//-----------------------------------------------------------------------------

Edit in TCppProgrammingGuidelines.docx

//-----------------------------------------------------------------------------

/\*! \page PAGE\_CPP\_PROG\_GUIDE **@@Programming\_guide\_for\_this\_project@@**

\if english

# \section CPG\_INTRO Introduction

In order to facilitate joint development in C++, it makes sense if programming guidelines are followed. Here are the most important guidelines for this C++ project.

\endif

\if german

# \section CPG\_INTRO Einleitung

Um eine gemeinsame Entwicklung in C++ zu erleichtern, ist es sinnvoll, wenn Programmierrichtlinien eingehalten werden. Hier sind die wichtigsten Vorgaben für dieses C++-Projekt aufgeführt.

\endif

\if english

# \section CPG\_NAMES Identifier

Uniform rules apply to the individual types of identifiers.

<br>

In principle, the English language should be used for all identifiers.

\endif

\if german

# \section CPG\_NAMES Bezeichn**er**

Für die einzelnen Typen von Bezeichnern gelten einheitliche Regeln.

<br>

Grundsätzlich soll für alle Bezeichner die englischer Sprache genutzt werden.

\endif

\if english

## \subsection CPG\_NAMES\_FILE file names

As a rule, each class should be declared in a header file and defined in a source code file (exceptions are template classes, inline methods and structs). Both files get the class name as name.

<br>

If several classes are declared in a header file and defined in a source code file, a meaningful module name should be chosen for the files.

\endif

\if german

## \subsection CPG\_NAMES\_FILE Dateinamen

In der Regel soll jede Klasse in einer Header-Datei deklariert und in einer Quelltext-Datei definiert werden (ausnahmen sind Template-Klassen, Inline-Methoden und Structs). Beide Dateien bekommen die Klassenbezeichnung als Namen.

<br>

Sollten mehrere Klassen in einer Header-Datei deklariert und in einer Quelltext-Datei definiert werden, sollte ein aussagekräftiger Modulname für die Dateien gewählt werden.

\endif

\if english

## \subsection CPG\_NAMES\_FILE\_EXT C++ file extensions

C++ header files and global type files get the extension <b>**.h**</b>.

<br>

C++ source code files get the extension <b>**.cpp**</b>.

\endif

\if german

## \subsection CPG\_NAMES\_FILE\_EXT C++ Dateiextensions

C++ Header-Dateien und globale Typ-Dateien erhalten die Extension <b>**.h**</b>.

<br>

C++ Quelltext-Dateien erhalten die Extension <b>**.cpp**</b>.

\endif

\if english

## \subsection CPG\_NAMES\_CLASS class names

Class names are always introduced with a capital letter at the beginning.

<br>

To signal an assignment to a project or library, a maximum of 3 letters can be used as abbreviations. The first of these must be an uppercase letter and the other two lowercase letters. The following class name starts again with an uppercase letter. The name should be chosen so that the class's job is clear. If it is a name consisting of several words, they should be written individually according to the CamelCase notation with leading capital letters ( e.g. <b>**'GloBaseMaker'**</b>).

<br>

Template classes get a <b>'T'</b> inserted before their name, after the project or library abbreviation (e.g. <b>**'GloTOndemandSet'**</b>).

<br>

Alternatively, an assignment to a project or to a library can be realized via a namespace. For the designation of the namespace only 3 lowercase letters should be used as an abbreviation (e.g. glo::BaseMaker). However, the repetition of an abbreviation befor the class name should then be avoided.

\endif

\if german

## \subsection CPG\_NAMES\_CLASS Klassen-Namen

Klassen-Namen werden grundsätzlich am Anfang mit einem Großbuchstaben eingeleitet.

