# XML Mill

1.0.0

Generated by Doxygen 1.7.6.1

Sun Apr 21 2013 15:04:17

# Contents


**1 Goblin Coding's XML Mill** **1**

1.1 Introduction . . . . . 1

1.2 Download . . . . . 1

**2 Namespace Index** **3**

2.1 Namespace List . . . . . 3

**3 Class Index** **5**

3.1 Class List . . . . . 5

**4 Namespace Documentation** **7**

4.1 GCGlobalSpace Namespace Reference . . . . . 7

4.1.1 Detailed Description . . . . . 8

4.1.2 Function Documentation . . . . . 8

4.1.2.1 setShowHelpButtons . . . . . 8

4.1.2.2 setShowTreeItemsVerbose . . . . . 8

4.1.2.3 showHelpButtons . . . . . 8

4.1.2.4 showTreeItemsVerbose . . . . . 8

4.1.3 Variable Documentation . . . . . 9

4.1.3.1 APPLICATION . . . . . 9

4.1.3.2 FONT . . . . . 9

4.1.3.3 FONTSIZE . . . . . 9

4.1.3.4 ORGANISATION . . . . . 9

4.2 GCMessageSpace Namespace Reference . . . . . 9

4.2.1 Detailed Description . . . . . 10

4.2.2 Enumeration Type Documentation . . . . . 10

4.2.2.1 ButtonCombo . . . . . 10
4.2.2.2 Buttons . . . . . 10
4.2.2.3 Icon . . . . . 10
4.2.3 Function Documentation . . . . . 11
4.2.3.1 forgetAllPreferences . . . . . 11
4.2.3.2 showErrorMessageBox . . . . . 11
4.2.3.3 userAccepted . . . . . 11

**5 Class Documentation** **13**

5.1 GCAddItemsForm Class Reference . . . . . 13
5.1.1 Detailed Description . . . . . 13
5.1.2 Constructor & Destructor Documentation . . . . . 13
5.1.2.1 GCAddItemsForm . . . . . 14
5.1.2.2 ~GCAddItemsForm . . . . . 14
5.2 GCAddSnippetsForm Class Reference . . . . . 14
5.2.1 Detailed Description . . . . . 15
5.2.2 Constructor & Destructor Documentation . . . . . 15
5.2.2.1 GCAddSnippetsForm . . . . . 15
5.2.2.2 ~GCAddSnippetsForm . . . . . 15
5.2.3 Member Function Documentation . . . . . 16
5.2.3.1 snippetAdded . . . . . 16
5.3 GCBatchProcessorHelper Class Reference . . . . . 16
5.3.1 Detailed Description . . . . . 17
5.3.2 Constructor & Destructor Documentation . . . . . 17
5.3.2.1 GCBatchProcessorHelper . . . . . 18
5.3.3 Member Function Documentation . . . . . 18
5.3.3.1 associatedElementsToUpdate . . . . . 18
5.3.3.2 attributeKeysToUpdate . . . . . 19
5.3.3.3 attributeValuesToUpdate . . . . . 19
5.3.3.4 elementAttributesToUpdate . . . . . 20
5.3.3.5 elementChildrenToUpdate . . . . . 20
5.3.3.6 elementsToUpdate . . . . . 21
5.3.3.7 newAssociatedElementsToAdd . . . . . 21
5.3.3.8 newAttributeKeysToAdd . . . . . 22

5.3.3.9 newAttributeValuesToAdd . . . 22
5.3.3.10 newElementAttributesToAdd . . . 23
5.3.3.11 newElementChildrenToAdd . . . 23
5.3.3.12 newElementsToAdd . . . 24
5.4 GCComboBox Class Reference . . . 25
5.4.1 Detailed Description . . . 25
5.4.2 Constructor & Destructor Documentation . . . 25
5.4.2.1 GCComboBox . . . 25
5.4.3 Member Function Documentation . . . 25
5.4.3.1 focusInEvent . . . 25
5.4.3.2 focusOutEvent . . . 26
5.4.3.3 mousePressEvent . . . 26
5.5 GCDataBaseInterface Class Reference . . . 26
5.5.1 Detailed Description . . . 28
5.5.2 Member Function Documentation . . . 28
5.5.2.1 activeSessionName . . . 29
5.5.2.2 addDatabase . . . 29
5.5.2.3 addElement . . . 29
5.5.2.4 addRootElement . . . 29
5.5.2.5 attributes . . . 30
5.5.2.6 attributeValues . . . 30
5.5.2.7 batchProcessDomDocument . . . 30
5.5.2.8 children . . . 31
5.5.2.9 connectionList . . . 32
5.5.2.10 containsKnownRootElement . . . 32
5.5.2.11 hasActiveSession . . . 32
5.5.2.12 instance . . . 32
5.5.2.13 isDocumentCompatible . . . 33
5.5.2.14 isInitialised . . . 34
5.5.2.15 isProfileEmpty . . . 34
5.5.2.16 isUniqueChildElement . . . 34
5.5.2.17 knownElements . . . 34
5.5.2.18 knownRootElements . . . 35
5.5.2.19 lastError . . . 35

5.5.2.20 removeAttribute . . . 35
5.5.2.21 removeChildElement . . . 36
5.5.2.22 removeDatabase . . . 36
5.5.2.23 removeElement . . . 36
5.5.2.24 removeRootElement . . . 36
5.5.2.25 setActiveDatabase . . . 37
5.5.2.26 updateAttributeValues . . . 37
5.5.2.27 updateElementAttributes . . . 37
5.5.2.28 updateElementChildren . . . 38
5.6 GCDBSessionManager Class Reference . . . 39
5.6.1 Detailed Description . . . 39
5.6.2 Constructor & Destructor Documentation . . . 40
5.6.2.1 GCDBSessionManager . . . 40
5.6.2.2 ∼GCDBSessionManager . . . 40
5.6.3 Member Function Documentation . . . 40
5.6.3.1 activeDatabaseChanged . . . 40
5.6.3.2 addExistingDatabase . . . 40
5.6.3.3 addNewDatabase . . . 41
5.6.3.4 removeDatabase . . . 41
5.6.3.5 reset . . . 42
5.6.3.6 selectActiveDatabase . . . 42
5.7 GCDomTreeWidget Class Reference . . . 42
5.7.1 Detailed Description . . . 44
5.7.2 Constructor & Destructor Documentation . . . 44
5.7.2.1 GCDomTreeWidget . . . 44
5.7.2.2 ∼GCDomTreeWidget . . . 45
5.7.3 Member Function Documentation . . . 45
5.7.3.1 activeCommentValue . . . 45
5.7.3.2 addItem . . . 45
5.7.3.3 allTreeWidgetItems . . . 46
5.7.3.4 appendSnippet . . . 46
5.7.3.5 clearAndReset . . . 47
5.7.3.6 cloneDocument . . . 47
5.7.3.7 dropEvent . . . 47

5.7.3.8 findItemPositionAmongDuplicates . . . 48
5.7.3.9 gcCurrentItem . . . 48
5.7.3.10 gcCurrentItemChanged . . . 49
5.7.3.11 gcCurrentItemSelected . . . 49
5.7.3.12 getIncludedTreeWidgetItems . . . 49
5.7.3.13 insertItem . . . 50
5.7.3.14 isBatchProcessSuccess . . . 51
5.7.3.15 isCurrentItemRoot . . . 51
5.7.3.16 isDocumentCompatible . . . 52
5.7.3.17 isEmpty . . . 52
5.7.3.18 keyPressEvent . . . 53
5.7.3.19 matchesRootName . . . 53
5.7.3.20 populateFromDatabase . . . 53
5.7.3.21 rebuildTreeWidget . . . 54
5.7.3.22 replaceItemsWithComment . . . 54
5.7.3.23 rootName . . . 55
5.7.3.24 setActiveCommentValue . . . 55
5.7.3.25 setAllCheckStates . . . 56
5.7.3.26 setContent . . . 56
5.7.3.27 setCurrentItemWithIndexMatching . . . 56
5.7.3.28 setShowTreeItemsVerbose . . . 56
5.7.3.29 toString . . . 57
5.8 GCHelpDialog Class Reference . . . 57
5.8.1 Detailed Description . . . 57
5.8.2 Constructor & Destructor Documentation . . . 58
5.8.2.1 GCHelpDialog . . . 58
5.8.2.2 ~GCHelpDialog . . . 58
5.9 GCMainWindow Class Reference . . . 58
5.9.1 Detailed Description . . . 58
5.9.2 Constructor & Destructor Documentation . . . 59
5.9.2.1 GCMainWindow . . . 59
5.9.2.2 ~GCMainWindow . . . 59
5.9.3 Member Function Documentation . . . 59
5.9.3.1 closeEvent . . . 59

5.10 GCMessageDialog Class Reference . . . . . 59
5.10.1 Detailed Description . . . . . 60
5.10.2 Constructor & Destructor Documentation . . . . . 60
5.10.2.1 GCMessageDialog . . . . . 60
5.10.2.2 ~GCMessageDialog . . . . . 60
5.11 GCPlainTextEdit Class Reference . . . . . 60
5.11.1 Detailed Description . . . . . 61
5.11.2 Member Function Documentation . . . . . 62
5.11.2.1 clearAndReset . . . . . 62
5.11.2.2 commentOut . . . . . 62
5.11.2.3 findTextRelativeToDuplicates . . . . . 62
5.11.2.4 keyPressEvent . . . . . 62
5.11.2.5 manualEditAccepted . . . . . 62
5.11.2.6 selectedIndex . . . . . 62
5.11.2.7 setContent . . . . . 62
5.11.2.8 wrapText . . . . . 63
5.12 GCRemoveItemsForm Class Reference . . . . . 63
5.12.1 Detailed Description . . . . . 63
5.12.2 Constructor & Destructor Documentation . . . . . 63
5.12.2.1 GCRemoveItemsForm . . . . . 63
5.12.2.2 ~GCRemoveItemsForm . . . . . 64
5.13 GCRestoreFilesForm Class Reference . . . . . 64
5.13.1 Detailed Description . . . . . 64
5.13.2 Constructor & Destructor Documentation . . . . . 64
5.13.2.1 GCRestoreFilesForm . . . . . 65
5.13.2.2 ~GCRestoreFilesForm . . . . . 65
5.14 GCSearchForm Class Reference . . . . . 65
5.14.1 Detailed Description . . . . . 65
5.14.2 Constructor & Destructor Documentation . . . . . 66
5.14.2.1 GCSearchForm . . . . . 66
5.14.2.2 ~GCSearchForm . . . . . 66
5.14.3 Member Function Documentation . . . . . 66
5.14.3.1 foundItem . . . . . 66
5.15 GCTreeWidgetItem Class Reference . . . . . 66

5.15.1 Detailed Description . . . . . 68
5.15.2 Constructor & Destructor Documentation . . . . . 68
5.15.2.1 GCTreeWidgetItem . . . . . 68
5.15.2.2 GCTreeWidgetItem . . . . . 68
5.15.3 Member Function Documentation . . . . . 68
5.15.3.1 attributeIncluded . . . . . 68
5.15.3.2 element . . . . . 69
5.15.3.3 elementExcluded . . . . . 69
5.15.3.4 excludeAttribute . . . . . 69
5.15.3.5 fixAttributeValues . . . . . 70
5.15.3.6 fixedValue . . . . . 70
5.15.3.7 gcChild . . . . . 70
5.15.3.8 gcParent . . . . . 71
5.15.3.9 includeAttribute . . . . . 71
5.15.3.10 incrementAttribute . . . . . 71
5.15.3.11 index . . . . . 71
5.15.3.12 insertGcChild . . . . . 72
5.15.3.13 name . . . . . 73
5.15.3.14 rename . . . . . 73
5.15.3.15 revertToFixedValues . . . . . 73
5.15.3.16 setExcludeElement . . . . . 74
5.15.3.17 setIncrementAttribute . . . . . 74
5.15.3.18 setIndex . . . . . 74
5.15.3.19 setVerbose . . . . . 75
5.15.3.20 toString . . . . . 75
5.16 XmlSyntaxHighlighter Class Reference . . . . . 76
5.16.1 Detailed Description . . . . . 76

# Chapter 1

# Goblin Coding's XML Mill

## 1.1 Introduction

Please note that this is not a user manual or "Help" documentation, but rather source documentation intended for use by developers or parties interested in the code.

If you are a user and want to know more about the application and its uses itself, the official site contains all the relevant information about this application.

Please also feel free to contact me for any reason whatsoever.

## 1.2 Download

If you haven't yet, please see the download page for a list of possible download options.

If you find any bugs or errors in the code, or typo's in the documentation, please use the contact form to let me know.

# Chapter 2

# Namespace Index

## 2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

GCGlobalSpace
Contains values and functions used throughout the application . . . 7
GCMessageSpace . . . . . 9

# Chapter 3

# Class Index

## 3.1 Class List

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


GCAddItemsForm
: Allows the user to add elements and attributes to the active database 13

GCAddSnippetsForm
: Allows the user to add whole snippets to the active document . . . . 14

GCBatchProcessorHelper
: Helper class assisting with batch updates to the database . . . . . 16

GCComboBox
: A custom combo box providing additional user selection information 25

GCDataBaseInterface
: Provides a Singleton interface to the SQLite databases used to profile XML documents . . . . . 26

GCDBSessionManager
: Responsible for managing database connections and active database sessions . . . . . 39

GCDomTreeWidget
: Specialist tree widget class consiting of GCTreeWidgetItems . . . . 42

GCHelpDialog
: Displays "Help" information . . . . . 57

GCMainWindow
: The main application window class . . . . . 58

GCMessageDialog
: Provides a user dialog prompt with the option to save the user's preference . . . . . 59

GCPlainTextEdit
: Specialist text edit class for displaying XML content in the XML Mill context . . . . . 60

GCRemoveItemsForm
: Allows the user to remove items from the active database . . . . . 63

GCRestoreFilesForm
Displays recovered files so that the user may decide whether or not he/she wants to save them . . . . . 64
GCSearchForm
Search through the current document for specific text . . . . . 65
GCTreeWidgetItem
Used in GCDomTreeWidget, each GCTreeWidgetItem can be associated with a QDomElement . . . . . 66
XmlSyntaxHighlighter
Original class was obtained here: http://qt.gitorious.org/qt/qt/blobs/HEAD/examples/xmlpatterns/shared/xmlsyntaxhighlighh . . . . . 76

# Chapter 4

# Namespace Documentation

## 4.1 GCGlobalSpace Namespace Reference

Contains values and functions used throughout the application.

