database file-path.

Referenced by BOOST_AUTO_TEST_CASE().

24.315.2.4 BOOST_GLOBAL_FIXTURE (UnitTestConfig)

24.315.2.5 BOOST_AUTO_TEST_CASE (slice_small_string)

Test the sliceing algorithm with a small string

Definition at line 71 of file SliceTestSuite.cpp.

References OPENTREP::QuerySlices::size(), X_SQLITE_DB_FP(), and X_XAPIAN_DB_FP().

24.316 SliceTestSuite.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 //
00003 // Query slice algorithm
00004 //
00005 // Author: Denis Arnaud
00006 // Date:   October 2012
00007 //
00008 // //////////////////////////////////////////////////////////////////////
00009 // STL
00010 #include <sstream>
00011 #include <fstream>
00012 #include <string>
00013 #include <list>
00014 // Boost
00015 #include <boost/filesystem.hpp>
00016 // Boost Unit Test Framework (UTF)
00017 #define BOOST_TEST_DYN_LINK
00018 #define BOOST_TEST_MAIN
00019 #define BOOST_TEST_MODULE SliceTestSuite
00020 #include <boost/test/unit_test.hpp>
00021 // OpenTrep
00022 #include <opentrep/bom/QuerySlices.hpp>
00023 #include <opentrep/basic/BasConst_OPENTREP_Service.hpp>
00024 #include <opentrep/OPENTREP_Service.hpp>
00025
00026 namespace boost_utf = boost::unit_test;
00027
00028 // (Boost) Unit Test XML Report
00029 std::ofstream utfReportStream ("SliceTestSuite_utfresults.xml");
00030
00034 struct UnitTestConfig {
00036   UnitTestConfig() {
00037     boost_utf::unit_test_log.set_stream (utfReportStream);
```

```
00038     boost_utf::unit_test_log.set_format (boost_utf::XML);
00039     boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00040     //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
00041   }
00042
00044   ~UnitTestConfig() {
00045   }
00046 };
00047
00048 // ////////////// Constants for the tests ///////////////
00052 const std::string X_XAPIAN_DB_FP (
      OPENTREP::DEFAULT_OPENTREP_XAPIAN_DB_FILEPATH);
00053
00057 const std::string X_SQLITE_DB_FP (
      OPENTREP::DEFAULT_OPENTREP_SQLITE_DB_FILEPATH);
00058
00059
00060 // ///////////////// Main: Unit Test Suite //////////////
00061
00062 // Set the UTF configuration (re-direct the output to a specific file)
00063 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00064
00065 // Start the test suite
00066 BOOST_AUTO_TEST_SUITE (master_test_suite)
00067
00068
00071 BOOST_AUTO_TEST_CASE (slice_small_string) {
00072
00073   // Output log File
00074   const std::string lLogFilename ("SliceTestSuite.log");
00075
00076   // Set the log parameters
00077   std::ofstream logOutputFile;
00078   // Open and clean the log outputfile
00079   logOutputFile.open (lLogFilename.c_str());
00080   logOutputFile.clear();
00081
00082   // Initialise the context
00083   const OPENTREP::TravelDBFilePath_T lTravelDBFilePath (
      X_XAPIAN_DB_FP);
00084   const OPENTREP::SQLiteDBFilePath_T lSQLiteDBFilePath (
      X_SQLITE_DB_FP);
00085   OPENTREP::OPENTREP_Service opentrepService (logOutputFile, lTravelDBFilePath,
00086                                               lSQLiteDBFilePath);
00087
00088   // A few sample strings
00089   const std::string lLax1Str = "los angeles";
00090   const std::string lLax2Str = "lso angeles";
00091   const std::string lRio1Str = "rio de janeiro";
00092   const std::string lRio2Str = "rio de janero";
00093   const std::string lRek1Str = "reikjavik";
00094   const std::string lRek2Str = "rekyavik";
00095   const std::string lSfoRio1Str = "san francisco rio de janeiro";
00096   const std::string lSfoRio2Str = "san francicso rio de janero";
00097   const std::string lSfoRio3Str = "sna francicso rio de janero";
00098   const std::string lChelseaStr = "chelsea municipal airport";
00099
00104   // Open the Xapian database
00105   Xapian::Database lXapianDatabase (lTravelDBFilePath);
00106
00107   // Create the query slices
00108   OPENTREP::QuerySlices lQuerySlices (lXapianDatabase, lSfoRio1Str);
00109
00110   //
00111   BOOST_CHECK_MESSAGE (lQuerySlices.size() == 5,
00112                        "The query ('" << lSfoRio1Str
00113                        << "') should contain 5 slices."
00114                        << " However, its size is " << lQuerySlices.size()
00115                        << ".");
00116
00117   // Create other query slices
00118   OPENTREP::QuerySlices lAnotherQuerySlices (lXapianDatabase, lChelseaStr);
00119
00120   //
00121   BOOST_CHECK_MESSAGE (lAnotherQuerySlices.size() == 3,
00122                        "The query ('" << lChelseaStr
00123                        << "') should contain 3 slices."
00124                        << " However, its size is " << lAnotherQuerySlices.size()
00125                        << ".");
00126
00127   // Close the Log outputFile
00128   logOutputFile.close();
00129 }
00130
00131 // End the test suite
00132 BOOST_AUTO_TEST_SUITE_END()
00133
```

24.317 test/opentrep/UnicodeTestSuite.cpp File Reference

