

libglademm 2.4

2.6.7

Generated by Doxygen 1.12.0

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 Glib Namespace Reference	9
5.2 Gnome Namespace Reference	9
5.3 Gnome::Glade Namespace Reference	9
6 Class Documentation	11
6.1 Gnome::Glade::VariablesMap Class Reference	11
6.1.1 Detailed Description	12
6.1.2 Member Typedef Documentation	12
6.1.2.1 type_mapWidgetsToVariables	12
6.1.3 Constructor & Destructor Documentation	12
6.1.3.1 VariablesMap()	12
6.1.3.2 ~VariablesMap()	12
6.1.4 Member Function Documentation	12
6.1.4.1 connect_widget() [1/4]	12
6.1.4.2 connect_widget() [2/4]	13
6.1.4.3 connect_widget() [3/4]	13
6.1.4.4 connect_widget() [4/4]	13
6.1.4.5 transfer_one_widget()	13
6.1.4.6 transfer_variables_to_widgets()	13
6.1.4.7 transfer_widgets_to_variables()	13
6.1.4.8 validate_widgets()	13
6.1.5 Member Data Documentation	14
6.1.5.1 m_mapWidgetsToVariables	14
6.1.5.2 m_refGlade	14
6.2 Gnome::Glade::Xml Class Reference	14
6.2.1 Member Typedef Documentation	16
6.2.1.1 Error	16
6.2.2 Constructor & Destructor Documentation	16
6.2.2.1 ~Xml()	16
6.2.2.2 Xml() [1/2]	16
6.2.2.3 Xml() [2/2]	16

6.2.3 Member Function Documentation	17
6.2.3.1 connect_clicked()	17
6.2.3.2 create()	17
6.2.3.3 create_from_buffer()	17
6.2.3.4 get_cwidget()	18
6.2.3.5 get_filename()	18
6.2.3.6 get_widget() [1/2]	18
6.2.3.7 get_widget() [2/2]	18
6.2.3.8 get_widget_checked()	19
6.2.3.9 get_widget_derived()	19
6.2.3.10 get_widget_name()	20
6.2.3.11 get_widget_prefix()	20
6.2.3.12 get_widget_tree()	20
6.2.3.13 gobj() [1/2]	20
6.2.3.14 gobj() [2/2]	20
6.2.3.15 gobj_copy()	20
6.2.3.16 lookup_type_vfunc()	20
6.2.3.17 relative_file()	20
6.2.3.18 reparent_widget()	21
6.2.4 Friends And Related Symbol Documentation	21
6.2.4.1 wrap()	21
6.3 Gnome::Glade::XmlError Class Reference	21
6.3.1 Constructor & Destructor Documentation	22
6.3.1.1 XmlError() [1/2]	22
6.3.1.2 ~XmlError()	22
6.3.1.3 XmlError() [2/2]	22
6.3.2 Member Function Documentation	22
6.3.2.1 operator=()	22
6.3.2.2 what()	22
7 File Documentation	23
7.1 init.h File Reference	23
7.2 variablesmap.h File Reference	23
7.3 xml.h File Reference	23
Index	25

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Glib	9
Gnome	9
Gnome::Glade	9

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Glib::Exception	
Gnome::Glade::XmlError	21
Glib::Object	
Gnome::Glade::Xml	14
Gnome::Glade::VariablesMap	11

Chapter 3

Class Index

3.1 Class List

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

Gnome::Glade::VariablesMap	
Associates named Glade widgets with member variables	11
Gnome::Glade::Xml	14
Gnome::Glade::XmlError	21

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

init.h	23
variablesmap.h	23
xml.h	23

Chapter 5

Namespace Documentation

5.1 Glib Namespace Reference

5.2 Gnome Namespace Reference

Namespaces

- namespace [Glade](#)

5.3 Gnome::Glade Namespace Reference

Classes

- class [VariablesMap](#)
Associates named [Glade](#) widgets with member variables.
- class [Xml](#)
- class [XmlError](#)

Functions

- `Glib::RefPtr< Gnome::Glade::Xml > wrap` (`GladeXML *object`, `bool take_copy=false`)
A `Glib::wrap()` method for this object.