### Functions

- bool showHelpButtons ()

  *Used by various forms to determine whether or not they must display their "Help" tool buttons.*
- void setShowHelpButtons (bool show)

  *Saves the user's "Help" button preference to the registry/ini/xml.*
- bool showTreeItemsVerbose ()

  *Used by GCTreeWidgetItem to determine whether or not it should show its element as "verbose".*
- void setShowTreeItemsVerbose (bool show)

  *Saves the user's tree item verbosity preference to the registry/ini/xml.*

### Variables

- const QString ORGANISATION = "William Hallatt"

  *Used when saving and loading settings to registry/XML/ini.*
- const QString APPLICATION = "XML Mill"

  *Used when saving and loading settings to registry/XML/ini.*
- const QString FONT = "Courier New"

  *Default font for displaying XML content (directly or via table and tree views).*
- const int FONTSIZE = 10

  *Default font size for displaying XML content (directly or via table and tree views).*

### 4.1.1 Detailed Description

Contains values and functions used throughout the application.

### 4.1.2 Function Documentation

#### 4.1.2.1 void GCGlobalSpace::setShowHelpButtons ( bool *show* )

Saves the user's "Help" button preference to the registry/ini/xml.

Definition at line 48 of file gcglobalspace.cpp.

#### 4.1.2.2 void GCGlobalSpace::setShowTreeItemsVerbose ( bool *show* )

Saves the user's tree item verbosity preference to the registry/ini/xml.

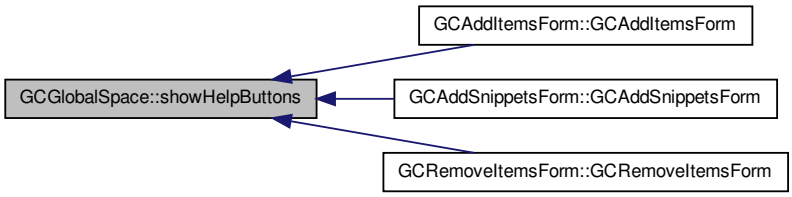
Definition at line 60 of file gcglobalspace.cpp.

#### 4.1.2.3 bool GCGlobalSpace::showHelpButtons ( )

Used by various forms to determine whether or not they must display their "Help" tool buttons.

Definition at line 42 of file gcglobalspace.cpp.

Here is the caller graph for this function:




#### 4.1.2.4 bool GCGlobalSpace::showTreeItemsVerbose ( )

Used by GCTreeWidgetItem to determine whether or not it should show its element as "verbose".

Definition at line 54 of file gcglobalspace.cpp.

### 4.1.3 Variable Documentation

#### 4.1.3.1 const QString GCGlobalSpace::APPLICATION = "XML Mill"

Used when saving and loading settings to registry/XML/ini.

Definition at line 62 of file gcglobalspace.h.

#### 4.1.3.2 const QString GCGlobalSpace::FONT = "Courier New"

Default font for displaying XML content (directly or via table and tree views).

Definition at line 67 of file gcglobalspace.h.

#### 4.1.3.3 const int GCGlobalSpace::FONTSIZE = 10

Default font size for displaying XML content (directly or via table and tree views).

Definition at line 70 of file gcglobalspace.h.

#### 4.1.3.4 const QString GCGlobalSpace::ORGANISATION = "William Hallatt"

Used when saving and loading settings to registry/XML/ini.

Definition at line 59 of file gcglobalspace.h.

## 4.2 GCMessageSpace Namespace Reference

### Enumerations

- enum Icon { NoIcon, Information, Warning, Critical, Question }

  *Determines the type of icon that will be set on the message dialog.*
- enum ButtonCombo { OKOnly, YesNo, OKCancel }

  *Determines the combination of buttons that will be shown.*
- enum Buttons { **Yes**, **No**, **OK**, **Cancel** }

  *Represents the individual buttons available.*

### Functions

- bool userAccepted (const QString &uniqueMessageKey, const QString &heading, const QString &text, ButtonCombo buttons, Buttons defaultButton, Icon icon=NoIcon, bool saveCancel=true)

  *This function will return the saved user preference (if there is one), or prompt the user for a decision and return the user's choice.*
- void forgetAllPreferences ()

*Deletes all saved dialog preferences from the registry/XML/ini files.*

- void showErrorMessageBox (QWidget *parent, const QString &message)

  *Displays a modal error message box with "message".*

### 4.2.1 Detailed Description

Responsible for the display of error messages and messages requiring user input with the option to remember the user's preference. Some message prompts displayed via this namespace contain the option to remember the user's preference. In cases where a user preference can be saved, GCMessageSpace will persist the changes to whatever medium exists on the platform it's running on (Windows registry, Mac XML, Unix ini).

This space is furthermore responsible for the display of all error messages, but not ALL messages (some messages always need to be shown and make more sense implemented in their respective classes)

### 4.2.2 Enumeration Type Documentation

#### 4.2.2.1 enum GCMessageSpace::ButtonCombo

Determines the combination of buttons that will be shown.

**Enumerator:**

***OKOnly*** Only the "OK" button should be made available.

***YesNo*** The buttons shown should be "Yes" and "No".

***OKCancel*** The buttons shown should be "OK" and "Cancel".

Definition at line 60 of file gcmessagespace.h.

#### 4.2.2.2 enum GCMessageSpace::Buttons

Represents the individual buttons available.

Definition at line 68 of file gcmessagespace.h.

#### 4.2.2.3 enum GCMessageSpace::Icon

Determines the type of icon that will be set on the message dialog.

**Enumerator:**

***NoIcon*** No icon will be shown.

***Information*** The message is of an informative nature.

***Warning*** The message is a warning.

***Critical*** The message contains critical information.

***Question*** The message is a question and requires user input.

Definition at line 50 of file gcmessagespace.h.

### 4.2.3 Function Documentation

#### 4.2.3.1 void GCMessageSpace::forgetAllPreferences ( )

Deletes all saved dialog preferences from the registry/XML/ini files.

Definition at line 222 of file gcmessagespace.cpp.

#### 4.2.3.2 void GCMessageSpace::showErrorMessageBox ( QWidget * *parent*, const QString & *message* )

Displays a modal error message box with "message".

Definition at line 231 of file gcmessagespace.cpp.

#### 4.2.3.3 bool GCMessageSpace::userAccepted ( const QString & *uniqueMessageKey*, const QString & *heading*, const QString & *text*, ButtonCombo *buttons*, Buttons *defaultButton*, Icon *icon* = `NoIcon`, bool *saveCancel* = `true` )

This function will return the saved user preference (if there is one), or prompt the user for a decision and return the user's choice.

**Parameters**

| | |
|---|---|
| *uniqueMessageKey* | - a unique name representing a specific message, this name is saved to the registry/xml/ini file |
| *heading* | - the message box header |
| *text* | - the actual message text |
| *buttons* | - the buttons that should be displayed for this particular message |
| *defaultButton* | - the button that should be highlighted as the default |
| *icon* | - the icon associated with this particular message |
| *saveCancel* | - if this value is set to "false", "Cancel"-ed user preferences will not be saved, irrespective of whether or not the user ticked the relevant box. |

Definition at line 157 of file gcmessagespace.cpp.

# Chapter 5

# Class Documentation

## 5.1 GCAddItemsForm Class Reference

Allows the user to add elements and attributes to the active database.

`#include <gcadditemsform.h>`

### Public Member Functions

- GCAddItemsForm (QWidget *parent=0)

  *Constructor.*
- ~GCAddItemsForm ()

  *Destructor.*

### 5.1.1 Detailed Description

Allows the user to add elements and attributes to the active database.

This form allows the user to add new elements and their associated attributes to the database. Although only one element can be added at a time (with or without attributes), all an element's attributes can be provided in one go through simply ensuring that each attribute appears on its own line in the input text edit.

The user will also be allowed to continue adding elements until "Done" is selected.

Finally, the Qt::WA_DeleteOnClose flag is set for all instances of this form. If you're not familiar with Qt, this means that Qt will delete this widget as soon as the widget accepts the close event (i.e. you don't need to worry about clean-up of dynamically created instances of this object).

Definition at line 54 of file gcadditemsform.h.

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 GCAddItemsForm::GCAddItemsForm ( QWidget * *parent* = 0 ) [explicit]

Constructor.

Definition at line 44 of file gcadditemsform.cpp.

Here is the call graph for this function:


#### 5.1.2.2 GCAddItemsForm::~GCAddItemsForm ( )

Destructor.

Definition at line 70 of file gcadditemsform.cpp.

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

- gcadditemsform.h
- gcadditemsform.cpp

## 5.2 GCAddSnippetsForm Class Reference

Allows the user to add whole snippets to the active document.

`#include <gcaddsnippetsform.h>`

### Signals

- void snippetAdded (GCTreeWidgetItem *, QDomElement)

  *Informs the listener that a new snippet has been added.*

### Public Member Functions

- GCAddSnippetsForm (const QString &elementName, GCTreeWidgetItem *parentItem, QWidget *parent=0)

  *Constructor.*
- ~GCAddSnippetsForm ()

  *Destructor.*

### 5.2.1 Detailed Description

Allows the user to add whole snippets to the active document.

This form allows the user to add multiple XML snippets of the same structure to the current document (with whichever default values the user specifies). It furthermore allows for the option to increment the default values for each snippet (i.e. if the user specifies "1" as an attribute value with the option to increment, then the next snippet generated will have "2" as the value for the same attribute and so on and so forth (strings will have the incremented value appended to the name). Only one element of each type can be inserted into any specific snippet as it makes no sense to insert multiple elements of the same type - for those use cases the user must create a smaller snippet subset.

Also, the Qt::WA_DeleteOnClose flag is set for all instances of this form. If you're unfamiliar with Qt, this means that Qt will delete this widget as soon as the widget accepts the close event (i.e. you don't need to worry about clean-up of dynamically created instances of this object).

Definition at line 60 of file gcaddsnippetsform.h.

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 GCAddSnippetsForm::GCAddSnippetsForm ( const QString & *elementName*, GCTreeWidgetItem * *parentItem*, QWidget * *parent* = 0 ) [explicit]

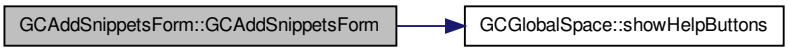
Constructor.

**Parameters**

| | |
|---|---|
| *elementName* | - the name of the element that will form the basis of the snippet, i.e. this element will be at the top of the snippet's DOM hierarchy. |
| *parentItem* | - the tree item and corresponding element in the active document to which the snippet will be added. |

Definition at line 48 of file gcaddsnippetsform.cpp.

Here is the call graph for this function:




#### 5.2.2.2 GCAddSnippetsForm::~GCAddSnippetsForm ( )

Destructor.

Definition at line 77 of file gcaddsnippetsform.cpp.

### 5.2.3 Member Function Documentation

#### 5.2.3.1 void GCAddSnippetsForm::snippetAdded ( GCTreeWidgetItem *, QDomElement ) [signal]

Informs the listener that a new snippet has been added.

The GCTreeWidgetItem thus emitted is the item to which the snippet must be added and the QDomElement is the element corresponding to this item. As always, we dependon QDomElement's shallow copy constructor and the GCTreeWidgetItem thus emitted is not owned by this class.

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

- gcaddsnippetsform.h
- gcaddsnippetsform.cpp

## 5.3 GCBatchProcessorHelper Class Reference

Helper class assisting with batch updates to the database.

`#include <gcbatchprocessorhelper.h>`

### Classes

- struct **ElementRecord**

  *Represents a single element's associated first level children, attributes and known attribute values.*

### Public Member Functions

- GCBatchProcessorHelper (const QDomDocument *domDoc, const QString &stringSeparator, const QStringList &knownElements, const QStringList &knownAttributes)

  *Constructor.*
- const QVariantList & newElementsToAdd () const

  *Returns a list of all the new element names that should be added to the database.*
- const QVariantList & newElementChildrenToAdd () const

  *Returns a list of lists of all new first level child element namess that should be added to the database.*
- const QVariantList & newElementAttributesToAdd () const

  *Returns a list of lists of all new associated attribute names that should be added to the database.*

- const QVariantList & elementsToUpdate () const

  *Returns a list of all the element names that should be updated.*
- const QVariantList & elementChildrenToUpdate () const

  *Returns a list of lists of all first level child element names corresponding to existing elements that should be updated.*
- const QVariantList & elementAttributesToUpdate () const

  *Returns a list of lists of all associated attribute names corresponding to existing elements that should be updated.*
- const QVariantList & newAttributeKeysToAdd () const

  *Returns a list of all the new attribute names that should be added to the database.*
- const QVariantList & newAssociatedElementsToAdd () const

  *Returns a list of all the new associated element names that should be added to the database.*
- const QVariantList & newAttributeValuesToAdd () const

  *Returns a list of lists of all new attribute values that should be added to the database.*
- const QVariantList & attributeKeysToUpdate () const

  *Returns a list of all the attribute keys that should be updated.*
- const QVariantList & associatedElementsToUpdate () const

  *Returns a list of all the associated element names corresponding to existing attribute keys that should be updated.*
- const QVariantList & attributeValuesToUpdate () const

  *Returns a list of lists of all attribute values associated with existing attributes that should be updated.*

### 5.3.1 Detailed Description

Helper class assisting with batch updates to the database.

The purpose of this class is to (1) extract all the elements and their associated attributes and attribute values from the DOM document passed in as parameter to the constructor and (2) to consolidate the lot into QVariantLists that can be used as bind variables for prepared queries intended to be executed in batches (that's quite a mouthful, see "execBatch" in the Qt documentation for more information on this topic).

