libyui

3.0.10

Generated by Doxygen 1.8.3.1

Fri Aug 30 2013 02:24:58

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 FSize Class Reference

```
#include <FSize.h>
```

Public Types

- enum Unit {
  B = 0, **K**, **M**, **G**,
  T }

Public Member Functions

- FSize (const long long size_r=0)
- FSize (const long long size_r, const Unit unit_r)
- FSize (const std::string &sizeStr, const Unit unit_r=B)
- operator long long () const
- FSize & **operator+=** (const long long rhs)
- FSize & **operator-=** (const long long rhs)
- FSize & **operator∗=** (const long long rhs)
- FSize & **operator/=** (const long long rhs)
- FSize & **operator++** ()
- FSize & **operator--** ()
- FSize **operator++** (int)
- FSize **operator--** (int)
- FSize & fillBlock (FSize blocksize_r=KB)
- FSize fullBlock (FSize blocksize_r=KB) const
- long long operator() (const Unit unit_r) const
- Unit bestUnit () const
- std::string form (const Unit unit_r, unsigned fw=0, unsigned prec=bestPrec, const bool showunit=true) const
- std::string form (unsigned fw=0, unsigned prec=bestPrec, const bool showunit=true) const
- std::string asString () const

Static Public Member Functions

- static long long factor (const Unit unit_r)
- static const char ∗ unit (const Unit unit_r)

Static Public Attributes

- static const long long **KB** = 1024
- static const long long **MB** = 1024 ∗ KB
- static const long long **GB** = 1024 ∗ MB
- static const long long **TB** = 1024 ∗ GB
- static const unsigned bestPrec = (unsigned)-1

3.1.1 Detailed Description

Store and operate on (file/package/partition) sizes (long long).

Definition at line 39 of file FSize.h.

3.1.2 Member Enumeration Documentation

3.1.2.1 enum FSize::Unit

The Units

Definition at line 46 of file FSize.h.

3.1.3 Constructor & Destructor Documentation

3.1.3.1 FSize::FSize (const long long *size_r* = `0`) `[inline]`

Construct from size in Byte.

Definition at line 95 of file FSize.h.

3.1.3.2 FSize::FSize (const long long *size_r*, const Unit *unit_r*) `[inline]`

Construct from size in certain unit. E.g. `FSize( 1, FSize::K )makes 1024 Byte.`

Definition at line 103 of file FSize.h.

3.1.3.3 FSize::FSize (const std::string & *sizeStr*, const Unit *unit_r* = `B`)

Construct from string containing a number in given unit.

Definition at line 34 of file FSize.cc.

3.1.4 Member Function Documentation

3.1.4.1 std::string FSize::asString ( ) const

Default string representation (precision 1 and unit appended).

Definition at line 122 of file FSize.cc.

Here is the call graph for this function:

3.1.4.2 FSize::Unit FSize::bestUnit ( ) const

Return the best unit for string representation.

Definition at line 66 of file FSize.cc.

3.1.4.3 static long long FSize::factor (const Unit *unit_r*) [inline], [static]

Return ammount of Byte in Unit.

Definition at line 65 of file FSize.h.

3.1.4.4 FSize & FSize::fillBlock (FSize *blocksize_r* = KB)

Adjust size to multiple of `blocksize_r`

Definition at line 46 of file FSize.cc.

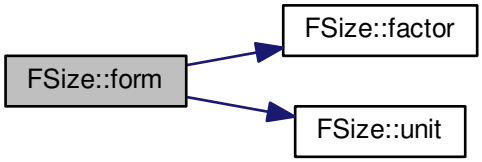
3.1.4.5 std::string FSize::form (const Unit *unit_r*, unsigned *fw* = 0, unsigned *prec* = bestPrec, const bool *showunit* = true) const

Return string representation in given Unit. Parameter `fw` and `prec` denote field width and precision as in a "%*.*f" printf format string. Avalue of `bestPrec` automatically picks an appropriate precision depending on the unit. If `showunit` ist true, the string representaion of Unit is *appendedseparated by a single blank.*

*If Unit is **B**yte, precision is set to zero.*

Definition at line 87 of file FSize.cc.

Here is the call graph for this function:

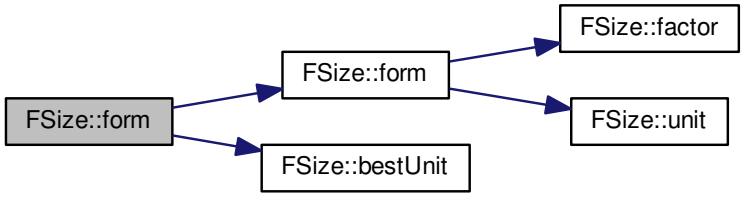


3.1.4.6 std::string FSize::form (unsigned *fw* = `0`, unsigned *prec* = **bestPrec**, const bool *showunit* = `true`) const [inline]

Return string representation in bestUnit.

Definition at line 169 of file FSize.h.

Here is the call graph for this function:



3.1.4.7 FSize FSize::fullBlock (FSize *blocksize_r* = `KB`) const [inline]

Return size adjusted to multiple of `blocksize_r`

Definition at line 136 of file FSize.h.

Here is the call graph for this function:

3.1.4.8 FSize::operator long long ( ) const [inline]

Conversion to long long

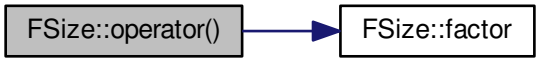
Definition at line 115 of file FSize.h.

3.1.4.9 long long FSize::operator() (const Unit *unit_r*) const [inline]

Return size in Unit (not rounded)

Definition at line 141 of file FSize.h.

Here is the call graph for this function:



3.1.4.10 static const char* FSize::unit (const Unit *unit_r*) [inline], [static]

String representation of Unit.

Definition at line 79 of file FSize.h.

3.1.5 Member Data Documentation

3.1.5.1 const unsigned FSize::bestPrec = (unsigned)-1 [static]

Used as precision argument to form(), the 'best' precision according to Unist is chosen.

Definition at line 152 of file FSize.h.

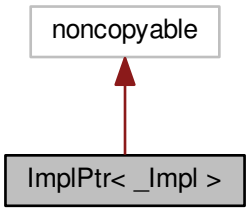
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/FSize.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/FSize.cc

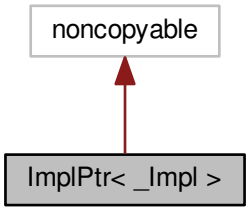
3.2 ImplPtr< _Impl > Class Template Reference

`#include <ImplPtr.h>`

Inheritance diagram for ImplPtr< _Impl >:



Collaboration diagram for ImplPtr< _Impl >:



Public Types

- typedef _Impl **element_type**

Public Member Functions

- **ImplPtr** (_Impl *impl_r=0)
- void **reset** (_Impl *impl_r=0)
- void **swap** (ImplPtr rhs)
- **operator bool** () const

- const _Impl & **operator**∗ () const
- const _Impl ∗ **operator-**> () const
- const _Impl ∗ **get** () const
- _Impl & **operator**∗ ()
- _Impl ∗ **operator-**> ()
- _Impl ∗ **get** ()

3.2.1 Detailed Description

template<class _Impl>class ImplPtr< _Impl >

Helper template class for implementation pointers (pointers to a private class or structure that hold the member variables of a higher-level class that is part of a public API).

This pointer class maintains constness of its parent class, i.e. if it is used in a const class the class this pointer points to will also be const.

This class automatically deletes the class it points to in its destructor.

Definition at line 42 of file ImplPtr.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/ImplPtr.h

3.3 OptimizeChanges Class Reference

```
#include <YWidget_OptimizeChanges.h>
```

Public Member Functions

- **OptimizeChanges** (YWidget &w)

3.3.1 Detailed Description

Helper class that calls startMultipleChanges() in its constructor and cares about the necessary call to doneMultiple-Changes() when it goes out of scope.

Definition at line 44 of file YWidget_OptimizeChanges.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWidget_OptimizeChanges.h

3.4 YWidget::OptimizeChanges Class Reference

```
#include <YWidget.h>
```

Public Member Functions

- **OptimizeChanges** (YWidget &w)

3.4.1 Detailed Description

Helper class that calls startMultipleChanges() in its constructor and cares about the necessary call to doneMultiple-Changes() when it goes out of scope.

Definition at line 45 of file YWidget.h.

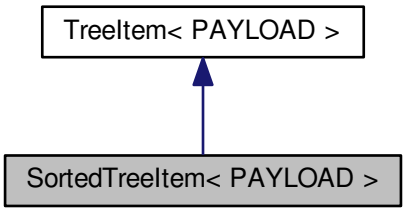
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWidget_OptimizeChanges.h

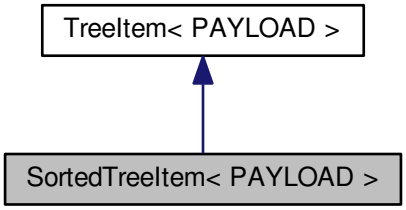
3.5 SortedTreeItem< PAYLOAD > Class Template Reference

```
#include <TreeItem.h>
```

Inheritance diagram for SortedTreeItem< PAYLOAD >:



Collaboration diagram for SortedTreeItem< PAYLOAD >:



Public Member Functions

- SortedTreeItem (PAYLOAD val, SortedTreeItem< PAYLOAD > *parentItem=0)
- virtual ~SortedTreeItem ()

- void insertChildSorted (SortedTreeItem< PAYLOAD > *newChild)
- SortedTreeItem< PAYLOAD > * parent () const
- SortedTreeItem< PAYLOAD > * next () const
- SortedTreeItem< PAYLOAD > * firstChild () const

Additional Inherited Members

3.5.1 Detailed Description

template<class PAYLOAD>class SortedTreeItem< PAYLOAD >

Template class for tree items that maintain sort order.

Class 'PAYLOAD' to provide operator<() in addition to what template 'TreeItem' requires.

Definition at line 191 of file TreeItem.h.

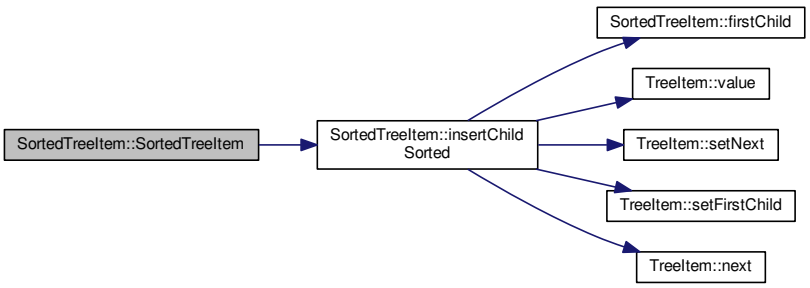
3.5.2 Constructor & Destructor Documentation

3.5.2.1 template<class PAYLOAD> SortedTreeItem< PAYLOAD >::SortedTreeItem (PAYLOAD *val*, SortedTreeItem< PAYLOAD > * *parentItem* = 0) [inline]

Constructor. Creates a new tree item with value "val" and inserts it in ascending sort order into the children list of "parent".

Definition at line 199 of file TreeItem.h.

Here is the call graph for this function:



3.5.2.2 template<class PAYLOAD> virtual SortedTreeItem< PAYLOAD >::~SortedTreeItem ( ) [inline], [virtual]

Destructor.

Definition at line 220 of file TreeItem.h.

3.5.3 Member Function Documentation

3.5.3.1 template<class PAYLOAD> SortedTreeItem<PAYLOAD>* SortedTreeItem< PAYLOAD >::firstChild ( ) const [inline]

Returns this item's first child or 0 if there is none.

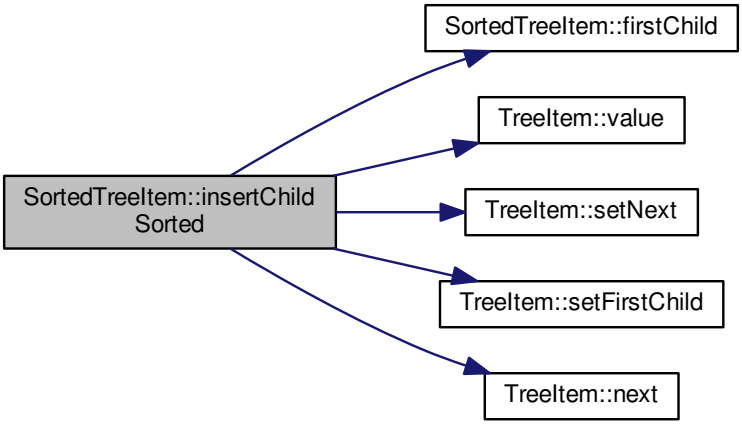
Definition at line 276 of file TreeItem.h.

3.5.3.2 template<class PAYLOAD> void SortedTreeItem< PAYLOAD >::insertChildSorted (SortedTreeItem< PAYLOAD > * *newChild*) [inline]

Insert a child into the internal children list in ascending sort order. Called from the new child's constructor, thus 'public'.

Definition at line 227 of file TreeItem.h.

Here is the call graph for this function:



3.5.3.3 template<class PAYLOAD> SortedTreeItem<PAYLOAD>* SortedTreeItem< PAYLOAD >::next ( ) const [inline]

Returns this item's next sibling or 0 if there is none.

Definition at line 270 of file TreeItem.h.

3.5.3.4 template<class PAYLOAD> SortedTreeItem<PAYLOAD>* SortedTreeItem< PAYLOAD >::parent ( ) const [inline]

Returns this item's parent or 0 if there is none.

Definition at line 264 of file TreeItem.h.

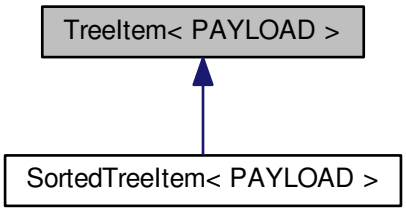
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/TreeItem.h

3.6 TreeItem< PAYLOAD > Class Template Reference

```
#include <TreeItem.h>
```

Inheritance diagram for TreeItem< PAYLOAD >:



Public Member Functions

- TreeItem (const PAYLOAD &val, TreeItem< PAYLOAD > *parent=0)
- virtual ~TreeItem ()
- const PAYLOAD & value () const
- void setValue (PAYLOAD newValue)
- TreeItem< PAYLOAD > * parent () const
- TreeItem< PAYLOAD > * next () const
- TreeItem< PAYLOAD > * firstChild () const
- void setParent (TreeItem< PAYLOAD > *newParent)
- void setNext (TreeItem< PAYLOAD > *newNext)
- void setFirstChild (TreeItem< PAYLOAD > *newFirstChild)
- void addChild (TreeItem< PAYLOAD > *newChild)

Protected Member Functions

- TreeItem (PAYLOAD val, bool autoAddChild, TreeItem< PAYLOAD > *parent=0)

Protected Attributes

- PAYLOAD **_value**
- TreeItem< PAYLOAD > * **_parent**
- TreeItem< PAYLOAD > * **_next**
- TreeItem< PAYLOAD > * **_firstChild**

3.6.1 Detailed Description

template<class PAYLOAD>class TreeItem< PAYLOAD >

Template class for tree items that can handle tree children in a generic way - firstChild(), next() and parent(). Each item stores one value of type 'PAYLOAD'.

Class 'PAYLOAD' needs to provide operator=().

Definition at line 40 of file TreeItem.h.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 template<class PAYLOAD> TreeItem< PAYLOAD >::TreeItem (const PAYLOAD & *val*, TreeItem< PAYLOAD > * *parent* = 0) [inline]

Constructor. Creates a new tree item with value "val" and inserts it (without maintaining any meaningful sort order!) into the children list of "parent".

Definition at line 49 of file TreeItem.h.

3.6.2.2 template<class PAYLOAD> TreeItem< PAYLOAD >::TreeItem (PAYLOAD *val*, bool *autoAddChild*, TreeItem< PAYLOAD > * *parent* = 0) [inline], [protected]

Constructor to be called for derived classes: Decide whether or not to automatically insert this item into the parent's children list. Useful for derived classes that want to maintain a specific sort order among children.

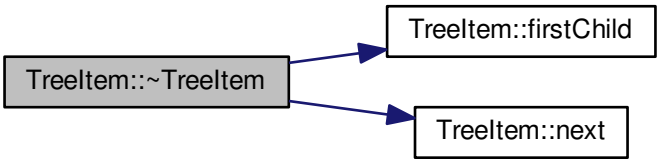
Definition at line 69 of file TreeItem.h.

3.6.2.3 template<class PAYLOAD> virtual TreeItem< PAYLOAD >::~TreeItem ( ) [inline], [virtual]

Destructor. Takes care of children - they will be deleted along with this item.

Definition at line 97 of file TreeItem.h.

Here is the call graph for this function:



3.6.3 Member Function Documentation

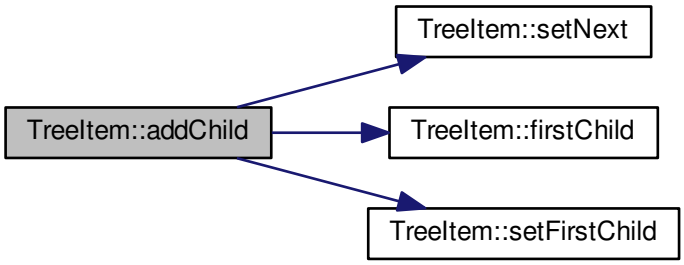
3.6.3.1 template<class PAYLOAD> void TreeItem< PAYLOAD >::addChild (TreeItem< PAYLOAD > * *newChild*) [inline]

Add a child to the internal children list - usually called from within the child's default constructor.

This default method does not maintain any meaningful sorting order - derived classes that require this might want to use the other constructor (with 'autoAddChild' set to 'false') take care of child insertion themselves.

Definition at line 165 of file TreeItem.h.

Here is the call graph for this function:



3.6.3.2 template<class PAYLOAD> TreeItem<PAYLOAD>* TreeItem< PAYLOAD >::firstChild ( ) const [inline]

Returns this item's first child or 0 if there is none.

Definition at line 137 of file TreeItem.h.

3.6.3.3 template<class PAYLOAD> TreeItem<PAYLOAD>* TreeItem< PAYLOAD >::next ( ) const [inline]

Returns this item's next sibling or 0 if there is none.

Definition at line 132 of file TreeItem.h.

3.6.3.4 template<class PAYLOAD> TreeItem<PAYLOAD>* TreeItem< PAYLOAD >::parent ( ) const [inline]

Returns this item's parent or 0 if there is none.

Definition at line 127 of file TreeItem.h.

3.6.3.5 template<class PAYLOAD> void TreeItem< PAYLOAD >::setFirstChild (TreeItem< PAYLOAD > * *newFirstChild*) [inline]

Sets this item's first child.

Definition at line 152 of file TreeItem.h.

3.6.3.6 template<class PAYLOAD> void TreeItem< PAYLOAD >::setNext (TreeItem< PAYLOAD > * *newNext*) [inline]

Sets this item's next sibling.

Definition at line 147 of file TreeItem.h.

3.6.3.7 template<class PAYLOAD> void TreeItem< PAYLOAD >::setParent (TreeItem< PAYLOAD > * *newParent*) [inline]

Sets this item's parent.

Definition at line 142 of file TreeItem.h.

3.6.3.8 template<class PAYLOAD> void TreeItem< PAYLOAD >::setValue (PAYLOAD *newValue*) [inline]

Set this item's value, the "payload".

If the sort order among children of one level is important, overwrite this method and change the sort order according to the new value. The template class itself never calls this.

Definition at line 122 of file TreeItem.h.

3.6.3.9 template<class PAYLOAD> const PAYLOAD& TreeItem< PAYLOAD >::value ( ) const [inline]

Returns this item's value, the "payload".

Definition at line 113 of file TreeItem.h.

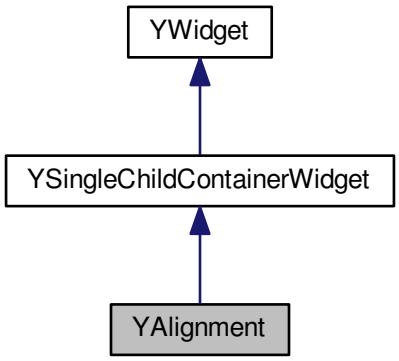
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/TreeItem.h

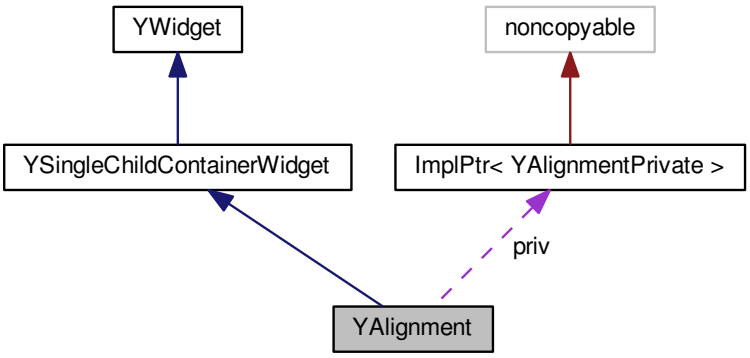
3.7 YAlignment Class Reference

```
#include <YAlignment.h>
```

Inheritance diagram for YAlignment:



Collaboration diagram for YAlignment:



Public Member Functions

- virtual ~YAlignment ()
- virtual const char * widgetClass () const
- YAlignmentType alignment (YUIDimension dim) const
- int leftMargin () const
- int rightMargin () const
- int topMargin () const
- int bottomMargin () const
- int totalMargins (YUIDimension dim) const

- void setLeftMargin (int margin)
- void setRightMargin (int margin)
- void setTopMargin (int margin)
- void setBottomMargin (int margin)
- int minWidth () const
- int minHeight () const
- void setMinWidth (int width)
- void setMinHeight (int height)
- virtual void setBackgroundPixmap (const std::string &pixmapFileName)
- std::string backgroundPixmap () const
- virtual void addChild (YWidget *child)
- virtual void moveChild (YWidget *child, int newx, int newy)=0
- virtual bool stretchable (YUIDimension dim) const
- virtual int preferredWidth ()
- virtual int preferredHeight ()
- virtual void setSize (int newWidth, int newHeight)

Protected Member Functions

- YAlignment (YWidget *parent, YAlignmentType horAlign, YAlignmentType vertAlign)

Protected Attributes

- ImplPtr< YAlignmentPrivate > **priv**

3.7.1 Detailed Description

Implementation of all the alignment widgets:

- Left, Right, HCenter,
- Top, Bottom, VCenter,
- HVCenter
- MinSize, MinWidth, MinHeight

Definition at line 41 of file YAlignment.h.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 YAlignment::YAlignment (YWidget * *parent*, YAlignmentType *horAlign*, YAlignmentType *vertAlign*) [protected]

Constructor.

Definition at line 76 of file YAlignment.cc.

3.7.2.2 YAlignment::~YAlignment ( ) [virtual]

Destructor.

Definition at line 86 of file YAlignment.cc.

3.7.3 Member Function Documentation

3.7.3.1 void YAlignment::addChild (YWidget * *child*) [virtual]

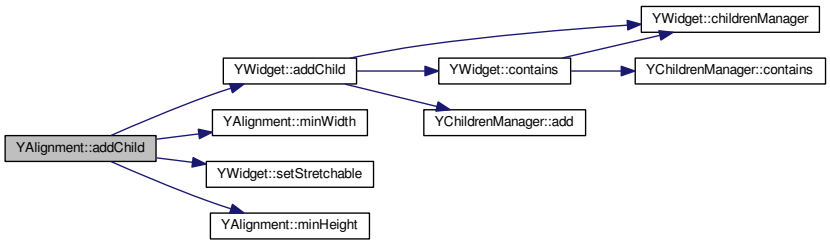
Add a child widget.

Reimplemented from YSingleChildContainerWidget to propagate stretchability down to the single child.

Reimplemented from YWidget.

Definition at line 177 of file YAlignment.cc.

Here is the call graph for this function:



3.7.3.2 YAlignmentType YAlignment::alignment (YUIDimension *dim*) const

Return the alignment in the specified dimension.

Definition at line 93 of file YAlignment.cc.

3.7.3.3 std::string YAlignment::backgroundPixmap ( ) const

Return the name of the background pixmap or an empty string, if there is none.

Definition at line 171 of file YAlignment.cc.

3.7.3.4 int YAlignment::bottomMargin ( ) const

Return the bottom margin in pixels, the distance between the bottom edge of this alignment and the bottom edge of the child widget.

Definition at line 117 of file YAlignment.cc.

3.7.3.5 int YAlignment::leftMargin ( ) const

Return the left margin in pixels, the distance between the left edge of this alignment and the left edge of the child widget.

Definition at line 99 of file YAlignment.cc.

3.7.3.6 int YAlignment::minHeight ( ) const

Return the minimum height of this alignment or 0 if none is set. preferredHeight() will never return less than this value.

Definition at line 153 of file YAlignment.cc.

3.7.3.7 int YAlignment::minWidth ( ) const

Return the minimum width of this alignment or 0 if none is set. preferredWidth() will never return less than this value.

Definition at line 147 of file YAlignment.cc.

3.7.3.8 virtual void YAlignment::moveChild (YWidget * *child*, int *newx*, int *newy*) [pure virtual]

Move a child widget to a new position.

3.7.3.9 int YAlignment::preferredHeight ( ) [virtual]

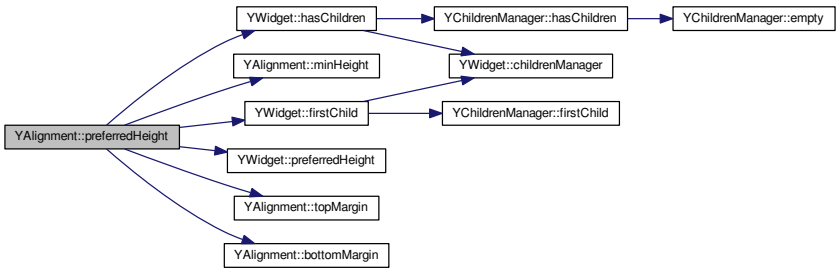
Preferred height of the widget.

Reimplemented from YWidget.

Reimplemented from YSingleChildContainerWidget.

Definition at line 207 of file YAlignment.cc.

Here is the call graph for this function:



3.7.3.10 int YAlignment::preferredWidth ( ) [virtual]

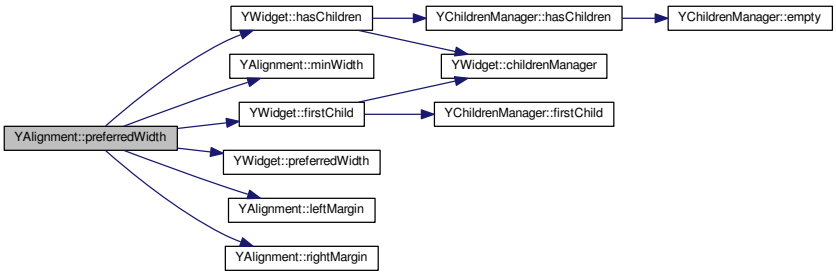
Preferred width of the widget.

Reimplemented from YWidget.

Reimplemented from YSingleChildContainerWidget.