Chapter 6

Class Documentation

6.1 Gnome::Glade::VariablesMap Class Reference

Associates named [Glade](#) widgets with member variables.

```
#include <variablesmap.h>
```

Public Member Functions

- [VariablesMap](#) (const Glib::RefPtr< [Glade::Xml](#) > &glade)
- virtual [~VariablesMap](#) ()
- virtual void [connect_widget](#) (const Glib::ustring &widget_name, bool &variable)
For ToggleButton (CheckBox and RadioButton)
- virtual void [connect_widget](#) (const Glib::ustring &widget_name, Glib::ustring &variable)
For Entry, ComboBoxEntry and SpinBox.
- virtual void [connect_widget](#) (const Glib::ustring &widget_name, double &variable)
For Scale (HScale and VScale)
- virtual void [connect_widget](#) (const Glib::ustring &widget_name, Glib::Date &variable)
For Calendar.
- virtual void [transfer_widgets_to_variables](#) ()
Transfer data from the widget to the variable.
- virtual void [transfer_variables_to_widgets](#) ()
Transfer data from the variable to the widget.

Protected Types

- typedef std::map< Gtk::Widget *, void * > [type_mapWidgetsToVariables](#)

Protected Member Functions

- virtual bool [validate_widgets](#) ()
Override this to validate the data that the user enters into the widgets.
- virtual void [transfer_one_widget](#) (Gtk::Widget *pWidget, bool to_variable)

Protected Attributes

- [type_mapWidgetsToVariables](#) `m_mapWidgetsToVariables`
- `Glib::RefPtr< Glade::Xml > m_refGlade`

6.1.1 Detailed Description

Associates named [Glade](#) widgets with member variables.

Use [connect_widget\(\)](#) to link the widgets with variables that will contain their data. Then use [transfer_widgets_to_variables\(\)](#) and [transfer_variables_to_widgets\(\)](#) to get or set all of the variables at once.

This is meant to be a bit like MFC's "Dialog Data Exchange and Validation".

The association of widget and member variables follow this mapping:

```
Gtk::Entry --> Glib::ustring Gtk::SpinBox --> Glib::ustring Gtk::ComboBoxEntry --> Glib::ustring Gtk::Scale -->
double Gtk::Calendar --> Glib::Date Gtk::CheckBox --> bool Gtk::RadioButton --> bool
```

6.1.2 Member Typedef Documentation

6.1.2.1 type_mapWidgetsToVariables

```
std::map<Gtk::Widget*, void*> Gnome::Glade::VariablesMap::type\_mapWidgetsToVariables [protected]
```

6.1.3 Constructor & Destructor Documentation

6.1.3.1 VariablesMap()

```
Gnome::Glade::VariablesMap::VariablesMap (
    const Glib::RefPtr< Glade::Xml > & glade) [explicit]
```

6.1.3.2 ~VariablesMap()

```
virtual Gnome::Glade::VariablesMap::~~VariablesMap () [virtual]
```

6.1.4 Member Function Documentation

6.1.4.1 connect_widget() [1/4]

```
virtual void Gnome::Glade::VariablesMap::connect_widget (
    const Glib::ustring & widget_name,
    bool & variable) [virtual]
```

For ToggleButton (CheckBox and RadioButton)

6.1.4.2 connect_widget() [2/4]

```
virtual void Gnome::Glade::VariablesMap::connect_widget (
    const Glib::ustring & widget_name,
    double & variable) [virtual]
```

For Scale (HScale and VScale)

6.1.4.3 connect_widget() [3/4]

```
virtual void Gnome::Glade::VariablesMap::connect_widget (
    const Glib::ustring & widget_name,
    Glib::Date & variable) [virtual]
```

For Calendar.

6.1.4.4 connect_widget() [4/4]

```
virtual void Gnome::Glade::VariablesMap::connect_widget (
    const Glib::ustring & widget_name,
    Glib::ustring & variable) [virtual]
```

For Entry, ComboBoxEntry and SpinBox.

6.1.4.5 transfer_one_widget()

```
virtual void Gnome::Glade::VariablesMap::transfer_one_widget (
    Gtk::Widget * pWidget,
    bool to_variable) [protected], [virtual]
```

6.1.4.6 transfer_variables_to_widgets()

```
virtual void Gnome::Glade::VariablesMap::transfer_variables_to_widgets () [virtual]
```

Transfer data from the variable to the widget.

6.1.4.7 transfer_widgets_to_variables()

```
virtual void Gnome::Glade::VariablesMap::transfer_widgets_to_variables () [virtual]
```

Transfer data from the widget to the variable.

6.1.4.8 validate_widgets()

```
virtual bool Gnome::Glade::VariablesMap::validate_widgets () [protected], [virtual]
```

Override this to validate the data that the user enters into the widgets.