The idea is not really to have a long-lived instance of this object in the calling object (i.e. it isn't intended to be used as a member variable, although it isn't prevented either), but rather to create a scoped local variable that should be created and set up as follows: Create an instance. Call the getters to retrieve the bind variable lists.

This class has also been specifically designed to be used in conjunction with GCDatabaseInterface.

Definition at line 57 of file gcbatchprocessorhelper.h.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 GCBatchProcessorHelper::GCBatchProcessorHelper ( const QDomDocument * *domDoc,* const QString & *stringSeparator,* const QStringList & *knownElements,* const QStringList & *knownAttributes* )

Constructor.

**Parameters**

| | |
|---|---|
| *domDoc* | - the DOM document from which all information will be extracted. |
| *string-Separator* | - the string sequence by which list elements in the database are separated (attribute values are stored as lists in the database, separated by a special character sequence. In other words, although the database sees a list of attribute values as a single string, we can extract the list elements later if we know which string sequence was used in the creation of the string list). This value should be unusual and unique. |
| *known-Elements* | - the list of elements known to the active database. If empty, all the elements in the DOM will be assumed to be new. |
| *known-Attributes* | - the list of attributes known to the active database, if empty, all the attributes in the DOM will be assumed to be new. |

Definition at line 35 of file gcbatchprocessorhelper.cpp.

### 5.3.3 Member Function Documentation

#### 5.3.3.1 const QVariantList & GCBatchProcessorHelper::associatedElementsToUpdate ( ) const

Returns a list of all the associated element names corresponding to existing attribute keys that should be updated.

All attributes are associated with specific elements (this allows us to save different values against attributes of the same name that are associated with different elements). Each item in this list is the specific element associated with an attribute in the "attribute keys to update" list.

**See also**

attributeKeysToUpdate()
attributeValuesToUpdate()

Definition at line 381 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:

#### 5.3.3.2 const QVariantList & GCBatchProcessorHelper::attributeKeysToUpdate ( ) const

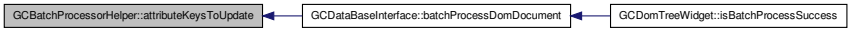
Returns a list of all the attribute keys that should be updated.

**See also**

associatedElementsToUpdate()
attributeValuesToUpdate()

Definition at line 374 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:




#### 5.3.3.3 const QVariantList & GCBatchProcessorHelper::attributeValuesToUpdate ( ) const

Returns a list of lists of all attribute values associated with existing attributes that should be updated.

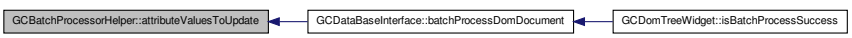
Each item in this list is a list of known attribute values corresponding to an attribute in the "attribute keys to update" list. In other words, for each item in the "attribute keys to update" list, there is a corresponding QVariant item in this list (with the same index number) that represents the list of the attribute's known values. Each QVariant consists of all these known values concatenated into a single string value with the individuals separated by the unique string separator that was passed in as constructor parameter.

**See also**

attributeKeysToUpdate()
associatedElementsToUpdate()

Definition at line 388 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:



#### 5.3.3.4 const QVariantList & GCBatchProcessorHelper::elementAttributesToUpdate ( ) const

Returns a list of lists of all associated attribute names corresponding to existing elements that should be updated.

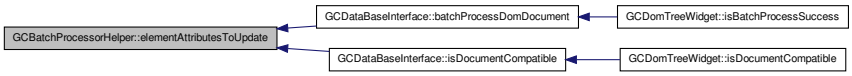
Each item in this list is a list of associated attributes corresponding to an element in the "elements to update" list. In other words, for each item in the "elements to update" list, there is a corresponding QVariant item in this list (with the same index number) that represents the list of the element's associated attributes. Each QVariant consists of all these associated attributes concatenated into a single string value with the individuals separated by the unique string separator that was passed in as constructor parameter. Where an element does not have associated attributes, a NULL QVariant value is added to the list to ensure that the indices of all lists are kept in synch.

**See also**

elementsToUpdate
elementChildrenToUpdate

Definition at line 347 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:




#### 5.3.3.5 const QVariantList & GCBatchProcessorHelper::elementChildrenToUpdate ( ) const

Returns a list of lists of all first level child element names corresponding to existing elements that should be updated.

Each item in this list is a list of first level child elements corresponding to an element in the "elements to update" list. In other words, for each item in the "elements to update" list, there is a corresponding QVariant item in this list (with the same index number) that represents the list of the element's first level children. Each QVariant consists of all these first level child elements concatenated into a single string value with the individuals separated by the unique string separator that was passed in as constructor parameter. Where an element does not have first level children, a NULL QVariant value is added to the list to ensure that the indices of all lists are kept in synch.

**See also**

elementsToUpdate
elementAttributesToUpdate

Definition at line 340 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:


#### 5.3.3.6 const QVariantList & GCBatchProcessorHelper::elementsToUpdate ( ) const

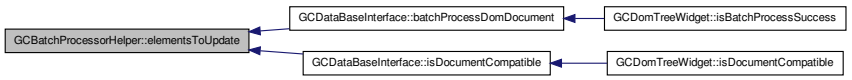
Returns a list of all the element names that should be updated.

**See also**

elementChildrenToUpdate
elementAttributesToUpdate

Definition at line 333 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:




#### 5.3.3.7 const QVariantList & GCBatchProcessorHelper::newAssociatedElementsToAdd ( ) const

Returns a list of all the new associated element names that should be added to the database.

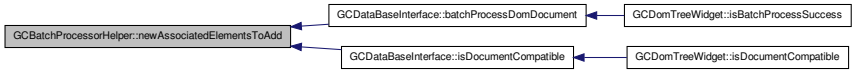
All attributes are associated with specific elements (this allows us to save different values against attributes of the same name that are associated with different elements). Each item in this list is the specific element associated with an attribute in the "new attribute keys to add" list.

**See also**

newAttributeKeysToAdd()
newAttributeValuesToAdd()

Definition at line 361 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:




#### 5.3.3.8 const QVariantList & GCBatchProcessorHelper::newAttributeKeysToAdd ( ) const

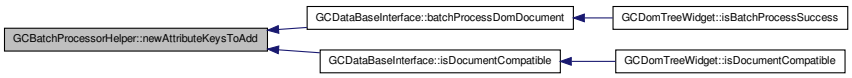
Returns a list of all the new attribute names that should be added to the database.

**See also**

newAssociatedElementsToAdd()
newAttributeValuesToAdd()

Definition at line 354 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:




#### 5.3.3.9 const QVariantList & GCBatchProcessorHelper::newAttributeValuesToAdd ( ) const

Returns a list of lists of all new attribute values that should be added to the database.

Each item in this list is a list of known attribute values corresponding to an attribute in the "new attribute keys to add" list. In other words, for each item in the "new attribute keys to add" list, there is a corresponding QVariant item in this list (with the same index number) that represents the list of the attribute's known values. Each QVariant consists of all these known values concatenated into a single string value with the individuals separated by the unique string separator that was passed in as constructor parameter.

**See also**

newAttributeKeysToAdd()
newAssociatedElementsToAdd()

Definition at line 367 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:


#### 5.3.3.10 const QVariantList & GCBatchProcessorHelper::newElementAttributesToAdd ( ) const

Returns a list of lists of all new associated attribute names that should be added to the database.

Each item in this list is a list of associated attributes corresponding to an element in the "new elements to add" list. In other words, for each item in the "new elements to add" list, there is a corresponding QVariant item in this list (with the same index number) that represents the list of the element's associated attributes. Each QVariant consists of all these associated attributes concatenated into a single string value with the individuals separated by the unique string separator that was passed in as constructor parameter. Where an element does not have associated attributes, a NULL QVariant value is added to the list to ensure that the indices of all lists are kept in synch.

**See also**

newElementsToAdd
newElementChildrenToAdd

Definition at line 326 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:


#### 5.3.3.11 const QVariantList & GCBatchProcessorHelper::newElementChildrenToAdd ( ) const

Returns a list of lists of all new first level child element namess that should be added to the database.

Each item in this list is a list of first level child elements corresponding to an element in the "new elements to add" list. In other words, for each item in the "new elements to add" list, there is a corresponding QVariant item in this list (with the same index)

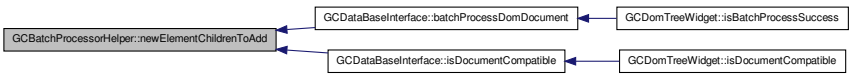
that represents the list of the element's first level children. Each QVariant consists of all these first level child elements concatenated into a single string value with the individuals separated by the unique string separator that was passed in as constructor parameter. Where an element does not have first level children, a NULL QVariant value is added to the list to ensure that the indices of all lists are kept in synch.

**See also**

newElementsToAdd
newElementAttributesToAdd

Definition at line 319 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:




#### 5.3.3.12 const QVariantList & GCBatchProcessorHelper::newElementsToAdd ( ) const

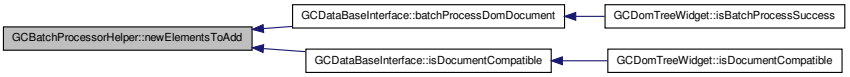
Returns a list of all the new element names that should be added to the database.

**See also**

newElementChildrenToAdd
newElementAttributesToAdd

Definition at line 312 of file gcbatchprocessorhelper.cpp.

Here is the caller graph for this function:




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

- gcbatchprocessorhelper.h
- gcbatchprocessorhelper.cpp

## 5.4 GCComboBox Class Reference

A custom combo box providing additional user selection information.

`#include <gccombobox.h>`

### Public Member Functions

- GCComboBox (QWidget *parent=0)

  *Constructor.*

### Protected Member Functions

- void mousePressEvent (QMouseEvent *e)

  *Re-eimplemented from QComboBox to emit the activated(int) signal.*
- void focusInEvent (QFocusEvent *e)

  *Re-eimplemented from QComboBox to emit the activated(int) signal.*
- void focusOutEvent (QFocusEvent *e)

  *Re-eimplemented from QComboBox to emit the currentIndexChanged(QString) signal.*

### 5.4.1 Detailed Description

A custom combo box providing additional user selection information.

The only reason this class exists is so that we may know when a combo box is activated. Initially I understood that the "activated" signal is emitted when a user clicks on a QComboBox (e.g. when the dropdown is expanded), but it turns out that this is not the case.

Definition at line 41 of file gccombobox.h.

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 GCComboBox::GCComboBox ( QWidget * *parent* = `0` ) `[explicit]`

Constructor.

Definition at line 33 of file gccombobox.cpp.

### 5.4.3 Member Function Documentation

#### 5.4.3.1 void GCComboBox::focusInEvent ( QFocusEvent * *e* ) `[protected]`

Re-eimplemented from QComboBox to emit the activated(int) signal.

Definition at line 48 of file gccombobox.cpp.

#### 5.4.3.2 void GCComboBox::focusOutEvent ( QFocusEvent * *e* ) [protected]

Re-eimplemented from QComboBox to emit the currentIndexChanged(QString) signal.

Definition at line 56 of file gccombobox.cpp.

#### 5.4.3.3 void GCComboBox::mousePressEvent ( QMouseEvent * *e* ) [protected]

Re-eimplemented from QComboBox to emit the activated(int) signal.

Definition at line 40 of file gccombobox.cpp.

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

- gccombobox.h
- gccombobox.cpp

## 5.5 GCDataBaseInterface Class Reference

Provides a Singleton interface to the SQLite databases used to profile XML documents.

`#include <gcdatabaseinterface.h>`

### Public Slots

- bool setActiveDatabase (const QString &dbName)

  *Sets the database connection corresponding to "dbName" as the active database.*
- bool addDatabase (const QString &dbName)

  *Adds "dbName" to the list of known database connections.*
- bool removeDatabase (const QString &dbName)

  *Removes "dbName" from the list of known database connections.*

### Public Member Functions

- bool isInitialised () const
- bool batchProcessDomDocument (const QDomDocument *domDoc) const

  *Batch process an entire DOM document.*
- bool addElement (const QString &element, const QStringList &children, const QStringList &attributes) const

  *Adds a single new element to the active database.*
- bool addRootElement (const QString &root) const

  *Marks an element as a known document root element.*
- bool updateElementChildren (const QString &element, const QStringList &children, bool replace=false) const

  *Updates the list of known first level children associated with "element" by appending the new children to the existing list (nothing is deleted).*

- bool updateElementAttributes (const QString &element, const QStringList &attributes, bool replace=false) const

  *Updates the list of known attributes associated with "element" by appending the new attributes to the existing list (nothing is deleted).*
- bool updateAttributeValues (const QString &element, const QString &attribute, const QStringList &attributeValues, bool replace=false) const

  *Updates the list of known attribute values that is associated with "element" and its corresponding "attribute" by appending the new attribute values to the existing list (nothing is deleted).*
- bool removeElement (const QString &element) const

  *Removes "element" from the active database.*
- bool removeRootElement (const QString &element) const

  *Removes "element" from the list of known root elements for the active database.*
- bool removeChildElement (const QString &element, const QString &child) const

  *Removes "child" from the list of first level element children associated with "element" in the active database.*
- bool removeAttribute (const QString &element, const QString &attribute) const

  *Removes "attribute" from the list of attributes associated with "element" in the active database.*
- bool hasActiveSession () const

  *Returns "true" if an active database session exists, "false" if not.*
- bool isProfileEmpty () const

  *Returns "true" if the active database is empty, "false" if not.*
- bool containsKnownRootElement (const QString &dbName, const QString &root) const

  *Returns "true" if the database named "dbName" knows about "root".*
- bool isUniqueChildElement (const QString &parentElement, const QString &element) const

  *Returns true if "element" is a child of "parentElement" only (i.e.*
- bool isDocumentCompatible (const QDomDocument ∗doc) const

  *Recursively scans the "doc"'s element hierarchy to ensure that all the document's elements, element relationships and attributes are known to the active profile.*
- QStringList knownElements () const

  *Returns a sorted (case sensitive, ascending) list of all the element names known to the current database connection (the active session).*
- QStringList children (const QString &element) const

  *Returns a sorted (case sensitive, ascending) list of all the first level children associated with "element" in the active database, or an empty QStringList if unsuccessful/none exist.*
- QStringList attributes (const QString &element) const

  *Returns an UNSORTED list of all the attribute names associated with "element" in the active database (the reason this list is unsorted is that all the other lists are used to populate combo boxes, where ordering makes sense, but this particular list is used to populate a table), or an empty QStringList if unsuccessful/none exist.*
- QStringList attributeValues (const QString &element, const QString &attribute) const

*Returns a sorted (case sensitive, ascending) list of all the attribute values associated with "element" and its corresponding "attribute" in the active database or, an empty QStringList if unsuccessful/none exist.*

- QStringList knownRootElements () const

  *Returns a sorted (case sensitive, ascending) list of all the document root elements known to the the active database.*

- QStringList connectionList () const

  *Returns a list of all known database connections.*

- const QString & lastError () const

  *Returns the last known error message.*

- QString activeSessionName () const

  *Returns the active database session if one exists, or an empty string if not.*

### Static Public Member Functions

- static GCDataBaseInterface * instance ()

  *Singleton accessor.*

### 5.5.1 Detailed Description

Provides a Singleton interface to the SQLite databases used to profile XML documents.