<br>

Um eine Zuordnung zu einem Projekt bzw. zu einer Bibliothek zu signalisieren, können maximal 3 Buchstaben als Kürzel genutzt werden. Davon muss der erste ein Groß- und die weiteren beiden Kleinbuchstaben sein. Die folgende Klassenbezeichnung fängt wieder mit einem Großbuchstaben an. Der Name sollte so gewählt werden, dass der Job der Klasse klar wird. Wenn es sich um eine Bezeichnung aus mehreren Wörtern handelt, sind diese einzeln entsprechend der CamelCase-Notation mit anführenden Großbuchstaben zu schreiben ( z.B. <b>**'GloBaseMaker'**</b>).

<br>

Template-Klassen bekommen vor ihren Namen, nach dem Projekt- bzw. Bibliothek-Kürzel ein <b>**'T'**</b> eingefügt (z.B. <b>**'GloTOndemandSet'**</b>).

<br>

Alternativ kann eine Zuordnung zu einem Projekt bzw. zu einer Bibliothek über einen Namensraum (namespace) realisiert werden. Für die Bezeichnung des Namensraums sollten maximal 3 Kleinbuchstaben als Kürzel genutzt werden ( z.B. <b>**'glo::BaseMaker'**</b>). Die Wiederholung eines Projektkürzels vor dem Klassen-Namen sollte dann aber vermieden werden.

\endif

\if english

## \subsection CPG\_NAMES\_MEMBER\_ATTR Attribute names in classes

The attributes of classes are always introduced with a <b>**'m\_'**</b> (stands for member). This is then followed by a controversial and not entirely consistent marking according to the <b>**Hungarian notation**</b> followed by a meaningful name that begins with a capital letter (for example, <b>**'m\_sApplicationName'**</b> for a string).

**\note** Static attributes are not marked separately.

\endif

\if german

## \subsection CPG\_NAMES\_MEMBER\_ATTR Attribut-Bezeichnung **in** Klassen

Die Attribute von Klassen werden grundsätzlich mit einem <b>**'m\_'**</b> (steht für Member) eingeleitet. Es folgt dann eine umstrittene und nicht ganz durchzuhaltende Kennzeichnung nach der <b>**ungarischen Notation**</b> gefolgt von einem aussagekräftigen Namen, welcher mit einem Großbuchstaben anfängt (z.B. <b>**'m\_sApplicationName'**</b> für einen String).

**\note** Statische Attribute werden nicht gesondert gekennzeichnet.

\endif

\if english

## \subsection CPG\_NAMES\_MEMBER\_METHOD Designation of methods in classes

The methods of classes are always introduced with a lowercase letter. Usually a verb should be used which describes the service of the method (e.g. <b>**'loadIconPath'**</b>. If the status of a class is to be queried, it has proven to be useful to start the method name with an 'is' or 'has' (e.g. <b>**'isStored'**</b> or <b>**'hasConnection'**</b>

**\note** Static methods are not marked separately.

\endif

\if german

## \subsection CPG\_NAMES\_MEMBER\_METHOD Methoden-Bezeichnung **in** Klassen

Die Methoden von Klassen werden grundsätzlich mit einem Kleinbuchstaben eingeleitet. Es sollte in der Regel ein Verb benutzt werden, welches die Dienstleistung der Methode beschreibt (z.B. <b>**'loadIconPath'**</b>. Wenn Zustände einer Klasse abgefragt werden sollen, hat sich bewährt, die Methodenbezeichnung mit einem 'is' oder 'has' zu beginnen (z.B. <b>**'isStored'**</b> oder <b>**'hasConnection'**</b>

**\note** Statische Methoden werden nicht gesondert gekennzeichnet.

\endif

\if english

## \subsection CPG\_NAMES\_GLOBAL Global variables

Global variables are always introduced with a <b>**'g\_'**</b> (stands for global). This is then followed by a controversial and not entirely consistent marking according to the <b>**Hungarian notation**</b> followed by a meaningful name that begins with a capital letter (e.g. <b>**'g\_sGloVersion'**</b> for a string).