The return value indicates whether the widgets' data is valid.

6.1.5 Member Data Documentation

6.1.5.1 m_mapWidgetsToVariables

`type_mapWidgetsToVariables` `Gnome::Glade::VariablesMap::m_mapWidgetsToVariables` [protected]

6.1.5.2 m_refGlade

`Glib::RefPtr<Glade::Xml>` `Gnome::Glade::VariablesMap::m_refGlade` [protected]

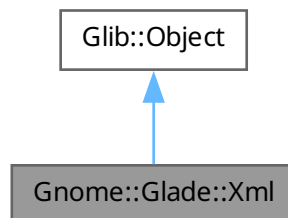
The documentation for this class was generated from the following file:

- [variablesmap.h](#)

6.2 Gnome::Glade::Xml Class Reference

```
#include <xml.h>
```

Inheritance diagram for Gnome::Glade::Xml:



Public Types

- typedef `Gnome::Glade::XmlError` `Error`

Public Member Functions

- virtual `~Xml ()`
- GladeXML * `gobj ()`
Provides access to the underlying C GObject.
- const GladeXML * `gobj () const`
Provides access to the underlying C GObject.
- GladeXML * `gobj_copy ()`
Provides access to the underlying C instance. The caller is responsible for unrefing it. Use when directly setting fields in structs.
- std::string `get_filename () const`
- Gtk::Widget * `get_widget (const Glib::ustring &name)`
Get a widget from the Glade file.
- template<class T_Widget >
T_Widget * `get_widget (const Glib::ustring &name, T_Widget *&widget)`
More convenient way of getting a widget from the Glade file.
- template<class T_Widget >
T_Widget * `get_widget_derived (const Glib::ustring &name, T_Widget *&widget)`
This provides a pointer to a widget whose details are specified in the Glade file, but which is implemented by your own derived class.
- Glib::ListHandle< Gtk::Widget * > `get_widget_prefix (const Glib::ustring &name)`
- void `reparent_widget (const Glib::ustring &name, Gtk::Container &container)`
Take the widget from the glade-generated container and put it in another container.
- std::string `relative_file (const std::string &filename) const`
- void `connect_clicked (const Glib::ustring &name, const sigc::slot< void > &slot_)`
Connect a Gtk::Button's clicked signal or a Gtk::MenuItem's activated signal to a slot.

Static Public Member Functions

- static Glib::RefPtr< Xml > `create (const std::string &filename, const Glib::ustring &root=Glib::ustring(), const Glib::ustring &domain=Glib::ustring())`
Loads a Glade XML file.
- static Glib::RefPtr< Xml > `create_from_buffer (const char *buffer, int size, const Glib::ustring &root=Glib::ustring(), const Glib::ustring &domain=Glib::ustring())`
Reads glade XML data from memory.
- static Glib::ustring `get_widget_name (Gtk::Widget &widget)`
- static Glib::RefPtr< Xml > `get_widget_tree (Gtk::Widget &widget)`

Protected Member Functions

- Xml (const std::string &filename, const Glib::ustring &root, const Glib::ustring &domain)
Loads a glade XML file.
- Xml (const char *buffer, int size, const Glib::ustring &root, const Glib::ustring &domain)
Reads glade XML data from memory.
- Gtk::Widget * `get_widget_checked (const Glib::ustring &name, GType type)`
- GtkWidget * `get_cwidget (const Glib::ustring &name)`
- virtual GType `lookup_type_vfunc (const Glib::ustring &classname)`

Related Symbols

(Note that these are not member symbols.)

- `Glib::RefPtr< Gnome::Glade::Xml > wrap` (`GladeXML *object`, `bool take_copy=false`)
A `Glib::wrap()` method for this object.

6.2.1 Member Typedef Documentation

6.2.1.1 Error

`Gnome::Glade::XmlError` `Gnome::Glade::Xml::Error`

6.2.2 Constructor & Destructor Documentation

6.2.2.1 ~Xml()

```
virtual Gnome::Glade::Xml::~~Xml () [virtual]
```

6.2.2.2 Xml() [1/2]

```
Gnome::Glade::Xml::Xml (
    const std::string & filename,
    const Glib::ustring & root,
    const Glib::ustring & domain) [protected]
```

Loads a glade XML file.

Exceptions

XmlError	
--------------------------	--

6.2.2.3 Xml() [2/2]

```
Gnome::Glade::Xml::Xml (
    const char * buffer,
    int size,
    const Glib::ustring & root,
    const Glib::ustring & domain) [protected]
```

Reads glade XML data from memory.