This class is designed to set up and manage embedded SQLite databases used to profile XML documents. Databases created by this class will consist of three tables:

"xmlelements" - accepts element names as unique primary keys and associates two fields with each record: "children" represents all the first level children of the element in question and "attributes" contain all the attributes known to be associated with the element (these will be ALL the children and attribute names ever associated with any particular unique element name in any particular database so it is best not to mix vastly different XML profiles in the same database).

"xmlattributes" - accepts an attribute name as primary key and references the unique element it is known to be associated with as foreign key. Only one additional field exists for each record: "attributevalues" contains all the values ever associated with this particular attribute when assigned to the specific element it references as foreign key. In other words, if element "x" is known to have had attribute "y" associated with it, then "attributevalues" will contain all the values ever assigned to "y" when associated with "x" across all XML profiles stored in a particular database.

"rootelements" - consists of a single field containing all known root elements stored in a specific database. If more than one XML profile has been loaded into the database in question, the database will have all their root elements listed in this table.

Definition at line 66 of file gcdatabaseinterface.h.

### 5.5.2 Member Function Documentation

#### 5.5.2.1 QString GCDataBaseInterface::activeSessionName ( ) const

Returns the active database session if one exists, or an empty string if not.

**See also**

hasActiveSession

Definition at line 710 of file gcdatabaseinterface.cpp.

#### 5.5.2.2 bool GCDataBaseInterface::addDatabase ( const QString & *dbName* ) [slot]

Adds "dbName" to the list of known database connections.

Definition at line 963 of file gcdatabaseinterface.cpp.

Here is the caller graph for this function:


#### 5.5.2.3 bool GCDataBaseInterface::addElement ( const QString & *element*, const QStringList & *children*, const QStringList & *attributes* ) const

Adds a single new element to the active database.

This function does nothing if an element with the same name already exists.

**Parameters**

| | |
|---|---|
| *element* | - the unique element name |
| *children* | - a list of the element's first level child elements' names |
| *attributes* | - a list of all the element's associated attribute names. |

Definition at line 306 of file gcdatabaseinterface.cpp.

#### 5.5.2.4 bool GCDataBaseInterface::addRootElement ( const QString & *root* ) const

Marks an element as a known document root element.

This function does nothing if the root already exists in the relevant table.

**Parameters**

| | |
|---|---|
| *root* | - the name of the unique root element |

Definition at line 346 of file gcdatabaseinterface.cpp.

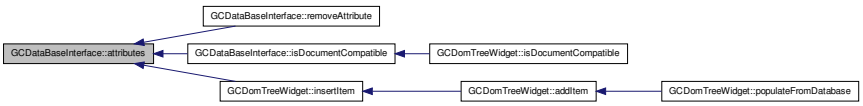
Here is the caller graph for this function:


#### 5.5.2.5 QStringList GCDataBaseInterface::attributes ( const QString & *element* ) const

Returns an UNSORTED list of all the attribute names associated with "element" in the active database (the reason this list is unsorted is that all the other lists are used to populate combo boxes, where ordering makes sense, but this particular list is used to populate a table), or an empty QStringList if unsuccessful/none exist.

Definition at line 844 of file gcdatabaseinterface.cpp.

Here is the caller graph for this function:




#### 5.5.2.6 QStringList GCDataBaseInterface::attributeValues ( const QString & *element*, const QString & *attribute* ) const

Returns a sorted (case sensitive, ascending) list of all the attribute values associated with "element" and its corresponding "attribute" in the active database or, an empty QStringList if unsuccessful/none exist.

Definition at line 865 of file gcdatabaseinterface.cpp.

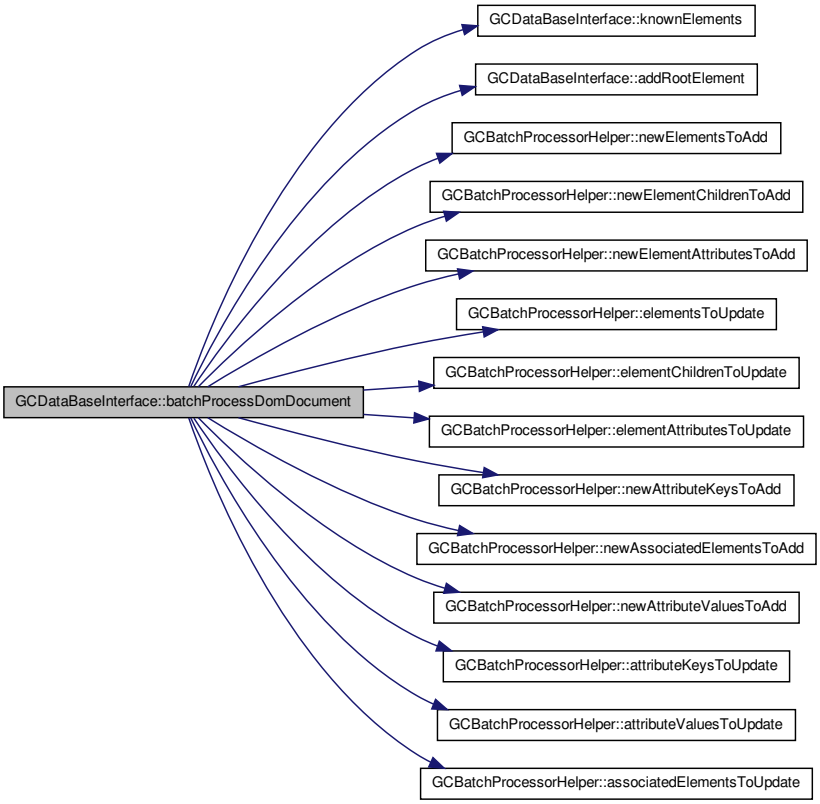
#### 5.5.2.7 bool GCDataBaseInterface::batchProcessDomDocument ( const QDomDocument * *domDoc* ) const

Batch process an entire DOM document.

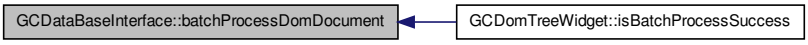
This function processes an entire DOM document by adding new (or updating existing) elements with their corresponding first level children and associated attributes and known attribute values to the active database in batches.

Definition at line 156 of file gcdatabaseinterface.cpp.

Here is the call graph for this function:




Here is the caller graph for this function:



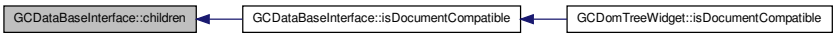

#### 5.5.2.8 QStringList GCDataBaseInterface::children ( const QString & *element* ) const

Returns a sorted (case sensitive, ascending) list of all the first level children associated with "element" in the active database, or an empty QStringList if unsuccessful/none

exist.

Definition at line 822 of file gcdatabaseinterface.cpp.

Here is the caller graph for this function:




#### 5.5.2.9 QStringList GCDataBaseInterface::connectionList ( ) const

Returns a list of all known database connections.

Definition at line 949 of file gcdatabaseinterface.cpp.

#### 5.5.2.10 bool GCDataBaseInterface::containsKnownRootElement ( const QString & *dbName,* const QString & *root* ) const

Returns "true" if the database named "dbName" knows about "root".

Definition at line 920 of file gcdatabaseinterface.cpp.

Here is the call graph for this function:


#### 5.5.2.11 bool GCDataBaseInterface::hasActiveSession ( ) const

Returns "true" if an active database session exists, "false" if not.

**See also**

activeSessionName()

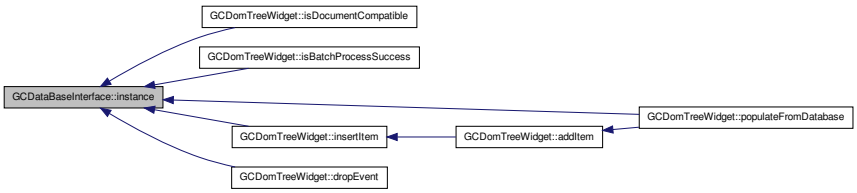
Definition at line 703 of file gcdatabaseinterface.cpp.

#### 5.5.2.12 GCDataBaseInterface * GCDataBaseInterface::instance ( ) [static]

Singleton accessor.

Definition at line 97 of file gcdatabaseinterface.cpp.

Here is the caller graph for this function:

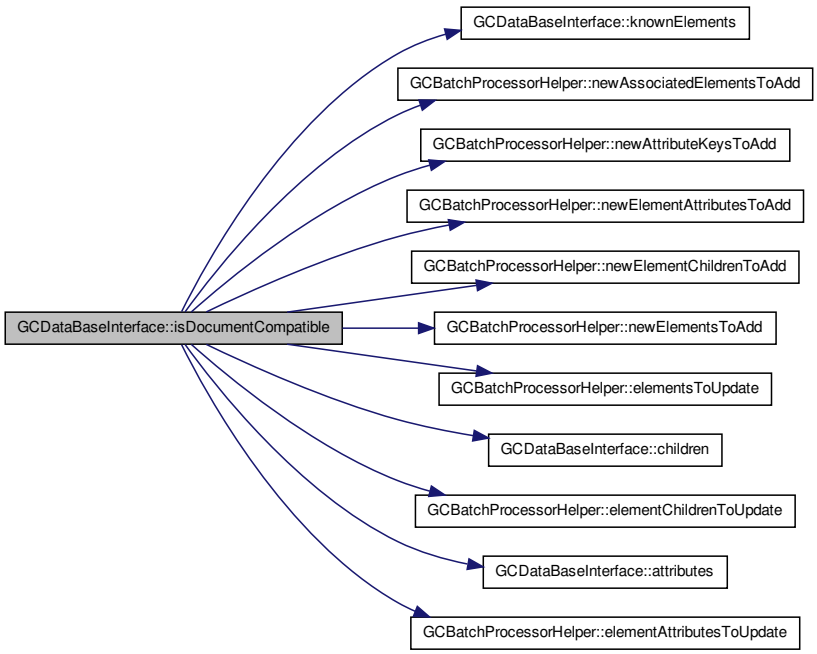



#### 5.5.2.13 bool GCDataBaseInterface::isDocumentCompatible ( const QDomDocument * *doc* ) const

Recursively scans the "doc"'s element hierarchy to ensure that all the document's elements, element relationships and attributes are known to the active profile.

Definition at line 747 of file gcdatabaseinterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:


#### 5.5.2.14 bool GCDataBaseInterface::isInitialised ( ) const

**Warning**

Call this function before using this interface for the first time to ensure that the known databases were initialised successfully.

Definition at line 149 of file gcdatabaseinterface.cpp.

#### 5.5.2.15 bool GCDataBaseInterface::isProfileEmpty ( ) const

Returns "true" if the active database is empty, "false" if not.

Definition at line 722 of file gcdatabaseinterface.cpp.

Here is the call graph for this function:


#### 5.5.2.16 bool GCDataBaseInterface::isUniqueChildElement ( const QString & *parentElement,* const QString & *element* ) const

Returns true if "element" is a child of "parentElement" only (i.e.

it doesn't exist in any other first level child list).

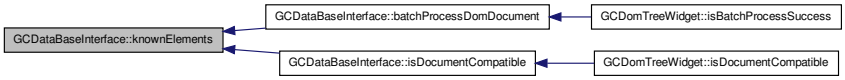
Definition at line 729 of file gcdatabaseinterface.cpp.

#### 5.5.2.17 QStringList GCDataBaseInterface::knownElements ( ) const

Returns a sorted (case sensitive, ascending) list of all the element names known to the current database connection (the active session).

Definition at line 802 of file gcdatabaseinterface.cpp.

Here is the caller graph for this function:

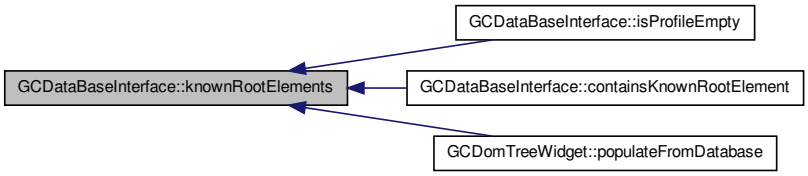



#### 5.5.2.18 QStringList GCDataBaseInterface::knownRootElements ( ) const

Returns a sorted (case sensitive, ascending) list of all the document root elements known to the the active database.

Definition at line 887 of file gcdatabaseinterface.cpp.

Here is the caller graph for this function:




#### 5.5.2.19 const QString & GCDataBaseInterface::lastError ( ) const

Returns the last known error message.