\endif

\if german

## \subsection CPG\_NAMES\_GLOBAL Globale Variablen

Globale Variablen werden grundsätzlich mit einem <b>**'g\_'**</b> (steht für global) eingeleitet. Es folgt dann eine umstrittene und nicht ganz durchzuhaltende Kennzeichnung nach der <b>**ungarischen Notation**</b> gefolgt von einem aussagekräftigen Namen, welcher mit einem Großbuchstaben anfängt (z.B. <b>**'g\_sGloVersion'**</b> für einen String).

\endif

\if english

## \subsection CPG\_NAMES\_LOCAL Local variables in functions and methods

Local variables are always introduced with a <b>**'t\_'**</b> (stands for temporary). This is then followed by a controversial and not entirely consistent marking according to the <b>**Hungarian notation**</b> followed by a meaningful name starting with a capital letter (e.g. <b>**'t\_iErr'**</b> for a local Int variable).

\endif

\if german

## \subsection CPG\_NAMES\_LOCAL Lokale Variablen in Funktionen und Methoden

Lokale Variablen werden grundsätzlich mit einem <b>**'t\_'**</b> (steht für temporär) eingeleitet. Es folgt dann eine umstrittene und nicht ganz durchzuhaltende Kennzeichnung nach der <b>**ungarischen Notation**</b> gefolgt von einem aussagekräftigen Namen, welcher mit einem Großbuchstaben anfängt (z.B. <b>**'t\_iErr'**</b> für eine lokale Int-Variable).

\endif

\if english

## \subsection CPG\_NAMES\_PARAM Parameter designation

Parameters are usually introduced with a controversial and not entirely consistent marking according to the <b>Hungarian notation</b>, followed by a meaningful name starting with a capital letter (e.g. <b>**'eLockMode'**</b> for an enum value).

\endif

\if german

## \subsection CPG\_NAMES\_PARAM Parameter-Bezeichnung

Parameter werden in der Regel mit einer umstrittenen und nicht ganz durchzuhaltende Kennzeichnung nach der <b>**ungarischen Notation**</b> eingeleitet, gefolgt von einem aussagekräftigen Namen, welcher mit einem Großbuchstaben anfängt (z.B. <b>**'eLockMode'**</b> für einen Enum-Wert).

\endif

\if english

# \section CPG\_FILE Files

To simplify the search for content, some file types are predefined or suggested.

\endif

\if german

# \section CPG\_FILE Dateien

Um die Suche nach Inhalten zu vereinfachen, sind einige Datei-Typen vorgegeben bzw. vorgeschlagen.

\endif

\if english

## \subsection CPG\_FILE\_CLASS Classes

As already mentioned under \ref CPG\_NAMES\_FILE "**File names**". Usually, each class should be declared in a header file and defined in a source code file (exceptions are template classes, inline methods and structs). Both files get the class name as name.

<br>

If several classes are declared in a header file and defined in a source code file, a meaningful module name should be chosen.

**\sa** \ref CPG\_NAMES\_FILE "**File names**" and \ref CPG\_NAMES\_FILE\_EXT C++ "**File extensions**"

\endif

\if german

## \subsection CPG\_FILE\_CLASS Klassen

Wie schon unter \ref CPG\_NAMES\_FILE "**Dateinamen**" erwähnt. In der Regel soll jede Klasse in einer Header-Datei deklariert und in einer Quelltext-Datei definiert werden (ausnahmen sind Template-Klassen, Inline-Methoden und Structs). Beide Dateien bekommen die Klassenbezeichnung als Namen.

<br>

Sollten mehrere Klassen in einer Header-Datei deklariert und in einer Quelltext-Datei definiert werden, sollte ein aussagekräftiger Modulname gewählt werden.