Exceptions

XmlError	
--------------------------	--

6.2.3 Member Function Documentation

6.2.3.1 connect_clicked()

```
void Gnome::Glade::Xml::connect_clicked (
    const Glib::ustring & name,
    const sigc::slot< void > & slot_)
```

Connect a Gtk::Button's clicked signal or a Gtk::MenuItem's activated signal to a slot.

For instance:

```
refXml->connect_button("button", sigc::mem_fun(*this, &ExampleWindow::on_button_clicked) );
```

Parameters

<i>name</i>	The name of the widget.
<i>pslot</i>	The slot to connect to.

6.2.3.2 create()

```
static Glib::RefPtr< Xml > Gnome::Glade::Xml::create (
    const std::string & filename,
    const Glib::ustring & root = Glib::ustring(),
    const Glib::ustring & domain = Glib::ustring()) [static]
```

Loads a [Glade](#) XML file.

This will instantiate the widgets in the XML file. You can use the root parameter to only instantiate a certain widget and its children. The returned [Xml](#) object keeps pointers to the instantiated widgets which you can retrieve with [get_widget\(\)](#).

Note that the [Xml](#) object does not delete the widgets it instantiates, but instead leaves the responsibility to you. See [get_widget\(\)](#). This means that you can safely let the [Xml](#) object go out of scope after you have retrieved the pointers you need from it.

Exceptions

XmlError	
--------------------------	--

6.2.3.3 create_from_buffer()

```
static Glib::RefPtr< Xml > Gnome::Glade::Xml::create_from_buffer (
    const char * buffer,
    int size,
    const Glib::ustring & root = Glib::ustring(),
    const Glib::ustring & domain = Glib::ustring()) [static]
```

Reads glade XML data from memory.

Exceptions

XmlError	
--------------------------	--

See also

[create\(\)](#)**6.2.3.4 get_cwidget()**

```
GtkWidget * Gnome::Glade::Xml::get_cwidget (
    const Glib::ustring & name) [protected]
```

6.2.3.5 get_filename()

```
std::string Gnome::Glade::Xml::get_filename () const
```

6.2.3.6 get_widget() [1/2]

```
Gtk::Widget * Gnome::Glade::Xml::get_widget (
    const Glib::ustring & name)
```

Get a widget from the [Glade](#) file.

For instance:

```
Gtk::Table* pTable = dynamic_cast<Gtk::Table*>(refXml->get_widget("mytable"));
```

Note that you are responsible for deleting top-level widgets (windows and dialogs). Other widgets are instantiated as managed so they will be deleted automatically if you add them to a container.

Parameters

<i>name</i>	The name of the widget.
-------------	-------------------------

Returns

A pointer to the widget, or 0 on failure.

6.2.3.7 get_widget() [2/2]

```
template<class T_Widget >
T_Widget * Gnome::Glade::Xml::get_widget (
    const Glib::ustring & name,
    T_Widget *& widget) [inline]
```

More convenient way of getting a widget from the [Glade](#) file.

It allows for a shorter syntax with less repetition. For instance:

```
Gtk::Table* pTable = 0;
refXml->get_widget("mytable", pTable);
```

This method prints a warning message to the console if the widget doesn't exist or has the wrong type, so you don't need to check that manually.

Note that you are responsible for deleting top-level widgets (windows and dialogs) instantiated by the [Xml](#) object. Other widgets are instantiated as managed so they will be deleted automatically if you add them to a container widget.

Parameters

<i>name</i>	The name of the widget.
-------------	-------------------------

Return values

<i>widget</i>	A pointer to the widget, or 0 on failure.
---------------	---

Returns

The value of *widget*.

6.2.3.8 get_widget_checked()

```
Gtk::Widget * Gnome::Glade::Xml::get_widget_checked (
    const Glib::ustring & name,
    GType type) [protected]
```

6.2.3.9 get_widget_derived()

```
template<class T_Widget >
T_Widget * Gnome::Glade::Xml::get_widget_derived (
    const Glib::ustring & name,
    T_Widget *& widget) [inline]
```

This provides a pointer to a widget whose details are specified in the [Glade](#) file, but which is implemented by your own derived class.