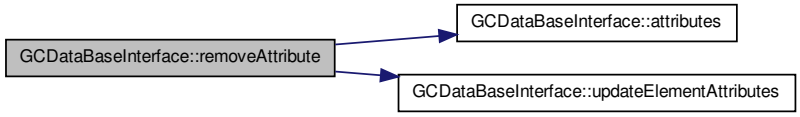
Definition at line 956 of file gcdatabaseinterface.cpp.

#### 5.5.2.20 bool GCDataBaseInterface::removeAttribute ( const QString & *element,* const QString & *attribute* ) const

Removes "attribute" from the list of attributes associated with "element" in the active database.

Definition at line 633 of file gcdatabaseinterface.cpp.

Here is the call graph for this function:

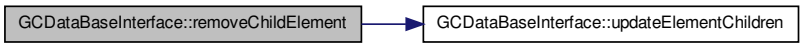



#### 5.5.2.21 bool GCDataBaseInterface::removeChildElement ( const QString & *element,* const QString & *child* ) const

Removes "child" from the list of first level element children associated with "element" in the active database.

Definition at line 615 of file gcdatabaseinterface.cpp.

Here is the call graph for this function:




#### 5.5.2.22 bool GCDataBaseInterface::removeDatabase ( const QString & *dbName* ) [slot]

Removes "dbName" from the list of known database connections.

Definition at line 1000 of file gcdatabaseinterface.cpp.

#### 5.5.2.23 bool GCDataBaseInterface::removeElement ( const QString & *element* ) const

Removes "element" from the active database.

Definition at line 583 of file gcdatabaseinterface.cpp.

#### 5.5.2.24 bool GCDataBaseInterface::removeRootElement ( const QString & *element* ) const

Removes "element" from the list of known root elements for the active database.

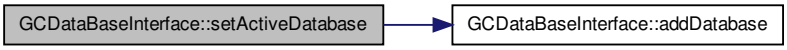
Definition at line 675 of file gcdatabaseinterface.cpp.

#### 5.5.2.25 bool GCDataBaseInterface::setActiveDatabase ( const QString & *dbName* ) [slot]

Sets the database connection corresponding to "dbName" as the active database.

Definition at line 1040 of file gcdatabaseinterface.cpp.

Here is the call graph for this function:




#### 5.5.2.26 bool GCDataBaseInterface::updateAttributeValues ( const QString & *element,* const QString & *attribute,* const QStringList & *attributeValues,* bool *replace* = false ) const

Updates the list of known attribute values that is associated with "element" and its corresponding "attribute" by appending the new attribute values to the existing list (nothing is deleted).

If "replace" is true, the existing values are replaced by those in the parameter list

**Parameters**

| | |
|---|---|
| *element* | - the unique name of the element to be updated |
| *attribute* | - the name of the associated attribute to te updated |
| *attribute-Values* | - a list of the attribute values associated with the attribute |
| *replace* | - if true, the attribute value list is replaced, if false, "attributeValues" is merged with the existing list. |

Definition at line 509 of file gcdatabaseinterface.cpp.

#### 5.5.2.27 bool GCDataBaseInterface::updateElementAttributes ( const QString & *element,* const QStringList & *attributes,* bool *replace* = false ) const

Updates the list of known attributes associated with "element" by appending the new attributes to the existing list (nothing is deleted).

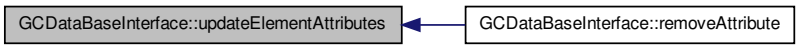
If "replace" is true, the existing values are replaced by those in the parameter list.

**Parameters**

| | |
|---|---|
| *element* | - the unique name of the element to be updated |
| *attributes* | - a list of the attribute names associated with the element |
| *replace* | - if true, the attribute list is replaced, if false, "attributes" is merged with the existing list. |

Definition at line 454 of file gcdatabaseinterface.cpp.

Here is the caller graph for this function:




#### 5.5.2.28 bool GCDataBaseInterface::updateElementChildren ( const QString & *element*, const QStringList & *children*, bool *replace* = `false` ) const

Updates the list of known first level children associated with "element" by appending the new children to the existing list (nothing is deleted).

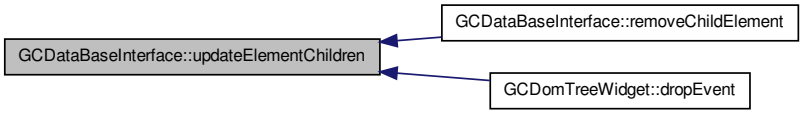
If "replace" is true, the existing values are replaced by those in the parameter list.

**Parameters**

| | |
|---|---|
| *element* | - the unique name of the element to be updated |
| *children* | - a list of the element's first level child elements' names |
| *replace* | - if true, the child list is replaced, if false, "children" is merged with the existing list. |

Definition at line 402 of file gcdatabaseinterface.cpp.

Here is the caller graph for this function:




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

- gcdatabaseinterface.h

- gcdatabaseinterface.cpp

## 5.6 GCDBSessionManager Class Reference

Responsible for managing database connections and active database sessions.

`#include <gcdbsessionmanager.h>`

### Public Slots

- void addExistingDatabase (const QString &currentRoot=QString())

  *Add an existing database from file.*
- void addNewDatabase (const QString &currentRoot=QString())

  *Create and add a new database.*

### Signals

- void activeDatabaseChanged (QString)

  *Emitted whenever the active database session is changed.*
- void reset ()

  *Emitted when the database change affects the current active document and informs the listener that the document must be reset.*

### Public Member Functions

- GCDBSessionManager (QWidget ∗parent=0)

  *Constructor.*
- ∼GCDBSessionManager ()

  *Destructor.*
- void selectActiveDatabase (const QString &currentRoot=QString())

  *Select a known database from the dropdown, or add a new or existing database from file.*
- void removeDatabase (const QString &currentRoot=QString())

  *Display the list of known databases that can be removed.*

### 5.6.1 Detailed Description

Responsible for managing database connections and active database sessions.

This class is responsible for managing database connections and active database sessions and will prompt the user to confirm actions or changes that may result in the current DOM doc being reset.

Definition at line 46 of file gcdbsessionmanager.h.

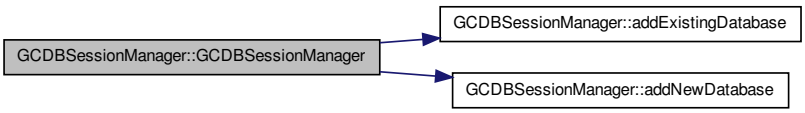
### 5.6.2 Constructor & Destructor Documentation

#### 5.6.2.1 GCDBSessionManager::GCDBSessionManager ( QWidget * *parent* = 0 ) [explicit]

Constructor.

Definition at line 39 of file gcdbsessionmanager.cpp.

Here is the call graph for this function:




#### 5.6.2.2 GCDBSessionManager::~GCDBSessionManager ( )

Destructor.

Definition at line 59 of file gcdbsessionmanager.cpp.

### 5.6.3 Member Function Documentation

#### 5.6.3.1 void GCDBSessionManager::activeDatabaseChanged ( QString ) [signal]

Emitted whenever the active database session is changed.

#### 5.6.3.2 void GCDBSessionManager::addExistingDatabase ( const QString & *currentRoot* = QString() ) [slot]

Add an existing database from file.

**Parameters**

| | |
|---|---|
| *currentRoot* | - used to determine whether or not the change will affect the active document (if not provided, the current document is assumed empty). |

Definition at line 102 of file gcdbsessionmanager.cpp.

Here is the caller graph for this function:


#### 5.6.3.3 void GCDBSessionManager::addNewDatabase ( const QString & *currentRoot* = QString() ) [slot]

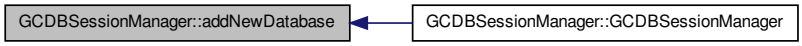
Create and add a new database.

**Parameters**

| | |
|---|---|
| *currentRoot* | - used to determine whether or not the change will affect the active document (if not provided, the current document is assumed empty). |

Definition at line 117 of file gcdbsessionmanager.cpp.

Here is the caller graph for this function:




#### 5.6.3.4 void GCDBSessionManager::removeDatabase ( const QString & *currentRoot* = QString() )

Display the list of known databases that can be removed.

**Parameters**

| | |
|---|---|
| *currentRoot* | - used to determine whether or not the change will affect the active document (if not provided, the current document is assumed empty). |

Definition at line 86 of file gcdbsessionmanager.cpp.

#### 5.6.3.5 void GCDBSessionManager::reset ( ) [signal]

Emitted when the database change affects the current active document and informs the listener that the document must be reset.

#### 5.6.3.6 void GCDBSessionManager::selectActiveDatabase ( const QString & *currentRoot* = QString() )

Select a known database from the dropdown, or add a new or existing database from file.

**Parameters**

| | |
|---|---|
| *currentRoot* | - used to determine whether or not the change will affect the active document (if not provided, the current document is assumed empty). |

Definition at line 66 of file gcdbsessionmanager.cpp.

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

- gcdbsessionmanager.h
- gcdbsessionmanager.cpp

## 5.7 GCDomTreeWidget Class Reference

Specialist tree widget class consiting of GCTreeWidgetItems.

`#include <gcdomtreewidget.h>`

### Public Slots

- void setCurrentItemWithIndexMatching (int index)

  *Finds the item with index matching "index" and sets it as the current tree item.*

### Signals

- void gcCurrentItemSelected (GCTreeWidgetItem *, int, bool)

  *Emitted when the current active item changes.*
- void gcCurrentItemChanged (GCTreeWidgetItem *, int)

  *Emitted when the current item's content changes.*

### Public Member Functions

- GCDomTreeWidget (QWidget *parent=0)

  *Constructor.*

- ~GCDomTreeWidget ()

  *Destructor.*
- GCTreeWidgetItem * gcCurrentItem () const

  *Returns the current item as a GCTreeWidgetItem.*
- void getIncludedTreeWidgetItems (QList< GCTreeWidgetItem * > &includedItems) const

  *Populates "includedItems" with all the GCTreeWidgetItems in the tree that have their "include" flags set.*
- const QList< GCTreeWidgetItem * > & allTreeWidgetItems () const

  *Returns a list of ALL the GCTreeWidgetItems currently in the tree.*
- int findItemPositionAmongDuplicates (const QString &nodeText, int itemIndex) const

  *Returns the position of "itemIndex" relative to that of ALL items matching "nodeText" (this is is not as odd as it sounds, it is possible that a DOM document may have multiple elements of the same name with matching attributes and attribute values).*
- QDomNode cloneDocument () const

  *Returns a deep copy of the underlying DOM document.*
- QString toString () const

  *Returns the DOM content as string.*
- QString rootName () const

  *Returns the name of the DOM document's root.*
- QString activeCommentValue () const

  *Returns the comment associated with the current element (if any).*
- void setActiveCommentValue (const QString &value)

  *Sets the value of the active comment node to "value".*
- bool setContent (const QString &text, QString *errorMsg=0, int *errorLine=0, int *errorColumn=0)

  *Sets the underlying DOM document's content.*
- bool isEmpty () const

  *Returns true if the widget and DOM is currently empty.*
- bool isCurrentItemRoot () const

  *Returns true if the current item is the one corresponding to the DOM document's root element.*
- bool matchesRootName (const QString &elementName) const

  *Returns true if "elementName" matches that of the DOM document's root.*
- bool isDocumentCompatible () const

  *Returns true if the underlying DOM document is compatible with the active DB session.*
- bool isBatchProcessSuccess () const

  *Returns true if batch processing of DOM content to the active DB was successful.*
- void rebuildTreeWidget ()

  *Rebuild the tree to conform to updated DOM content.*
- void appendSnippet (GCTreeWidgetItem *parentItem, QDomElement childElement)

  *Creates and adds tree widget items for each element in the parameter element hierarchy.*

- void replaceItemsWithComment (const QList< int > &indices, const QString &comment)

  *Removes the items with indices matching those in the parameter list from the tree as well as from the DOM document.*
- void updateItemNames (const QString &oldName, const QString &newName)

  *Update all the tree widget items with text "oldName" to text "newName".*
- void populateFromDatabase (const QString &baseElementName=QString())

  *This function starts the recursive process of populating the tree widget with items consisting of the element hierarchy starting at "baseElementName".*
- void addItem (const QString &element, bool toParent=false)

  *Adds a new item and corresponding DOM element node named "element".*
- void insertItem (const QString &elementName, int index, bool toParent=false)

  *Adds a new item and corresponding DOM element node named "elementName" and inserts the new tree widget item into position "index" of the current item.*
- void setAllCheckStates (Qt::CheckState state)

  *Iterates through the tree and sets all items' check states to "state".*
- void setShowTreeItemsVerbose (bool verbose)

  *Iterates through the tree and set all items' "verbose" flags to "show".*
- void clearAndReset ()

  *Clears and resets the tree as well as the underlying DOM document.*

### Protected Member Functions

- void dropEvent (QDropEvent *event)

  *Re-implemented from QTreeWidget.*
- void keyPressEvent (QKeyEvent *event)

  *Re-implemented from QTreeWidget.*

## 5.7.1 Detailed Description

Specialist tree widget class consiting of GCTreeWidgetItems.

This class wraps an underlying DOM document and manages its items (GCTreeWidgetItems) based on changes to the DOM, but also manages the DOM based on changes made to its items. Perhaps a better way of describing this relationship is to say that this class is, in effect, a non-textual visual representation of a DOM document.

Definition at line 49 of file gcdomtreewidget.h.

## 5.7.2 Constructor & Destructor Documentation

### 5.7.2.1 GCDomTreeWidget::GCDomTreeWidget ( QWidget * *parent* = 0 )

Constructor.

Definition at line 44 of file gcdomtreewidget.cpp.

#### 5.7.2.2 GCDomTreeWidget::~GCDomTreeWidget ( )

Destructor.

Definition at line 85 of file gcdomtreewidget.cpp.

### 5.7.3 Member Function Documentation

#### 5.7.3.1 QString GCDomTreeWidget::activeCommentValue ( ) const

Returns the comment associated with the current element (if any).