**\sa** \ref CPG\_NAMES\_FILE "**Dateinamen**" und \ref CPG\_NAMES\_FILE\_EXT C++ "**Dateiextensions**"

\endif

\if english

## \subsection CPG\_FILE\_TYPEDEF Global project types

The global constants, enumerations and structures are stored in a header file. The name of this file should be the project or library abbreviation with a following 'Types' ( e.g. <b>**'GloTypes'**</b>).

<br>

If constants, enumerations and structures only affect one class, you must always consider whether they belong in the respective classes.

\endif

\if german

## \subsection CPG\_FILE\_TYPEDEF Globale Projekttypen

Die globalen Konstanten, Enumerationen und Strukturen werden in einer Header-Datei untergebracht. Der Name dieser Datei sollte das Projekt- bzw. Bibliothekskürzel mit einem folgenden 'Types' sein ( z.B. <b>**'GloTypes'**</b>).

<br>

Wenn Konstanten, Enumerationen und Strukturen nur eine Klasse betreffen, ist immer abzuwägen, ob diese nicht in die jeweilige Klassen gehören.

\endif

\if english

## \subsection CPG\_FILE\_ERRORS Global error codes

Global error codes for a project are stored in a header file. The name of this file should be the project or library abbreviation with a following 'Errors' ( e.g. <b>**'GloErrors'**</b>).

\endif

\if german

## \subsection CPG\_FILE\_ERRORS Globale Fehlerkodes

Für ein Projekt globale Fehlerkodes werden in einer Header-Datei untergebracht. Der Name dieser Datei sollte das Projekt- bzw. Bibliothekskürzel mit einem folgenden 'Errors' sein ( z.B. <b>**'GloErrors'**</b>).

\endif

\if english

## \subsection CPG\_FILE\_TODO **T**o-do list

A ToList <b>**'ToDo.txt'**</b> is placed in the respective C ++ project directory. In this to-do list, individual requirements and things still to be done are listed with priority according to the manner of a Scrum backlog.

\endif

\if german

## \subsection CPG\_FILE\_TODO ToDo-Liste

Eine ToListe <b>**'ToDo.txt'**</b> kommt in das jeweilige C++ Projektverzeichnis. In dieser ToDo-Liste werden nach Manier eines Scrum-Backlogs einzelne Anforderungen und noch zu erledigende Dinge priorisiert aufgelistet.

\endif

\if english

# \section CPG\_FORMAT Formatting

In order to simplify the recognition of content, some formatting is predefined or suggested.

\endif

\if german

# \section CPG\_FORMAT Formattierung

Um das Erkennen von Inhalten zu vereinfachen, sind einige Formatierungen vorgegeben bzw. vorgeschlagen.

\endif

\if english

## \subsection CPG\_FORMAT\_UTF8 UTF8

Every text file with special characters must be saved in UTF8 format so that the special characters are handled uniformly.

\endif

\if german

## \subsection CPG\_FORMAT\_UTF8 UTF8

Jede Textdatei mit Sonderzeichen muss im UTF8-Format gespeichert sein, damit die Sonderzeichen einheitlich behandelt werden.

\endif

\if english

## \subsection CPG\_FORMAT\_INDENTS Indentations

Each logical section in the source code should be indented. Indentations are in 2 character increments. This indentation should not be defined as tab characters.

\endif

\if german

## \subsection CPG\_FORMAT\_INDENTS Einrückungen

Jeder logische Abschnitt im Quelltext sollte eingerückt sein. Einrückungen erfolgen in Schritten von 2 Zeichen. Diese Einrückung soll nicht als Tabulatorzeichen definiert sein.

\endif

\if english

## \subsection CPG\_FORMAT\_BRACES Curly brackets

The characters <b>**'{'**</b> and <b>**'}'**</b> always stand alone in a line. Pairs of brackets are always in one column. It is recommended to also bracket single-line statements in <b>**'{'**</b> and <b>**'}'**</b>.