```
#include <sstream>
#include <fstream>
#include <string>
#include <list>
#include <boost/test/unit_test.hpp>
#include <opentrep/basic/OTransliterator.hpp>
```

Classes

- struct UnitTestConfig

Macros

- #define BOOST_TEST_DYN_LINK
- #define BOOST_TEST_MAIN
- #define BOOST_TEST_MODULE UnicodeTestSuite

Functions

- std::ofstream utfReportStream ("UnicodeTestSuite_utfresults.xml")
- BOOST_GLOBAL_FIXTURE (UnitTestConfig)
- BOOST_AUTO_TEST_CASE (unicode_strings)

24.317.1 Macro Definition Documentation

24.317.1.1 #define BOOST_TEST_DYN_LINK

Definition at line 15 of file UnicodeTestSuite.cpp.

24.317.1.2 #define BOOST_TEST_MAIN

Definition at line 16 of file UnicodeTestSuite.cpp.

24.317.1.3 #define BOOST_TEST_MODULE UnicodeTestSuite

Definition at line 17 of file UnicodeTestSuite.cpp.

24.317.2 Function Documentation

24.317.2.1 std::ofstream utfReportStream ("UnicodeTestSuite_utfresults.xml")

Referenced by UnitTestConfig::UnitTestConfig().

24.317.2.2 BOOST_GLOBAL_FIXTURE (UnitTestConfig)

24.317.2.3 BOOST_AUTO_TEST_CASE (unicode_strings)

Test the Unicode transformation algorithm

Definition at line 56 of file UnicodeTestSuite.cpp.

References OPENTREP::OTransliterator::normalise().

24.318 UnicodeTestSuite.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 //
00003 // Unicode-based transformations
00004 //
00005 // Author: Denis Arnaud
00006 // Date:   September 2012
00007 //
00008 // ////////////////////////////////////////////////////////////////////
00009 // STL
00010 #include <sstream>
00011 #include <fstream>
00012 #include <string>
00013 #include <list>
00014 // Boost Unit Test Framework (UTF)
00015 #define BOOST_TEST_DYN_LINK
00016 #define BOOST_TEST_MAIN
00017 #define BOOST_TEST_MODULE UnicodeTestSuite
00018 #include <boost/test/unit_test.hpp>
00019 // OpenTrep
00020 #include <opentrep/basic/OTransliterator.hpp>
00021
00022 namespace boost_utf = boost::unit_test;
00023
00024 // (Boost) Unit Test XML Report
00025 std::ofstream utfReportStream ("UnicodeTestSuite_utfresults.xml");
00026
00030 struct UnitTestConfig {
00032   UnitTestConfig() {
00033     boost_utf::unit_test_log.set_stream (utfReportStream);
00034     boost_utf::unit_test_log.set_format (boost_utf::XML);
00035     boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00036     //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
00037   }
00038
00040   ~UnitTestConfig() {
00041   }
00042 };
00043
00044
00045 // ///////////////// Main: Unit Test Suite ///////////////
00046
00047 // Set the UTF configuration (re-direct the output to a specific file)
00048 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00049
00050 // Start the test suite
00051 BOOST_AUTO_TEST_SUITE (master_test_suite)
00052
00053
00056 BOOST_AUTO_TEST_CASE (unicode_strings) {
00057
00058   // Output log File
00059   std::string lLogFilename ("UnicodeTestSuite.log");
00060
00061   // Set the log parameters
00062   std::ofstream logOutputFile;
00063   // Open and clean the log outputfile
00064   logOutputFile.open (lLogFilename.c_str());
00065   logOutputFile.clear();
00066
00067   // Unicode transliterator
00068   OPENTREP::OTransliterator lTransliterator;
00069
00070   // European
00071   const std::string lFRStr = "À côté de Nice Côte d'Azur";
00072   const std::string lFRTranslitStr = lTransliterator.normalise (lFRStr);
00073   const std::string lFRTargetStr = "a cote de nice cote d azur";
00074   const std::string lHUStr = "San Franciscó-i nemzetközi repülőtér";
00075   const std::string lHUTranslitStr = lTransliterator.normalise (lHUStr);
00076   const std::string lHUTargetStr = "san francisco i nemzetkozi repuloter";
00077   const std::string lGRStr = " ";
00078   const std::string lGRTranslitStr = lTransliterator.normalise (lGRStr);
00079   const std::string lGRTargetStr = "alphabetikos katalogos";