If no comment precedes the current element, an empty string is returned.

**See also**

setActiveCommentValue

Definition at line 120 of file gcdomtreewidget.cpp.

#### 5.7.3.2 void GCDomTreeWidget::addItem ( const QString & *element*, bool *toParent* = `false` )

Adds a new item and corresponding DOM element node named "element".

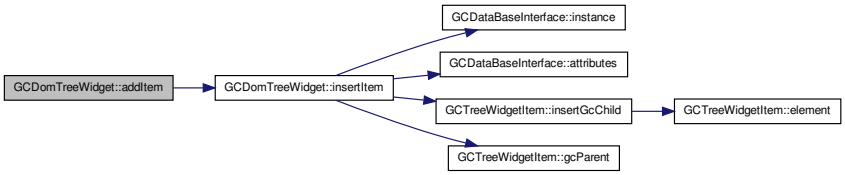
If the tree is empty, the new item will be added to the invisible root (i.e. as header item), otherwise it will be added as a child of the current item. The new item is also set as the current item. If "toParent" is true, the new item will be added as a child to the current item's parent (i.e. as a sibling to the current item).

**See also**

insertItem

Definition at line 495 of file gcdomtreewidget.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:


#### 5.7.3.3 const QList< GCTreeWidgetItem * > & GCDomTreeWidget::allTreeWidgetItems ( ) const

Returns a list of ALL the GCTreeWidgetItems currently in the tree.

**See also**

getIncludedTreeWidgetItems

Definition at line 186 of file gcdomtreewidget.cpp.

#### 5.7.3.4 void GCDomTreeWidget::appendSnippet ( GCTreeWidgetItem * *parentItem*, QDomElement *childElement* )

Creates and adds tree widget items for each element in the parameter element hierarchy.

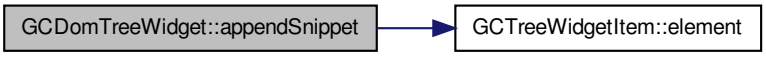
The process starts by appending "childElement" to "parentItem's" corresponding element and then recursively creates and adds items with associated elements corresponding to "childElement's" element hierarchy.

**See also**

processNextElement

Definition at line 317 of file gcdomtreewidget.cpp.

Here is the call graph for this function:

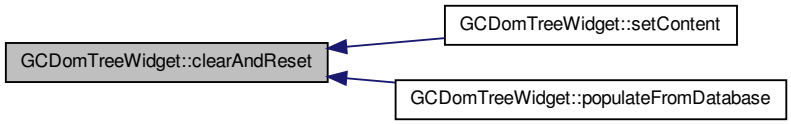


#### 5.7.3.5 void GCDomTreeWidget::clearAndReset ( )

Clears and resets the tree as well as the underlying DOM document.

Definition at line 987 of file gcdomtreewidget.cpp.

Here is the caller graph for this function:




#### 5.7.3.6 QDomNode GCDomTreeWidget::cloneDocument ( ) const

Returns a deep copy of the underlying DOM document.

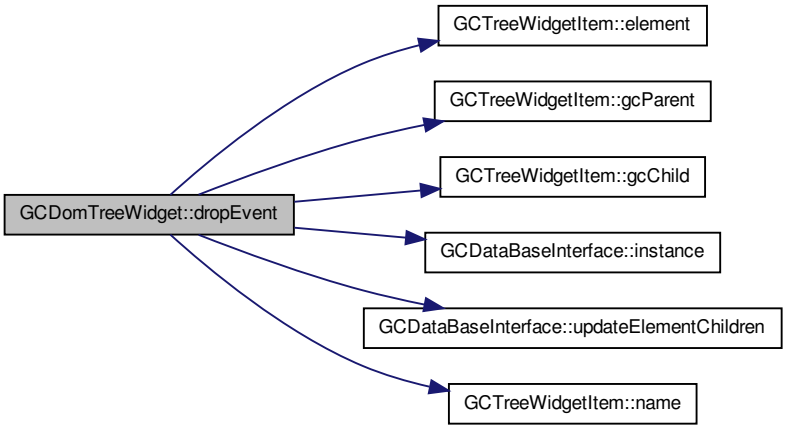
Definition at line 99 of file gcdomtreewidget.cpp.

#### 5.7.3.7 void GCDomTreeWidget::dropEvent ( QDropEvent * *event* ) [protected]

Re-implemented from QTreeWidget.

Definition at line 683 of file gcdomtreewidget.cpp.

Here is the call graph for this function:




#### 5.7.3.8 int GCDomTreeWidget::findItemPositionAmongDuplicates ( const QString & *nodeText*, int *itemIndex* ) const

Returns the position of "itemIndex" relative to that of ALL items matching "nodeText" (this is is not as odd as it sounds, it is possible that a DOM document may have multiple elements of the same name with matching attributes and attribute values).

Definition at line 193 of file gcdomtreewidget.cpp.

Here is the call graph for this function:


#### 5.7.3.9 GCTreeWidgetItem * GCDomTreeWidget::gcCurrentItem ( ) const

Returns the current item as a GCTreeWidgetItem.

Definition at line 92 of file gcdomtreewidget.cpp.

Here is the caller graph for this function:


#### 5.7.3.10 void GCDomTreeWidget::gcCurrentItemChanged ( GCTreeWidgetItem *, int ) [signal]

Emitted when the current item's content changes.

**See also**

emitGcCurrentItemChanged
gcCurrentItemSelected

#### 5.7.3.11 void GCDomTreeWidget::gcCurrentItemSelected ( GCTreeWidgetItem *, int , bool ) [signal]

Emitted when the current active item changes.

**See also**

emitGcCurrentItemSelected
gcCurrentItemChanged

#### 5.7.3.12 void GCDomTreeWidget::getIncludedTreeWidgetItems ( QList< GCTreeWidgetItem * > & *includedItems* ) const

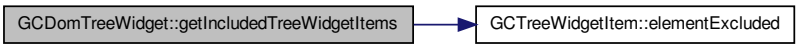
Populates "includedItems" with all the GCTreeWidgetItems in the tree that have their "include" flags set.

**See also**

allTreeWidgetItems

Definition at line 171 of file gcdomtreewidget.cpp.

Here is the call graph for this function:




#### 5.7.3.13 void GCDomTreeWidget::insertItem ( const QString & *elementName,* int *index,* bool *toParent* = `false` )

Adds a new item and corresponding DOM element node named "elementName" and inserts the new tree widget item into position "index" of the current item.

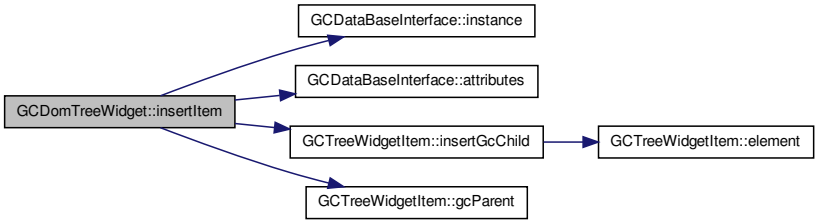
If the tree is empty, the new item will be added to the invisible root. The new item is also set as the current item. If "toParent" is true, the new item will be added as a child to the current item's parent (i.e. as a sibling to the current item).

**See also**

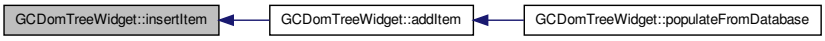
addItem

Definition at line 516 of file gcdomtreewidget.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

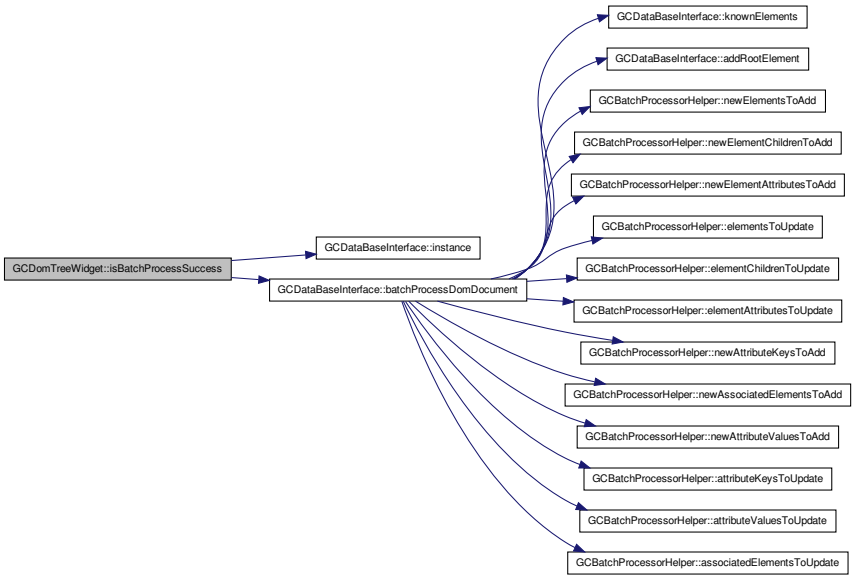



#### 5.7.3.14 bool GCDomTreeWidget::isBatchProcessSuccess ( ) const

Returns true if batch processing of DOM content to the active DB was successful.

Definition at line 275 of file gcdomtreewidget.cpp.

Here is the call graph for this function:




#### 5.7.3.15 bool GCDomTreeWidget::isCurrentItemRoot ( ) const

Returns true if the current item is the one corresponding to the DOM document's root element.

**See also**

matchesRootName
rootName

Definition at line 249 of file gcdomtreewidget.cpp.

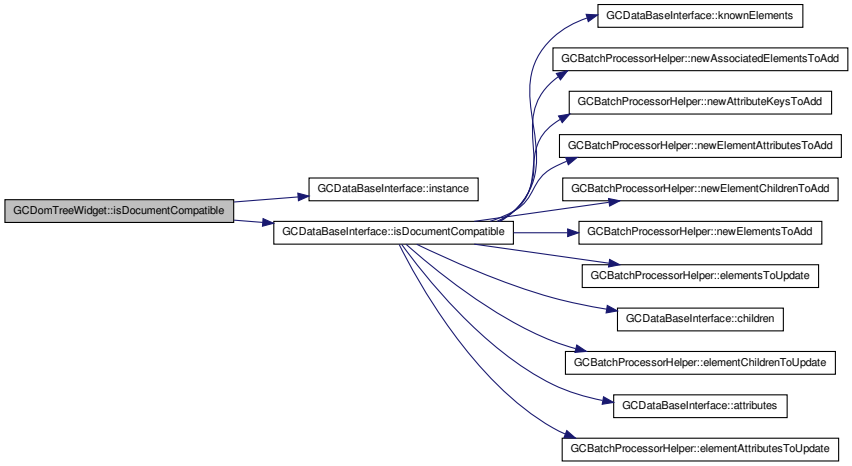
Here is the call graph for this function:


#### 5.7.3.16 bool GCDomTreeWidget::isDocumentCompatible ( ) const

Returns true if the underlying DOM document is compatible with the active DB session.

Definition at line 268 of file gcdomtreewidget.cpp.

Here is the call graph for this function:




#### 5.7.3.17 bool GCDomTreeWidget::isEmpty ( ) const

Returns true if the widget and DOM is currently empty.

Definition at line 242 of file gcdomtreewidget.cpp.

#### 5.7.3.18 void GCDomTreeWidget::keyPressEvent ( QKeyEvent * *event* ) [protected]

Re-implemented from QTreeWidget.

Definition at line 762 of file gcdomtreewidget.cpp.

#### 5.7.3.19 bool GCDomTreeWidget::matchesRootName ( const QString & *elementName* ) const

Returns true if "elementName" matches that of the DOM document's root.

**See also**

isCurrentItemRoot
rootName

Definition at line 261 of file gcdomtreewidget.cpp.

#### 5.7.3.20 void GCDomTreeWidget::populateFromDatabase ( const QString & *baseElementName* = QString() )

This function starts the recursive process of populating the tree widget with items consisting of the element hierarchy starting at "baseElementName".

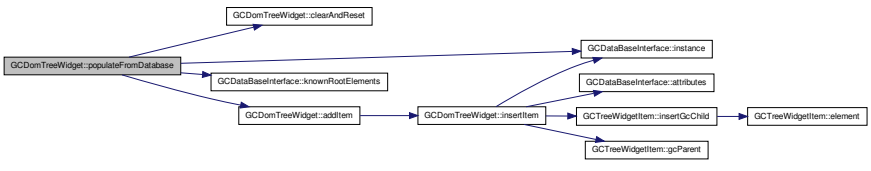
If "baseElementName" is empty, a complete hierarchy of the current active profile will be constructed. This method also automatically clears and resets GCDomTreeWidget's state, expands the entire tree, sets the first top level item as current and emits the "gcCurrentItemSelected" signal.

**See also**

processNextElement

Definition at line 422 of file gcdomtreewidget.cpp.

Here is the call graph for this function:



#### 5.7.3.21 void GCDomTreeWidget::rebuildTreeWidget ( )

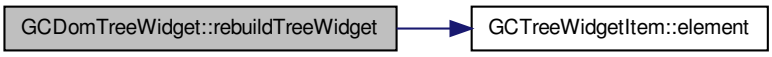
Rebuild the tree to conform to updated DOM content.

**See also**

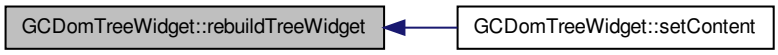
processNextElement

Definition at line 296 of file gcdomtreewidget.cpp.

Here is the call graph for this function:




Here is the caller graph for this function:

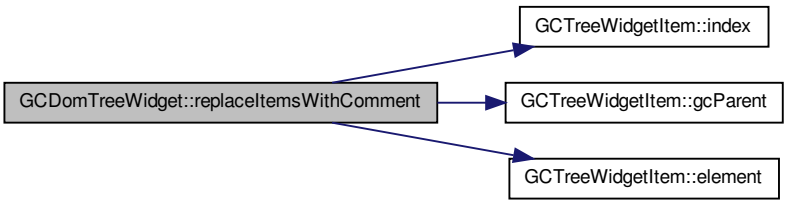



#### 5.7.3.22 void GCDomTreeWidget::replaceItemsWithComment ( const QList< int > & *indices*, const QString & *comment* )

Removes the items with indices matching those in the parameter list from the tree as well as from the DOM document.