Example:

\endif

\if german

## \subsection CPG\_FORMAT\_BRACES Geschweifte Klammern

Die Zeichen <b>**'{'**</b> und <b>**'}'**</b> stehen immer alleine in einer Zeile. Klammerpaare stehen immer in einer Spalte. Es wird empfohlen, auch einzeilige Anweisungen in <b>**'{'**</b> und <b>**'}'**</b> einzuklammern.

Beispiel:

\endif

\code{.cpp}

int foo::bar( int iValue )

{

int t\_iErr = 0;

if ( iValue )

{

t\_iErr = this->doSomething( iValue );

}

return t\_iErr;

}

\endcode

\if english

## \subsection CPG\_FORMAT\_GROUPING Grouping

To support clarity, logical units are divided by comments and interruptions with lines.

Example:

\endif

\if german

## \subsection CPG\_FORMAT\_GROUPING Gruppierung

Um die Übersichtlichkeit zu unterstützen, werden logische Einheiten durch Kommentare und Unterbrechungen mit Linien eingeteilt.

Beispiel:

\endif

\code{.unparsed}

...

protected:

//============== Attributes

//-------------------------------------------------------------------------

/\*!

\if english

Objectpointer.

\endif

\if german

Zeiger auf Objekt.

\endif

\*/

T \* m\_pT;

//-------------------------------------------------------------------------

/\*!

\if english

The reference count.

\endif

\if german

Der Referenzzähler.

\endif

\*/

EuSharedPtrRefCount \* m\_pRefCount;

//-------------------------------------------------------------------------

public:

//============= Constructors, destructor

//-------------------------------------------------------------------------

/\*!

\if english

Default constructor can take object.

\param[in] pT Pointer doe object (optional).

\endif

\if german

Standard-Konstruktor kann Objekt übernehmen.

\param[in] pT Zeiger auf Objekt (optional)

\endif

\*/

EuSharedPtr(T \* pT = NULL);

//-------------------------------------------------------------------------

...

\endcode

\if english

## \subsection CPG\_FORMAT\_LINELENGTH Line length

The line length should not exceed 80 characters. There are cases where this cannot be adhered to.

<br>

Recommendations for line breaks if the line length is exceeded.

- Parameter transfer after a comma. If you do, it is best to transfer all parameters into one column:

\code{.cpp}

int foo::bar( int iParamName1,

int iParamName2,

int iParamName3,

int iParamName4,

int iParamName5,

int iParamName6 )

\endcode

- For arithmetic expressions before an operator:

\code{.cpp}

int iResult = ( ( iParamName1 + iParamName2 + iParamName3 + iParamName4 )

\* iParamName5 ) - iParamName6;

\endcode

- In the case of an extensive if statement or while and do-while loops:

\code{.cpp}

if ( ( iParamName1 && iParamName2 )

|| ( iParamName3 && iParamName4 )

|| ( iParamName5 && iParamName6 )

{

...

}

\endcode

\endif

\if german

## \subsection CPG\_FORMAT\_LINELENGTH Zeilenlänge

Die Zeilenlänge sollte 80 Zeichen nicht überschreiten. Es gibt Fälle, wo dieses nicht eingehalten werden kann.

<br>

Empfehlungen für Zeilenumbrüche, wenn die Zeilenlänge überschritten wird.

- Parameterübergabe nach einem Komma. Wenn schon, dann am besten alle Parameter in eine Spalte:

\code{.cpp}

int foo::bar( int iParamName1,

int iParamName2,

int iParamName3,

int iParamName4,

int iParamName5,

int iParamName6 )

\endcode

- Bei arithmetischen Ausdrücken vor einem Operator:

\code{.cpp}

int iResult = ( ( iParamName1 + iParamName2 + iParamName3 + iParamName4 )

\* iParamName5 ) - iParamName6;

\endcode

- Bei einer umfangreichen if-Anweisung bzw. while- und do-while-Schleifen:

\code{.cpp}

if ( ( iParamName1 && iParamName2 )

|| ( iParamName3 && iParamName4 )

|| ( iParamName5 && iParamName6 ) )

{

...