Definition at line 195 of file YAlignment.cc.

Here is the call graph for this function:



3.7.3.11 int YAlignment::rightMargin ( ) const

Return the right margin in pixels, the distance between the right edge of this alignment and the right edge of the child widget.

Definition at line 105 of file YAlignment.cc.

3.7.3.12 void YAlignment::setBackgroundPixmap (const std::string & *pixmapFileName*) [virtual]

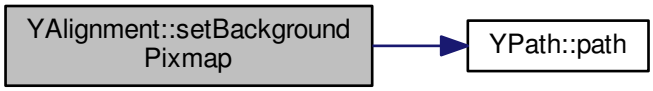
Set a background pixmap.

Derived classes may want to overwrite this.

This parent method should be called first in the overwritten method to ensure path expansion is done as specified (prepend the theme path ("/usr/share/libyui/theme/") if the path doesn't start with "/" or ").

Definition at line 334 of file YAlignment.cc.

Here is the call graph for this function:



3.7.3.13 void YAlignment::setBottomMargin (int *margin*)

Set the bottom margin in pixels.

Definition at line 141 of file YAlignment.cc.

3.7.3.14 void YAlignment::setLeftMargin (int *margin*)

Set the left margin in pixels.

Definition at line 123 of file YAlignment.cc.

3.7.3.15 void YAlignment::setMinHeight (int *height*)

Set the minimum height to return for preferredHeight().

Definition at line 165 of file YAlignment.cc.

3.7.3.16 void YAlignment::setMinWidth (int *width*)

Set the minimum width to return for preferredWidth().

Definition at line 159 of file YAlignment.cc.

3.7.3.17 void YAlignment::setRightMargin (int *margin*)

Set the right margin in pixels.

Definition at line 129 of file YAlignment.cc.

3.7.3.18 void YAlignment::setSize (int *newWidth,* int *newHeight*) [virtual]

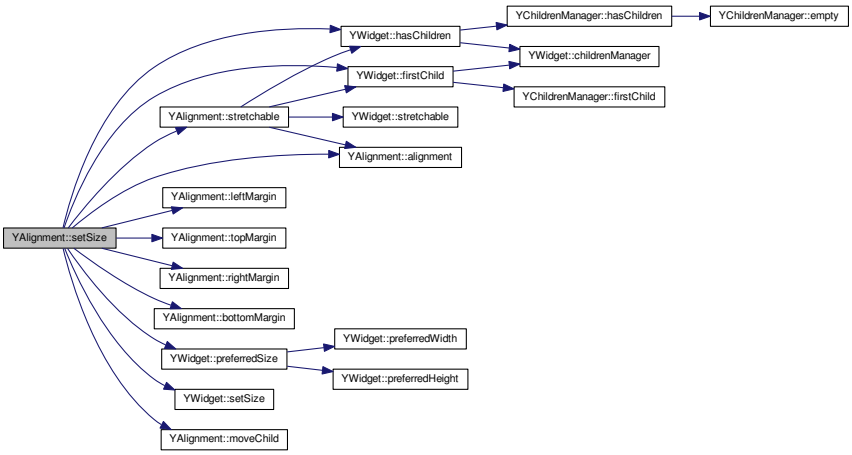
Set the current size and move the child widget according to its alignment.

Derived classes should reimplement this, but call this base class function in their own implementation.

Reimplemented from YSingleChildContainerWidget.

Definition at line 219 of file YAlignment.cc.

Here is the call graph for this function:



3.7.3.19 void YAlignment::setTopMargin (int *margin*)

Set the top margin in pixels.

Definition at line 135 of file YAlignment.cc.

3.7.3.20 bool YAlignment::stretchable (YUIDimension *dim*) const [virtual]

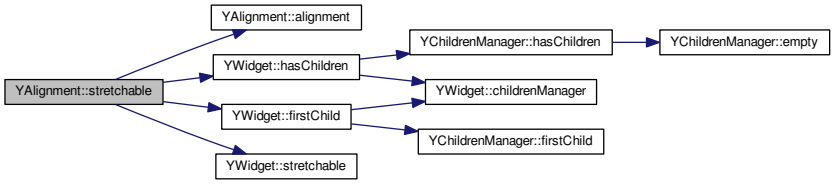
Return this widget's stretchability. Reimplemented from YWidget.

In an aligned dimension the widget is always stretchable. In an unchanged dimension the widget is stretchable if the child is stretchable.

Reimplemented from YSingleChildContainerWidget.

Definition at line 186 of file YAlignment.cc.

Here is the call graph for this function:



3.7.3.21 int YAlignment::topMargin ( ) const

Return the top margin in pixels, the distance between the top edge of this alignment and the top edge of the child widget.

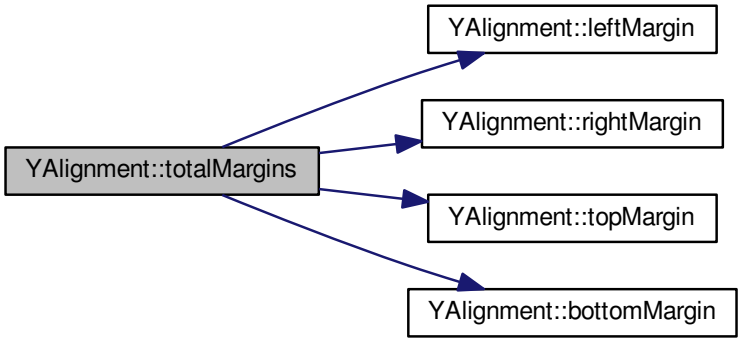
Definition at line 111 of file YAlignment.cc.

3.7.3.22 int YAlignment::totalMargins (YUIDimension *dim*) const

Return the sum of all margins in the specified dimension.

Definition at line 326 of file YAlignment.cc.

Here is the call graph for this function:



3.7.3.23 const char * YAlignment::widgetClass ( ) const [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

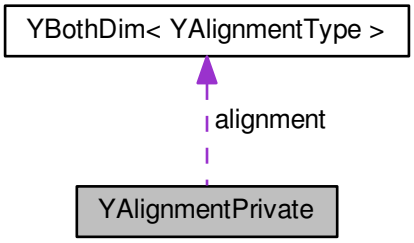
Definition at line 353 of file YAlignment.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YAlignment.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YAlignment.cc

3.8 YAlignmentPrivate Struct Reference

Collaboration diagram for YAlignmentPrivate:



Public Member Functions

- YAlignmentPrivate (YAlignmentType horAlign, YAlignmentType vertAlign)

Public Attributes

- int **leftMargin**
- int **rightMargin**
- int **topMargin**
- int **bottomMargin**
- int **minWidth**
- int **minHeight**
- std::string **backgroundPixmap**
- YBothDim< YAlignmentType > **alignment**

3.8.1 Detailed Description

Definition at line 38 of file YAlignment.cc.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 YAlignmentPrivate::YAlignmentPrivate (YAlignmentType *horAlign*, YAlignmentType *vertAlign*) [inline]

Constructor.

Definition at line 43 of file YAlignment.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YAlignment.cc

3.9 YApplication Class Reference

```
#include <YApplication.h>
```

Public Member Functions

- YWidget * findWidget (YWidgetID *id, bool doThrow=true) const
- virtual std::string iconBasePath () const
- virtual void setIconBasePath (const std::string &newIconBasePath)
- YIconLoader * **iconLoader** ()
- int defaultFunctionKey (const std::string &label) const
- void setDefaultFunctionKey (const std::string &label, int fkey)
- void clearDefaultFunctionKeys ()
- virtual void setLanguage (const std::string &language, const std::string &encoding=std::string())
- std::string language (bool stripEncoding=false) const
- virtual std::string glyph (const std::string &glyphSymbolName)
- virtual std::string askForExistingDirectory (const std::string &startDir, const std::string &headline)=0

- virtual std::string askForExistingFile (const std::string &startWith, const std::string &filter, const std::string &headline)=0
- virtual std::string askForSaveFileName (const std::string &startWith, const std::string &filter, const std::string &headline)=0
- virtual bool openContextMenu (const YItemCollection &itemCollection)
- virtual void setProductName (const std::string &productName)
- std::string productName () const
- virtual int deviceUnits (YUIDimension dim, float layoutUnits)
- virtual float layoutUnits (YUIDimension dim, int deviceUnits)
- virtual void setReverseLayout (bool reverse)
- bool reverseLayout () const
- virtual void busyCursor ()
- virtual void normalCursor ()
- virtual void makeScreenShot (const std::string &fileName)
- virtual void beep ()
- virtual void redrawScreen ()
- virtual void initConsoleKeyboard ()
- virtual void setConsoleFont (const std::string &console_magic, const std::string &font, const std::string &screen_map, const std::string &unicode_map, const std::string &language)
- virtual int runInTerminal (const std::string &command)
- virtual int **displayWidth** ()=0
- virtual int **displayHeight** ()=0
- virtual int **displayDepth** ()=0
- virtual long **displayColors** ()=0
- virtual int **defaultWidth** ()=0
- virtual int **defaultHeight** ()=0
- virtual bool **isTextMode** ()=0
- virtual bool **hasImageSupport** ()=0
- virtual bool **hasIconSupport** ()=0
- virtual bool **hasAnimationSupport** ()=0
- virtual bool **hasFullUtf8Support** ()=0
- virtual bool **richTextSupportsTable** ()=0
- virtual bool **leftHandedMouse** ()=0
- virtual bool **hasWizardDialogSupport** ()
- virtual void setApplicationTitle (const std::string &title)
- virtual const std::string & applicationTitle () const
- virtual void setApplicationIcon (const std::string &icon)
- virtual const std::string & applicationIcon () const

Protected Member Functions

- YApplication ()
- virtual ~YApplication ()

Friends

- class **YUI**

3.9.1 Detailed Description

Class for application-wide values and functions. This is a singleton. Access and create it via the static functions in YUI.

Definition at line 44 of file YApplication.h.

3.9.2 Constructor & Destructor Documentation

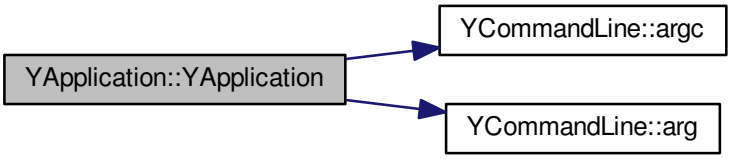
3.9.2.1 YApplication::YApplication ( ) [protected]

Constructor.

Use YUI::app() to get the singleton for this class.

Definition at line 60 of file YApplication.cc.

Here is the call graph for this function:



3.9.2.2 YApplication::~YApplication ( ) [protected], [virtual]

Destructor.

Definition at line 71 of file YApplication.cc.

3.9.3 Member Function Documentation

3.9.3.1 const std::string & YApplication::applicationIcon ( ) const [virtual]

Get the application Icon

Default icon is an empty string

Definition at line 270 of file YApplication.cc.

3.9.3.2 const std::string & YApplication::applicationTitle ( ) const [virtual]

Get the application title

Default title is the running command (argv[0])

Definition at line 261 of file YApplication.cc.

3.9.3.3 virtual std::string YApplication::askForExistingDirectory (const std::string & *startDir,* const std::string & *headline*) [pure virtual]

Open a directory selection box and prompt the user for an existing directory.

'startDir' is the initial directory that is displayed.

'headline' is an explanatory text for the directory selection box. Graphical UIs may omit that if no window manager is running.

Returns the selected directory name or an empty string if the user canceled the operation.

Derived classes are required to implement this.

3.9.3.4 virtual std::string YApplication::askForExistingFile (const std::string & *startWith,* const std::string & *filter,* const std::string & *headline*) [pure virtual]

Open a file selection box and prompt the user for an existing file.

'startWith' is the initial directory or file.

'filter' is one or more blank-separated file patterns, e.g. "∗.png ∗.jpg"

'headline' is an explanatory text for the file selection box. Graphical UIs may omit that if no window manager is running.

Returns the selected file name or an empty string if the user canceled the operation.

Derived classes are required to implement this.

3.9.3.5 virtual std::string YApplication::askForSaveFileName (const std::string & *startWith,* const std::string & *filter,* const std::string & *headline*) [pure virtual]

Open a file selection box and prompt the user for a file to save data to. Automatically asks for confirmation if the user selects an existing file.

'startWith' is the initial directory or file.

'filter' is one or more blank-separated file patterns, e.g. "∗.png ∗.jpg"

'headline' is an explanatory text for the file selection box. Graphical UIs may omit that if no window manager is running.

Returns the selected file name or an empty string if the user canceled the operation.

Derived classes are required to implement this.

3.9.3.6 virtual void YApplication::beep ( ) [inline],[virtual]

Beep. This default implementation does nothing.

Definition at line 305 of file YApplication.h.

3.9.3.7 virtual void YApplication::busyCursor ( ) [inline],[virtual]

Change the (mouse) cursor to indicate busy status. This default implementation does nothing.

Definition at line 287 of file YApplication.h.

3.9.3.8 void YApplication::clearDefaultFunctionKeys ( )

Clear all previous label-to-function-key mappings.

Definition at line 159 of file YApplication.cc.

3.9.3.9 int YApplication::defaultFunctionKey (const std::string & *label*) const

Return the default function key number for a widget with the specified label or 0 if there is none. Any keyboard shortcuts that may be contained in 'label' are stripped away before any comparison.

The basic idea behind this concept is to have an easy default mapping from buttons etc. with the same semantics to function keys:

"OK" -> F10 "Accept" -> F10 "Yes" -> F10 "Next" -> F10

"Cancel" -> F9 "No" -> F9 ...

This function returns 10 for F10, F for F9 etc.; 0 means "no function key".

Definition at line 136 of file YApplication.cc.

Here is the call graph for this function:

3.9.3.10 int YApplication::deviceUnits (YUIDimension *dim*, float *layoutUnits*) [virtual]

Convert logical layout spacing units into device dependent units. A default size dialog is assumed to be 80x25 layout spacing units.

Derived classes may want to reimplement this method.

Definition at line 235 of file YApplication.cc.

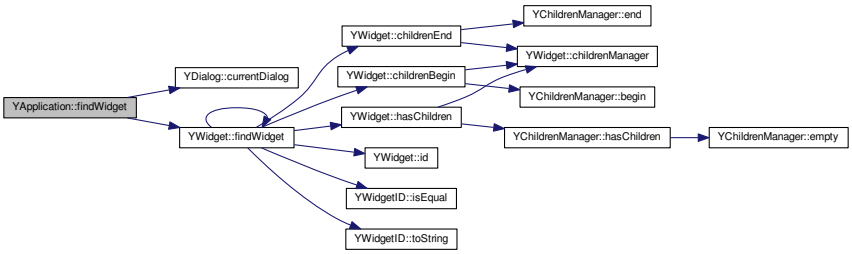
3.9.3.11 YWidget * YApplication::findWidget (YWidgetID * *id*, bool *doThrow* = true) const

Find a widget in the topmost dialog by its ID.

If there is no widget with that ID (or no dialog at all), this function throws a YUIWidgetNotFoundException if 'doThrow' is 'true'. It returns 0 if 'doThrow' is 'false'.

Definition at line 78 of file YApplication.cc.

Here is the call graph for this function:



3.9.3.12 std::string YApplication::glyph (const std::string & *glyphSymbolName*) [virtual]

Return a string for a named glyph:

YUIGlyph_ArrowLeft YUIGlyph_ArrowRight YUIGlyph_ArrowUp YUIGlyph_ArrowDown YUIGlyph_CheckMark YUIGlyph_BulletArrowRight YUIGlyph_BulletCircle YUIGlyph_BulletSquare

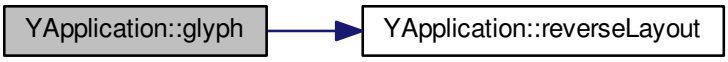
Using this is discouraged in new applications. This method is available for backward compatibility.

This default implementation returns simple textual representations for each glyph simbol (e.g., "->" for YUIGlyphArrorRight).

Derived classes are free to overwrite this. It does not make sense to call this base class method in a new implementation.

Definition at line 208 of file YApplication.cc.

Here is the call graph for this function:



3.9.3.13 std::string YApplication::iconBasePath ( ) const [virtual]

Get the base path for icons used by the UI. Selection widgets like YSelectionBox, YComboBox, etc. or YWizard prepend this to icon specifications that don't use an absolute path.

Definition at line 90 of file YApplication.cc.

3.9.3.14 virtual void YApplication::initConsoleKeyboard ( ) [inline], [virtual]

Initialize the (text) console keyboard. This default implementation does nothing.

Definition at line 322 of file YApplication.h.

3.9.3.15 std::string YApplication::language (bool *stripEncoding* = `false`) const

Return the current language from the locale environment ($LANG). If 'stripEncoding' is true, any encoding (".utf8" etc.) is removed.

Definition at line 184 of file YApplication.cc.

3.9.3.16 float YApplication::layoutUnits (YUIDimension *dim*, int *deviceUnits*) `[virtual]`

Convert device dependent units into logical layout spacing units. A default size dialog is assumed to be 80x25 layout spacing units.

Derived classes may want to reimplement this method.

Definition at line 242 of file YApplication.cc.

Here is the call graph for this function:

3.9.3.17 virtual void YApplication::makeScreenShot (const std::string & *fileName*) `[inline]`, `[virtual]`

Make a screen shot and save it to the specified file. This default implementation does nothing.

Definition at line 299 of file YApplication.h.

3.9.3.18 virtual void YApplication::normalCursor ( ) `[inline]`, `[virtual]`

Change the (mouse) cursor back from busy status to normal. This default implementation does nothing.

Definition at line 293 of file YApplication.h.

3.9.3.19 bool YApplication::openContextMenu (const YItemCollection & *itemCollection*) `[virtual]`

Open a context menu for a widget

'itemCollection' describes the menu structure

Returns true on success (otherwise false).

Derived classes are free to overwrite this.

Definition at line 226 of file YApplication.cc.

3.9.3.20 std::string YApplication::productName ( ) const

Set the current product name ("openSUSE", "SLES", ...).

Definition at line 116 of file YApplication.cc.

3.9.3.21 virtual void YApplication::redrawScreen ( ) [inline], [virtual]

Redraw the screen. This default implementation does nothing.

Definition at line 316 of file YApplication.h.

3.9.3.22 bool YApplication::reverseLayout ( ) const

Returns 'true' if widget geometry should be reversed for languages that have right-to-left writing direction (Arabic, Hebrew).

Definition at line 129 of file YApplication.cc.

3.9.3.23 int YApplication::runInTerminal (const std::string & *command*) [virtual]

Run a shell command (typically an interactive program using NCurses) in a terminal (window).

This is useful for text UIs (e.g., NCurses) that need special preparation prior to running an NCurses-based application and special clean-up afterwards.

This default implementation logs an error and returns -1.

Definition at line 249 of file YApplication.cc.

3.9.3.24 void YApplication::setApplicationIcon (const std::string & *icon*) [virtual]

Set the application Icon

Definition at line 266 of file YApplication.cc.

3.9.3.25 void YApplication::setApplicationTitle (const std::string & *title*) [virtual]

Set the application title

Definition at line 256 of file YApplication.cc.

3.9.3.26 virtual void YApplication::setConsoleFont (const std::string & *console_magic,* const std::string & *font,* const std::string & *screen_map,* const std::string & *unicode_map,* const std::string & *language*) [inline], [virtual]

Set the (text) console font according to the current encoding etc. See the setfont(8) command and the console HowTo for details.

This default implementation does nothing.

Definition at line 330 of file YApplication.h.

3.9.3.27 void YApplication::setDefaultFunctionKey (const std::string & *label,* int *fkey*)

Add a mapping from the specified label to the specified F-key number. This is the counterpart to defaultFunctionKey().

This only affects widgets that are created after this call.

Definition at line 149 of file YApplication.cc.

Here is the call graph for this function:

3.9.3.28 void YApplication::setIconBasePath (const std::string & *newIconBasePath*) [virtual]

Set the icon base path.

Definition at line 97 of file YApplication.cc.

3.9.3.29 void YApplication::setLanguage (const std::string & *language,* const std::string & *encoding* = std::string()) [virtual]

Set language and encoding for the locale environment ($LANG).

This affects UI-internal translations (e.g. for predefined dialogs like file selection), encoding and fonts.

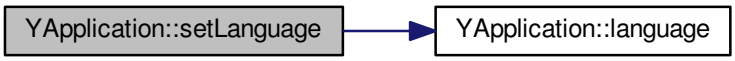
'language' is the ISO short code ("de_DE", "en_US", ...).

'encoding' an (optional) encoding ("utf8", ...) that will be appended if present.

Derived classes can overwrite this method, but they should call this base class method at the beginning of the new implementation.

Definition at line 166 of file YApplication.cc.

Here is the call graph for this function:



3.9.3.30 void YApplication::setProductName (const std::string & *productName*) [virtual]

Set the current product name ("openSUSE", "SLES", ...). This name will be expanded in help texts when the entity is used.

Derived classes can overwrite this method, but they should call this base class method in the new implementation.

Definition at line 109 of file YApplication.cc.

Here is the call graph for this function:

3.9.3.31 void YApplication::setReverseLayout (bool *reverse*) [virtual]

Set reverse layout for Arabic / Hebrew support.

Derived classes can overwrite this method, but they should call this base class method in the new implementation.

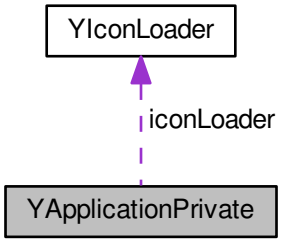
Definition at line 123 of file YApplication.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YApplication.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YApplication.cc

3.10 YApplicationPrivate Struct Reference

Collaboration diagram for YApplicationPrivate:



Public Attributes

- std::string **productName**
- bool **reverseLayout**
- std::string **applicationTitle**
- std::string **applicationIcon**
- YFunctionKeyMap **defaultFunctionKey**
- YIconLoader * **iconLoader**

3.10.1 Detailed Description

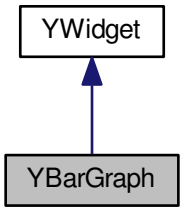
Definition at line 44 of file YApplication.cc.

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

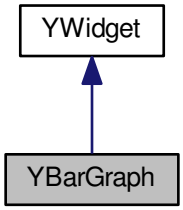
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YApplication.cc

3.11 YBarGraph Class Reference

Inheritance diagram for YBarGraph:



Collaboration diagram for YBarGraph:



Public Member Functions

- virtual ~YBarGraph ()
- virtual const char * widgetClass () const
- void addSegment (const YBarGraphSegment &segment)
- void deleteAllSegments ()
- int segments ()

- const YBarGraphSegment & segment (int segmentIndex) const
- void setValue (int segmentIndex, int newValue)
- void setLabel (int segmentIndex, const std::string &newLabel)
- void setSegmentColor (int segmentIndex, const YColor &color)
- void setTextColor (int segmentIndex, const YColor &color)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Protected Member Functions

- YBarGraph (YWidget *parent)
- virtual void doUpdate ()=0

Friends

- class **YBarGraphMultiUpdate**

3.11.1 Detailed Description

Definition at line 36 of file YBarGraph.h.

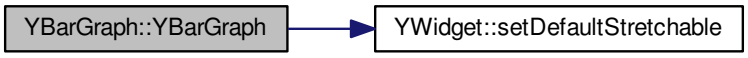
3.11.2 Constructor & Destructor Documentation

3.11.2.1 YBarGraph::YBarGraph (YWidget * *parent*) [protected]

Constructor.

Definition at line 67 of file YBarGraph.cc.

Here is the call graph for this function:



3.11.2.2 YBarGraph::~YBarGraph ( ) [virtual]

Destructor.

Definition at line 76 of file YBarGraph.cc.

3.11.3 Member Function Documentation

3.11.3.1 void YBarGraph::addSegment (const YBarGraphSegment & *segment*)

Add one segment.

If the segment's background and text colors are not explicitly specified, the YBarGraph widget will assign them from a list of (at least 5 different) color sets.

When adding multiple segments, use a YBarGraphMultiUpdate object for improved performance to hold back display updates until all segments are added.

Definition at line 96 of file YBarGraph.cc.

3.11.3.2 void YBarGraph::deleteAllSegments ( )

Delete all segments.

Definition at line 104 of file YBarGraph.cc.

3.11.3.3 virtual void YBarGraph::doUpdate ( ) [protected],[pure virtual]

Perform a display update after any change to any of the segments.

Derived classes are required to implement this.

3.11.3.4 YPropertyValue YBarGraph::getProperty (const std::string & *propertyName*) [virtual]

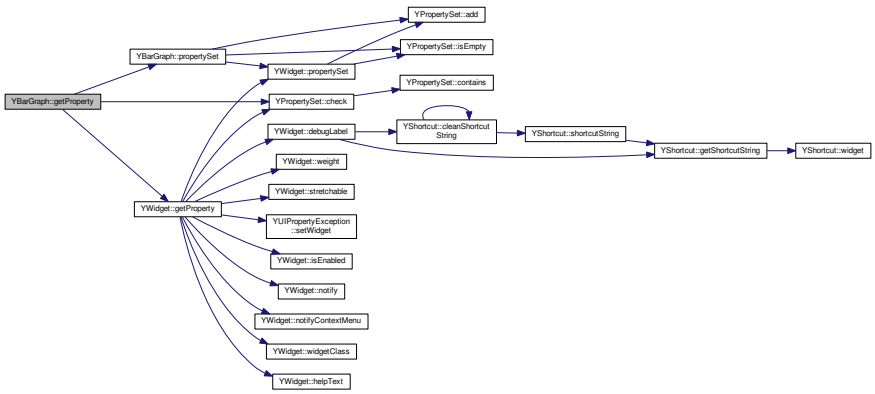
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 211 of file YBarGraph.cc.

Here is the call graph for this function:



3.11.3.5 const YPropertySet & YBarGraph::propertySet ( ) [virtual]

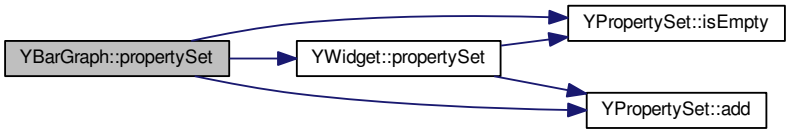
Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 174 of file YBarGraph.cc.

Here is the call graph for this function:



3.11.3.6 const YBarGraphSegment & YBarGraph::segment (int *segmentIndex*) const

Return the segment with the specified index (from 0 on).

This will throw an exception if there are not this many segments.

Definition at line 112 of file YBarGraph.cc.

3.11.3.7 int YBarGraph::segments ( )

Return the current number of segments.

Definition at line 121 of file YBarGraph.cc.

3.11.3.8 void YBarGraph::setLabel (int *segmentIndex*, const std::string & *newLabel*)

Set the label of the segment with the specified index (from 0 on). Use %1 as a placeholder for the current value.

This will throw an exception if there are not this many segments.

Note: Use a YBarGraphMultiUpdate object for improved performance when doing multiple changes at the same time.

Definition at line 138 of file YBarGraph.cc.