Your class must have a constructor like so:

```
DerivedDialog::DerivedDialog(BaseObjectType* cobject, const Glib::RefPtr<Gnome::Glade::Xml>& refGlade)
: Gtk::Dialog(cobject) //Calls the base class constructor
```

For instance:

```
Gtk::DerivedBox* pBox = 0;
refXml->get_widget_derived("mybox", pBox);
```

Parameters

<i>name</i>	The name of the widget.
-------------	-------------------------

Return values

<i>widget</i>	A pointer to the widget, or 0 on failure.
---------------	---

Returns

The value of *widget*.

6.2.3.10 get_widget_name()

```
static Glib::ustring Gnome::Glade::Xml::get_widget_name (
    Gtk::Widget & widget) [static]
```

6.2.3.11 get_widget_prefix()

```
Glib::ListHandle< Gtk::Widget * > Gnome::Glade::Xml::get_widget_prefix (
    const Glib::ustring & name)
```

6.2.3.12 get_widget_tree()

```
static Glib::RefPtr< Xml > Gnome::Glade::Xml::get_widget_tree (
    Gtk::Widget & widget) [static]
```

6.2.3.13 gobj() [1/2]

```
GladeXML * Gnome::Glade::Xml::gobj () [inline]
```

Provides access to the underlying C GObject.

6.2.3.14 gobj() [2/2]

```
const GladeXML * Gnome::Glade::Xml::gobj () const [inline]
```

Provides access to the underlying C GObject.

6.2.3.15 gobj_copy()

```
GladeXML * Gnome::Glade::Xml::gobj_copy ()
```

Provides access to the underlying C instance. The caller is responsible for unrefing it. Use when directly setting fields in structs.

6.2.3.16 lookup_type_vfunc()

```
virtual GType Gnome::Glade::Xml::lookup_type_vfunc (
    const Glib::ustring & classname) [protected], [virtual]
```

- interface for changing the custom widget handling */

6.2.3.17 relative_file()

```
std::string Gnome::Glade::Xml::relative_file (
    const std::string & filename) const
```

6.2.3.18 reparent_widget()

```
void Gnome::Glade::Xml::reparent_widget (
    const Glib::ustring & name,
    Gtk::Container & container)
```

Take the widget from the glade-generated container and put it in another container.

6.2.4 Friends And Related Symbol Documentation

6.2.4.1 wrap()

```
Glib::RefPtr< Gnome::Glade::Xml > wrap (
    GladeXML * object,
    bool take_copy = false) [related]
```

A Glib::wrap() method for this object.

Parameters

<i>object</i>	The C instance.
<i>take_copy</i>	False if the result should take ownership of the C instance. True if it should take a new copy or ref.

Returns

A C++ instance that wraps this C instance.

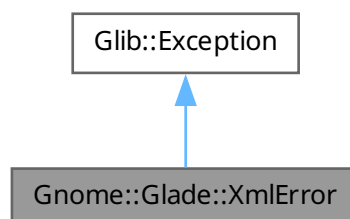
The documentation for this class was generated from the following file:

- [xml.h](#)

6.3 Gnome::Glade::XmlError Class Reference

```
#include <xml.h>
```

Inheritance diagram for Gnome::Glade::XmlError:



Public Member Functions

- [XmlError](#) (const Glib::ustring &message)
- virtual [~XmlError](#) () throw ()
- [XmlError](#) (const [XmlError](#) &other)
- [XmlError](#) & [operator=](#) (const [XmlError](#) &other)
- virtual Glib::ustring [what](#) () const

6.3.1 Constructor & Destructor Documentation

6.3.1.1 XmlError() [1/2]

```
Gnome::Glade::XmlError::XmlError (  
    const Glib::ustring & message) [explicit]
```

6.3.1.2 ~XmlError()

```
virtual Gnome::Glade::XmlError::~~XmlError () throw ( ) [virtual]
```

6.3.1.3 XmlError() [2/2]

```
Gnome::Glade::XmlError::XmlError (  
    const XmlError & other)
```

6.3.2 Member Function Documentation

6.3.2.1 operator=()

```
XmlError & Gnome::Glade::XmlError::operator= (  
    const XmlError & other)
```

6.3.2.2 what()

```
virtual Glib::ustring Gnome::Glade::XmlError::what () const [virtual]
```

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

- [xml.h](#)

Chapter 7

File Documentation

7.1 init.h File Reference

Namespaces

- namespace [Gnome](#)
- namespace [Gnome::Glade](#)

7.2 variablesmap.h File Reference

Classes

- class [Gnome::Glade::VariablesMap](#)
Associates named [Glade](#) widgets with member variables.