}

\endcode

\endif

\if english

## \subsection CPG\_FORMAT\_VARDECL Declaration of variables

Individual logically related declarations and assignments should be in the same column.

\endif

\if german

## \subsection CPG\_FORMAT\_VARDECL Variablen Deklaration

Einzelne logisch zusammenhängende Deklarationen und Zuweisungen sollten in der selben Spalte stehen.

\endif

\code{.cpp}

int t\_iMyValue1,

t\_iMySecondValue;

MyClass \* t\_pMyClass = 0;

t\_iMyValue1 = 0;

t\_iMySecondValue = 0;

t\_pMyClass = new MyClass();

\endcode

\if english

# \section CPG\_COMMENT Comments

Comments and documentation are necessary to keep the source code understandable.

<br>

Comments in the source code should be written in English or translated if possible. Block or line comments can be used as required.

<br>

The documentation should be available in several languages and meet the requirements of <b>doxygen</b>.

**\sa** \ref CPG\_DOCU\_DOXYGEN "**Documentation with doxygen**".

\endif

\if german

# \section CPG\_COMMENT Kommentare

Kommentare und Dokumentationen sind notwendig, um den Quelltext verständlich zu halten.

<br>

Kommentare im Quelltext sollten in englischer Sprache verfasst sein bzw. bei Gelegenheit übersetzt werden. Je nach Bedarf können Block- oder Zeilenkommentare genutzt werden.

<br>

Die Dokumentation soll in mehreren Sprachen vorhanden sein und den Vorgaben von <b>**doxygen**</b> genügen.

**\sa** \ref CPG\_DOCU\_DOXYGEN "**Dokumentation mit doxygen**".

\endif

\if english

# \section CPG\_DOCU\_DOXYGEN Documentation with doxygen

The documentation of the C++ projects is realized with <b>doxygen</b>. Here <b>doxygen</b> is not described as such, but how and where it is used.

<br>

If there should be a language-specific documentation, the respective documentation text is enclosed between <b>**'\\if german'**</b> (for german) and <b>**'\\endif'**</b>. For the English language this would be, for example, <b>**'\\if english'**</b> and <b>**'\\endif'**</b>. This should make it possible for every programmer to use the respective OpenSource products or to work on the project, regardless of which language(s) he or she speaks.

\endif

\if german

# \section CPG\_DOCU\_DOXYGEN Dokumentation mit doxygen

Die Dokumentation der C++ Projekte wird mittels <b>**doxygen**</b> realisiert. Hier wird nicht <b>**doxygen**</b> als solches beschrieben, sondern wie und wo es angewendet wird.

<br>

Wenn es eine sprachspezifische Dokumentation geben soll, wird der jeweilige Dokumentationstexte zwischen <b>**'\\if german'**</b> (für deutsch) und <b>**'\\endif'**</b> eingefasst. Für die englische Sprache wäre das z.B. <b>**'\\if english'**</b> und <b>**'\\endif'**</b>. Das soll ermöglichen, dass jeder Programmierer und jede Programmiererin unabhängig davon welche Sprache(n) beherrscht werden, die jeweiligen OpenSource-Produkte nutzen bzw. am Projekt mitarbeiten können.

**\sa** <a href="http://www.doxygen.org" target="\_blank" rel="noopener">http://www.doxygen.org</a>

\endif

\if english

## \subsection CPG\_DOCU\_HFILE Header file

Each header file gets a hint about the license after the include blocker. A short description can be multilingual; the license text is always in English.

<br><br>

Example for class 'GloLocalThread'.

\endif

\if german

## \subsection CPG\_DOCU\_HFILE Header-Datei

Jede Header-Datei bekommt nach dem Include-Blocker einen Hinweis auf die Lizenz. Eine kurze Beschreibung kann mehrsprachig sein; der Lizenztext ist grundsätzlich englisch.