3.11.3.9 bool YBarGraph::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

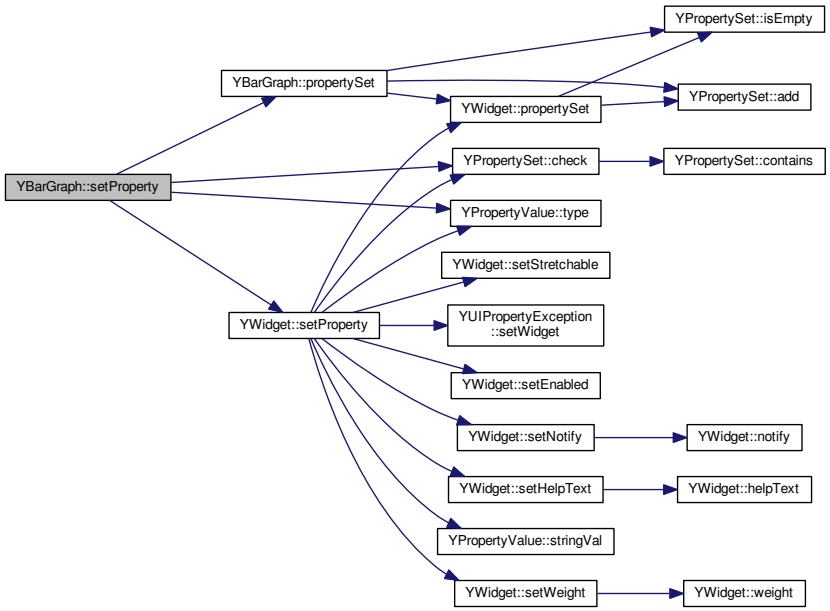
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 195 of file YBarGraph.cc.

Here is the call graph for this function:



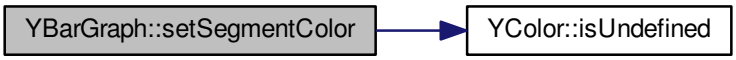
3.11.3.10 void YBarGraph::setSegmentColor (int *segmentIndex*, const YColor & *color*)

Set the background color of the segment with the specified index (from 0 on).

This will throw an exception if there are not this many segments or if the color is undefined.

Definition at line 148 of file YBarGraph.cc.

Here is the call graph for this function:



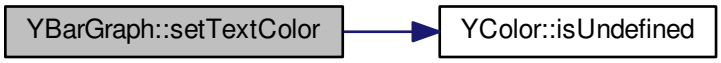
3.11.3.11 void YBarGraph::setTextColor (int *segmentIndex*, const YColor & *color*)

Set the text color of the segment with the specified index (from 0 on).

This will throw an exception if there are not this many segments or if the color is undefined.

Definition at line 161 of file YBarGraph.cc.

Here is the call graph for this function:



3.11.3.12 void YBarGraph::setValue (int *segmentIndex*, int *newValue*)

Set the value of the segment with the specifie index (from 0 on).

This will throw an exception if there are not this many segments.

Note: Use a YBarGraphMultiUpdate object for improved performance when doing multiple changes at the same time.

Definition at line 128 of file YBarGraph.cc.

3.11.3.13 virtual const char* YBarGraph::widgetClass ( ) const [inline], [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 56 of file YBarGraph.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBarGraph.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBarGraph.cc

3.12 YBarGraphMultiUpdate Class Reference

```
#include <YBarGraph.h>
```

Public Member Functions

- YBarGraphMultiUpdate (YBarGraph *barGraph)
- ~YBarGraphMultiUpdate ()

3.12.1 Detailed Description

Helper class for multiple updates to a YBarGraph widget: This will hold back display updates until this object goes out of scope.

Definition at line 280 of file YBarGraph.h.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 YBarGraphMultiUpdate::YBarGraphMultiUpdate (YBarGraph * *barGraph*)

Constructor.

This will make the corresponding YBarGraph widget hold back any pending display updates (due to changed values, labels, or colors) until this object is destroyed (goes out of scope).

Create objects of this class on the stack (as local variables) and simply let them go out of scope.

Example:

{ YBarGraphMultiUpdate multiUpdate(myBarGraph); myBarGraph->setValue(0, 42); // No display update yet myBarGraph->setValue(1, 84); // No display update yet myBarGraph->setValue(2, 21); // No display update yet

} // multiUpdate goes out of scope, will trigger display update now

Definition at line 226 of file YBarGraph.cc.

3.12.2.2 YBarGraphMultiUpdate::~YBarGraphMultiUpdate ( )

Destructor.

This will trigger display updates of the corresponding YBarGraph widget if any are necessary.

Definition at line 235 of file YBarGraph.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBarGraph.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBarGraph.cc

3.13 YBarGraphPrivate Struct Reference

Public Attributes

- std::vector< YBarGraphSegment > **segments**
- bool **updatesPending**
- bool **postponeUpdates**

3.13.1 Detailed Description

Definition at line 52 of file YBarGraph.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBarGraph.cc

3.14 YBarGraphSegment Class Reference

```
#include <YBarGraph.h>
```

Public Member Functions

- YBarGraphSegment (int value=0, const std::string &label=std::string(), const YColor &segmentColor=YColor(), const YColor &textColor=YColor())
- int value () const
- void setValue (int newValue)
- std::string label () const
- void setLabel (const std::string &newLabel)
- YColor segmentColor () const
- bool hasSegmentColor () const
- void setSegmentColor (const YColor &color)
- YColor textColor () const
- bool hasTextColor () const
- void setTextColor (const YColor &color)

3.14.1 Detailed Description

Helper class to describe one segment of a YBarGraph.

Definition at line 181 of file YBarGraph.h.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 YBarGraphSegment::YBarGraphSegment (int *value* = 0, const std::string & *label* = std::string(), const YColor & *segmentColor* = YColor(), const YColor & *textColor* = YColor()) [inline]

Constructor.

'value' is the initial value of this segment.

'label' is the label text in the segment. Use %1 as a placeholder for the current value.

'segmentColor' is the background color of this segment.

'textColor' is the color for the label text.

The YBarGraph widget will automatically assign some default colors (one of at least 5 different ones) if none are specified.

Definition at line 199 of file YBarGraph.h.

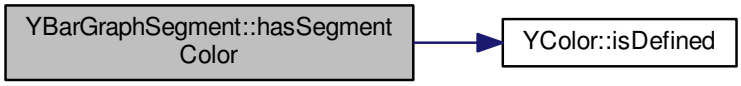
3.14.3 Member Function Documentation

3.14.3.1 bool YBarGraphSegment::hasSegmentColor ( ) const [inline]

Return 'true' if this segment's background color is defined, i.e. it has a real RGB value and was not just created with the default constructor.

Definition at line 241 of file YBarGraph.h.

Here is the call graph for this function:

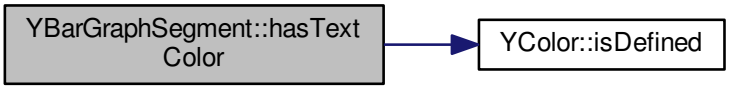


3.14.3.2 bool YBarGraphSegment::hasTextColor ( ) const [inline]

Return 'true' if this segment's text color is defined, i.e. it has a real RGB value and was not just created with the default constructor.

Definition at line 258 of file YBarGraph.h.

Here is the call graph for this function:



3.14.3.3 std::string YBarGraphSegment::label ( ) const [inline]

Return the current text label of this segment. Any %1 placeholder will be returned as %1 (not expanded).

Definition at line 223 of file YBarGraph.h.

3.14.3.4 YColor YBarGraphSegment::segmentColor ( ) const [inline]

Return the segment background color.

Definition at line 234 of file YBarGraph.h.

3.14.3.5 void YBarGraphSegment::setLabel (const std::string & *newLabel*) [inline]

Set the text label of this segment. Use %1 as a placeholder for the current value.

Definition at line 229 of file YBarGraph.h.

3.14.3.6 void YBarGraphSegment::setSegmentColor (const YColor & *color*) [inline]

Set this segment's background color.

Definition at line 246 of file YBarGraph.h.

3.14.3.7 void YBarGraphSegment::setTextColor (const YColor & *color*) [inline]

Set this segment's text color.

Definition at line 263 of file YBarGraph.h.

3.14.3.8 void YBarGraphSegment::setValue (int *newValue*) [inline]

Set the value of this segment.

Definition at line 217 of file YBarGraph.h.

3.14.3.9 YColor YBarGraphSegment::textColor ( ) const [inline]

Return this segment's text color.

Definition at line 251 of file YBarGraph.h.

3.14.3.10 int YBarGraphSegment::value ( ) const [inline]

Return the current value of this segment.

Definition at line 212 of file YBarGraph.h.

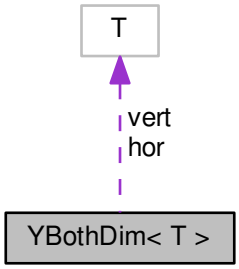
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBarGraph.h

3.15 YBothDim< T > Class Template Reference

```
#include <YBothDim.h>
```

Collaboration diagram for YBothDim< T >:



Public Member Functions

- YBothDim (T hor, T vert)
- YBothDim ()
- T & operator[] (YUIDimension dim)
- const T & operator[] (YUIDimension dim) const

Public Attributes

- T **vert**
- T **hor**

3.15.1 Detailed Description

template<typename T>class YBothDim< T >

Template class for two-dimensional entities, such as

- width, height
- x_pos, y_pos
- hStretchable, vStretchable

Precondition: type T needs to have a default constructor (which all simple types like int, long, bool have).

Definition at line 41 of file YBothDim.h.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 template<typename T> YBothDim< T >::YBothDim (T *hor*, T *vert*) [inline]

Constructor with explicit initialization for both values

Definition at line 52 of file YBothDim.h.

3.15.2.2 template<typename T> YBothDim< T >::YBothDim ( ) [inline]

Default constructor (calls T default constructor for both values)

Definition at line 60 of file YBothDim.h.

3.15.3 Member Function Documentation

3.15.3.1 template<typename T> T& YBothDim< T >::operator[] (YUIDimension *dim*) [inline]

operator[] for alternative access via myVar[YD_HORIZ] Please note that this returns a non-const reference, so this can be used as an lvalue (e.g., in assignments)

Definition at line 68 of file YBothDim.h.

3.15.3.2 template<typename T> const T& YBothDim< T >::operator[] (YUIDimension *dim*) const [inline]

Same as above for const objects

Definition at line 84 of file YBothDim.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBothDim.h

3.16 YBuiltinCaller Class Reference

```
#include <YBuiltinCaller.h>
```

Public Member Functions

- virtual void call ()=0

3.16.1 Detailed Description

Abstract base class for transparently calling a built-in function. Derived classes will want to add some methods to store the function to be called, arguments to that function and its result and to retrieve the result when needed.

See YCPBuiltinCaller.h for an implementation.

Definition at line 37 of file YBuiltinCaller.h.

3.16.2 Member Function Documentation

3.16.2.1 virtual void YBuiltinCaller::call ( ) [pure virtual]

Call the built-in. This will be called in the UI thread with appropriate syncing between the threads.

Derived classes might want to store the result of the call in a member variable in this class so it can later be queried.

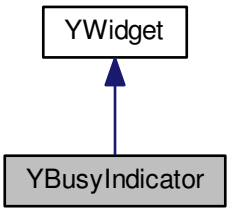
Derived classes are required to implement this method.

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

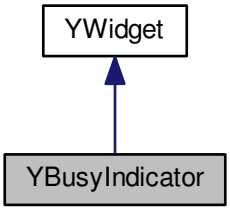
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBuiltinCaller.h

3.17 YBusyIndicator Class Reference

Inheritance diagram for YBusyIndicator:



Collaboration diagram for YBusyIndicator:



Public Member Functions

- virtual ~YBusyIndicator ()
- virtual const char * widgetClass () const
- std::string label ()
- virtual void setLabel (const std::string &label)
- int timeout () const
- virtual void setTimeout (int newTimeout)
- bool alive () const

- virtual void setAlive (bool newAlive)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Protected Member Functions

- YBusyIndicator (YWidget *parent, const std::string &label, int timeout=1000, bool alive=true)

3.17.1 Detailed Description

Definition at line 33 of file YBusyIndicator.h.

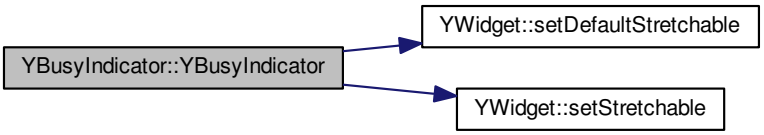
3.17.2 Constructor & Destructor Documentation

3.17.2.1 YBusyIndicator::YBusyIndicator (YWidget * *parent*, const std::string & *label*, int *timeout* = 1000, bool *alive* = true) [protected]

Constructor.

Definition at line 52 of file YBusyIndicator.cc.

Here is the call graph for this function:



3.17.2.2 YBusyIndicator::~YBusyIndicator ( ) [virtual]

Destructor.

Definition at line 66 of file YBusyIndicator.cc.

3.17.3 Member Function Documentation

3.17.3.1 bool YBusyIndicator::alive ( ) const

Return whether busy indicator is alive or in stalled stated.

Definition at line 104 of file YBusyIndicator.cc.

3.17.3.2 YPropertyValue YBusyIndicator::getProperty (const std::string & *propertyName*) [virtual]

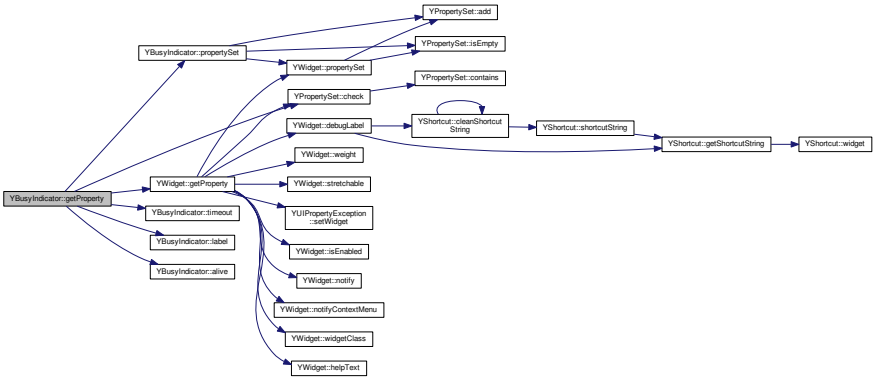
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 149 of file YBusyIndicator.cc.

Here is the call graph for this function:



3.17.3.3 std::string YBusyIndicator::label ( )

Get the label (the caption above the progress bar).

Definition at line 72 of file YBusyIndicator.cc.

3.17.3.4 const YPropertySet & YBusyIndicator::propertySet ( ) [virtual]

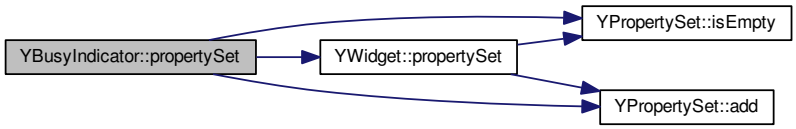
Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 110 of file YBusyIndicator.cc.

Here is the call graph for this function:



3.17.3.5 void YBusyIndicator::setAlive (bool *newAlive*) [virtual]

Send a keep alive message to prevent BusyIndicator from changing to 'stalled' state.

Derived classes should reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 99 of file YBusyIndicator.cc.

Here is the call graph for this function:

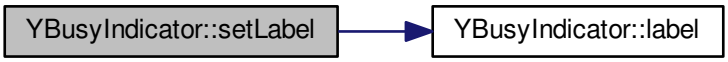
3.17.3.6 void YBusyIndicator::setLabel (const std::string & *label*) [virtual]

Set the label (the caption above the progress bar).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 78 of file YBusyIndicator.cc.

Here is the call graph for this function:



3.17.3.7 bool YBusyIndicator::setProperty (const std::string & *propertyName,* const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

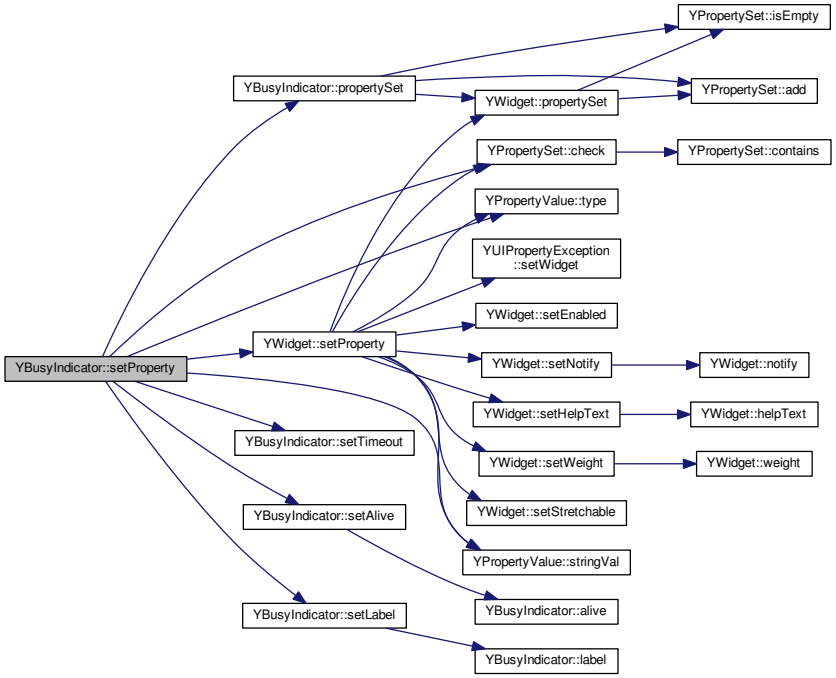
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 132 of file YBusyIndicator.cc.

Here is the call graph for this function:



3.17.3.8 void YBusyIndicator::setTimeout (int *newTimeout*) [virtual]

Set the timeout in milliseconds after that the widget shows 'stalled' when no new tick is received.

Derived classes should reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 90 of file YBusyIndicator.cc.

3.17.3.9 int YBusyIndicator::timeout ( ) const

Return the current timeout in milliseconds.

Definition at line 84 of file YBusyIndicator.cc.

3.17.3.10 virtual const char* YBusyIndicator::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 54 of file YBusyIndicator.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBusyIndicator.h

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBusyIndicator.cc

3.18 YBusyIndicatorPrivate Struct Reference

Public Member Functions

- **YBusyIndicatorPrivate** (const std::string &label, int timeout, bool alive)

Public Attributes

- std::string **label**
- int **timeout**
- bool **alive**

3.18.1 Detailed Description

Definition at line 33 of file YBusyIndicator.cc.

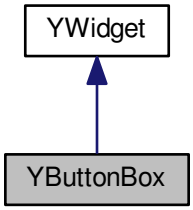
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YBusyIndicator.cc

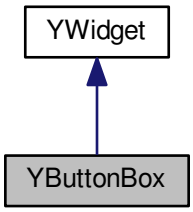
3.19 YButtonBox Class Reference

```
#include <YButtonBox.h>
```

Inheritance diagram for YButtonBox:



Collaboration diagram for YButtonBox:



Public Member Functions

- virtual ~YButtonBox ()
- virtual const char * widgetClass () const
- virtual void setMargins (const YButtonBoxMargins &margins)
- YButtonBoxMargins margins () const
- virtual void doLayout (int width, int height)
- YPushButton * findButton (YButtonRole role)
- void sanityCheck ()
- void setSanityCheckRelaxed (bool relax=true)
- bool sanityCheckRelaxed () const
- virtual int preferredWidth ()
- virtual int preferredHeight ()
- virtual void setSize (int newWidth, int newHeight)
- virtual bool stretchable (YUIDimension dimension) const

Static Public Member Functions

- static void setLayoutPolicy (const YButtonBoxLayoutPolicy &layoutPolicy)
- static YButtonBoxLayoutPolicy layoutPolicy ()
- static YButtonBoxLayoutPolicy kdeLayoutPolicy ()
- static YButtonBoxLayoutPolicy gnomeLayoutPolicy ()
- static void setDefaultMargins (const YButtonBoxMargins &margins)
- static YButtonBoxMargins defaultMargins ()

Protected Member Functions

- YButtonBox (YWidget *parent)
- virtual std::vector
  < YPushButton * > buttonsByButtonOrder ()
- int maxChildSize (YUIDimension dim) const
- int totalChildrenWidth () const
- virtual void moveChild (YWidget *child, int newX, int newY)=0
- int preferredWidth (bool equalSizeButtons)

Friends

- class **YButtonBoxPrivate**

3.19.1 Detailed Description

Container widget for dialog buttons that abstracts the button order depending on the desktop environment.

KDE and Windows arrange dialog buttons like this:

```
[OK] [Apply] [Cancel] [Custom1] [Custom2] ... [Help]

[Continue] [Cancel]

[Yes] [No]
```

GNOME and MacOS arrange them like this:

```
[Help] [Custom1] [Custom2] ... [Apply] [Cancel] [OK]

[Cancel] [Continue]

[No] [Yes]
```

This class provides the abstraction to use whatever layout is more appropriate in the current environment. The application creates the buttons as child widgets of a YButtonBox (rather than a YHBox) and leaves the button order to the YButtonBox.

Each of the standard buttons ([OK], [Apply], [Cancel], [Help]) needs to have a button role properly assigned.

If set up properly (see YApplication::setDefaultFunctionKey()), known button labels will be assigned an appropriate role:

```
[OK]                        F10
[Continue] -> [OK]  F10
[Yes]      -> [OK]  F10
[Accept]   -> [OK]  F10
[Next]     -> [OK]  F10

[Cancel]                    F9
[No]       -> [Cancel]      F9

[Help]                      F1
```

Buttons with nonstandard labels that act in such a role need to be explicitly assigned that role:

```
[Print ] [Cancel] [Help]
[Delete] [Cancel] [Help]
```

Those [Print] or [Delete] buttons act as [OK] buttons (the "yes, do it" action of that dialog). Call YPushButton::setButtonRole(YOkButton) explicitly for them.

YButtonBox widgets only accept YPushButton child widgets. Otherwise an exception is thrown.

If there is more than one button, one of the child buttons needs to have the [OK] role, and another needs to have the [Cancel] role. Otherwise an exception is thrown.

Definition at line 148 of file YButtonBox.h.

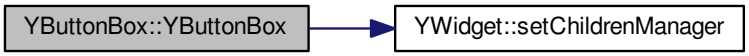
3.19.2 Constructor & Destructor Documentation

3.19.2.1 YButtonBox::YButtonBox (YWidget * *parent*) [protected]

Constructor.

Definition at line 66 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.2.2 YButtonBox::~YButtonBox ( ) [virtual]

Destructor.

Definition at line 75 of file YButtonBox.cc.

3.19.3 Member Function Documentation

3.19.3.1 std::vector< YPushButton * > YButtonBox::buttonsByButtonOrder ( ) [protected], [virtual]

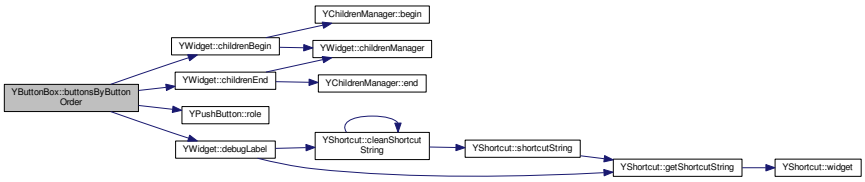
Return the button children sorted (left to right) by the current button order (from the layout policy).

This default implementation handles KDE and Gnome button orders. It is used in the default doLayout() method.

This may throw exceptions if there are non-button children or if there are multiple buttons with any of the standard button roles (except YCustomButton, of course).

Definition at line 410 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.3.2 YButtonBoxMargins YButtonBox::defaultMargins ( ) [static]

Return the default margins for all future YButtonBox widgets.

Definition at line 132 of file YButtonBox.cc.

3.19.3.3 void YButtonBox::doLayout (int *width*, int *height*) [virtual]

Lay out the button box and its children (its buttons). This is where the button order is implemented.

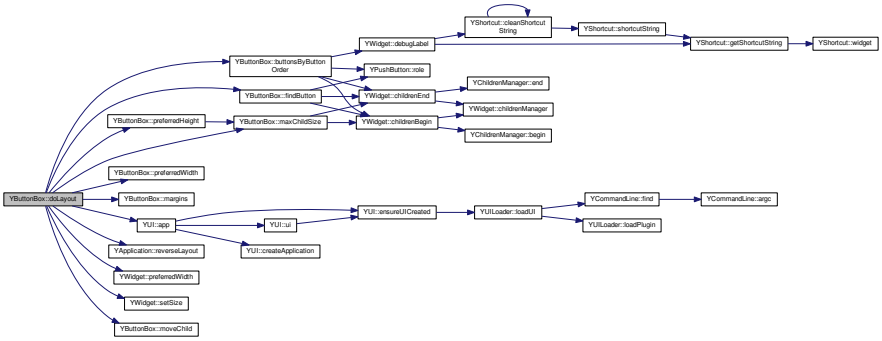
This method is called by the default setSize() method. It uses YButtonBox::layoutPolicy() and the specified margins (defaultMargins unless changed with setMargins()). It moves the buttons to their position with moveChild().

This all should work reasonably in all cases (Qt-UI with KDE button order, Gtk-UI with Gnome button order, NCurses-UI with KDE button order).

Still, derived classes can reimplement this. It does not make very much sense to call this default method in a new implementation.

Definition at line 161 of file YButtonBox.cc.

Here is the call graph for this function:

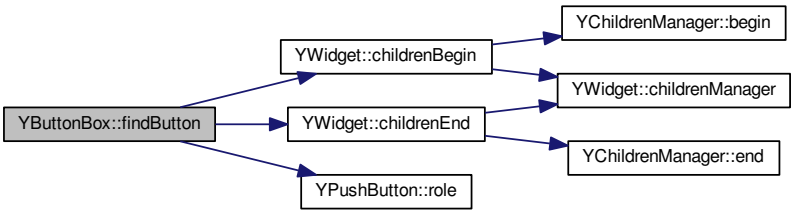


3.19.3.4 YPushButton * YButtonBox::findButton (YButtonRole *role*)

Search the child widgets for the first button with the specified role. Return the button or 0 if there is no button with that role.

Definition at line 574 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.3.5 YButtonBoxLayoutPolicy YButtonBox::gnomeLayoutPolicy ( ) [static]

Predefined layout policy for GNOME-like behaviour.

Definition at line 110 of file YButtonBox.cc.

3.19.3.6 YButtonBoxLayoutPolicy YButtonBox::kdeLayoutPolicy ( ) [static]

Predefined layout policy for KDE-like behaviour.

Definition at line 96 of file YButtonBox.cc.

3.19.3.7 YButtonBoxLayoutPolicy YButtonBox::layoutPolicy ( ) [static]

Return the layout policy.

Definition at line 89 of file YButtonBox.cc.

3.19.3.8 YButtonBoxMargins YButtonBox::margins ( ) const

Return the margins of this YButtonBox.

Notice that those are only the desired margins; if there is not enough space, margins and spacings will be reduced before buttons are cut off.

