Base Library

Overview

The Base library is a library for C++ that allows you to create your own modules in the QC4OpenX framework. With it, it's very easy to create your own CheckerBundles or ReportModules. The library consists of two parts, which are described in the following sections. The source code and an exemplary CheckerBundle can be found in folder `examples/src`.

The Configuration Format (`config_format`)

Configurations contain all necessary input parameters and parameterization for a CheckerBundle. Configurations can be created by the user and stored persistently as an XML file. A CheckerBundle should accept as single parameter an XML file containing the corresponding configuration (see User defined modules). The configuration XML file follows the schema file `doc/schema/config_format.xsd`.

To comfortably read out configurations, corresponding classes have been defined in the Base library. Reading out a configuration is simple, as shown in the following snippet.

```
cConfiguration configuration;
cConfiguration::ParseFromXML(&pMyConfiguration, strFilepath))
```

A configuration file can contain the parameters of multiple CheckerBundles and ReportModules. Therefore you should extract those parameters which correspond to your own bundle/module. For this we query the corresponding part from the configuration by the method GetCheckerBundleByName. Parameterization can thus be easily transferred from one parameter container (cParameterContainer) to another. Parameterization are nothing more than mappings from a keyword to a value and therefore allow easy overwriting and reuse.

```
cConfigurationCheckerBundle* checkerBundleConfig =
configuration.GetCheckerBundleByName("myChecker");
if (nullptr != checkerBundleConfig)
{
   inputParams.Overwrite(checkerBundleConfig->GetParams());
}
```

The Report Format (`report_format`)

The results that a Checker or CheckerBundle defines as output are stored in a report file. Reports contain information about the defects found, which are called issues. At least a description and an identifier is assigned to an issue. Additional meta information can be added to an issue: file location (cFileLocation), XML file location (cXMLLocation) or road information (cRoadLocation). This information is relevant for the ReportModule, for example, to point out meaningful errors in a GUI. The report XML file has to follow the schema file doc/schema/xqar_report_format.xsd.

```
myChecker->AddIssue(new cIssue("errorDescription", ERROR_LVL))
myChecker->AddIssue(new cIssue("take a closer look", INFO_LVL,
(cExtendedInformation*) new cFileLocation(3, 0, "This is row 3, column 0.")));
```

Issues are always assigned to exactly one checker. A log level is assigned to an issue: if it represents an error: ERROR_LVL, if it's just an information: INFO_LVL.

All CheckerBundles must contain at least one Checker. Checkers can be created or queried using a factory pattern:

```
auto pXSDCheckerBundle = new cCheckerBundle("myCheckerBundle");
cChecker* pOSCVersionChecker = pXSDCheckerBundle-
>CreateChecker("xoscVersionChecker", "Checks the validity of an xosc version.");
pOSCVersionChecker->AddIssue(new cIssue("Version not supported.", ERROR_LVL));
```

In a ResultContainer, all CheckerBundles are added. The ResultContainer is responsible for reading, writing and managing the data.

```
pResultContainer->AddCheckerBundle(pXSDCheckerBundle);
pResultContainer->WriteResults("resultFile"));
```

All objects in a ResultContainer are managed automatically. A release of the objects is not necessary and can be done with a `cResultContainer::Clear` or in the destructor.

Mapping `XMLLocation` to `FileLocation`

- **XmlLocation**: Addressing in an XML file based on an XPath expression
- **FileLocation**: A reference to a file with row and column

Some issues include a XML location to indicate where the issue occurred. During result pooling, a file location is determined from the XML location and saved as well.

The definition of the XML location is based on a XPath expression. Almost the complete XPath 2. 0 Standard is supported.

The XML processing is based on Qt 5. Therefore there are minor restrictions to the standard: Qt 5 Restrictions.