<br><br>

Beispiel für Klasse 'GloLocalThread'.

\endif

\code{.unparsed}

#ifndef INC\_GLOLOCALTHREAD\_H

#define INC\_GLOLOCALTHREAD\_H

//-----------------------------------------------------------------------------

/\*!

\file GloLocalThread.h

\brief

\if english

Header for GloLocalThread

\endif

\if german

Header für GloLocalThread

\endif

\author Helmut Jakoby

\copyright © 2020 Helmut Jakoby

<!-- @@TermsOfLicense BEGIN -->

<b>Terms of License</b>

This file is part of the GlobalObjects module.

GlobalObjects is free software:

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.

<b>GNU Affero General Public License Usage</b><br>

You can redistribute it and/or modify it under the terms of the GNU Affero

General Public License as published by the Free Software Foundation; either

version 3 of the License, or (at your option) any later version. Please review

the following information to ensure the GNU Affero General Public License

requirements will be met: hhttps://www.gnu.org/licenses/agpl-3.0.en.html .

<!-- @@TermsOfLicense END -->\*/

//-----------------------------------------------------------------------------

\endcode

\if english

The file and class name, the author name and copyright entry as well as the project or module name are adjusted.

\endif

\if german

Der Datei- und Klassenname, der Autorenname und der copyright-Eintrag sowie der Projekt- bzw. Modulname werden angepasst.

\endif

\if english

## \subsection CPG\_DOCU\_CLASS Classes

The class is described after any necessary forward declarations and includes directly above the class declaration.

<br><br>

Example for template class 'GloTOndemandSet':

\endif

\if german

## \subsection CPG\_DOCU\_CLASS Klassen

Nach den ggf. nötigen Forward-Deklarationen und Includes direkt über der Klassen-Deklaration wird die Klasse beschrieben.

<br><br>

Beispiel für Template-Klasse 'GloTOndemandSet':

\endif

\code{.unparsed}

...

#include "GloBasePersistent.h"

//-----------------------------------------------------------------------------

/\*! \brief

\if english

A type-safe set of references to persistent objects in the database (intended

as an attribute for persistent objects).

Unlike a GloTPointerSet (where the referenced objects are 'loaded'), the

GloTOndemandSet is a set of references to persistent objects using the object

ID. This means that the objects are not loaded from the database with the

referencing object, but can be reloaded if required.

It is always useful when an object references large objects that do not

necessarily have to be available in memory.

\endif

\if german

Ein typsicherer Set von Referenzen auf persistente Objekte in der Datenbank

(gedacht als Attribut für persistente Objekte).

Im Gegensatz zu einem GloTPointerSet (bei dem die referenzierten Objekte

'mitgeladen' werden), ist der GloTOndemandSet ein Set von Referenzen über

die Objekt-ID auf persistente Objekt. Es werden die Objekte also nicht mit

dem referenzierenden Objekt aus der Datenbank geladen, sondern können bei

Bedarf nachgeladen werden

Sinnvoller Einsatz ist immer dann gegeben, wenn ein Objekt große Objekte

referenziert, die nicht im jeden Fall im Speicher verfügbare sein müssen.

\endif

\copyright © 1995-2020 Helmut Jakoby

\author Helmut Jakoby

\*/

template < class T > class GloTOndemandSet : public GloOndemandSet

{

...

\endcode

\if english

## \subsection CPG\_DOCU\_ATTRIBUTE Attributes

The description text for attributes is described directly in front of it as shown in the example.

<br><br>

Example of an attribute 'GloObjID m\_RefObjID':

\endif

\if german

## \subsection CPG\_DOCU\_ATTRIBUTE Attribute

Der Beschreibungstext für Attribute wird direkt vor diesem wie im Beispiel dargestellt beschrieben.

<br><br>

Beispiel für ein Attribut 'GloObjID m\_RefObjID':

\endif

\code{.unparsed}

...