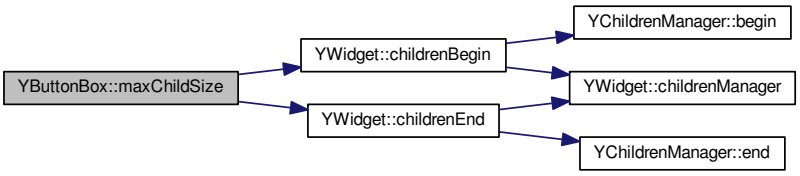
Definition at line 146 of file YButtonBox.cc.

3.19.3.9 int YButtonBox::maxChildSize (YUIDimension *dim*) const [protected]

Return the (preferred) size of the biggest child widget in the specified dimension.

Definition at line 527 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.3.10 virtual void YButtonBox::moveChild (YWidget * *child*, int *newX*, int *newY*) [protected], [pure virtual]

Move a child to a new position. This is used in doLayout().

Derived classes are required to implement this.

3.19.3.11 int YButtonBox::preferredHeight ( ) [virtual]

Preferred height of the widget.

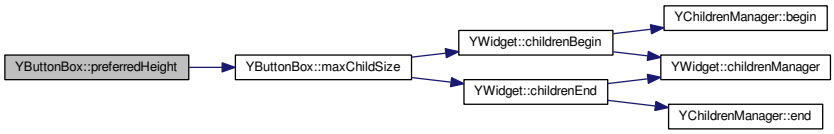
Reimplemented from YWidget. This default method returns the height of the highest child plus the top and bottom margins.

Derived classes can reimplement this method. It does not make very much sense to call this base class method in a new implementation.

Implements YWidget.

Definition at line 516 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.3.12 int YButtonBox::preferredWidth ( ) [virtual]

Preferred width of the widget.

Reimplemented from YWidget. This default method returns the sum of the the widths of all child widgets plus the left and right margins plus the spacings.

Derived classes can reimplement this method. It does not make very much sense to call this base class method in a new implementation.

Implements YWidget.

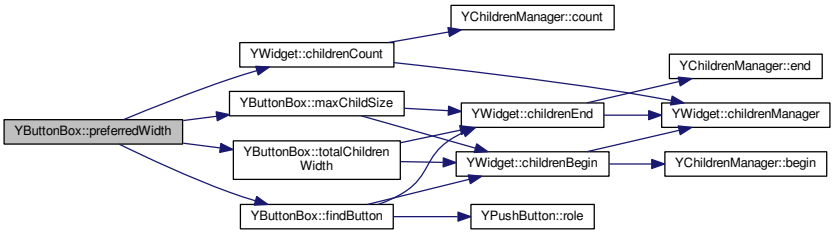
Definition at line 509 of file YButtonBox.cc.

3.19.3.13 int YButtonBox::preferredWidth (bool *equalSizeButtons*) [protected]

Calculate the preferred with with or without trying to enforce buttons of equal size.

Definition at line 483 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.3.14 void YButtonBox::sanityCheck ( )

Sanity check: Check if all child widgets have the correct widget class and if the button roles are assigned correctly.

A YButtonBox with more than one button is required to have one YOKButton and ony YCancelButton. Neither button role may be assigned more than once.

This method may throw exceptions:

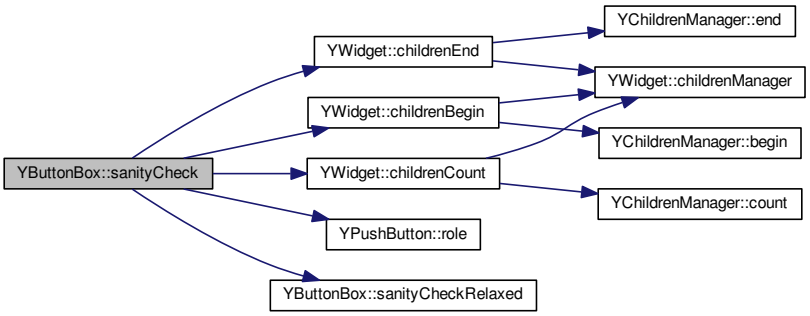
- YUIButtonRoleMismatchException
- YUIInvalidChildException (wrong widget class)

This cannot be done as child widgets are inserted since this is done from the child widgets' constructors, so virtual methods or dynamic_cast don't work at that point.

This is called in the default setSize() method (just before doLayout()), so any of the above errors are caught only at that time. Applications are free to call this before that time to make error handling more transparent.

Definition at line 605 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.3.15 bool YButtonBox::sanityCheckRelaxed ( ) const

Return 'true' if sanity checks are currently relaxed, 'false' if not.

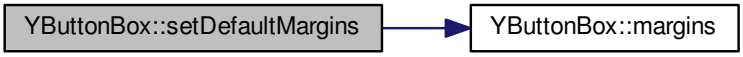
Definition at line 598 of file YButtonBox.cc.

3.19.3.16 void YButtonBox::setDefaultMargins (const YButtonBoxMargins & *margins*) [static]

Set the default margins for all future YButtonBox widgets.

Definition at line 125 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.3.17 void YButtonBox::setLayoutPolicy (const YButtonBoxLayoutPolicy & *layoutPolicy*) [static]

Set the button policy for all future YButtonBox widgets: Button order, alignment if there is any excess space, whether or not to give all buttons the same size.

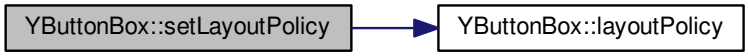
You might want to use one of the predefined static methods: YButtonBox::kdeLayoutPolicy(), YButtonBox::gnomeLayoutPolicy().

The default doLayout() method uses those values.

Notice that there is intentionally no way to set this differently for each individual YButtonBox: This would defeat the purpose of consistency (with the desktop environment this application is running in), which is the reason for having this widget class.

Definition at line 82 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.3.18 void YButtonBox::setMargins (const YButtonBoxMargins & *margins*) [virtual]

Set the margins for this YButtonBox.

Derived classes are free to reimplement this, but they should call this base class method in the new method.

Definition at line 139 of file YButtonBox.cc.

Here is the call graph for this function:

3.19.3.19 void YButtonBox::setSanityCheckRelaxed (bool *relax* = `true`)

Relax the sanity check done in sanityCheck(): Do not enforce that there has to be a YOKButton and a YCancelButton if there is more than one button.

In very rare cases, it might be neccessary to have a less stringent sanity check: There are some very few legitimate cases for having a YButtonBox with multiple buttons, yet without a YCancelButton.

Examples:

```
...message...
 <Countdown>
 [OK] [Stop]

...message...
[OK] [Details]
```

In those cases, it makes sense to relax the sanity check.

Definition at line 591 of file YButtonBox.cc.

3.19.3.20 void YButtonBox::setSize (int *newWidth*, int *newHeight*) `[virtual]`

Sets the size of the YButtonBox.

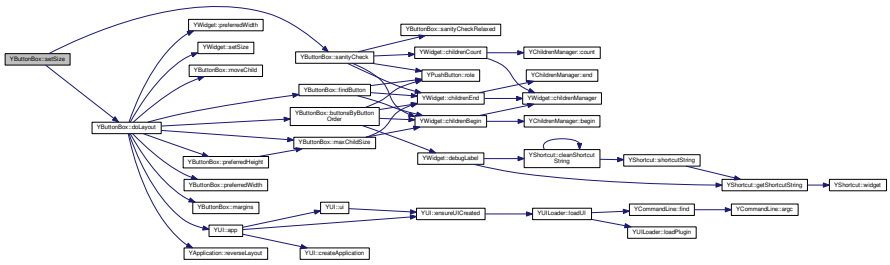
Derived classes can reimplement this, but this base class method should be called in the reimplemented function.

Reimplemented from YWidget.

Implements YWidget.

Definition at line 153 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.3.21 bool YButtonBox::stretchable (YUIDimension *dimension*) const [virtual]

Returns the stretchability of the YButtonBox. YButtonBox widgets are horizontally stretchable by default. How any excess space is used is specified in the layout policy.

Reimplemented from YWidget.

Reimplemented from YWidget.

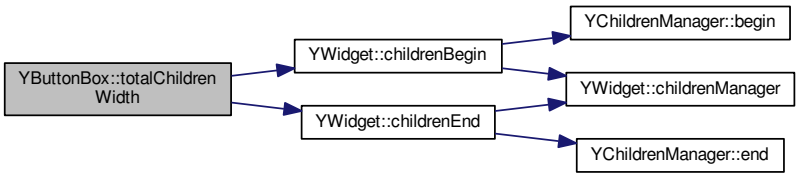
Definition at line 559 of file YButtonBox.cc.

3.19.3.22 int YButtonBox::totalChildrenWidth ( ) const [protected]

Return the sum of the (preferred) widths of all child widgets.

Definition at line 543 of file YButtonBox.cc.

Here is the call graph for this function:



3.19.3.23 virtual const char* YButtonBox::widgetClass ( ) const [inline], [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 169 of file YButtonBox.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YButtonBox.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YButtonBox.cc

3.20 YButtonBoxLayoutPolicy Struct Reference

```
#include <YButtonBox.h>
```

Public Attributes

- YButtonOrder **buttonOrder**
- bool **equalSizeButtons**
- bool **addExcessSpaceToHelpButtonExtraMargin**
- YAlignmentType **alignment** [YUIAllDimensions]

3.20.1 Detailed Description

Helper class: Layout policy for YButtonBox widgets. This is used in the default YButtonBox::doLayout() method.

Definition at line 41 of file YButtonBox.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YButtonBox.h

3.21 YButtonBoxMargins Struct Reference

```
#include <YButtonBox.h>
```

Public Attributes

- int **left**
- int **right**
- int **top**
- int **bottom**
- int **spacing**
- int **helpButtonExtraSpacing**

3.21.1 Detailed Description

Helper class: Margins for YButtonBox widgets. All sizes are in UI-dependent units, i.e. in pixels.

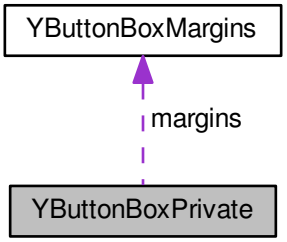
Definition at line 65 of file YButtonBox.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YButtonBox.h

3.22 YButtonBoxPrivate Struct Reference

Collaboration diagram for YButtonBoxPrivate:



Public Member Functions

- YButtonBoxPrivate ()

Public Attributes

- bool **sanityCheckRelaxed**
- YButtonBoxMargins **margins**

3.22.1 Detailed Description

Definition at line 45 of file YButtonBox.cc.

3.22.2 Constructor & Destructor Documentation

3.22.2.1 YButtonBoxPrivate::YButtonBoxPrivate ( ) [inline]

Constructor

Definition at line 50 of file YButtonBox.cc.

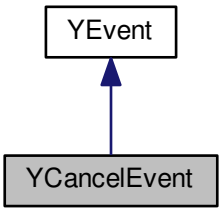
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YButtonBox.cc

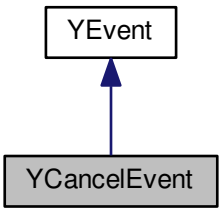
3.23 YCancelEvent Class Reference

```
#include <YEvent.h>
```

Inheritance diagram for YCancelEvent:



Collaboration diagram for YCancelEvent:



Protected Member Functions

- virtual ~YCancelEvent ()

Additional Inherited Members

3.23.1 Detailed Description

Event to be returned upon closing a dialog with the window manager close button (or Alt-F4)

Definition at line 305 of file YEvent.h.

3.23.2 Constructor & Destructor Documentation

3.23.2.1 virtual YCancelEvent::~YCancelEvent ( ) [inline],[protected],[virtual]

Protected destructor - events can only be deleted via YDialog::deleteEvent(). The associated dialog will take care of this event and delete it when appropriate.

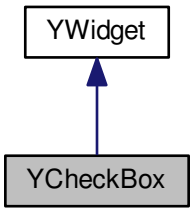
Definition at line 318 of file YEvent.h.

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

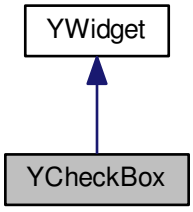
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEvent.h

3.24 YCheckBox Class Reference

Inheritance diagram for YCheckBox:



Collaboration diagram for YCheckBox:



Public Member Functions

- virtual ~YCheckBox ()
- virtual const char * widgetClass () const
- virtual YCheckBoxState value ()=0
- virtual void setValue (YCheckBoxState state)=0
- bool isChecked ()
- void setChecked (bool checked=true)
- bool dontCare ()

- void setDontCare ()
- std::string label () const
- virtual void setLabel (const std::string &label)
- bool useBoldFont () const
- virtual void setUseBoldFont (bool bold=true)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)
- const char * userInputProperty ()

Protected Member Functions

- YCheckBox (YWidget *parent, const std::string &label)

3.24.1 Detailed Description

Definition at line 43 of file YCheckBox.h.

3.24.2 Constructor & Destructor Documentation

3.24.2.1 YCheckBox::YCheckBox (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

Definition at line 45 of file YCheckBox.cc.

3.24.2.2 YCheckBox::~YCheckBox ( ) [virtual]

Destructor.

Definition at line 53 of file YCheckBox.cc.

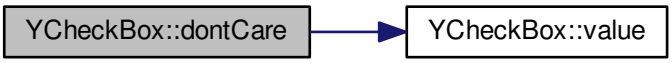
3.24.3 Member Function Documentation

3.24.3.1 bool YCheckBox::dontCare ( ) [inline]

Simplified access to tri-state ("don't care").

Definition at line 109 of file YCheckBox.h.

Here is the call graph for this function:



3.24.3.2 YPropertyValue YCheckBox::getProperty (const std::string & *propertyName*) [virtual]

Get a property. Reimplemented from YWidget.

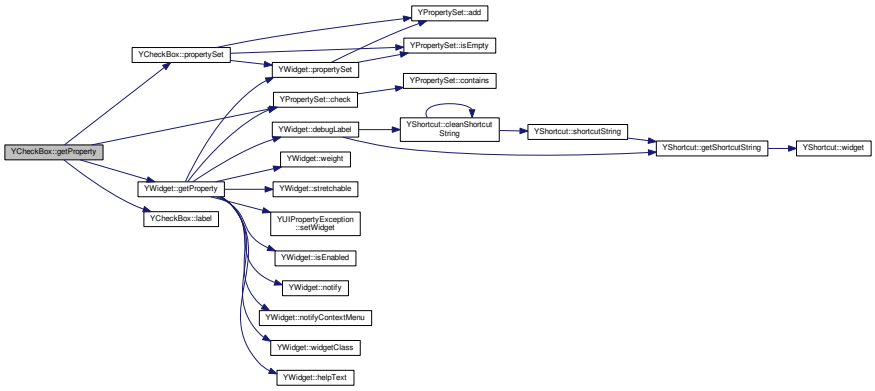
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from YWidget.

Definition at line 121 of file YCheckBox.cc.

Here is the call graph for this function:

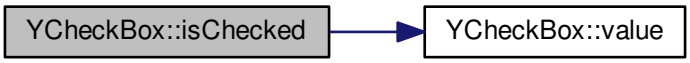


3.24.3.3 bool YCheckBox::isChecked ( ) [inline]

Simplified access to value(): Return 'true' if the CheckBox is checked.

Definition at line 98 of file YCheckBox.h.

Here is the call graph for this function:



3.24.3.4 std::string YCheckBox::label ( ) const

Get the label (the text on the CheckBox).

Definition at line 65 of file YCheckBox.cc.

3.24.3.5 const YPropertySet & YCheckBox::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property set upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 84 of file YCheckBox.cc.

Here is the call graph for this function:

3.24.3.6 void YCheckBox::setChecked (bool *checked* = true) [inline]

Simplified access to setValue(): Check of uncheck the CheckBox.

Definition at line 103 of file YCheckBox.h.

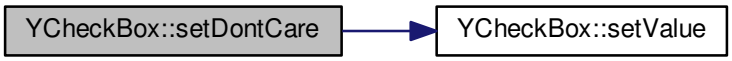
Here is the call graph for this function:

3.24.3.7 void YCheckBox::setDontCare ( ) [inline]

Simplified access to setting tri-state ("don't care").

Definition at line 114 of file YCheckBox.h.

Here is the call graph for this function:



3.24.3.8 void YCheckBox::setLabel (const std::string & *label*) [virtual]

Set the label (the text on the CheckBox).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 59 of file YCheckBox.cc.

3.24.3.9 bool YCheckBox::setProperty (const std::string & *propertyName,* const **YPropertyValue** & *val*) [virtual]

Set a property. Reimplemented from YWidget.

This method may throw exceptions, for example

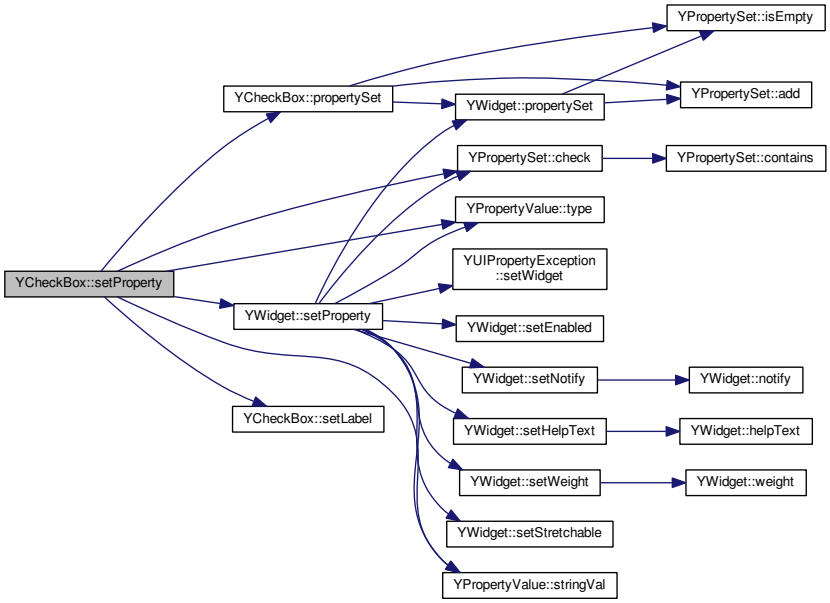
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 105 of file YCheckBox.cc.

Here is the call graph for this function:



3.24.3.10 virtual void YCheckBox::setShortcutString (const std::string & *str*) [inline], [virtual]

Set the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 187 of file YCheckBox.h.

Here is the call graph for this function:

3.24.3.11 void YCheckBox::setUseBoldFont (bool *bold* = true) [virtual]

Indicate whether or not a bold font should be used.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 77 of file YCheckBox.cc.

3.24.3.12 virtual void YCheckBox::setValue (YCheckBoxState *state*) [pure virtual]

Set the CheckBox value (on/off/don't care).

Derived classes are required to implement this.

3.24.3.13 virtual std::string YCheckBox::shortcutString ( ) const [inline], [virtual]

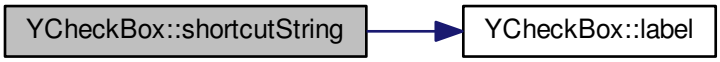
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 180 of file YCheckBox.h.

Here is the call graph for this function:



3.24.3.14 bool YCheckBox::useBoldFont ( ) const

Returns 'true' if a bold font should be used.

Definition at line 71 of file YCheckBox.cc.

3.24.3.15 const char* YCheckBox::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 194 of file YCheckBox.h.

3.24.3.16 virtual YCheckBoxState YCheckBox::value ( ) [pure virtual]

Get the current value:

YCheckBox_on CheckBox is checked YCheckBox_off CheckBox is unchecked

YCheckBox_dont_care tri-state: CheckBox is greyed out, neither checked nor unchecked

The user cannot set YCheckBox_dont_care directly. This status is always only set from the outside, usually because a setting cannot be clearly determined. For example, a checkbox

```
[ ] Read only
```

would be set to "don't care" (by the application, not directly by the user) when it is to display the read-only state of a group of files where some are read-only and some are writeable.

Derived classes are required to implement this function. (Intentionally not const)

3.24.3.17 virtual const char* YCheckBox::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

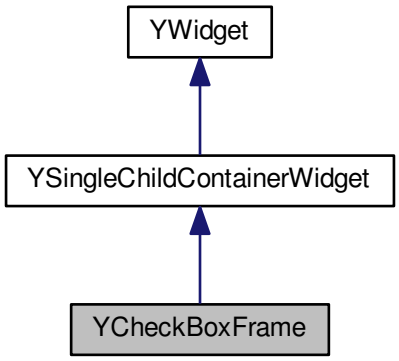
Definition at line 61 of file YCheckBox.h.

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

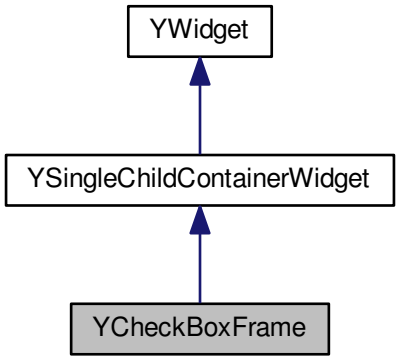
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YCheckBox.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YCheckBox.cc

3.25 YCheckBoxFrame Class Reference

Inheritance diagram for YCheckBoxFrame:



Collaboration diagram for YCheckBoxFrame:



Public Member Functions

- YCheckBoxFrame (YWidget *parent, const std::string &label, bool checked)
- virtual ~YCheckBoxFrame ()
- virtual const char * widgetClass () const
- std::string label () const
- virtual void setLabel (const std::string &label)
- virtual void setValue (bool isChecked)=0
- virtual bool value ()=0
- bool autoEnable () const
- virtual void setAutoEnable (bool autoEnable)
- bool invertAutoEnable () const
- virtual void setInvertAutoEnable (bool invertAutoEnable)
- void handleChildrenEnablement (bool isChecked)
- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)
- const char * userInputProperty ()
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Additional Inherited Members

3.25.1 Detailed Description

Definition at line 35 of file YCheckBoxFrame.h.

3.25.2 Constructor & Destructor Documentation

3.25.2.1 YCheckBoxFrame::YCheckBoxFrame (YWidget * *parent,* const std::string & *label,* bool *checked*)

Constructor.

Definition at line 49 of file YCheckBoxFrame.cc.

3.25.2.2 YCheckBoxFrame::~YCheckBoxFrame ( ) [virtual]

Destructor.

Definition at line 59 of file YCheckBoxFrame.cc.

3.25.3 Member Function Documentation

3.25.3.1 bool YCheckBoxFrame::autoEnable ( ) const

Handle children enabling/disabling automatically based on the CheckBoxFrame's check box?

Definition at line 75 of file YCheckBoxFrame.cc.

3.25.3.2 YPropertyValue YCheckBoxFrame::getProperty (const std::string & *propertyName*) [virtual]

Get a property. Reimplemented from YWidget.

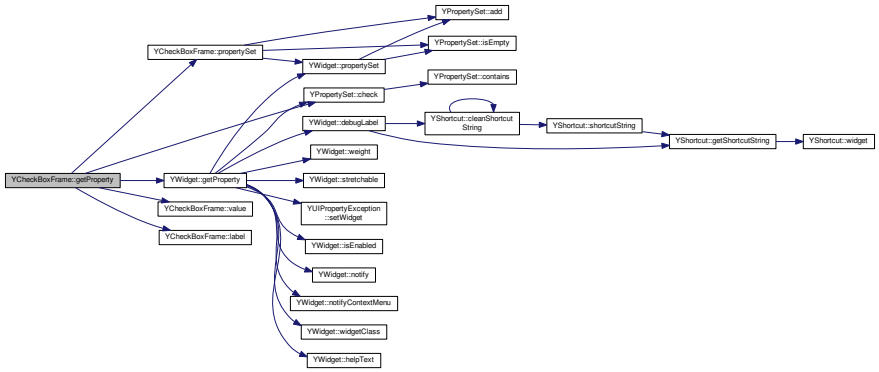
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from YWidget.

Definition at line 149 of file YCheckBoxFrame.cc.

Here is the call graph for this function:



3.25.3.3 void YCheckBoxFrame::handleChildrenEnablement (bool *isChecked*)

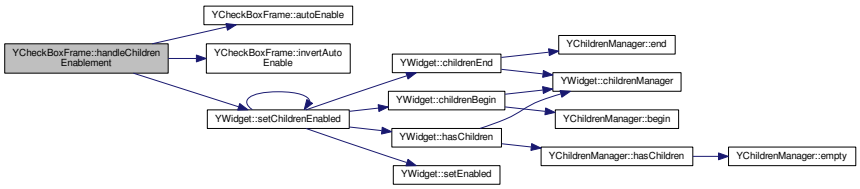
Handle enabling/disabling of child widgets based on 'isChecked' (the current status of the check box) and autoEnable() and invertAutoEnable().

Derived classes should call this when the check box status changes rather than try to handle it on their level.

This method also needs to be called after new child widgets are added to establish the initial enabled or disabled state of the child widgets.

Definition at line 98 of file YCheckBoxFrame.cc.

Here is the call graph for this function:



3.25.3.4 bool YCheckBoxFrame::invertAutoEnable ( ) const

Invert the meaning of the CheckBoxFrame's check box, i.e., disable child widgets when checked?

Definition at line 86 of file YCheckBoxFrame.cc.

3.25.3.5 std::string YCheckBoxFrame::label ( ) const

Return the label text on the CheckBoxFrame.

Definition at line 65 of file YCheckBoxFrame.cc.

3.25.3.6 const YPropertySet & YCheckBoxFrame::propertySet ( ) [virtual]

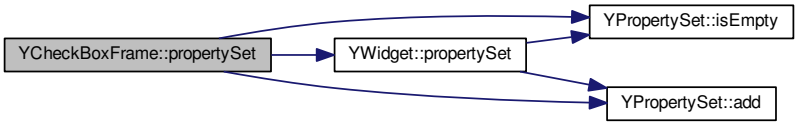
Return this class's property set. This also initializes the property set upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 112 of file YCheckBoxFrame.cc.

Here is the call graph for this function:



3.25.3.7 void YCheckBoxFrame::setAutoEnable (bool *autoEnable*) [virtual]

Change autoEnabled flag.

Derived classes are free to overload this, but they should call this base class function in the overloaded function.

Definition at line 80 of file YCheckBoxFrame.cc.

Here is the call graph for this function:

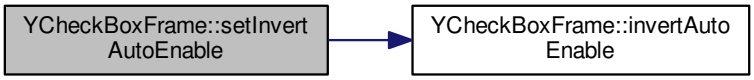
3.25.3.8 void YCheckBoxFrame::setInvertAutoEnable (bool *invertAutoEnable*) [virtual]

Change invertAutonEnable flag.

Derived classes are free to overload this, but they should call this base class function in the overloaded function.

Definition at line 91 of file YCheckBoxFrame.cc.

Here is the call graph for this function:



3.25.3.9 void YCheckBoxFrame::setLabel (const std::string & *label*) [virtual]

Change the label text on the CheckBoxFrame.

Derived classes should overload this, but call this base class function in the overloaded function.

Definition at line 70 of file YCheckBoxFrame.cc.