Definition at line 328 of file gcdomtreewidget.cpp.

Here is the call graph for this function:




#### 5.7.3.23 QString GCDomTreeWidget::rootName ( ) const

Returns the name of the DOM document's root.

**See also**

currentItemIsRoot
matchesRootName

Definition at line 113 of file gcdomtreewidget.cpp.

#### 5.7.3.24 void GCDomTreeWidget::setActiveCommentValue ( const QString & *value* )

Sets the value of the active comment node to "value".

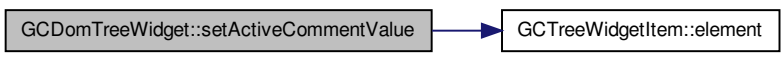
If the active element doesn't have an associated comment, a comment node is created.

**See also**

activeCommentValue

Definition at line 139 of file gcdomtreewidget.cpp.

Here is the call graph for this function:



#### 5.7.3.25 void GCDomTreeWidget::setAllCheckStates ( Qt::CheckState *state* )

Iterates through the tree and sets all items' check states to "state".

Definition at line 576 of file gcdomtreewidget.cpp.

#### 5.7.3.26 bool GCDomTreeWidget::setContent ( const QString & *text*, QString * *errorMsg* = 0, int * *errorLine* = 0, int * *errorColumn* = 0 )

Sets the underlying DOM document's content.

If successful, a recursive DOM tree traversal is kicked off in order to populate the tree widget with the information contained in the active DOM document.

**See also**

processNextElement

Definition at line 223 of file gcdomtreewidget.cpp.

Here is the call graph for this function:


#### 5.7.3.27 void GCDomTreeWidget::setCurrentItemWithIndexMatching ( int *index* ) [slot]

Finds the item with index matching "index" and sets it as the current tree item.

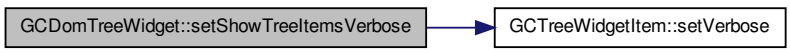
Definition at line 560 of file gcdomtreewidget.cpp.

#### 5.7.3.28 void GCDomTreeWidget::setShowTreeItemsVerbose ( bool *verbose* )

Iterates through the tree and set all items' "verbose" flags to "show".

Definition at line 593 of file gcdomtreewidget.cpp.

Here is the call graph for this function:




#### 5.7.3.29 QString GCDomTreeWidget::toString ( ) const

Returns the DOM content as string.

Definition at line 106 of file gcdomtreewidget.cpp.

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

- gcdomtreewidget.h
- gcdomtreewidget.cpp

## 5.8 GCHelpDialog Class Reference

Displays "Help" information.

`#include <gchelpdialog.h>`

### Public Member Functions

- GCHelpDialog (const QString &text, QWidget ∗parent=0)

  *Constructor.*
- ∼GCHelpDialog ()

  *Destructor.*

### 5.8.1 Detailed Description

Displays "Help" information.

The Qt::WA_DeleteOnClose flag is set for all instances of this form. If you're unfamiliar with Qt, this means that Qt will delete this widget as soon as the widget accepts the close event (i.e. you don't need to worry about clean-up of dynamically created instances of this object).

Definition at line 47 of file gchelpdialog.h.

### 5.8.2 Constructor & Destructor Documentation

#### 5.8.2.1 GCHelpDialog::GCHelpDialog ( const QString & *text*, QWidget * *parent* = 0 ) [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *text* | - the "Help" text that should be displayed. |

Definition at line 34 of file gchelpdialog.cpp.

#### 5.8.2.2 GCHelpDialog::~GCHelpDialog ( )

Destructor.

Definition at line 46 of file gchelpdialog.cpp.

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

- gchelpdialog.h
- gchelpdialog.cpp

## 5.9 GCMainWindow Class Reference

The main application window class.

`#include <gcmainwindow.h>`

### Public Member Functions

- GCMainWindow (QWidget *parent=0)

  *Constructor.*
- ~GCMainWindow ()

  *Destructor.*

### Protected Member Functions

- void closeEvent (QCloseEvent *event)

  *Re-implemented from QMainWindow.*

### 5.9.1 Detailed Description

The main application window class.

All the code refers to "databases" whereas all the user prompts reference "profiles". - This is deliberate. In reality, everything is persisted to SQLite database files, but a friend suggested that end users may be intimidated by the use of the word "database" (especially if they aren't necessarily technically inclined) and that "profile" may be less scary and I agreed :)

Definition at line 82 of file gcmainwindow.h.

### 5.9.2 Constructor & Destructor Documentation

#### 5.9.2.1 GCMainWindow::GCMainWindow ( QWidget * *parent* = 0 ) [explicit]

Constructor.

Definition at line 75 of file gcmainwindow.cpp.

#### 5.9.2.2 GCMainWindow::~GCMainWindow ( )

Destructor.

Definition at line 158 of file gcmainwindow.cpp.

### 5.9.3 Member Function Documentation

#### 5.9.3.1 void GCMainWindow::closeEvent ( QCloseEvent * *event* ) [protected]

Re-implemented from QMainWindow.

Queries user to save before closing and saves the user's "Options" preferences to settings.

Definition at line 165 of file gcmainwindow.cpp.

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

- gcmainwindow.h
- gcmainwindow.cpp

## 5.10 GCMessageDialog Class Reference

Provides a user dialog prompt with the option to save the user's preference.

### Public Member Functions

- GCMessageDialog (bool *remember, const QString &heading, const QString &text, GCMessageSpace::ButtonCombo buttons, GCMessageSpace::Buttons defaultButton, GCMessageSpace::Icon icon=GCMessageSpace::NoIcon)

*Constructor.*

- ~GCMessageDialog ()

  *Destructor.*

### 5.10.1 Detailed Description

Provides a user dialog prompt with the option to save the user's preference.

Definition at line 38 of file gcmessagespace.cpp.

### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 GCMessageDialog::GCMessageDialog ( bool * *remember*, const QString & *heading*, const QString & *text*, GCMessageSpace::ButtonCombo *buttons*, GCMessageSpace::Buttons *defaultButton*, GCMessageSpace::Icon *icon* = GCMessageSpace::NoIcon ) [inline, explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *remember* | - this flag should be passed in from the calling object and will be set when the user checks the relevant box |
| *heading* | - the message box header |
| *text* | - the actual message text |
| *buttons* | - the buttons that should be displayed for this particular message |
| *default-Button* | - the button that should be highlighted as the default |
| *icon* | - the icon associated with this particular message. |

Definition at line 51 of file gcmessagespace.cpp.

#### 5.10.2.2 GCMessageDialog::~GCMessageDialog ( ) [inline]

Destructor.

Definition at line 119 of file gcmessagespace.cpp.

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

- gcmessagespace.cpp

## 5.11 GCPlainTextEdit Class Reference

Specialist text edit class for displaying XML content in the XML Mill context.

`#include <gcplaintextedit.h>`

### Public Slots

- void wrapText (bool wrap)

  *Sets the necessary flags on the text edit to wrap or unwrap text as per user preference.*

### Signals

- void selectedIndex (int)

  *Emitted when the user clicks or otherwise moves within the text edit.*
- void commentOut (const QList< int > &, const QString &)

  *Emitted whenever a selection has been commented out.*
- void manualEditAccepted ()

  *Emitted whenever a selection must be "uncommented" or deleted.*

### Public Member Functions

- **GCPlainTextEdit** (QWidget *parent=0)
- void setContent (const QString &text)

  *Use instead of "setPlainText" as it improves performance significantly (especially for larger documents).*
- void findTextRelativeToDuplicates (const QString &text, int relativePos)

  *Finds the "relativePos"'s occurrence of "text" within the active document.*
- void clearAndReset ()

  *Resets the internal state of GCPlainTextEdit.*

### Protected Member Functions

- void keyPressEvent (QKeyEvent *e)

  *Re-implemented from QPlainTextEdit.*

#### 5.11.1 Detailed Description

Specialist text edit class for displaying XML content in the XML Mill context.

Provides functionality with which to comment out or uncomment XML selections and keeps track of which XML nodes are currently under investigation (based on cursor positions).

Definition at line 43 of file gcplaintextedit.h.

### 5.11.2 Member Function Documentation

#### 5.11.2.1 void GCPlainTextEdit::clearAndReset ( )

Resets the internal state of GCPlainTextEdit.

Definition at line 136 of file gcplaintextedit.cpp.

#### 5.11.2.2 void GCPlainTextEdit::commentOut ( const QList< int > & , const QString & ) [signal]

Emitted whenever a selection has been commented out.

The parameter list contains the indices corresponding to the items that should be removed from the tree widget.

#### 5.11.2.3 void GCPlainTextEdit::findTextRelativeToDuplicates ( const QString & *text*, int *relativePos* )

Finds the "relativePos"'s occurrence of "text" within the active document.

Definition at line 120 of file gcplaintextedit.cpp.

#### 5.11.2.4 void GCPlainTextEdit::keyPressEvent ( QKeyEvent * *e* ) [protected]

Re-implemented from QPlainTextEdit.

Definition at line 435 of file gcplaintextedit.cpp.

#### 5.11.2.5 void GCPlainTextEdit::manualEditAccepted ( ) [signal]

Emitted whenever a selection must be "uncommented" or deleted.

#### 5.11.2.6 void GCPlainTextEdit::selectedIndex ( int ) [signal]

Emitted when the user clicks or otherwise moves within the text edit.

**See also**

emitSelectedIndex

#### 5.11.2.7 void GCPlainTextEdit::setContent ( const QString & *text* )

Use instead of "setPlainText" as it improves performance significantly (especially for larger documents).

Definition at line 105 of file gcplaintextedit.cpp.

#### 5.11.2.8 void GCPlainTextEdit::wrapText ( bool *wrap* ) [slot]

Sets the necessary flags on the text edit to wrap or unwrap text as per user preference.

Definition at line 421 of file gcplaintextedit.cpp.

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

- gcplaintextedit.h
- gcplaintextedit.cpp

## 5.12 GCRemoveItemsForm Class Reference

Allows the user to remove items from the active database.

`#include <gcremoveitemsform.h>`

### Public Member Functions

- GCRemoveItemsForm (QWidget *parent=0)

  *Constructor.*
- ~GCRemoveItemsForm ()

  *Destructor.*

### 5.12.1 Detailed Description

Allows the user to remove items from the active database.

All changes made via this form are irreversible and will be executed immediately against the active database.

The Qt::WA_DeleteOnClose flag is set for all instances of this form. If you're unfamiliar with Qt, this means that Qt will delete this widget as soon as the widget accepts the close event (i.e. you don't need to worry about clean-up of dynamically created instances of this object).

Definition at line 52 of file gcremoveitemsform.h.

### 5.12.2 Constructor & Destructor Documentation

#### 5.12.2.1 GCRemoveItemsForm::GCRemoveItemsForm ( QWidget * *parent* = 0 ) [explicit]

Constructor.

Definition at line 40 of file gcremoveitemsform.cpp.

Here is the call graph for this function:


#### 5.12.2.2 GCRemoveItemsForm::~GCRemoveItemsForm ( )

Destructor.

Definition at line 69 of file gcremoveitemsform.cpp.

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

- gcremoveitemsform.h
- gcremoveitemsform.cpp

## 5.13 GCRestoreFilesForm Class Reference

Displays recovered files so that the user may decide whether or not he/she wants to save them.

`#include <gcrestorefilesform.h>`

### Public Member Functions

- GCRestoreFilesForm (const QStringList &tempFiles, QWidget *parent=0)

  *Constructor.*
- ~GCRestoreFilesForm ()

  *Destructor.*

### 5.13.1 Detailed Description

Displays recovered files so that the user may decide whether or not he/she wants to save them.

Definition at line 40 of file gcrestorefilesform.h.

### 5.13.2 Constructor & Destructor Documentation

#### 5.13.2.1 GCRestoreFilesForm::GCRestoreFilesForm ( const QStringList & *tempFiles*, QWidget * *parent* = 0 ) [explicit]

Constructor.

Definition at line 43 of file gcrestorefilesform.cpp.

#### 5.13.2.2 GCRestoreFilesForm::~GCRestoreFilesForm ( )

Destructor.

Definition at line 67 of file gcrestorefilesform.cpp.

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

- gcrestorefilesform.h
- gcrestorefilesform.cpp

## 5.14 GCSearchForm Class Reference

Search through the current document for specific text.

`#include <gcsearchform.h>`

### Signals

- void foundItem (GCTreeWidgetItem *)

  *Emitted when the search string is found in the document.*

### Public Member Functions

- GCSearchForm (const QList< GCTreeWidgetItem * > &items, const QString &docContents, QWidget *parent=0)

  *Constructor.*
- ~GCSearchForm ()

  *Destructor.*

### 5.14.1 Detailed Description

Search through the current document for specific text.

The Qt::WA_DeleteOnClose flag is set for all instances of this form. If you're unfamiliar with Qt, this means that Qt will delete this widget as soon as the widget accepts the close event (i.e. you don't need to worry about clean-up of dynamically created instances of this object).

Definition at line 50 of file gcsearchform.h.

### 5.14.2 Constructor & Destructor Documentation

#### 5.14.2.1 GCSearchForm::GCSearchForm ( const QList< GCTreeWidgetItem * > & *items*, const QString & *docContents*, QWidget * *parent* = 0 ) [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *elements* | - a list of all the elements in the active document. |
| *docContents* | - the string representation of the active document's DOM content. |

Definition at line 51 of file gcsearchform.cpp.

#### 5.14.2.2 GCSearchForm::~GCSearchForm ( )

Destructor.

Definition at line 78 of file gcsearchform.cpp.

### 5.14.3 Member Function Documentation

#### 5.14.3.1 void GCSearchForm::foundItem ( GCTreeWidgetItem * ) [signal]

Emitted when the search string is found in the document.