//============== Attribute

//-------------------------------------------------------------------------

/\*!

\if english

The reference to a persistent object as object ID.

\endif

\if german

Die Referenz auf ein persistentes Objekt als Objekt-ID.

\endif

\*/

GloObjID m\_RefObjID;

//-------------------------------------------------------------------------

...

\endcode

\if english

## \subsection CPG\_DOCU\_ENUMS Enumerations

The description text for enumerations is described as shown in the example.

<br><br>

Example 1 for an Enum 'EnWatchStyle':

\endif

\if german

## \subsection CPG\_DOCU\_ENUMS Aufzählungen (enums)

Der Beschreibungstext für Aufzählungen wird wie im Beispiel dargestellt beschrieben.

<br><br>

Beispiel 1 für eine Enum 'EnWatchStyle':

\endif

\code{.unparsed}

...

//============== Enums

//---------------------------------------------------------------------

/\*!

\if english

What is being observed, an AllSet or an object?

\endif

\if german

Was wird beobachtet, ein AllSet oder ein Objekt?

\endif

\*/

enum EnWatchStyle

{

eUNDEF = 0, /\*!<

\if english

Undefined.

\endif

\if german

Undefiniert

\endif

\*/

eALLSET, /\*!<

\if english

An AllSet will be observed

\endif

\if german

Es wird ein AllSet beobachtet

\endif

\*/

eOBJECT /\*!<

\if english

An object will be observed

\endif

\if german

Es wird ein Objekt beobachtet

\endif

\*/

};

...

\endcode

<br>

\if english

Example 2 for an Enum 'EnOpenClose':

\endif

\if german

Beispiel 2 für eine Enum 'EnOpenClose':

\endif

\code{.unparsed}

...

//============== Enums

//-------------------------------------------------------------------------

/\*!

\if english

The thread can open or close a database.

\endif

\if german

Der Thread kann eine Datenbank öffnen oder schliessen.

\endif

\*/

enum EnOpenClose

{

/\*!

\if english

The database will be opened.

\endif

\if german

Die Datenbank wird geöffnet.

\endif

\*/

eOpen,

/\*!

\if english

The database will be closed.

\endif

\if german

Die Datenbank wird geschlossen.

\endif

\*/

eClose

};

...

\endcode

\if english

## \subsection CPG\_DOCU\_METHOD Functions and methods

The description text for functions and methods is described as shown in the example.

\endif

\if german

## \subsection CPG\_DOCU\_METHOD Funktionen und Methoden

Der Beschreibungstext für Funktionen bzw. Methoden wird direkt vor dieser wie im Beispiel dargestellt beschrieben.

<br><br>

Beispiel für eine Methode 'insert(...)':

\endif

\code{.unparsed}

...

//-------------------------------------------------------------------------

/\*!

\if english

Inserts a persistent object into the set behind the current position. If

the set objects are locked by the set, the object to be inserted is locked

with the lock mode of the set. If the set objects are observed by the set,

the object to be inserted is observed with the observation mode of the set.

\param[in] spNewObj The persistent object to be inserted in a GloSharedPtr.

\return A return value < 0 indicates an error.

\endif

\if german

Fügt ein persistentes Objekt in die Set hinter der aktuellen Position ein.

Wenn die Setobjekte durch die Set gesperrt sind, wird das einzufügende

Objekt mit den Sperrmodus des Sets gesperrt. Wenn die Setobjekte durch

die Set beobachtet werden, wird das einzufügende Objekt mit den

Beobachtungsmodus des Sets beobachtet.

\param[in] spNewObj Das einzufügende persistente Objekt in einem GloSharedPtr.

\return Ein Rückgabewert < 0 zeigt einen Fehler an.

\endif

\*/

virtual int insert( GloSharedPtr<T> spNewObj );

//-------------------------------------------------------------------------

...

\endcode

\*/