Here is the call graph for this function:

3.25.3.10 bool YCheckBoxFrame::setProperty (const std::string & *propertyName,* const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

This method may throw exceptions, for example

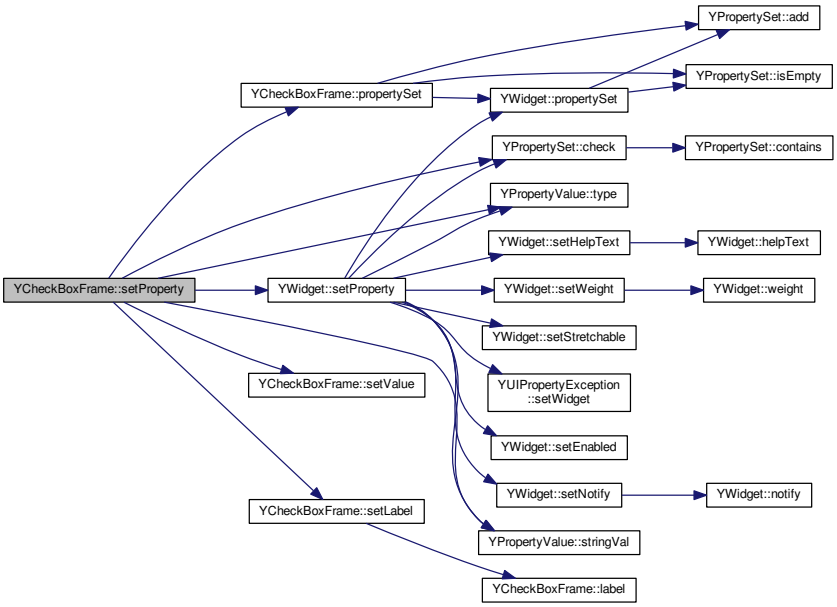
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 133 of file YCheckBoxFrame.cc.

Here is the call graph for this function:



3.25.3.11 virtual void YCheckBoxFrame::setShortcutString (const std::string & *str*) [inline],[virtual]

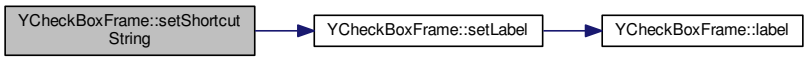
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 136 of file YCheckBoxFrame.h.

Here is the call graph for this function:



3.25.3.12 virtual void YCheckBoxFrame::setValue (bool *isChecked*) [pure virtual]

Check or uncheck the CheckBoxFrame's check box.

Derived classes are required to implement this.

3.25.3.13 virtual std::string YCheckBoxFrame::shortcutString ( ) const [inline], [virtual]

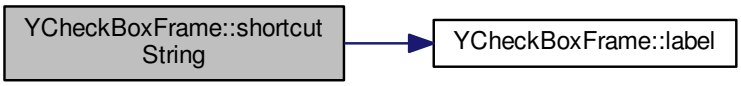
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 129 of file YCheckBoxFrame.h.

Here is the call graph for this function:



3.25.3.14 const char* YCheckBoxFrame::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 143 of file YCheckBoxFrame.h.

3.25.3.15 virtual bool YCheckBoxFrame::value ( ) [pure virtual]

Get the status of the CheckBoxFrame's check box.

Derived classes are required to implement this.

3.25.3.16 virtual const char* YCheckBoxFrame::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 54 of file YCheckBoxFrame.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YCheckBoxFrame.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YCheckBoxFrame.cc

3.26 YCheckBoxFramePrivate Struct Reference

Public Member Functions

- **YCheckBoxFramePrivate** (const std::string &label)

Public Attributes

- std::string **label**
- bool **autoEnable**
- bool **invertAutoEnable**

3.26.1 Detailed Description

Definition at line 33 of file YCheckBoxFrame.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YCheckBoxFrame.cc

3.27 YCheckBoxPrivate Struct Reference

Public Member Functions

- **YCheckBoxPrivate** (const std::string &label)

Public Attributes

- std::string **label**
- bool **useBoldFont**

3.27.1 Detailed Description

Definition at line 33 of file YCheckBox.cc.

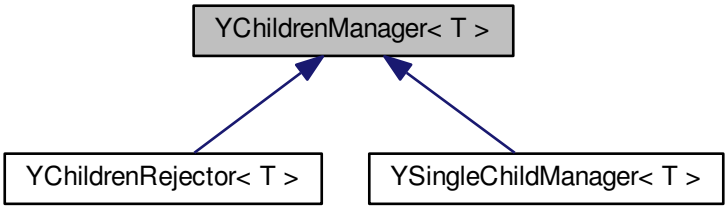
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YCheckBox.cc

3.28 YChildrenManager< T > Class Template Reference

```
#include <YChildrenManager.h>
```

Inheritance diagram for YChildrenManager< T >:



Public Types

- typedef std::list< T * > **ChildrenList**

Public Member Functions

- YChildrenManager (T *containerParent)
- virtual ~YChildrenManager ()
- bool hasChildren () const
- bool empty () const
- int count () const
- ChildrenList::const_iterator begin () const
- ChildrenList::const_iterator end () const
- ChildrenList::const_reverse_iterator rbegin () const
- ChildrenList::const_reverse_iterator rend () const
- T * firstChild ()
- T * lastChild ()
- virtual void add (T *child)
- virtual void remove (T *child)
- virtual void clear ()
- bool contains (T *child) const
- T * container () const

Protected Attributes

- T * **_container**
- ChildrenList **_children**

3.28.1 Detailed Description

template<class T>class YChildrenManager< T >

Abstract base template class for children management, such as child widgets.

Definition at line 37 of file YChildrenManager.h.

3.28.2 Constructor & Destructor Documentation

3.28.2.1 template<class T > YChildrenManager< T >::YChildrenManager (T * *containerParent*) [inline]

Constructor.

'containerParent' is the class whose children are managed.

Definition at line 46 of file YChildrenManager.h.

3.28.2.2 template<class T > virtual YChildrenManager< T >::~YChildrenManager ( ) [inline], [virtual]

Destructor.

Definition at line 53 of file YChildrenManager.h.

3.28.3 Member Function Documentation

3.28.3.1 template<class T > virtual void YChildrenManager< T >::add (T * *child*) [inline], [virtual]

Add a new child.

This may throw exceptions if more children are added than the class whose children are handled (the associated widget) can handle.

Reimplemented in YChildrenRejector< T >, and YSingleChildManager< T >.

Definition at line 116 of file YChildrenManager.h.

3.28.3.2 template<class T > ChildrenList::const_iterator YChildrenManager< T >::begin ( ) const [inline]

Return an iterator that points to the first child.

Definition at line 76 of file YChildrenManager.h.

3.28.3.3 template<class T > virtual void YChildrenManager< T >::clear ( ) [inline], [virtual]

Remove all children. This only removes the children from the children manager's list; it does not delete them.

Definition at line 130 of file YChildrenManager.h.

3.28.3.4 template<class T > T* YChildrenManager< T >::container ( ) const [inline]

Returns the associated container, i.e. the object whose children are handled here.

Definition at line 148 of file YChildrenManager.h.

3.28.3.5 template<class T > bool YChildrenManager< T >::contains (T * *child*) const [inline]

Check if the children list contains the specified child. Returns 'true' if the children list contains the child, 'false' otherwise.

Definition at line 138 of file YChildrenManager.h.

3.28.3.6 template<class T> int YChildrenManager<T>::count ( ) const [inline]

Returns the number of children.

Definition at line 71 of file YChildrenManager.h.

3.28.3.7 template<class T> bool YChildrenManager<T>::empty ( ) const [inline]

Check if the children list is empty, i.e. if there are no children.

Definition at line 66 of file YChildrenManager.h.

3.28.3.8 template<class T> ChildrenList::const_iterator YChildrenManager<T>::end ( ) const [inline]

Return an iterator that points after the last child.

Definition at line 82 of file YChildrenManager.h.

3.28.3.9 template<class T> T* YChildrenManager<T>::firstChild ( ) [inline]

Returns the first child or 0 if there is none. Useful mostly for children managers that handle only one child.

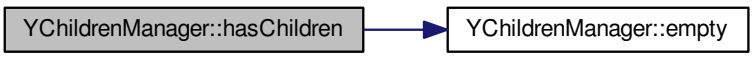
Definition at line 101 of file YChildrenManager.h.

3.28.3.10 template<class T> bool YChildrenManager<T>::hasChildren ( ) const [inline]

Check if there are any children.

Definition at line 61 of file YChildrenManager.h.

Here is the call graph for this function:



3.28.3.11 template<class T> T* YChildrenManager<T>::lastChild ( ) [inline]

Returns the last child or 0 if there is none.

Definition at line 107 of file YChildrenManager.h.

3.28.3.12 template<class T> ChildrenList::const_reverse_iterator YChildrenManager<T>::rbegin ( ) const [inline]

Return a reverse iterator that points to the last child.

Definition at line 88 of file YChildrenManager.h.

3.28.3.13 template<class T > virtual void YChildrenManager< T >::remove (T * *child*) [inline], [virtual]

Remove a child. This only removes the child from the children manager's list; it does not delete it.

Definition at line 123 of file YChildrenManager.h.

3.28.3.14 template<class T > ChildrenList::const_reverse_iterator YChildrenManager< T >::rend ( ) const [inline]

Return a reverse iterator that points before the first child.

Definition at line 94 of file YChildrenManager.h.

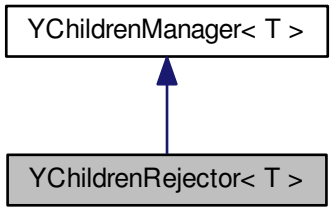
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YChildrenManager.h

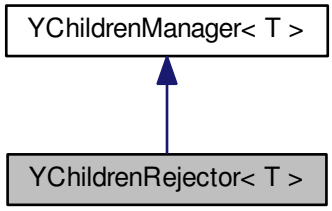
3.29 YChildrenRejector< T > Class Template Reference

`#include <YChildrenManager.h>`

Inheritance diagram for YChildrenRejector< T >:



Collaboration diagram for YChildrenRejector< T >:



Public Member Functions

- YChildrenRejector (T *containerParent)
- virtual void add (T *child)

Additional Inherited Members

3.29.1 Detailed Description

template<class T>class YChildrenRejector< T >

Children manager that rejects all children.

Useful for widget classes that can't handle children such as YPushButton, YSelectionBox etc.

Definition at line 202 of file YChildrenManager.h.

3.29.2 Constructor & Destructor Documentation

3.29.2.1 template<class T > YChildrenRejector< T >::YChildrenRejector (T * containerParent) [inline]

Constructor.

Definition at line 208 of file YChildrenManager.h.

3.29.3 Member Function Documentation

3.29.3.1 template<class T > virtual void YChildrenRejector< T >::add (T * child) [inline], [virtual]

Add a new child.

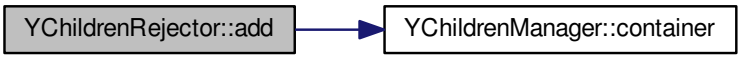
Reimplemented from YChildrenManager.

Since this class is designed to reject children, this always throws a YUITooManyChildrenException.

Reimplemented from YChildrenManager< T >.

Definition at line 220 of file YChildrenManager.h.

Here is the call graph for this function:



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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YChildrenManager.h

3.30 YCodeLocation Class Reference

```
#include <YUIException.h>
```

Public Member Functions

- YCodeLocation (const std::string &file_r, const std::string &func_r, int line_r)
- YCodeLocation ()
- std::string file () const
- std::string func () const
- int line () const
- std::string asString () const

Friends

- std::ostream & operator<< (std::ostream &str, const YCodeLocation &obj)

3.30.1 Detailed Description

Helper class for UI exceptions: Store *FILE*, *FUNCTION* and *LINE*. Construct this using the YUI_EXCEPTION_CODE_-LOCATION macro.

Definition at line 213 of file YUIException.h.

3.30.2 Constructor & Destructor Documentation

3.30.2.1 YCodeLocation::YCodeLocation (const std::string & *file_r*, const std::string & *func_r*, int *line_r*) [inline]

Constructor. Commonly called using the YUI_EXCEPTION_CODE_LOCATION macro.

Definition at line 220 of file YUIException.h.

3.30.2.2 YCodeLocation::YCodeLocation ( ) [inline]

Default constructor.

Definition at line 231 of file YUIException.h.

3.30.3 Member Function Documentation

3.30.3.1 std::string YCodeLocation::asString ( ) const

Returns the location in normalized string format.

Definition at line 41 of file YUIException.cc.

3.30.3.2 std::string YCodeLocation::file ( ) const [inline]

Returns the source file name where the exception occured.

Definition at line 238 of file YUIException.h.

3.30.3.3 std::string YCodeLocation::func ( ) const [inline]

Returns the name of the function where the exception occured.

Definition at line 243 of file YUIException.h.

3.30.3.4 int YCodeLocation::line ( ) const [inline]

Returns the source line number where the exception occured.

Definition at line 248 of file YUIException.h.

3.30.4 Friends And Related Function Documentation

3.30.4.1 std::ostream& operator<< (std::ostream & *str*, const YCodeLocation & *obj*) [friend]

Stream output

YCodeLocation stream output

Definition at line 56 of file YUIException.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.cc

3.31 YColor Class Reference

```
#include <YColor.h>
```

Public Member Functions

- YColor (uchar red, uchar green, uchar blue)
- YColor ()
- uchar red () const
- uchar green () const
- uchar blue () const
- bool isUndefined () const
- bool isDefined () const

3.31.1 Detailed Description

Helper class to define an RGB color.

Definition at line 34 of file YColor.h.

3.31.2 Constructor & Destructor Documentation

3.31.2.1 YColor::YColor (uchar *red,* uchar *green,* uchar *blue*) [inline]

Constructor.

Definition at line 40 of file YColor.h.

3.31.2.2 YColor::YColor ( ) [inline]

Default constructor: Create "undefined" color.

Definition at line 50 of file YColor.h.

3.31.3 Member Function Documentation

3.31.3.1 uchar YColor::blue ( ) const [inline]

Return the blue component (0: none, 255: bright blue).

Definition at line 68 of file YColor.h.

3.31.3.2 uchar YColor::green ( ) const [inline]

Return the green component (0: none, 255: bright green).

Definition at line 63 of file YColor.h.

3.31.3.3 bool YColor::isDefined ( ) const [inline]

Return 'true' if this color is defined.

Definition at line 78 of file YColor.h.

3.31.3.4 bool YColor::isUndefined ( ) const [inline]

Return 'true' if this color is undefined.

Definition at line 73 of file YColor.h.

3.31.3.5 uchar YColor::red ( ) const [inline]

Return the red component (0: none, 255: bright red).

Definition at line 58 of file YColor.h.

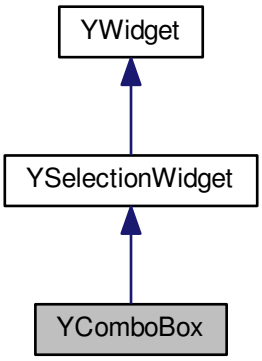
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YColor.h

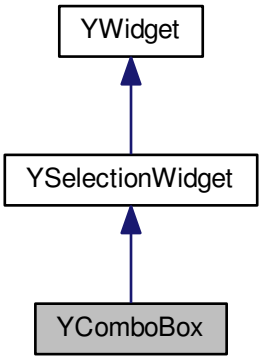
3.32 YComboBox Class Reference

```
#include <YComboBox.h>
```

Inheritance diagram for YComboBox:



Collaboration diagram for YComboBox:



Public Member Functions

- virtual ~YComboBox ()
- virtual const char * widgetClass () const
- bool editable () const
- std::string value ()

- void setValue (const std::string &newText)
- virtual YItem * selectedItem ()
- virtual YItemCollection selectedItems ()
- virtual void selectItem (YItem *item, bool selected=true)
- std::string validChars ()
- virtual void setValidChars (const std::string &validChars)
- int inputMaxLength () const
- virtual void setInputMaxLength (int numberOfChars)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- const char * userInputProperty ()

Protected Member Functions

- YComboBox (YWidget *parent, const std::string &label, bool editable)
- virtual std::string text ()=0
- virtual void setText (const std::string &newText)=0

3.32.1 Detailed Description

ComboBox (a.k.a. "drop down box", "drop down selection"):

A widget with a drop-down list of predefined values to select from. Optionally, this widget can be created in "editable" mode which means that the user can freely enter any text.

In non-editable mode, a ComboBox works very much like a SelectionBox that uses fewer screen space. In that mode, it is recommended to use selectedItem() to retrieve its current value and selectItem() to set it.

In editable mode, a ComboBox is more like an InputField with a list to pick predefined values from (for less typing). In that mode, it is recommended to use value() and setValue().

In either mode, it might be dangerous to use the iterators the (itemsBegin(), itemsEnd()) the base class (YSelectionWidget) provides to find the currently selected item: The items' "selected" flag may or may not be up to date. YComboBox::selectedItem() makes sure they are up to date.

Definition at line 53 of file YComboBox.h.

3.32.2 Constructor & Destructor Documentation

3.32.2.1 YComboBox::YComboBox (YWidget * *parent*, const std::string & *label*, bool *editable*) [protected]

Constructor.

'editable' means the user can freely enter any value without being restricted to the items of the ComboBox's list.

Definition at line 49 of file YComboBox.cc.

3.32.2.2 YComboBox::~YComboBox ( ) [virtual]

Destructor.

Definition at line 58 of file YComboBox.cc.

3.32.3 Member Function Documentation

3.32.3.1 bool YComboBox::editable ( ) const

Return 'true' if this ComboBox is editable, i.e. if the user can freely enter any value without being restricted to the items of the ComboBox's list.

Notice that this can only be set in the constructor.

Definition at line 64 of file YComboBox.cc.

3.32.3.2 YPropertyValue YComboBox::getProperty (const std::string & *propertyName*) [virtual]

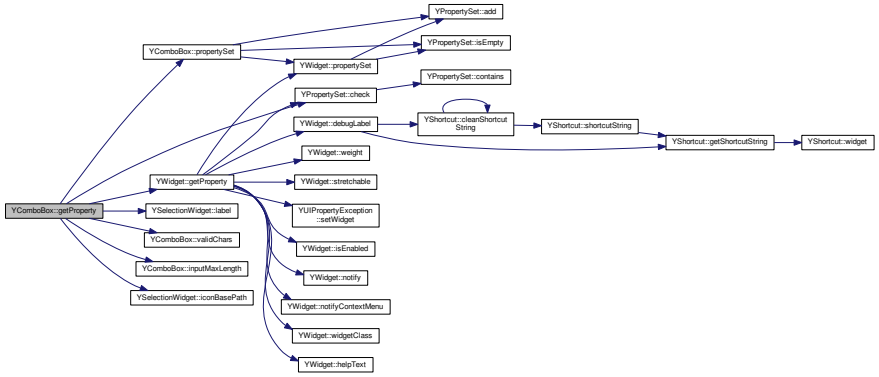
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 220 of file YComboBox.cc.

Here is the call graph for this function:



3.32.3.3 int YComboBox::inputMaxLength ( ) const

The maximum input length, i.e., the maximum number of characters the user can enter. -1 means no limit.

This is only meaningful for if the ComboBox is editable.

Definition at line 82 of file YComboBox.cc.

3.32.3.4 const YPropertySet & YComboBox::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 172 of file YComboBox.cc.

Here is the call graph for this function:

3.32.3.5 YItem * YComboBox::selectedItem ( ) [virtual]

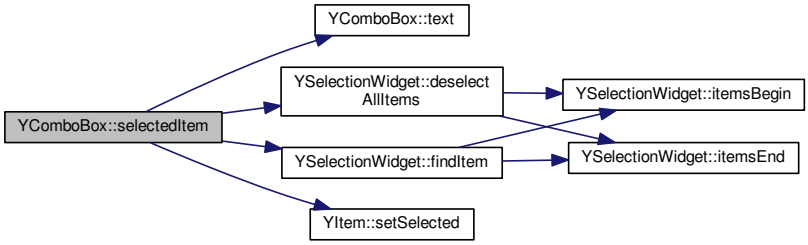
Return the (first) selected item or 0 if none is selected or if this ComboBox is editable and the user entered something that does not match any of the ComboBox's list items (in that case, use value() instead).

Reimplemented from YSelectionWidget for better reliability: This will compare an editable ComboBox's user input against the text labels of all items and try to return an item if there is any match.

Reimplemented from YSelectionWidget.

Definition at line 136 of file YComboBox.cc.

Here is the call graph for this function:



3.32.3.6 YItemCollection YComboBox::selectedItems ( ) [virtual]

Return all selected items.

This is not particularly useful for ComboBoxes since there can be no more than one selected item anyway; * better use selectedItem() or value() instead.

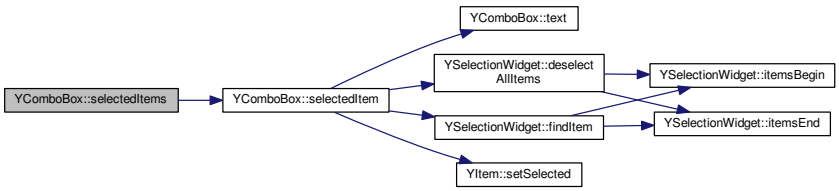
This function does not transfer ownership of those items to the caller, so don't try to delete them!

Reimplemented from YSelectionWidget for better reliability.

Reimplemented from YSelectionWidget.

Definition at line 157 of file YComboBox.cc.

Here is the call graph for this function:



3.32.3.7 void YComboBox::selectItem (YItem * *item*, bool *selected* = `true`) [virtual]

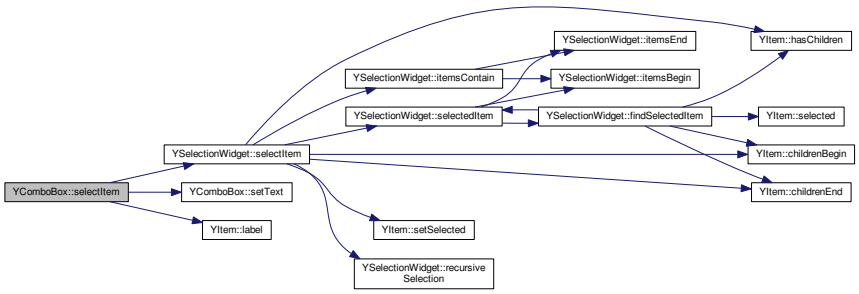
Select or deselect an item. See also setValue().

Reimplemented from YSelectionWidget.

Reimplemented from YSelectionWidget.

Definition at line 122 of file YComboBox.cc.

Here is the call graph for this function:



3.32.3.8 void YComboBox::setInputMaxLength (int *numberOfChars*) [virtual]

Set the maximum input length, i.e., the maximum number of characters the user can enter. -1 means no limit.

This is only meaningful for if the ComboBox is editable.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 88 of file YComboBox.cc.

3.32.3.9 bool YComboBox::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

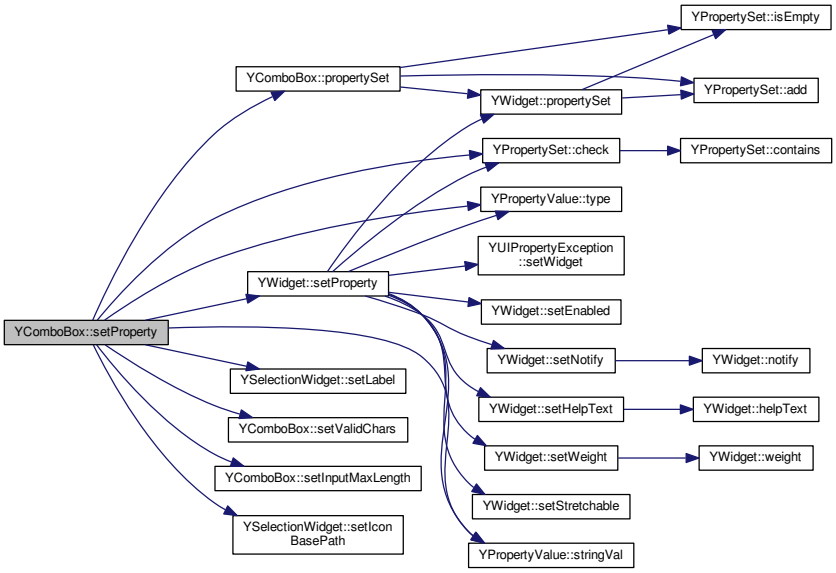
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 200 of file YComboBox.cc.

Here is the call graph for this function:



3.32.3.10 virtual void YComboBox::setText (const std::string & *newText*) [protected],[pure virtual]

Set this ComboBox's current value as text.

Called internally whenever the content is to change programmatically. Don't call setValue() or selectItem() from here.

Derived classes are required to implement this function.

3.32.3.11 void YComboBox::setValidChars (const std::string & *validChars*) [virtual]

Set the valid input characters. No input validation is performed (i.e., the user can enter anything) if this is empty.

This is only meaningful for if the ComboBox is editable.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 76 of file YComboBox.cc.

3.32.3.12 void YComboBox::setValue (const std::string & *newText*)

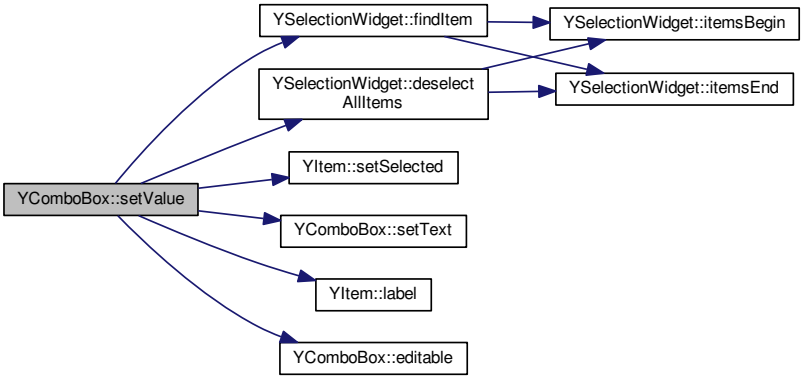
Set the value of this ComboBox by string: Try to find a list item with that label and select it.

If there is no matching list item, editable ComboBoxes will set their input field to that text. Non-editable ComboBoxes will throw an exception.

See also selectItem().

Definition at line 100 of file YComboBox.cc.

Here is the call graph for this function:



3.32.3.13 virtual std::string YComboBox::text ( ) [protected],[pure virtual]

Return this ComboBox's current value as text.

Called internally from value(), selectedItem() and related.

Derived classes are required to implement this function.

3.32.3.14 const char* YComboBox::userInputProperty ( ) [inline],[virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 211 of file YComboBox.h.

3.32.3.15 std::string YComboBox::validChars ( )

Get the valid input characters. No input validation is performed (i.e., the user can enter anything) if this is empty.

This is only meaningful for if the ComboBox is editable.

Definition at line 70 of file YComboBox.cc.

3.32.3.16 std::string YComboBox::value ( )

Return the value of this ComboBox:

The text of a list item if the user (or the appplication) selected a list item or the content of the ComboBox's input field if the ComboBox is editable and the user (or the application) entered text there.

See also YComboBox::selectedItem().

Definition at line 94 of file YComboBox.cc.

Here is the call graph for this function:

3.32.3.17 virtual const char* YComboBox::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YSelectionWidget.