The item emitted in this signal will contain the matched string in either its corresponding element's name, or the name of an associated attribute or attribute value.

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

- gcsearchform.h
- gcsearchform.cpp

## 5.15 GCTreeWidgetItem Class Reference

Used in GCDomTreeWidget, each GCTreeWidgetItem can be associated with a QDomElement.

`#include <gctreewidgetitem.h>`

### Public Member Functions

- GCTreeWidgetItem (QDomElement element)

  *Constructor.*
- GCTreeWidgetItem (QDomElement element, int index)

  *Constructor.*

- GCTreeWidgetItem * gcParent () const

  *Returns the parent item as a GCTreeWidgetItem.*
- GCTreeWidgetItem * gcChild (int index) const

  *Returns the child item at "index" as a GCTreeWidgetItem.*
- QDomElement element () const

  *Returns the associated element via QDomElement's default shallow copy constructor.*
- void setExcludeElement (bool exclude)

  *Sets the "exclude" flag (used to determine if the element must be included in GCDomTreeWidget's DOM document).*
- bool elementExcluded () const

  *Returns "true" if the element should be excluded from the active document.*
- void excludeAttribute (const QString &attribute)

  *Removes "attribute" from the underlying DOM element.*
- void includeAttribute (const QString &attribute, const QString &value)

  *Includes "attribute" with "value" in the underlying DOM element.*
- bool attributeIncluded (const QString &attribute) const

  *Returns true if the underlying element contains "attribute".*
- void setIncrementAttribute (const QString &attribute, bool increment)

  *This function is only used in GCAddSnippetsForm.*
- bool incrementAttribute (const QString &attribute) const

  *This function is only used in GCAddSnippetsForm.*
- void fixAttributeValues ()

  *This function is only used in GCAddSnippetsForm.*
- QString fixedValue (const QString &attribute) const

  *This function is only used in GCAddSnippetsForm.*
- void revertToFixedValues ()

  *This function is only used in GCAddSnippetsForm.*
- QString toString () const

  *Provides a string representation of the element, its attributes and attribute values (including brackets and other XML characters).*
- void setIndex (int index)

  *Sets the item's index to "index".*
- int index () const

  *Returns the index associated with this element.*
- void rename (const QString &newName)

  *Renames the element to "newName".*
- QString name () const

  *Returns the element name.*
- void setVerbose (bool verbose)

  *Sets the item's element display as "verbose".*
- void insertGcChild (int index, GCTreeWidgetItem *item)

  *Inserts "item" at "index" and ensures that the corresponding DOM element is also inserted in the correct position (relative to the item's siblings).*

### 5.15.1 Detailed Description

Used in GCDomTreeWidget, each GCTreeWidgetItem can be associated with a QDomElement.

Can be associated with a QDomElement in order to provide additional information in the XML Mill context regarding whether or not the element or any of its attributes should be excluded from the DOM document being built. This item DOES NOT OWN the QDomElement (hence all the non const return values...QDomElement's default copy constructor is shallow).

Definition at line 46 of file gctreewidgetitem.h.

### 5.15.2 Constructor & Destructor Documentation

#### 5.15.2.1 GCTreeWidgetItem::GCTreeWidgetItem ( QDomElement *element* ) [explicit]

Constructor.

Associates "element" with this item.

Definition at line 33 of file gctreewidgetitem.cpp.

#### 5.15.2.2 GCTreeWidgetItem::GCTreeWidgetItem ( QDomElement *element,* int *index* ) [explicit]

Constructor.

Associates "element" with this item and assigns it an "index" which is used to determine the underlying DOM element's relative position within the DOM (roughly corresponding to "line numbers").

Definition at line 40 of file gctreewidgetitem.cpp.

### 5.15.3 Member Function Documentation

#### 5.15.3.1 bool GCTreeWidgetItem::attributeIncluded ( const QString & *attribute* ) const

Returns true if the underlying element contains "attribute".

**See also**

excludeAttribute
includeAttribute

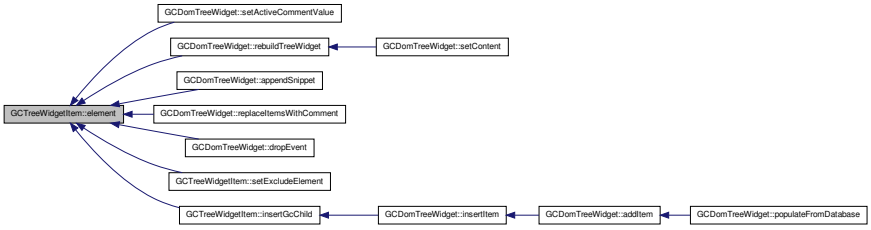
Definition at line 150 of file gctreewidgetitem.cpp.

#### 5.15.3.2 QDomElement GCTreeWidgetItem::element ( ) const

Returns the associated element via QDomElement's default shallow copy constructor.

Definition at line 96 of file gctreewidgetitem.cpp.

Here is the caller graph for this function:




#### 5.15.3.3 bool GCTreeWidgetItem::elementExcluded ( ) const

Returns "true" if the element should be excluded from the active document.

**See also**

setExcludeElement

Definition at line 122 of file gctreewidgetitem.cpp.

Here is the caller graph for this function:


#### 5.15.3.4 void GCTreeWidgetItem::excludeAttribute ( const QString & *attribute* )

Removes "attribute" from the underlying DOM element.

**See also**

includeAttribute
attributeIncluded

Definition at line 129 of file gctreewidgetitem.cpp.

#### 5.15.3.5 void GCTreeWidgetItem::fixAttributeValues ( )

This function is only used in GCAddSnippetsForm.

Takes a snapshot of the current attribute values so that element attributes may be updated on each snippet iteration without forgetting what the underlying value was.

**See also**

incrementAttribute
setIncrementAttribute
fixedValue
revertToFixedValues

Definition at line 180 of file gctreewidgetitem.cpp.

#### 5.15.3.6 QString GCTreeWidgetItem::fixedValue ( const QString & *attribute* ) const

This function is only used in GCAddSnippetsForm.

Returns the fixed value saved against "attribute".

**See also**

incrementAttribute
setIncrementAttribute
fixAttributeValues
revertToFixedValues

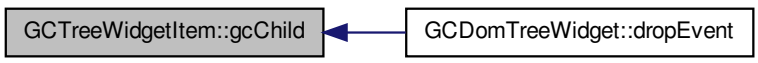
Definition at line 195 of file gctreewidgetitem.cpp.

#### 5.15.3.7 GCTreeWidgetItem * GCTreeWidgetItem::gcChild ( int *index* ) const

Returns the child item at "index" as a GCTreeWidgetItem.

Definition at line 89 of file gctreewidgetitem.cpp.

Here is the caller graph for this function:

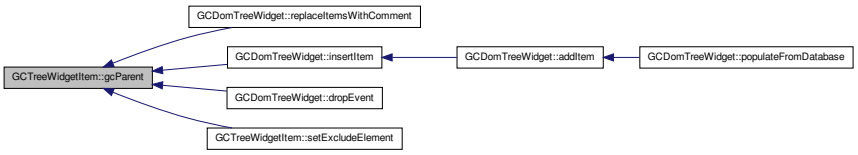


#### 5.15.3.8 GCTreeWidgetItem * GCTreeWidgetItem::gcParent ( ) const

Returns the parent item as a GCTreeWidgetItem.

Definition at line 82 of file gctreewidgetitem.cpp.

Here is the caller graph for this function:




#### 5.15.3.9 void GCTreeWidgetItem::includeAttribute ( const QString & *attribute,* const QString & *value* )

Includes "attribute" with "value" in the underlying DOM element.

**See also**

excludeAttribute
attributeIncluded

Definition at line 139 of file gctreewidgetitem.cpp.

#### 5.15.3.10 bool GCTreeWidgetItem::incrementAttribute ( const QString & *attribute* ) const

This function is only used in GCAddSnippetsForm.

Returns true if "attribute" must be incremented automatically.

**See also**

setIncrementAttribute
fixAttributeValues
fixedValue
revertToFixedValues

Definition at line 173 of file gctreewidgetitem.cpp.

#### 5.15.3.11 int GCTreeWidgetItem::index ( ) const

Returns the index associated with this element.

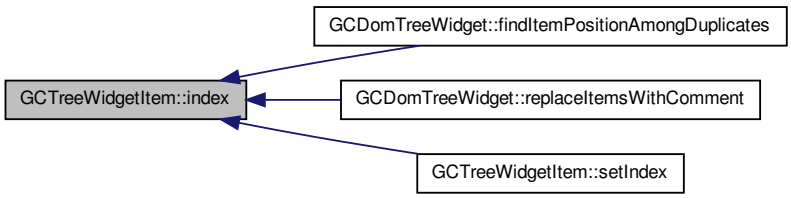
Indices in this context are rough indications of an element's relative position within the DOM document (approximating "line numbers").

**See also**

setIndex

Definition at line 270 of file gctreewidgetitem.cpp.

Here is the caller graph for this function:

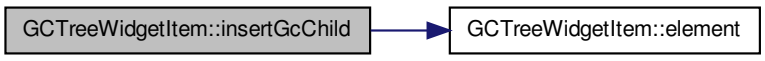



#### 5.15.3.12 void GCTreeWidgetItem::insertGcChild ( int *index*, GCTreeWidgetItem * *item* )

Inserts "item" at "index" and ensures that the corresponding DOM element is also inserted in the correct position (relative to the item's siblings).

Definition at line 305 of file gctreewidgetitem.cpp.

Here is the call graph for this function:



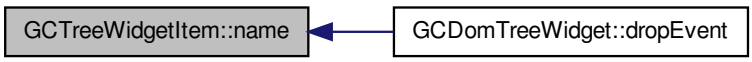

Here is the caller graph for this function:

#### 5.15.3.13 QString GCTreeWidgetItem::name ( ) const

Returns the element name.

Definition at line 285 of file gctreewidgetitem.cpp.

Here is the caller graph for this function:

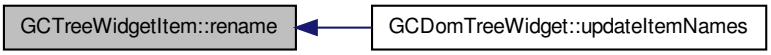



#### 5.15.3.14 void GCTreeWidgetItem::rename ( const QString & *newName* )

Renames the element to "newName".

Definition at line 277 of file gctreewidgetitem.cpp.

Here is the caller graph for this function:




#### 5.15.3.15 void GCTreeWidgetItem::revertToFixedValues ( )

This function is only used in GCAddSnippetsForm.

Reverts to the attribute values set with the "fixedAttributeValues" call.

**See also**

setIncrementAttribute
incrementAttribute
fixAttributeValues
fixedValue

Definition at line 202 of file gctreewidgetitem.cpp.

#### 5.15.3.16 void GCTreeWidgetItem::setExcludeElement ( bool *exclude* )

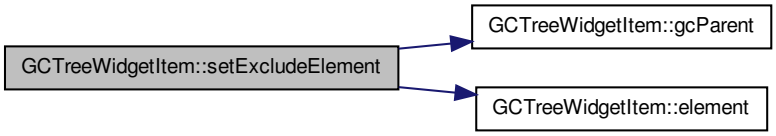
Sets the "exclude" flag (used to determine if the element must be included in GCDomTreeWidget's DOM document).

**See also**

elementExcluded

Definition at line 103 of file gctreewidgetitem.cpp.

Here is the call graph for this function:




#### 5.15.3.17 void GCTreeWidgetItem::setIncrementAttribute ( const QString & *attribute,* bool *increment* )

This function is only used in GCAddSnippetsForm.

Adds "attribute" to a list of attributes whose values must be incremented when multiple snippets are added to the active DOM. The reason this functionality was added is due to the complications inherent to the default shallow copy constructors of QDomAttr (I originally tried to use maps confined to GCSnippetForm objects, but to no avail).

**See also**

incrementAttribute
fixAttributeValues
fixedValue
revertToFixedValues

Definition at line 157 of file gctreewidgetitem.cpp.

#### 5.15.3.18 void GCTreeWidgetItem::setIndex ( int *index* )

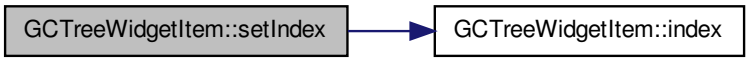
Sets the item's index to "index".

**See also**

index

Definition at line 263 of file gctreewidgetitem.cpp.

Here is the call graph for this function:




#### 5.15.3.19 void GCTreeWidgetItem::setVerbose ( bool *verbose* )

Sets the item's element display as "verbose".

When "verbose", the entire node is displayed (element attributes and values), otherwise only the element name is displayed.

**See also**

setDisplayText

Definition at line 297 of file gctreewidgetitem.cpp.

Here is the caller graph for this function:


#### 5.15.3.20 QString GCTreeWidgetItem::toString ( ) const

Provides a string representation of the element, its attributes and attribute values (including brackets and other XML characters).

Definition at line 212 of file gctreewidgetitem.cpp.

Here is the caller graph for this function:


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

- gctreewidgetitem.h
- gctreewidgetitem.cpp

## 5.16 XmlSyntaxHighlighter Class Reference

Original class was obtained here: http://qt.gitorious.org/qt/qt/blobs/-HEAD/examples/xmlpatterns/shared/xmlsyntaxhighlighter.h.

```
#include <xmlsyntaxhighlighter.h>
```

### Classes

- struct **HighlightingRule**

### Public Member Functions

- **XmlSyntaxHighlighter** (QTextDocument *parent=0)

### Protected Member Functions

- virtual void **highlightBlock** (const QString &text)

### 5.16.1 Detailed Description

Original class was obtained here: http://qt.gitorious.org/qt/qt/blobs/-HEAD/examples/xmlpatterns/shared/xmlsyntaxhighlighter.h.

This version has some alterations made to the regular expressions.

Definition at line 52 of file xmlsyntaxhighlighter.h.

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

- xmlsyntaxhighlighter.h
- xmlsyntaxhighlighter.cpp