Namespaces

- namespace [Gnome](#)
- namespace [Gnome::Glade](#)

7.3 xml.h File Reference

Classes

- class [Gnome::Glade::XmlError](#)
- class [Gnome::Glade::Xml](#)

Namespaces

- namespace [Gnome](#)
- namespace [Gnome::Glade](#)
- namespace [Glib](#)

Index

- ~VariablesMap
 - Gnome::Glade::VariablesMap, [12](#)
- ~Xml
 - Gnome::Glade::Xml, [16](#)
- ~XmlError
 - Gnome::Glade::XmlError, [22](#)
- connect_clicked
 - Gnome::Glade::Xml, [17](#)
- connect_widget
 - Gnome::Glade::VariablesMap, [12](#), [13](#)
- create
 - Gnome::Glade::Xml, [17](#)
- create_from_buffer
 - Gnome::Glade::Xml, [17](#)
- Error
 - Gnome::Glade::Xml, [16](#)
- get_cwidget
 - Gnome::Glade::Xml, [18](#)
- get_filename
 - Gnome::Glade::Xml, [18](#)
- get_widget
 - Gnome::Glade::Xml, [18](#)
- get_widget_checked
 - Gnome::Glade::Xml, [19](#)
- get_widget_derived
 - Gnome::Glade::Xml, [19](#)
- get_widget_name
 - Gnome::Glade::Xml, [19](#)
- get_widget_prefix
 - Gnome::Glade::Xml, [20](#)
- get_widget_tree
 - Gnome::Glade::Xml, [20](#)
- Glib, [9](#)
- Gnome, [9](#)
- Gnome::Glade, [9](#)
- Gnome::Glade::VariablesMap, [11](#)
 - ~VariablesMap, [12](#)
 - connect_widget, [12](#), [13](#)
 - m_mapWidgetsToVariables, [14](#)
 - m_refGlade, [14](#)
 - transfer_one_widget, [13](#)
 - transfer_variables_to_widgets, [13](#)
 - transfer_widgets_to_variables, [13](#)
 - type_mapWidgetsToVariables, [12](#)
 - validate_widgets, [13](#)
 - VariablesMap, [12](#)
- Gnome::Glade::Xml, [14](#)
 - ~Xml, [16](#)
 - connect_clicked, [17](#)
 - create, [17](#)
 - create_from_buffer, [17](#)
 - Error, [16](#)
 - get_cwidget, [18](#)
 - get_filename, [18](#)
 - get_widget, [18](#)
 - get_widget_checked, [19](#)
 - get_widget_derived, [19](#)
 - get_widget_name, [19](#)
 - get_widget_prefix, [20](#)
 - get_widget_tree, [20](#)
 - gobj, [20](#)
 - gobj_copy, [20](#)
 - lookup_type_vfunc, [20](#)
 - relative_file, [20](#)
 - reparent_widget, [20](#)
 - wrap, [21](#)
 - Xml, [16](#)
- Gnome::Glade::XmlError, [21](#)
 - ~XmlError, [22](#)
 - operator=, [22](#)
 - what, [22](#)
 - XmlError, [22](#)
- gobj
 - Gnome::Glade::Xml, [20](#)
- gobj_copy
 - Gnome::Glade::Xml, [20](#)
- init.h, [23](#)
- lookup_type_vfunc
 - Gnome::Glade::Xml, [20](#)
- m_mapWidgetsToVariables
 - Gnome::Glade::VariablesMap, [14](#)
- m_refGlade
 - Gnome::Glade::VariablesMap, [14](#)
- operator=
 - Gnome::Glade::XmlError, [22](#)
- relative_file
 - Gnome::Glade::Xml, [20](#)
- reparent_widget
 - Gnome::Glade::Xml, [20](#)
- transfer_one_widget
 - Gnome::Glade::VariablesMap, [13](#)
- transfer_variables_to_widgets

- Gnome::Glade::VariablesMap, [13](#)
- transfer_widgets_to_variables
 - Gnome::Glade::VariablesMap, [13](#)
- type_mapWidgetsToVariables
 - Gnome::Glade::VariablesMap, [12](#)
- validate_widgets
 - Gnome::Glade::VariablesMap, [13](#)
- VariablesMap
 - Gnome::Glade::VariablesMap, [12](#)
- variablesmap.h, [23](#)
- what
 - Gnome::Glade::XmlError, [22](#)
- wrap
 - Gnome::Glade::Xml, [21](#)
- Xml
 - Gnome::Glade::Xml, [16](#)
- xml.h, [23](#)
- XmlError
 - Gnome::Glade::XmlError, [22](#)