Definition at line 74 of file YComboBox.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YComboBox.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YComboBox.cc

3.33 YComboBoxPrivate Struct Reference

Public Member Functions

- **YComboBoxPrivate** (bool editable)

Public Attributes

- bool **editable**
- std::string **validChars**
- int **inputMaxLength**

3.33.1 Detailed Description

Definition at line 34 of file YComboBox.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YComboBox.cc

3.34 YCommandLine Class Reference

```
#include <YCommandLine.h>
```

Public Member Functions

- YCommandLine ()
- ~YCommandLine ()
- int argc () const
- char ** argv () const
- int size () const
- std::string arg (int index) const
- std::string operator[] (int index) const
- void add (const std::string &arg)
- void remove (int index)
- void replace (int index, const std::string &arg)
- int find (const std::string &argName) const

3.34.1 Detailed Description

Utility class to access /proc/<pid>/cmdline to retrieve argc and argv

Definition at line 37 of file YCommandLine.h.

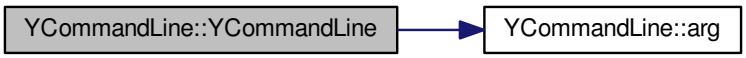
3.34.2 Constructor & Destructor Documentation

3.34.2.1 YCommandLine::YCommandLine ( )

Constructor. This will read /proc/<pid>/cmdline of this process.

Definition at line 48 of file YCommandLine.cc.

Here is the call graph for this function:



3.34.2.2 YCommandLine::~YCommandLine ( )

Destructor.

Definition at line 71 of file YCommandLine.cc.

3.34.3 Member Function Documentation

3.34.3.1 void YCommandLine::add (const std::string & *arg*)

Add a command line argument (at the end of the existing ones).

Definition at line 102 of file YCommandLine.cc.

3.34.3.2 std::string YCommandLine::arg (int *index*) const

Return command line argument no. 'index' (from 0 on).

This might throw an YUIIndexOutOfRangeException.

Definition at line 109 of file YCommandLine.cc.

3.34.3.3 int YCommandLine::argc ( ) const

Return the number of arguments in the command line. Remember that the command itself (the binary of the process) is included, so a value of 1 (not 0!) means "no additional arguments".

Definition at line 78 of file YCommandLine.cc.

3.34.3.4 char ** YCommandLine::argv ( ) const

Return the arguments in a C compatible fashion: An array of pointers to characters. The data are copied with strdup(), so they are valid beyond the life time of this object (but OTOH should be released with free() at some point).

Definition at line 85 of file YCommandLine.cc.

Here is the call graph for this function:

3.34.3.5 int YCommandLine::find (const std::string & *argName*) const

Find a command line argument 'argName' ("-display" etc.). Notice that leading minus signs must be specified in 'argName'. Since argv[0] is the program name, the search starts from argv[1].

Return the position of 'argName' (from 0 on) or -1 if not found.

Definition at line 136 of file YCommandLine.cc.

Here is the call graph for this function:

3.34.3.6 std::string YCommandLine::operator[] (int *index*) const [inline]

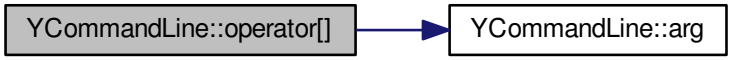
Return command line argument no. 'index' (from 0 on) as operator[]:

for (int i=0; i < cmdLine.argc(); i++) cout << cmdLine[i] << std::endl;

This might throw an YUIIndexOutOfRangeException.

Definition at line 85 of file YCommandLine.h.

Here is the call graph for this function:



3.34.3.7 void YCommandLine::remove (int *index*)

Remove command line argument no. 'index' (from 0 on).

This might throw an YUIIndexOutOfRangeException.

Definition at line 118 of file YCommandLine.cc.

3.34.3.8 void YCommandLine::replace (int *index*, const std::string & *arg*)

Replace command line argument no. 'index' (from 0 on) with 'arg'.

This might throw an YUIIndexOutOfRangeException.

Definition at line 127 of file YCommandLine.cc.

3.34.3.9 int YCommandLine::size ( ) const [inline]

Alias for argc() for those who like a more C++ -like syntax.

Definition at line 68 of file YCommandLine.h.

Here is the call graph for this function:

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YCommandLine.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YCommandLine.cc

3.35 YCommandLinePrivate Struct Reference

Public Attributes

- std::vector< std::string > **args**

3.35.1 Detailed Description

Definition at line 39 of file YCommandLine.cc.

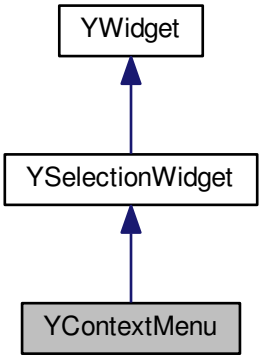
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YCommandLine.cc

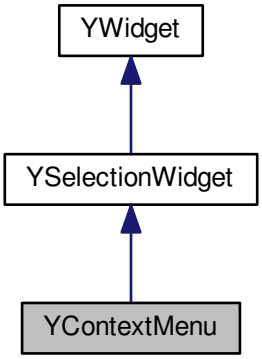
3.36 YContextMenu Class Reference

```
#include <YContextMenu.h>
```

Inheritance diagram for YContextMenu:



Collaboration diagram for YContextMenu:



Public Member Functions

- virtual ~YContextMenu ()
- virtual const char * widgetClass () const
- virtual void rebuildMenuTree ()=0
- virtual void addItems (const YItemCollection &itemCollection)
- virtual void addItem (YItem *item_disown)
- virtual void deleteAllItems ()
- void resolveShortcutConflicts ()

- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Protected Member Functions

- YContextMenu ()
- YMenuItem * findMenuItem (int index)
- YMenuItem * findMenuItem (int index, YItemConstIterator begin, YItemConstIterator end)
- YMenuItem * itemAt (int index)

3.36.1 Detailed Description

ContextMenu: Similar to PushButton, but with several actions: Upon clicking on a ContextMenu (or activating it with the keyboard), a pop-up menu opens where the user can activate an action. Menu items in that pop-up menu can have submenus (that will pop up in separate pop-up menus).

Internally, this widget is more similar to the Tree widget. The difference is that it does not keep a "selected" status, but triggers an action right away, just like a PushButton. Like PushButton, ContextMenu sends an event right away when the user selects an item (clicks on a menu item or activates it with the keyboard). Items that have a submenu never send an event, they simply open their submenu when activated.

Definition at line 48 of file YContextMenu.h.

3.36.2 Constructor & Destructor Documentation

3.36.2.1 YContextMenu::YContextMenu ( ) [protected]

Constructor.

'label' is the user-visible text on the button (not above it like all other SelectionWidgets).

Definition at line 46 of file YContextMenu.cc.

3.36.2.2 YContextMenu::~YContextMenu ( ) [virtual]

Destructor.

Definition at line 55 of file YContextMenu.cc.

3.36.3 Member Function Documentation

3.36.3.1 void YContextMenu::addItem (YItem * *item_disown*) [virtual]

Add one item. This widget assumes ownership of the item object and will delete it in its destructor.

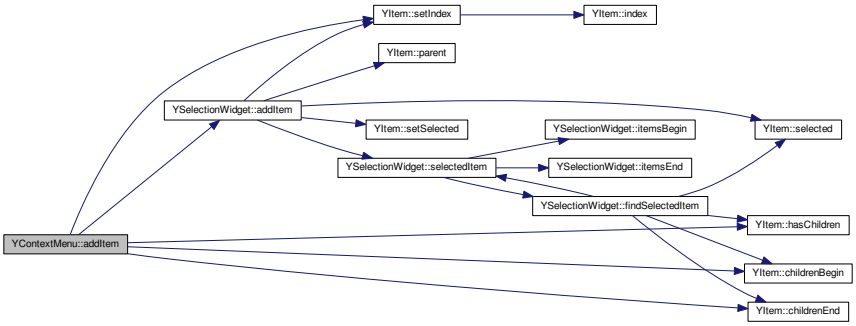
This reimplementation will an index to the item that is unique for all items in this ContextMenu. That index can be used later with findMenuItem() to find the item by that index.

Reimplemented from YSelectionWidget.

Reimplemented from YSelectionWidget.

Definition at line 71 of file YContextMenu.cc.

Here is the call graph for this function:



3.36.3.2 void YContextMenu::addItems (const YItemCollection & *itemCollection*) [virtual]

Add multiple items. For some UIs, this can be more efficient than calling addItem() multiple times. This function also automatically calls resolveShortcutConflicts() and rebuildMenuTree() at the end.

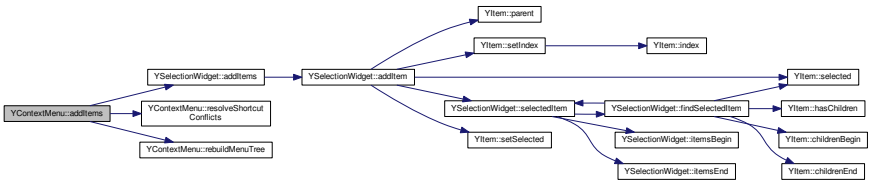
Derived classes can overwrite this function, but they should call this base class function at the end of the new implementation.

Reimplemented from YSelectionWidget.

Reimplemented from YSelectionWidget.

Definition at line 62 of file YContextMenu.cc.

Here is the call graph for this function:



3.36.3.3 void YContextMenu::deleteAllItems ( ) [virtual]

Delete all items.

Reimplemented from YSelectionWidget.

Reimplemented from YSelectionWidget.

Definition at line 97 of file YContextMenu.cc.

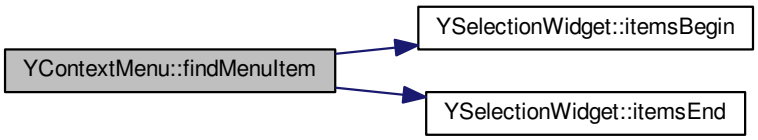
Here is the call graph for this function:

3.36.3.4 YMenuItem * YContextMenu::findMenuItem (int *index*) [protected]

Recursively find the first menu item with the specified index. Returns 0 if there is no such item.

Definition at line 105 of file YContextMenu.cc.

Here is the call graph for this function:



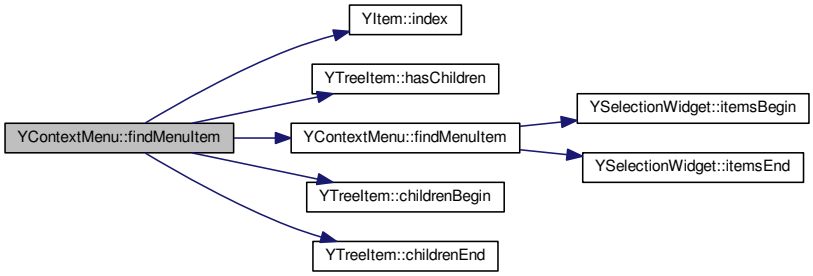
3.36.3.5 YMenuItem * YContextMenu::findMenuItem (int *index*, YItemConstIterator *begin*, YItemConstIterator *end*) [protected]

Recursively find the first menu item with the specified index from iterator 'begin' to iterator 'end'.

Returns 0 if there is no such item.

Definition at line 112 of file YContextMenu.cc.

Here is the call graph for this function:



3.36.3.6 YPropertyValue YContextMenu::getProperty (const std::string & *propertyName*) [virtual]

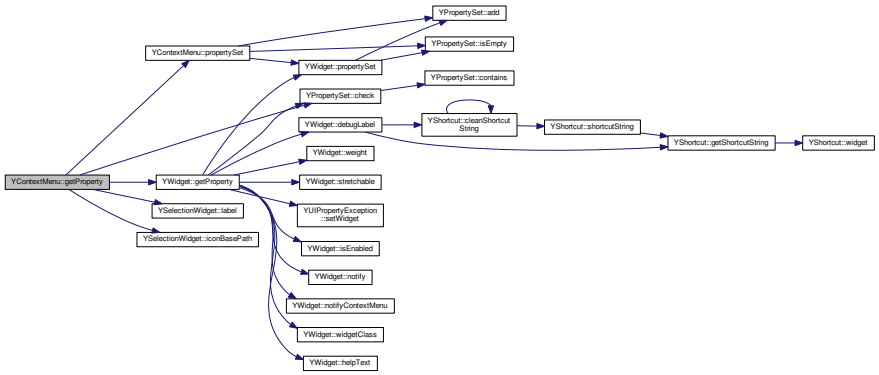
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 192 of file YContextMenu.cc.

Here is the call graph for this function:



3.36.3.7 YMenuItem* YContextMenu::itemAt (int *index*) [inline],[protected]

Alias for findMenuItem(). Reimplemented to ensure consistent behaviour with YSelectionWidget::itemAt().

Definition at line 170 of file YContextMenu.h.

Here is the call graph for this function:

3.36.3.8 const YPropertySet & YContextMenu::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 153 of file YContextMenu.cc.

Here is the call graph for this function:

3.36.3.9 virtual void YContextMenu::rebuildMenuTree ( ) [pure virtual]

Rebuild the displayed menu tree from the internally stored YMenuItems.

The application should call this (once) after all items have been added with addItem(). YContextMenu::addItems() calls this automatically.

Derived classes are required to implement this.

3.36.3.10 void YContextMenu::resolveShortcutConflicts ( )

Resolve keyboard shortcut conflicts: Change shortcuts of menu items if there are duplicates in the respective menu level.

This has to be called after all items are added, but before rebuildMenuTree() (see above). YContextMenu::addItems() calls this automatically.

Definition at line 138 of file YContextMenu.cc.

3.36.3.11 bool YContextMenu::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

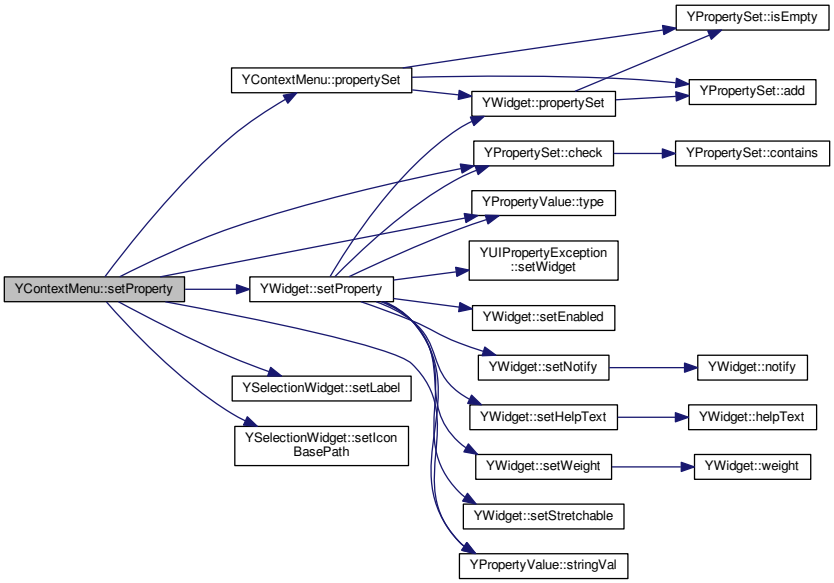
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 175 of file YContextMenu.cc.

Here is the call graph for this function:



3.36.3.12 virtual const char* YContextMenu::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YSelectionWidget.

Definition at line 69 of file YContextMenu.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YContextMenu.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YContextMenu.cc

3.37 YContextMenuPrivate Struct Reference

Public Attributes

- int **nextSerialNo**

3.37.1 Detailed Description

Definition at line 34 of file YContextMenu.cc.

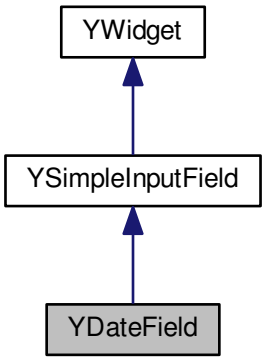
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YContextMenu.cc

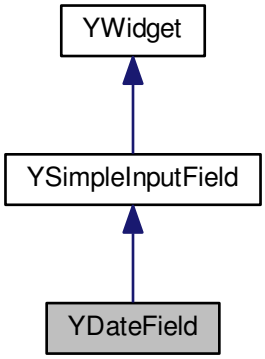
3.38 YDateField Class Reference

```
#include <YDateField.h>
```

Inheritance diagram for YDateField:



Collaboration diagram for YDateField:



Public Member Functions

- virtual ~YDateField ()
- virtual const char * widgetClass () const

Protected Member Functions

- YDateField (YWidget *parent, const std::string &label)

3.38.1 Detailed Description

Input field for entering a date.

Derived classes are required to implement: value() setValue()

For both methods the date is formatted as "YYYY-MM-DD". See YSimpleInputField.h for more details.

Definition at line 42 of file YDateField.h.

3.38.2 Constructor & Destructor Documentation

3.38.2.1 YDateField::YDateField (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

Definition at line 43 of file YDateField.cc.

3.38.2.2 YDateField::~YDateField ( ) [virtual]

Destructor.

Definition at line 51 of file YDateField.cc.

3.38.3 Member Function Documentation

3.38.3.1 virtual const char* YDateField::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 60 of file YDateField.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDateField.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDateField.cc

3.39 YDateFieldPrivate Struct Reference

Public Attributes

- bool **dummy**

3.39.1 Detailed Description

Definition at line 32 of file YDateField.cc.

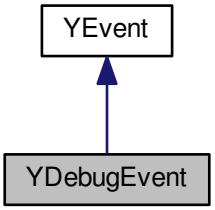
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDateField.cc

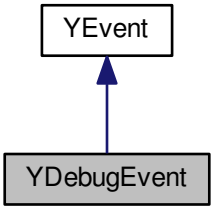
3.40 YDebugEvent Class Reference

```
#include <YEvent.h>
```

Inheritance diagram for YDebugEvent:



Collaboration diagram for YDebugEvent:



Protected Member Functions

- virtual ~YDebugEvent ()

Additional Inherited Members

3.40.1 Detailed Description

Event to be returned upon closing a dialog with the window manager close button (or Alt-F4)

Definition at line 326 of file YEvent.h.

3.40.2 Constructor & Destructor Documentation

3.40.2.1 virtual YDebugEvent::~YDebugEvent ( ) [inline], [protected], [virtual]

Protected destructor - events can only be deleted via YDialog::deleteEvent(). The associated dialog will take care of this event and delete it when appropriate.

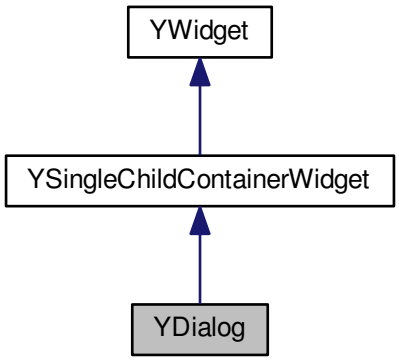
Definition at line 338 of file YEvent.h.

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

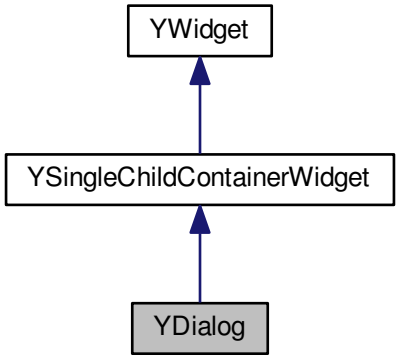
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEvent.h

3.41 YDialog Class Reference

Inheritance diagram for YDialog:



Collaboration diagram for YDialog:



Public Member Functions

- virtual const char ∗ widgetClass () const
- void open ()
- bool isOpen () const
- YEvent ∗ waitForEvent (int timeout_millisec=0)
- YEvent ∗ pollEvent ()

- bool isTopmostDialog () const
- bool destroy (bool doThrow=true)
- void setInitialSize ()
- void recalcLayout ()
- YDialogType dialogType () const
- bool isMainDialog ()
- YDialogColorMode colorMode () const
- void checkShortcuts (bool force=false)
- void postponeShortcutCheck ()
- bool shortcutCheckPostponed () const
- YPushButton * defaultButton () const
- void deleteEvent (YEvent *event)
- void addEventFilter (YEventFilter *eventFilter)
- void removeEventFilter (YEventFilter *eventFilter)
- virtual void highlight (YWidget *child)
- virtual void setDefaultButton (YPushButton *defaultButton)
- virtual void activate ()=0

Static Public Member Functions

- static bool deleteTopmostDialog (bool doThrow=true)
- static void deleteAllDialogs ()
- static void deleteTo (YDialog *dialog)
- static int openDialogsCount ()
- static YDialog * currentDialog (bool doThrow=true)
- static YDialog * topmostDialog (bool doThrow=true)
- static void showText (const std::string &text, bool richText=false)
- static bool showHelpText (YWidget *widget)

Protected Member Functions

- YDialog (YDialogType dialogType, YDialogColorMode colorMode=YDialogNormalColor)
- virtual ~YDialog ()
- virtual void openInternal ()=0
- virtual YEvent * waitForEventInternal (int timeout_millisec)=0
- virtual YEvent * pollEventInternal ()=0
- YEvent * filterInvalidEvents (YEvent *event)
- YEvent * callEventFilters (YEvent *event)
- void deleteEventFilters ()

Static Protected Attributes

- static std::stack< YDialog * > _dialogStack

3.41.1 Detailed Description

Definition at line 41 of file YDialog.h.

3.41.2 Constructor & Destructor Documentation

3.41.2.1 YDialog::YDialog (YDialogType *dialogType*, YDialogColorMode *colorMode* = `YDialogNormalColor`) `[protected]`

Constructor.

'dialogType' is one of YMainDialog or YPopupDialog.

'colorMode' can be set to YDialogWarnColor to use very bright "warning" colors or YDialogInfoColor to use more prominent, yet not quite as bright as "warning" colors. Use both only very rarely.

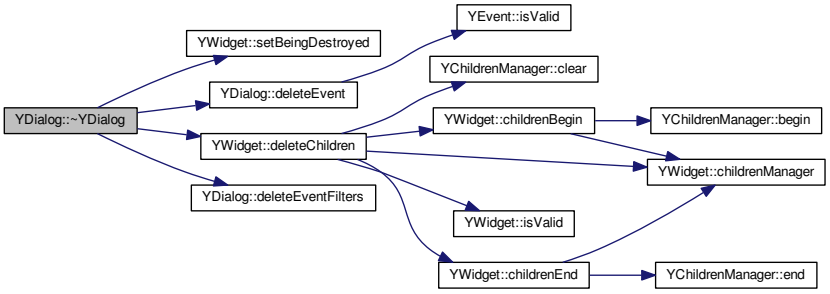
Definition at line 111 of file YDialog.cc.

3.41.2.2 YDialog::~YDialog ( ) `[protected],[virtual]`

Destructor. Don't delete a dialog directly, use YDialog::deleteTopmostDialog() or YDialog::destroy().

Definition at line 127 of file YDialog.cc.

Here is the call graph for this function:



3.41.3 Member Function Documentation

3.41.3.1 virtual void YDialog::activate ( ) `[pure virtual]`

Activate this dialog: Make sure that it is shown as the topmost dialog of this application and that it can receive input.

Derived classes are required to implement this.

3.41.3.2 void YDialog::addEventFilter (YEventFilter * *eventFilter*)

Add an event filter. This can be useful to catch certain types of events before they are delivered to the application. All event filters are called (in unspecified order) in waitForEvent(). Each one may consume an event, pass it through unchanged, or replace it with a newly created event.

Normally, an YEventFilter should be created on the heap with 'new'. In that case, the dialog's destructor will take care of deleting it.

In rare cases it might make sense to create an YEventFilter on the stack (as a local variable) and rely on that variable to go out of scope and be destroyed before the dialog gets destroyed. But that may be risky.

Notice that applications never need to call this function: YEventFilter does it automatically in its constructor.

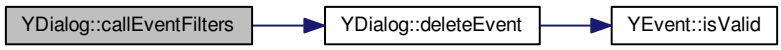
Definition at line 560 of file YDialog.cc.

3.41.3.3 YEvent * YDialog::callEventFilters (YEvent * *event*) [protected]

Call the installed event filters.

Definition at line 594 of file YDialog.cc.

Here is the call graph for this function:



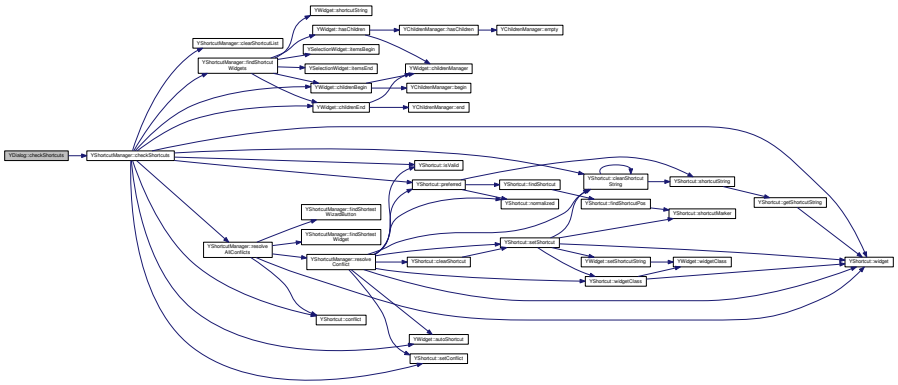
3.41.3.4 void YDialog::checkShortcuts (bool *force* = false)

Checks the keyboard shortcuts of widgets in this dialog unless shortcut checks are postponed or 'force' is 'true'.

A forced shortcut check resets postponed checking.

Definition at line 279 of file YDialog.cc.

Here is the call graph for this function:



3.41.3.5 YDialogColorMode YDialog::colorMode ( ) const

Return this dialog's color mode.

Definition at line 258 of file YDialog.cc.

3.41.3.6 YDialog * YDialog::currentDialog (bool *doThrow* = `true`) [static]

Return the current (topmost) dialog.

If there is none, throw a YUINoDialogException if 'doThrow' is 'true' and return 0 if 'doThrow' is false.

Definition at line 491 of file YDialog.cc.

3.41.3.7 YPushButton * YDialog::defaultButton ( ) const

Return this dialog's default button: The button that is activated when the user hits [Return] anywhere in this dialog. Note that this is not the same as the button that currently has the keyboard focus.

This might return 0 if there is no default button.

Definition at line 297 of file YDialog.cc.

3.41.3.8 void YDialog::deleteAllDialogs ( ) [static]

Delete all open dialogs.

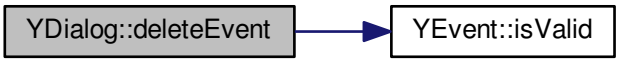
Definition at line 522 of file YDialog.cc.

3.41.3.9 void YDialog::deleteEvent (YEvent * *event*)

Delete an event.

Definition at line 468 of file YDialog.cc.

Here is the call graph for this function:



3.41.3.10 void YDialog::deleteEventFilters ( ) [protected]

Delete all (remaining) event filters.

Definition at line 197 of file YDialog.cc.

3.41.3.11 void YDialog::deleteTo (YDialog * *dialog*) [static]

Delete all dialogs from the topmost to the one specified.

Definition at line 532 of file YDialog.cc.

3.41.3.12 bool YDialog::deleteTopmostDialog (bool *doThrow* = `true`) [static]

Delete the topmost dialog.