00080   const std::string lRUStr = " «», ";
00081   const std::string lRUTranslitStr = lTransliterator.normalise (lRUStr);
00082   const std::string lRUTargetStr = "aeroport annaba biologiceskom";
00083   const std::string lUKStr = "";
00084   const std::string lUKTranslitStr = lTransliterator.normalise (lUKStr);
00085   const std::string lUKTargetStr = "biologiceskom";
00086   const std::string lTRStr = "San Francisco Uluslararası Havalimanı";
00087   const std::string lTRTranslitStr = lTransliterator.normalise (lTRStr);
00088   const std::string lTRTargetStr = "san francisco uluslararası havalimanı";
00089
00090   // Africa - Middle-East
00091   const std::string lARStr = "    ";
00092   const std::string lARTranslitStr = lTransliterator.normalise (lARStr);
```

```
00093   const std::string lARTargetStr = "mtar san fransyskw aldwly";
00094   const std::string lFAStr = "    ";
00095   const std::string lFATranslitStr = lTransliterator.normalise (lFAStr);
00096   const std::string lFATargetStr = "frwdgah byn almlly sanfransyskw";
00097
00098   // Asia
00099   const std::string lZHStr = "";
00100   const std::string lZHTranslitStr = lTransliterator.normalise (lZHStr);
00101   const std::string lZHTargetStr = "jiu jin shan guo ji ji chang";
00102   const std::string lJPStr = "";
00103   const std::string lJPTranslitStr = lTransliterator.normalise (lJPStr);
00104   const std::string lJPTargetStr = "sanfuranshisuko guo ji kong gang";
00105   const std::string lKOStr = " ";
00106   const std::string lKOTranslitStr = lTransliterator.normalise (lKOStr);
00107   const std::string lKOTargetStr = "saenpeulansiseuko gugjegonghang";
00108   const std::string lTHStr = "";
00109   const std::string lTHTranslitStr = lTransliterator.normalise (lTHStr);
00110   const std::string lTHTargetStr = "tha xakasyan nana chat san f ran ss ko";
00111   const std::string lVIStr = "Sân bay quc t San Francisco";
00112   const std::string lVITranslitStr = lTransliterator.normalise (lVIStr);
00113   const std::string lVITargetStr = "san bay quoc te san francisco";
00114
00115   // French (accentuated Latin)
00116   logOutputFile << "[FR] " << lFRStr << " -> " << lFRTranslitStr << std::endl;
00117   BOOST_CHECK_MESSAGE (lFRTranslitStr == lFRTargetStr,
00118                        "The French transliterated string for '" << lFRStr
00119                        << "' should be '" << lFRTargetStr << "'. "
00120                        << "However, it is '" << lFRTranslitStr << "'.");
00121
00122   // Hungarian (accentuated Latin)
00123   logOutputFile << "[HU] " << lHUStr << " -> " << lHUTranslitStr << std::endl;
00124   BOOST_CHECK_MESSAGE (lHUTranslitStr == lHUTargetStr,
00125                        "The Hungarian transliterated string for '" << lHUStr
00126                        << "' should be '" << lHUTargetStr << "'. "
00127                        << "However, it is '" << lHUTranslitStr << "'.");
00128
00129   // Greek
00130   logOutputFile << "[GR] " << lGRStr << " -> " << lGRTranslitStr << std::endl;
00131   BOOST_CHECK_MESSAGE (lGRTranslitStr == lGRTargetStr,
00132                        "The Greek transliterated string for '" << lGRStr
00133                        << "' should be '" << lGRTargetStr << "'. "
00134                        << "However, it is '" << lGRTranslitStr << "'.");
00135
00136   // Russian (Cyrillic)
00137   logOutputFile << "[RU] " << lRUStr << " -> " << lRUTranslitStr << std::endl;
00138   BOOST_CHECK_MESSAGE (lRUTranslitStr == lRUTargetStr,
00139                        "The Russian transliterated string for '" << lRUStr
00140                        << "' should be '" << lRUTargetStr << "'. "
00141                        << "However, it is '" << lRUTranslitStr << "'.");
00142
00143   // Ukrainian (Cyrillic)