Will throw a YUINoDialogException if there is no dialog and 'doThrow' is 'true'.

This is equivalent to YDialog::currentDialog()->destroy().

Returns 'true' if there is another open dialog after deleting, 'false' if there is none.

Definition at line 505 of file YDialog.cc.

3.41.3.13 bool YDialog::destroy (bool *doThrow* = `true`)

Close and delete this dialog (and all its children) if it is the topmost dialog. If this is not the topmost dialog, this will throw an exception if 'doThrow' is true (default).

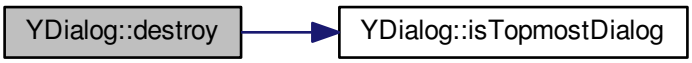
Remember that all pointers to the dialog and its children will be invalid after this operation.

This is intentionally not named close() since close() would not imply that the dialog and its children are deleted.

Returns 'true' upon success, 'false' upon failure.

Definition at line 212 of file YDialog.cc.

Here is the call graph for this function:



3.41.3.14 YDialogType YDialog::dialogType ( ) const

Return this dialog's type (YMainDialog / YPopupDialog /YWizardDialog).

Definition at line 233 of file YDialog.cc.

3.41.3.15 YEvent * YDialog::filterInvalidEvents (YEvent * *event*) [protected]

Filter out invalid events: Return 0 if the event does not belong to this dialog or the unchanged event if it does. Silently discard events from widgets that have become invalid.

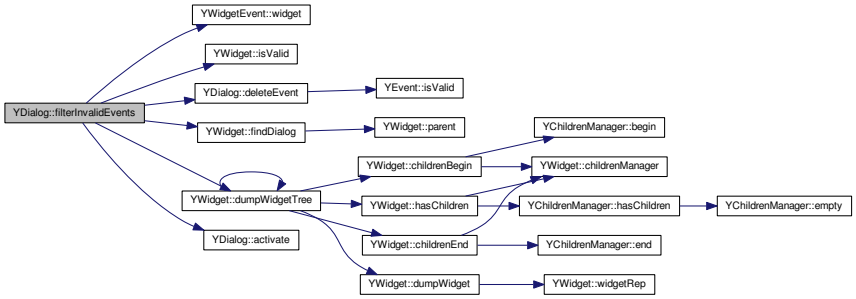
This may legitimately happen if some widget triggered an event yet nobody cared for that event (i.e. called UserInput() or PollInput()) and the widget has been destroyed meanwhile.

Silently discard events from all but the current (topmost) dialog.

This may happen even here even though the specific UI should have taken care about that: Events may still be in the queue. They might have been valid (i.e. belonged to the topmost dialog) when they arrived, but maybe simply nobody has evaluated them.

Definition at line 404 of file YDialog.cc.

Here is the call graph for this function:



3.41.3.16 virtual void YDialog::highlight (YWidget * *child*) [inline], [virtual]

Highlight a child widget of this dialog. This is meant for debugging: YDialogSpy and similar uses.

No more than one widget can be highlighted at any one time in the same dialog. Highlighting another widget unhighlights a previously highlighted widget. 0 means 'unhighlight the last highlighted widget, but don't highlight any other'.

This default implementation does nothing.

Definition at line 306 of file YDialog.h.

3.41.3.17 bool YDialog::isMainDialog ( )

Return 'true' if this dialog is a dialog of main dialog size: YMainDialog or YWizardDialog.

Definition at line 240 of file YDialog.cc.

3.41.3.18 bool YDialog::isOpen ( ) const

Return 'true' if open() has already been called for this dialog.

Definition at line 177 of file YDialog.cc.

3.41.3.19 bool YDialog::isTopmostDialog ( ) const

Return 'true' if this dialog is the topmost dialog.

Definition at line 184 of file YDialog.cc.

3.41.3.20 void YDialog::open ( )

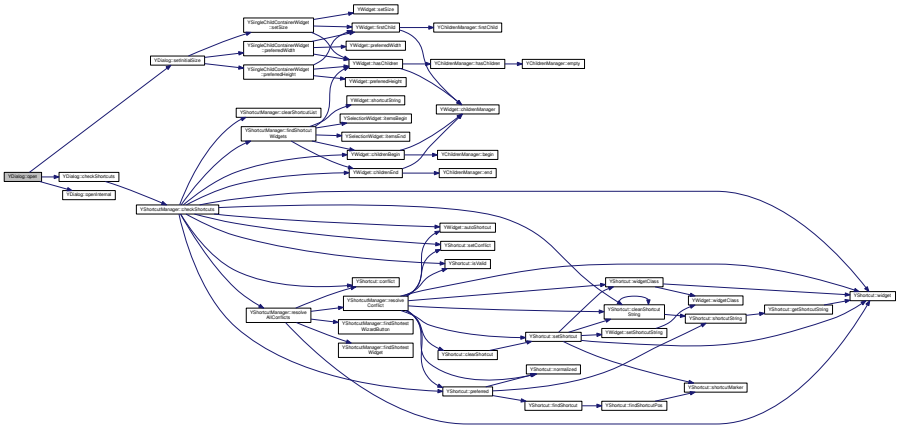
Open a newly created dialog: Finalize it and make it visible on the screen.

Applications should call this once after all children are created. If the application doesn't do this, it will be done automatically upon the next call of YDialog::waitForEvent() (or related). This is OK if YDialog::waitForEvent() is called immediately after creating the dialog anyway. If it is not, the application might appear sluggish to the user.

Derived classes are free to reimplement this, but they should call this base class method in the new implementation.

Definition at line 163 of file YDialog.cc.

Here is the call graph for this function:



3.41.3.21 int YDialog::openDialogsCount ( ) [static]

Returns the number of currently open dialogs (from 1 on), i.e., the depth of the dialog stack.

Definition at line 553 of file YDialog.cc.

3.41.3.22 virtual void YDialog::openInternal ( ) [protected], [pure virtual]

Internal open() method. This is called (exactly once during the life time of the dialog) in open().

Derived classes are required to implement this to do whatever is necessary to make this dialog visible on the screen.

3.41.3.23 YEvent * YDialog::pollEvent ( )

Check if a user event is pending. If there is one, return it. If there is none, do not wait for one - return 0.

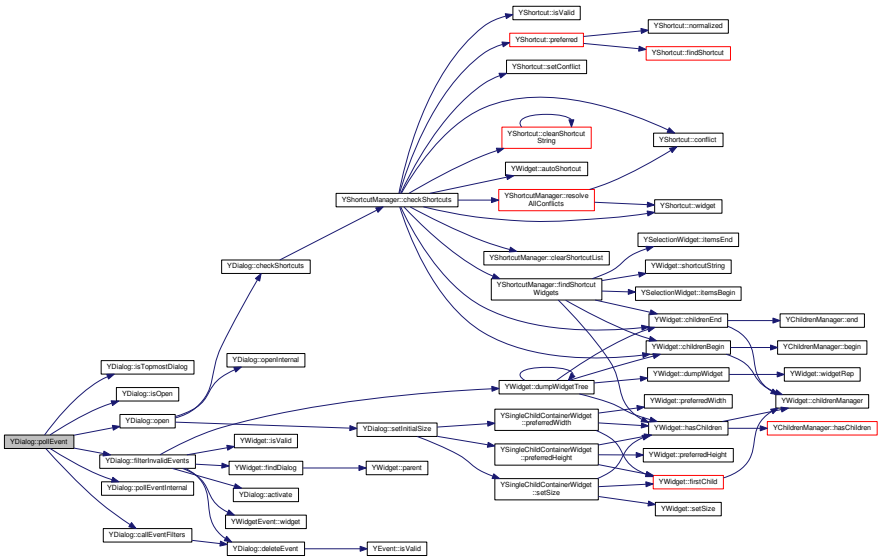
If open() has not been called for this dialog until now, it is called now.

The dialog retains ownership of the event and will delete it upon the next call to waitForEvent() or pollEvent() or when the dialog is deleted. This also means that the return value of this function can safely be ignored without fear of memory leaks.

If this dialog is not the topmost dialog, an exception is thrown.

Definition at line 379 of file YDialog.cc.

Here is the call graph for this function:



3.41.3.24 virtual YEvent* YDialog::pollEventInternal ( ) [protected], [pure virtual]

Check if a user event is pending. If there is one, return it. If there is none, do not wait for one - return 0.

Derived classes are required to implement this.

3.41.3.25 void YDialog::postponeShortcutCheck ( )

From now on, postpone keyboard shortcut checks - i.e. normal (not forced) checkKeyboardShortcuts() will do nothing. Reset this mode by forcing a shortcut check with checkKeyboardShortcuts(true).

Definition at line 265 of file YDialog.cc.

3.41.3.26 void YDialog::recalcLayout ( )

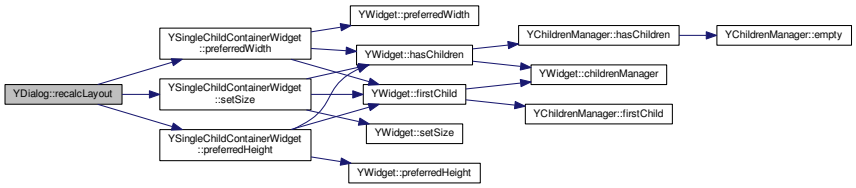
Recalculate the layout of the dialog and of all its children after children have been added or removed or if any of them changed its preferred width of height.

This is a very expensive operation. Call it only when really necessary. YDialog::open() includes a call to YDialog::setInitialSize() which does the same.

The basic idea behind this function is to call it when the dialog changed after it (and its children hierarchy) was initially created.

Definition at line 330 of file YDialog.cc.

Here is the call graph for this function:



3.41.3.27 void YDialog::removeEventFilter (YEventFilter * *eventFilter*)

Remove an event filter.

Notice that applications never need to call this function: YEventFilter does it automatically in its destructor.

Definition at line 582 of file YDialog.cc.

3.41.3.28 void YDialog::setDefaultButton (YPushButton * *defaultButton*) [virtual]

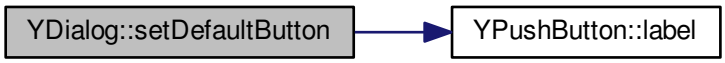
Set this dialog's default button (the button that is activated when the user hits [Return] anywhere in this dialog). 0 means no default button.

There should be no more than one default button in a dialog.

Derived classes are free to overwrite this method, but they should call this base class method in the new implementation.

Definition at line 304 of file YDialog.cc.

Here is the call graph for this function:

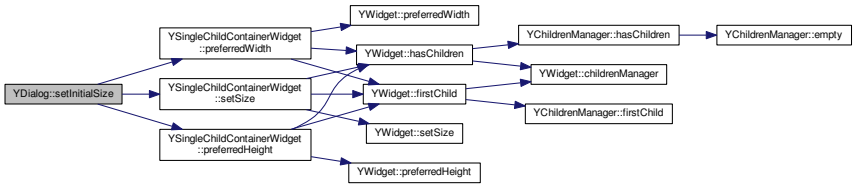


3.41.3.29 void YDialog::setInitialSize ( )

Set the initial dialog size, depending on dialogType: YMainDialog dialogs get the UI's "default main window" size, YPopupDialog dialogs use their content's preferred size.

Definition at line 318 of file YDialog.cc.

Here is the call graph for this function:



3.41.3.30 bool YDialog::shortcutCheckPostponed ( ) const

Return whether or not shortcut checking is currently postponed.

Definition at line 272 of file YDialog.cc.

3.41.3.31 bool YDialog::showHelpText (YWidget * *widget*) [static]

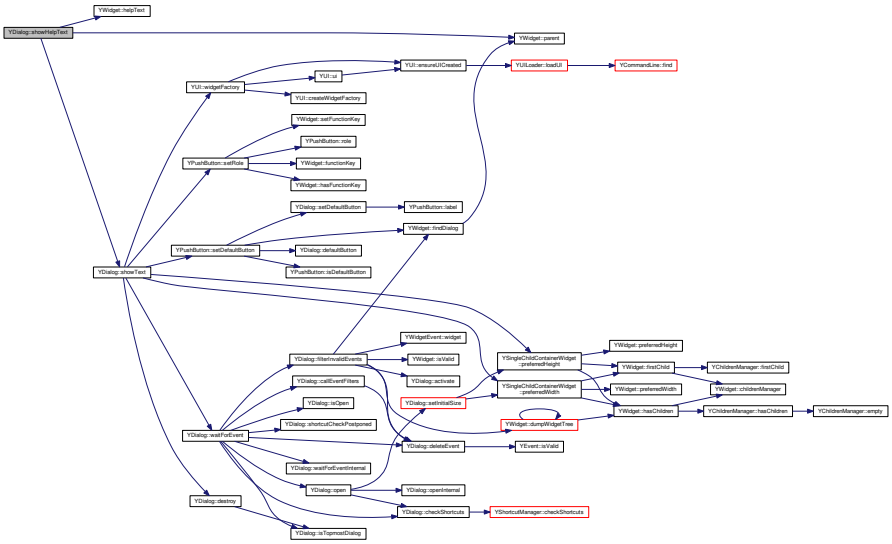
Show the help text for the specified widget. If it doesn't have one, traverse up the widget hierarchy until there is one.

If there is a help text, it is displayed in a pop-up dialog with a local event loop.

This returns 'true' on success (there was a help text) and 'false' on failure (no help text).

Definition at line 660 of file YDialog.cc.

Here is the call graph for this function:

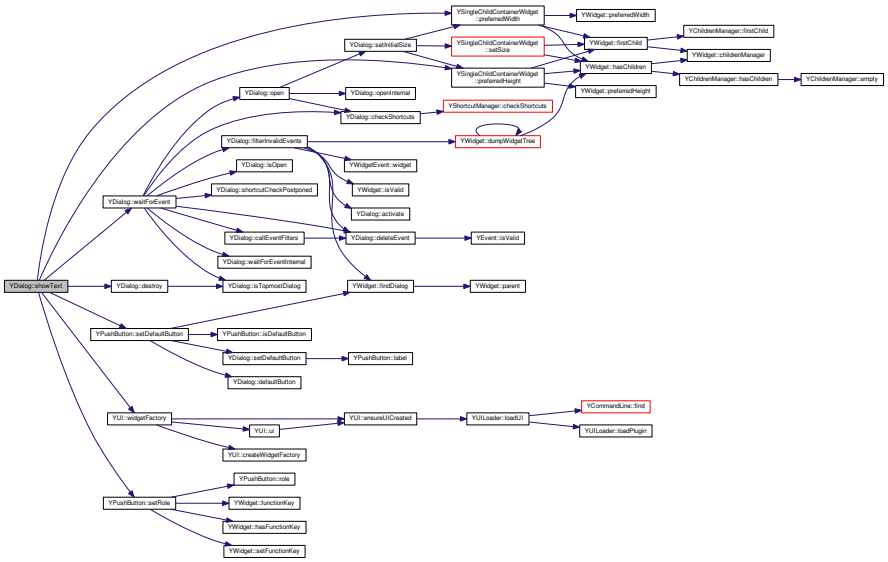


3.41.3.32 void YDialog::showText (const std::string & *text*, bool *richText* = `false`) [static]

Show the specified text in a pop-up dialog with a local event loop. This is useful for help texts. 'richText' indicates if YRichText formatting should be applied.

Definition at line 614 of file YDialog.cc.

Here is the call graph for this function:



3.41.3.33 static YDialog* YDialog::topmostDialog (bool *doThrow* = `true`) [inline], [static]

Alias for currentDialog().

Definition at line 194 of file YDialog.h.

Here is the call graph for this function:

3.41.3.34 YEvent * YDialog::waitForEvent (int *timeout_millisec* = `0`)

Wait for a user event. In most cases, this means waiting until the user has clicked on a button in this dialog. If any widget has its 'notify' flag set (`opt` (notify) in YCP, setNotify(true) in C++), an action on such a widget will also make waitForEvent() return.

If the specified timeout elapses without any user event, a YTimeoutEvent will be returned. 0 means no timeout (wait forever).

If open() has not been called for this dialog until now, it is called now.

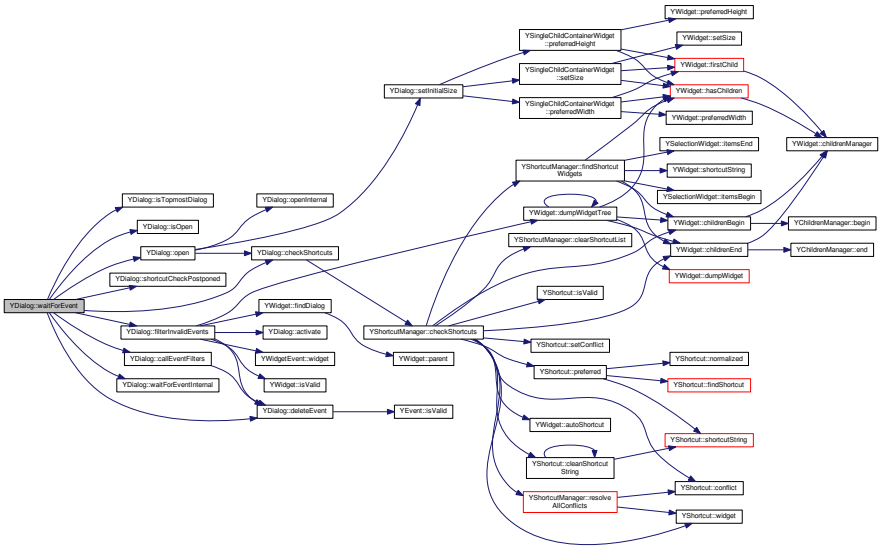
The dialog retains ownership of the event and will delete it upon the next call to waitForEvent() or pollEvent() or when the dialog is deleted. This also means that the return value of this function can safely be ignored without fear of memory leaks.

Applications can create YEventFilters to act upon some events before they are delivered to the application. Each event filter of this dialog is called (in undefined order) in waitForEvent(). An event filter can consume an event (in which case waitForEvent() will return to its internal event loop), pass it through unchanged, or even replace it with a new event. Refer to the YEventFilter documentation for more details.

If this dialog is not the topmost dialog, an exception is thrown.

Definition at line 339 of file YDialog.cc.

Here is the call graph for this function:



3.41.3.35 virtual YEvent* YDialog::waitForEventInternal (int *timeout_millisec*) [protected], [pure virtual]

Wait for a user event.

Derived classes are required to implement this.

3.41.3.36 virtual const char* YDialog::widgetClass ( ) const [inline], [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 68 of file YDialog.h.

3.41.4 Member Data Documentation

3.41.4.1 std::stack< YDialog * > YDialog::_dialogStack [static], [protected]

Stack holding all currently existing dialogs.

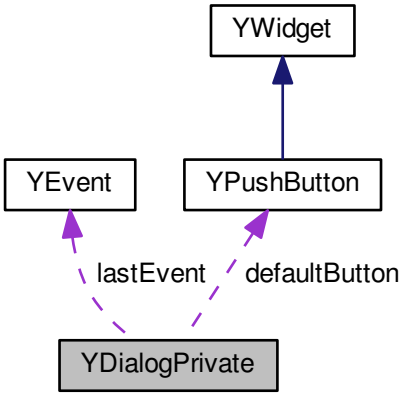
Definition at line 393 of file YDialog.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDialog.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDialog.cc
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUI.cc

3.42 YDialogPrivate Struct Reference

Collaboration diagram for YDialogPrivate:



Public Member Functions

- **YDialogPrivate** (YDialogType dialogType, YDialogColorMode colorMode)

Public Attributes

- YDialogType **dialogType**
- YDialogColorMode **colorMode**
- bool **shortcutCheckPostponed**
- YPushButton * **defaultButton**
- bool **isOpen**
- YEvent * **lastEvent**
- YEventFilterList **eventFilterList**

3.42.1 Detailed Description

Definition at line 55 of file YDialog.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDialog.cc

3.43 YDialogSpy Class Reference

```
#include <YDialogSpy.h>
```

Public Member Functions

- void showProperties ()
- void hideProperties ()
- bool propertiesShown () const

Static Public Member Functions

- static void showDialogSpy (YDialog *dialog=0)

Protected Member Functions

- YDialogSpy (YDialog *dialog=0)
- virtual ~YDialogSpy ()
- void exec ()
- void showProperties (YWidget *widget)

3.43.1 Detailed Description

An interactive dialog debugger: Show the structure and content of a dialog and its widgets.

This can be invoked by special key combinations: Ctrl-Alt-Shift-Y in the Qt UI

Definition at line 43 of file YDialogSpy.h.

3.43.2 Constructor & Destructor Documentation

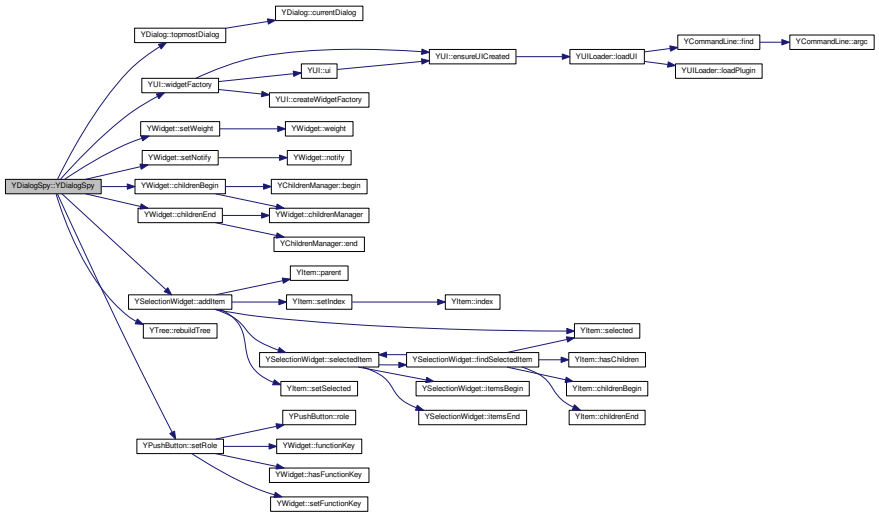
3.43.2.1 YDialogSpy::YDialogSpy (YDialog * *dialog* = 0) [protected]

Constructor: Create a YDialogSpy for the specified dialog. 0 means "use the topmost dialog".

In most cases it is more useful to use the static showDialogSpy() method rather than create this dialog directly.

Definition at line 128 of file YDialogSpy.cc.

Here is the call graph for this function:



3.43.2.2 YDialogSpy::~YDialogSpy ( ) [protected],[virtual]

Destructor.

Definition at line 163 of file YDialogSpy.cc.

Here is the call graph for this function:

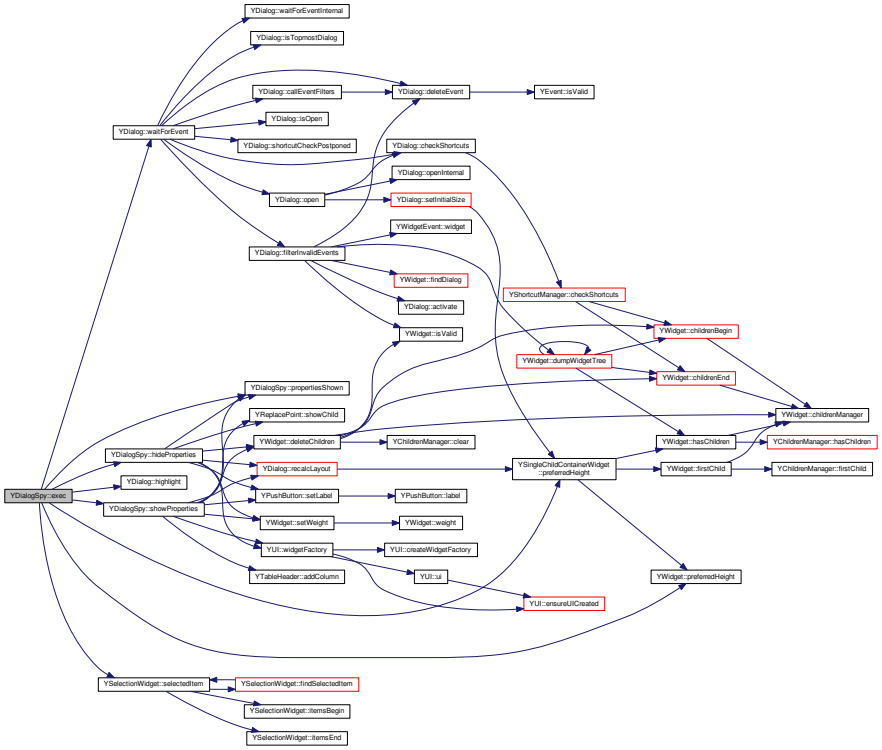
3.43.3 Member Function Documentation

3.43.3.1 void YDialogSpy::exec ( ) [protected]

Execute the event loop. This will only return when the user closes the YDialogSpy dialog.

Definition at line 289 of file YDialogSpy.cc.

Here is the call graph for this function:

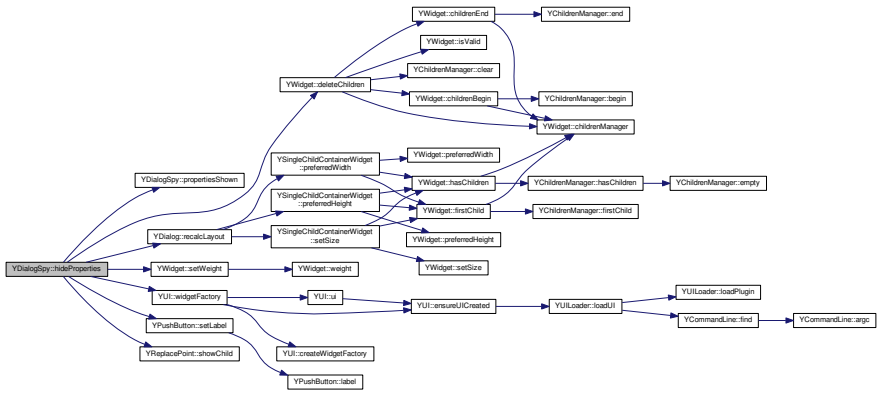


3.43.3.2 void YDialogSpy::hideProperties ( )

Hide the "Properties" sub-window.

Definition at line 202 of file YDialogSpy.cc.

Here is the call graph for this function:



3.43.3.3 bool YDialogSpy::propertiesShown ( ) const

Return 'true' if the "Properties" sub-window is currently shown, 'false' if not.