00144   logOutputFile << "[UK] " << lUKStr << " -> " << lUKTranslitStr << std::endl;
00145   BOOST_CHECK_MESSAGE (lUKTranslitStr == lUKTargetStr,
00146                        "The Ukrainian transliterated string for '" << lUKStr
00147                        << "' should be '" << lUKTargetStr << "'. "
00148                        << "However, it is '" << lUKTranslitStr << "'.");
00149
00150   // Turkish
00151   logOutputFile << "[TR] " << lTRStr << " -> " << lTRTranslitStr << std::endl;
00152   BOOST_CHECK_MESSAGE (lTRTranslitStr == lTRTargetStr,
00153                        "The Turkish transliterated string for '" << lTRStr
00154                        << "' should be '" << lTRTargetStr << "'. "
00155                        << "However, it is '" << lTRTranslitStr << "'.");
00156
00157   // Arabic
00158   logOutputFile << "[AR] " << lARStr << " -> " << lARTranslitStr << std::endl;
00159   BOOST_CHECK_MESSAGE (lARTranslitStr == lARTargetStr,
00160                        "The Arabic transliterated string for '" << lARStr
00161                        << "' should be '" << lARTargetStr << "'. "
00162                        << "However, it is '" << lARTranslitStr << "'.");
00163
00164   // Iranian (Farsi)
00165   logOutputFile << "[FA] " << lFAStr << " -> " << lFATranslitStr << std::endl;
00166   BOOST_CHECK_MESSAGE (lFATranslitStr == lFATargetStr,
00167                        "The Arabic transliterated string for '" << lFAStr
00168                        << "' should be '" << lFATargetStr << "'. "
00169                        << "However, it is '" << lFATranslitStr << "'.");
00170
00171   // Chinese (Han)
00172   logOutputFile << "[ZH] " << lZHStr << " -> " << lZHTranslitStr << std::endl;
00173   BOOST_CHECK_MESSAGE (lZHTranslitStr == lZHTargetStr,
00174                        "The Chinese transliterated string for '" << lZHStr
00175                        << "' should be '" << lZHTargetStr << "'. "
00176                        << "However, it is '" << lZHTranslitStr << "'.");
00177
00178   // Japanese (Katakana)
00179   logOutputFile << "[JP] " << lJPStr << " -> " << lJPTranslitStr << std::endl;
```

```
00180   BOOST_CHECK_MESSAGE (lJPTranslitStr == lJPTargetStr,
00181                        "The Japanese transliterated string for '" << lJPStr
00182                        << "' should be '" << lJPTargetStr << "'. "
00183                        << "However, it is '" << lJPTranslitStr << "'.");
00184
00185   // Korean (Hangul)
00186   logOutputFile << "[KO] " << lKOStr << " -> " << lKOTranslitStr << std::endl;
00187   BOOST_CHECK_MESSAGE (lKOTranslitStr == lKOTargetStr,
00188                        "The Korean transliterated string for '" << lKOStr
00189                        << "' should be '" << lKOTargetStr << "'. "
00190                        << "However, it is '" << lKOTranslitStr << "'.");
00191
00192   // Thai
00193   logOutputFile << "[TH] " << lTHStr << " -> " << lTHTranslitStr << std::endl;
00194   BOOST_CHECK_MESSAGE (lTHTranslitStr == lTHTargetStr,
00195                        "The Korean transliterated string for '" << lTHStr
00196                        << "' should be '" << lTHTargetStr << "'. "
00197                        << "However, it is '" << lTHTranslitStr << "'.");
00198
00199   // Vietnamese
00200   logOutputFile << "[VI] " << lVIStr << " -> " << lVITranslitStr << std::endl;
00201   BOOST_CHECK_MESSAGE (lVITranslitStr == lVITargetStr,
00202                        "The Korean transliterated string for '" << lVIStr
00203                        << "' should be '" << lVITargetStr << "'. "
00204                        << "However, it is '" << lVITranslitStr << "'.");
00205
00206
00207   // Close the Log outputFile
00208   logOutputFile.close();
00209 }
00210
00211 // End the test suite
00212 BOOST_AUTO_TEST_SUITE_END()
00213
```