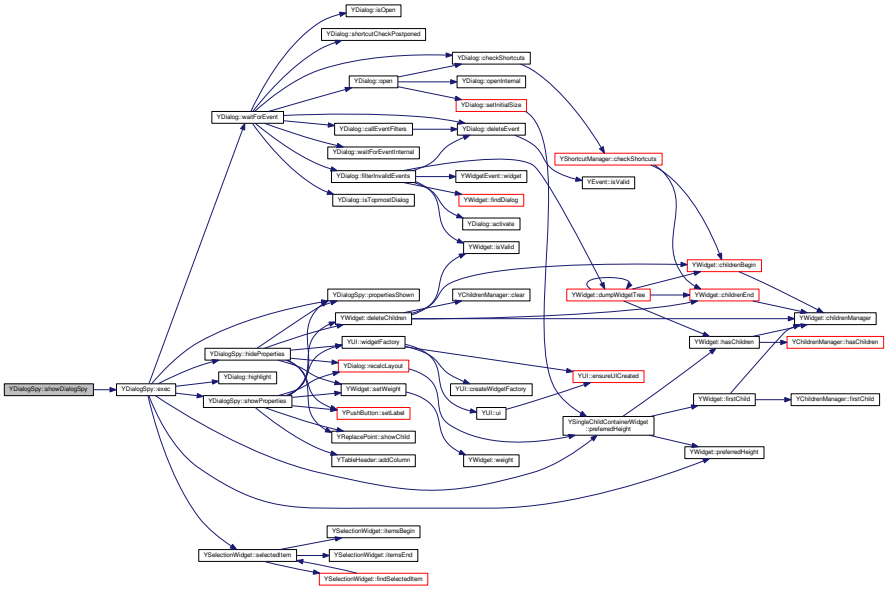
Definition at line 170 of file YDialogSpy.cc.

3.43.3.4 void YDialogSpy::showDialogSpy (YDialog * *dialog* = 0) [static]

Show a YDialogSpy for the specified dialog. 0 means "use the topmost dialog". This will return only when the user closes the YDialogSpy dialog.

Definition at line 336 of file YDialogSpy.cc.

Here is the call graph for this function:

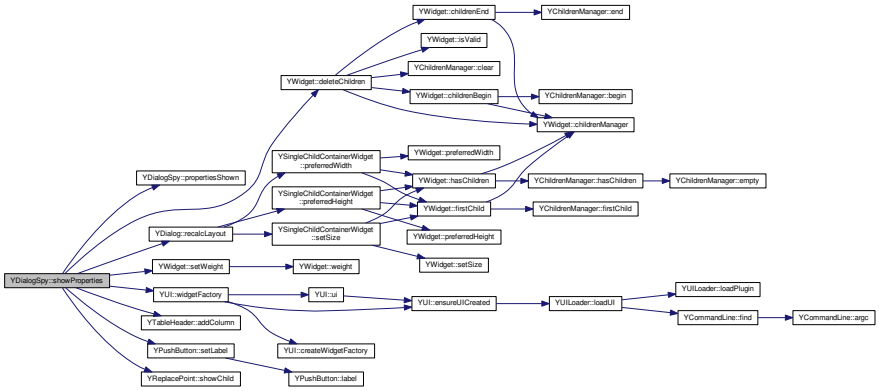


3.43.3.5 void YDialogSpy::showProperties ( )

Show the "Properties" sub-window.

Definition at line 176 of file YDialogSpy.cc.

Here is the call graph for this function:

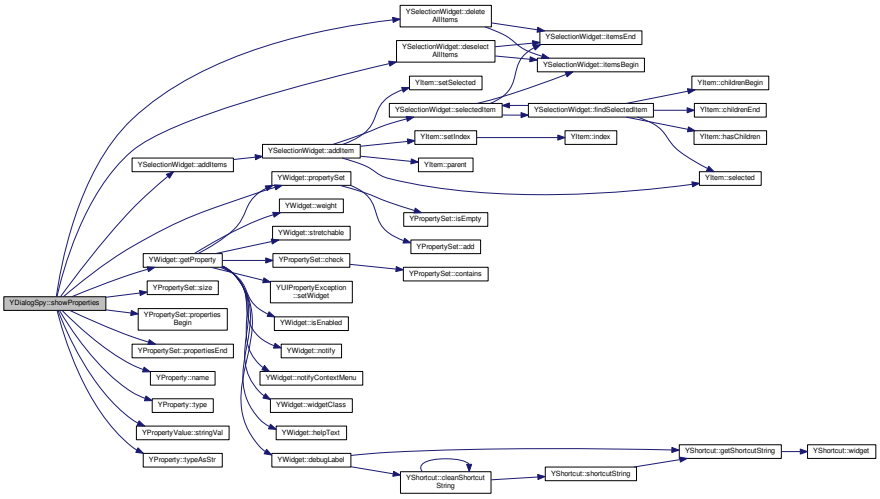


3.43.3.6 void YDialogSpy::showProperties (YWidget * *widget*) [protected]

Show the properties of the specified widget if the "Properties" sub-window is currently shown.

Definition at line 218 of file YDialogSpy.cc.

Here is the call graph for this function:

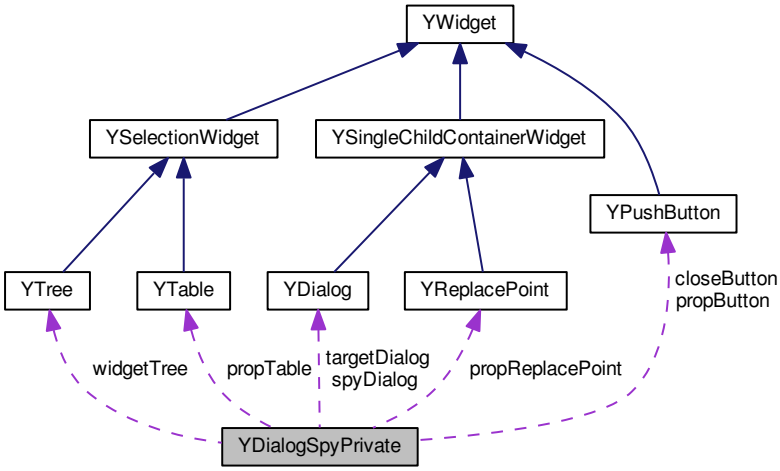


The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDialogSpy.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDialogSpy.cc

3.44 YDialogSpyPrivate Struct Reference

Collaboration diagram for YDialogSpyPrivate:



Public Attributes

- YDialog * **targetDialog**
- YDialog * **spyDialog**
- YTree * **widgetTree**
- YPushButton * **propButton**
- YReplacePoint * **propReplacePoint**
- YTable * **propTable**
- YPushButton * **closeButton**

3.44.1 Detailed Description

Definition at line 105 of file YDialogSpy.cc.

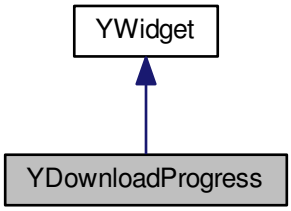
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDialogSpy.cc

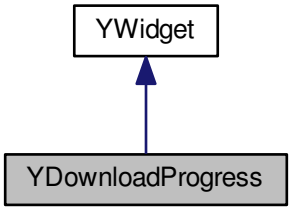
3.45 YDownloadProgress Class Reference

```
#include <YDownloadProgress.h>
```

Inheritance diagram for YDownloadProgress:



Collaboration diagram for YDownloadProgress:



Public Member Functions

- virtual ~YDownloadProgress ()
- virtual const char * widgetClass () const
- std::string label () const
- virtual void setLabel (const std::string &label)
- std::string filename () const
- virtual void setFilename (const std::string &filename)
- YFileSize_t expectedSize () const
- virtual void setExpectedSize (YFileSize_t newSize)
- virtual YFileSize_t currentFileSize () const
- int currentPercent () const
- int value () const
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Protected Member Functions

- YDownloadProgress (YWidget *parent, const std::string &label, const std::string &filename, YFileSize_t expectedSize)

3.45.1 Detailed Description

DownloadProgress: A progress bar that monitors downloading a file by repeatedly polling its size up to its expected size.

Definition at line 37 of file YDownloadProgress.h.

3.45.2 Constructor & Destructor Documentation

3.45.2.1 YDownloadProgress::YDownloadProgress (YWidget * *parent*, const std::string & *label*, const std::string & *filename*, YFileSize_t *expectedSize*) [protected]

Constructor.

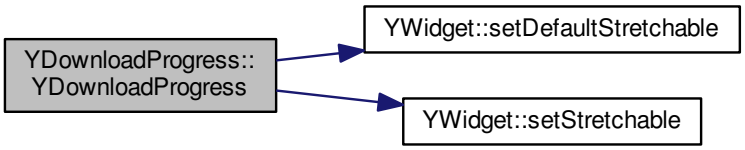
'label' is the label above the progress bar.

'filename' is the name (with path) of the file being monitored.

'expectedSize' is the expected size of the file in bytes.

Definition at line 52 of file YDownloadProgress.cc.

Here is the call graph for this function:



3.45.2.2 YDownloadProgress::~YDownloadProgress ( ) [virtual]

Destructor.

Definition at line 66 of file YDownloadProgress.cc.

3.45.3 Member Function Documentation

3.45.3.1 YFileSize_t YDownloadProgress::currentFileSize ( ) const [virtual]

Return the current size of the file that is being downloaded or 0 if this file doesn't exist (yet).

This default implementation returns the 'st_size' field of a stat() system call on the file. This should be useful for most implementations.

Definition at line 130 of file YDownloadProgress.cc.

3.45.3.2 int YDownloadProgress::currentPercent ( ) const

Return the percentage (0..100) of the file being downloaded so far.

Definition at line 115 of file YDownloadProgress.cc.

Here is the call graph for this function:

3.45.3.3 YFileSize_t YDownloadProgress::expectedSize ( ) const

Return the expected file size.

Definition at line 101 of file YDownloadProgress.cc.

3.45.3.4 std::string YDownloadProgress::filename ( ) const

Return the name of the file that is being monitored.

Definition at line 87 of file YDownloadProgress.cc.

3.45.3.5 YPropertyValue YDownloadProgress::getProperty (const std::string & *propertyName*) [virtual]

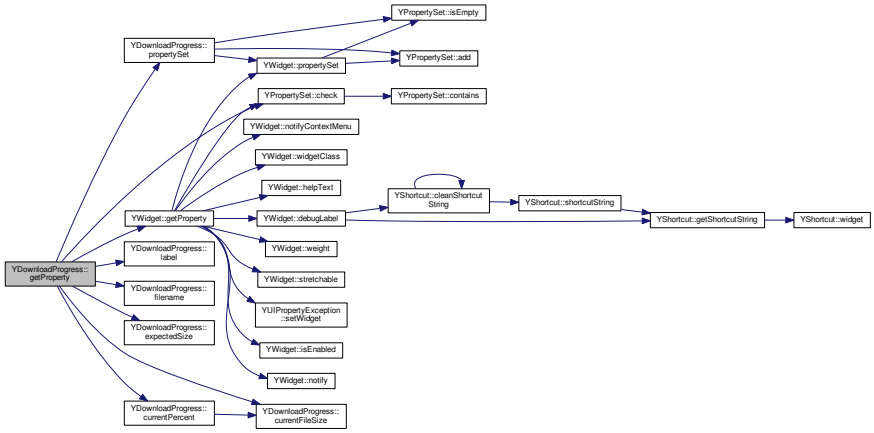
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 185 of file YDownloadProgress.cc.

Here is the call graph for this function:



3.45.3.6 std::string YDownloadProgress::label ( ) const

Get the label (the text above the progress bar).

Definition at line 73 of file YDownloadProgress.cc.

3.45.3.7 const YPropertySet & YDownloadProgress::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 142 of file YDownloadProgress.cc.

Here is the call graph for this function:

3.45.3.8 void YDownloadProgress::setExpectedSize (YFileSize_t *newSize*) [virtual]

Set the expected file size.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 108 of file YDownloadProgress.cc.

3.45.3.9 void YDownloadProgress::setFilename (const std::string & *filename*) [virtual]

Set the name of a new file to monitor.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 94 of file YDownloadProgress.cc.

Here is the call graph for this function:

3.45.3.10 void YDownloadProgress::setLabel (const std::string & *label*) [virtual]

Set the label (the text above the progress bar).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 80 of file YDownloadProgress.cc.

Here is the call graph for this function:

3.45.3.11 bool YDownloadProgress::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

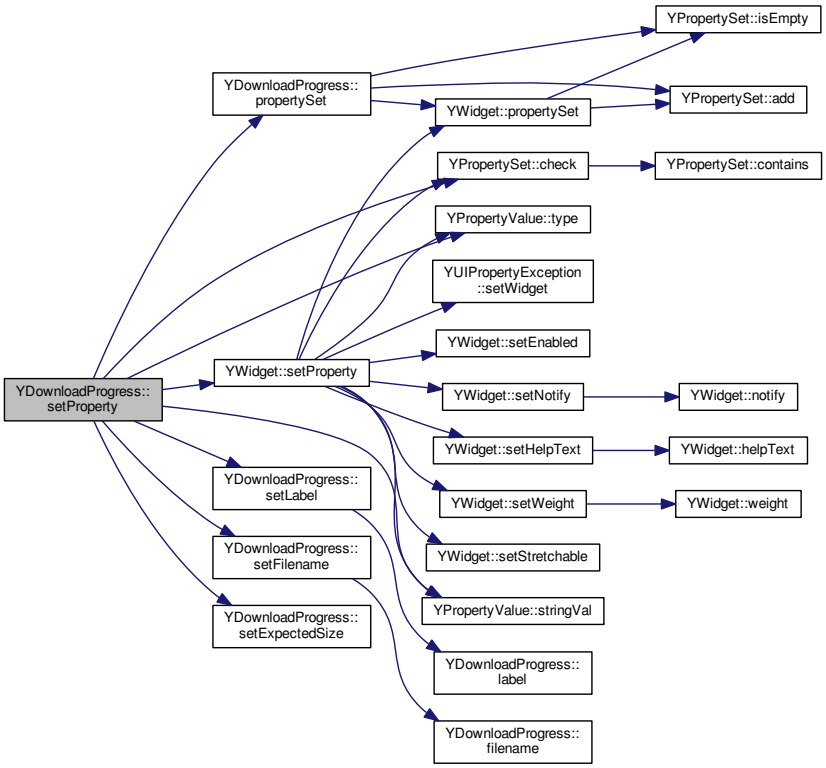
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 168 of file YDownloadProgress.cc.

Here is the call graph for this function:

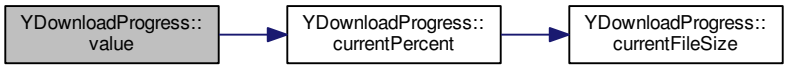


3.45.3.12 int YDownloadProgress::value ( ) const [inline]

Alias for currentPercent().

Definition at line 121 of file YDownloadProgress.h.

Here is the call graph for this function:



3.45.3.13 virtual const char* YDownloadProgress::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 63 of file YDownloadProgress.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDownloadProgress.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDownloadProgress.cc

3.46 YDownloadProgressPrivate Struct Reference

Public Member Functions

- **YDownloadProgressPrivate** (const std::string &label, const std::string &filename, YFileSize_t expectedSize)

Public Attributes

- std::string **label**
- std::string **filename**
- YFileSize_t **expectedSize**

3.46.1 Detailed Description

Definition at line 36 of file YDownloadProgress.cc.

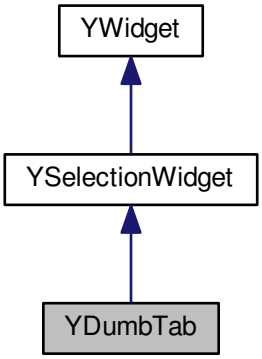
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDownloadProgress.cc

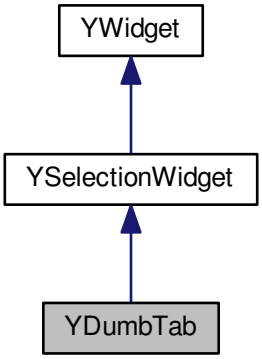
3.47 YDumbTab Class Reference

```
#include <YDumbTab.h>
```

Inheritance diagram for YDumbTab:



Collaboration diagram for YDumbTab:



Public Member Functions

- virtual ~YDumbTab ()
- virtual const char * widgetClass () const
- virtual void addItem (YItem *item)
- virtual void shortcutChanged ()
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)
- virtual bool stretchable (YUIDimension dim) const
- virtual std::string debugLabel () const

Protected Member Functions

- YDumbTab (YWidget *parent)

3.47.1 Detailed Description

DumbTab: A very simple tab widget that can display and switch between a number of tabs, but will only deliver the "user clicked on tab " event very much like a PushButton does. Actually exchanging the content of the tab is left to the application.

DumbTab accepts a single child widget.

Definition at line 40 of file YDumbTab.h.

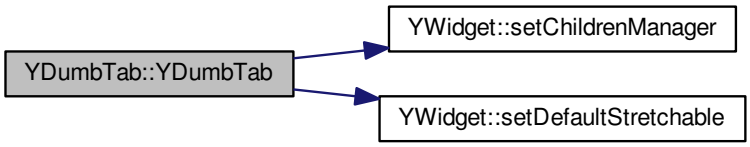
3.47.2 Constructor & Destructor Documentation

3.47.2.1 YDumbTab::YDumbTab (YWidget * *parent*) [protected]

Constructor.

Definition at line 45 of file YDumbTab.cc.

Here is the call graph for this function:



3.47.2.2 YDumbTab::~YDumbTab ( ) [virtual]

Destructor.

Definition at line 59 of file YDumbTab.cc.

3.47.3 Member Function Documentation

3.47.3.1 void YDumbTab::addItem (YItem * *item*) [virtual]

Add an item (a tab page).

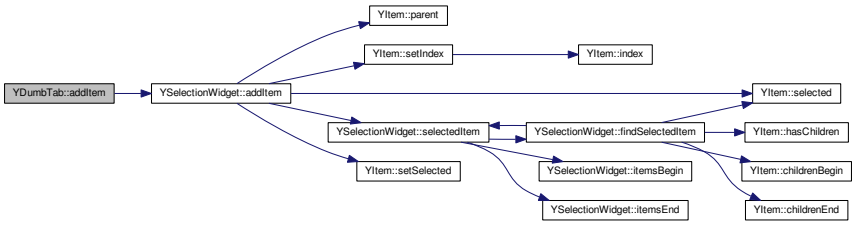
Reimplemented from YSelectionWidget.

Derived classes can overwrite this function, but they should call this base class function in the new implementation.

Reimplemented from YSelectionWidget.

Definition at line 66 of file YDumbTab.cc.

Here is the call graph for this function:



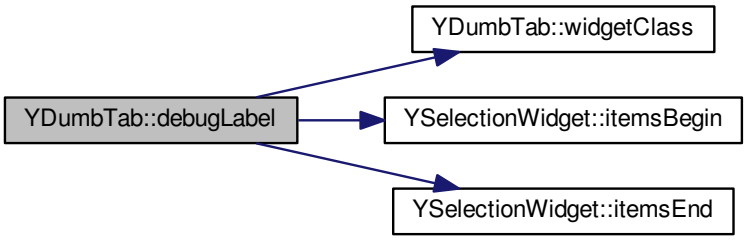
3.47.3.2 std::string YDumbTab::debugLabel ( ) const [virtual]

Descriptive label for debugging. Derived from this widget's only child (if there is one).

Reimplemented from YWidget.

Definition at line 83 of file YDumbTab.cc.

Here is the call graph for this function:



3.47.3.3 YPropertyValue YDumbTab::getProperty (const std::string & *propertyName*) [virtual]

Get a property. Reimplemented from YWidget.

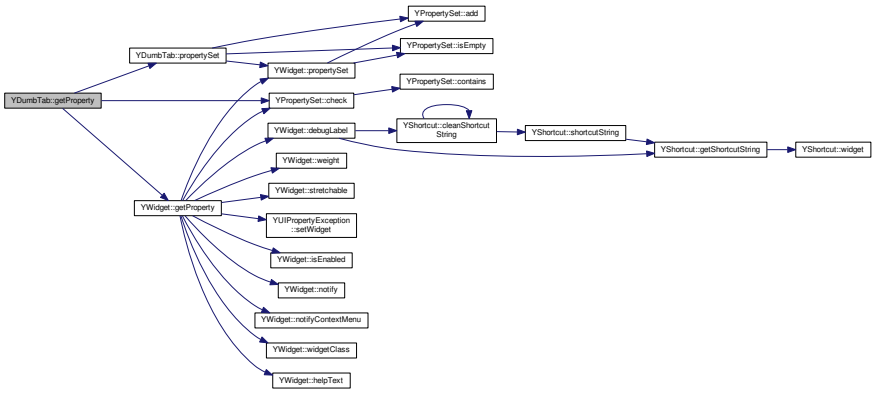
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from YWidget.

Definition at line 139 of file YDumbTab.cc.

Here is the call graph for this function:



3.47.3.4 const YPropertySet & YDumbTab::propertySet ( ) [virtual]

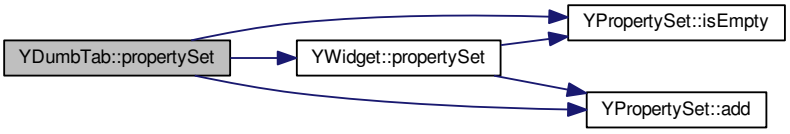
Return this class's property set. This also initializes the property set upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 100 of file YDumbTab.cc.

Here is the call graph for this function:



3.47.3.5 bool YDumbTab::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

This method may throw exceptions, for example

- if there is no property with that name

- if the expected type and the type mismatch

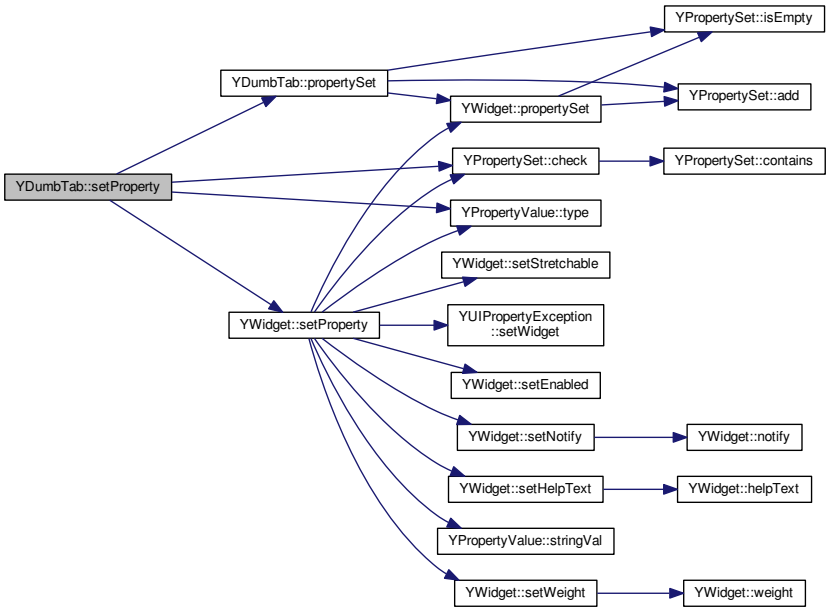
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 122 of file YDumbTab.cc.

Here is the call graph for this function:



3.47.3.6 virtual void YDumbTab::setShortcutString (const std::string & *str*) [inline], [virtual]

Set the string of this widget that holds the keyboard shortcut. Since YDumbTab doesn't have a shortcut for the widget itself (only for the tab pages, i.e. the items), this will simply trigger a shortcutChanged() notification.

Reimplemented from YSelectionWidget.

Reimplemented from YSelectionWidget.

Definition at line 130 of file YDumbTab.h.

Here is the call graph for this function:

3.47.3.7 virtual void YDumbTab::shortcutChanged ( ) [inline], [virtual]

Notification that any shortcut of any item was changed by the shortcut conflict manager.

Derived classes should reimplement this.

Definition at line 76 of file YDumbTab.h.

3.47.3.8 virtual std::string YDumbTab::shortcutString ( ) const [inline], [virtual]

Get the string of this widget that holds the keyboard shortcut. Notice that since YDumbTab has one shortcut for each tab page (for each item), this is not meaningful for this widget class.

Check YItemShortcut in YShortcut.{cc,h} for more details.

Reimplemented from YSelectionWidget.

Reimplemented from YSelectionWidget.

Definition at line 120 of file YDumbTab.h.

3.47.3.9 bool YDumbTab::stretchable (YUIDimension *dim*) const [virtual]

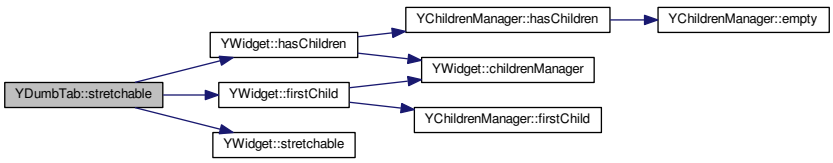
Returns 'true' if this widget is stretchable in the specified dimension. In this case, the stretchability of the single child is returned.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 73 of file YDumbTab.cc.

Here is the call graph for this function:



3.47.3.10 virtual const char* YDumbTab::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YSelectionWidget.

Definition at line 58 of file YDumbTab.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDumbTab.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDumbTab.cc

3.48 YDumbTabPrivate Struct Reference

Public Attributes

- bool **dummy**

3.48.1 Detailed Description

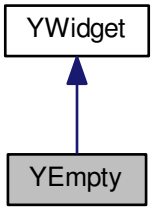
Definition at line 34 of file YDumbTab.cc.

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

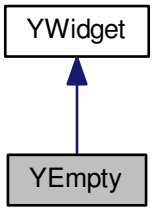
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDumbTab.cc

3.49 YEmpty Class Reference

Inheritance diagram for YEmpty:



Collaboration diagram for YEmpty:



Public Member Functions

- virtual ~YEmpty ()
- virtual const char * widgetClass () const
- virtual int preferredWidth ()
- virtual int preferredHeight ()

Protected Member Functions

- YEmpty (YWidget *parent)

3.49.1 Detailed Description

Definition at line 35 of file YEmpty.h.

3.49.2 Constructor & Destructor Documentation

3.49.2.1 YEmpty::YEmpty (YWidget * *parent*) [protected]

Constructor.

Definition at line 36 of file YEmpty.cc.

3.49.2.2 YEmpty::~YEmpty ( ) [virtual]

Destructor.

Definition at line 44 of file YEmpty.cc.

3.49.3 Member Function Documentation

3.49.3.1 int YEmpty::preferredHeight ( ) [virtual]

Preferred height of the widget.

Reimplemented from YWidget.

Implements YWidget.

Definition at line 56 of file YEmpty.cc.

3.49.3.2 int YEmpty::preferredWidth ( ) [virtual]

Preferred width of the widget.

Reimplemented from YWidget.

Implements YWidget.

Definition at line 50 of file YEmpty.cc.

3.49.3.3 virtual const char* YEmpty::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 53 of file YEmpty.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEmpty.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEmpty.cc

3.50 YEmptyPrivate Struct Reference

Public Attributes

- bool **dummy**

3.50.1 Detailed Description

Definition at line 28 of file YEmpty.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEmpty.cc

3.51 YEnvVar Class Reference

```
#include <YEnvVar.h>
```

Public Member Functions

- YEnvVar (const std::string &name=std::string())
- std::string name () const
- bool isSet () const
- std::string value () const
- bool isEqual (const std::string &str, bool caseSensitive=false) const
- bool operator== (const std::string &str) const
- bool contains (const std::string &str, bool caseSensitive=false) const

3.51.1 Detailed Description

Helper class to represent an environment variable and its value.

Definition at line 36 of file YEnvVar.h.

3.51.2 Constructor & Destructor Documentation

3.51.2.1 YEnvVar::YEnvVar (const std::string & *name* = `std::string()`)

Constructor: Retrieve the environment variable 'name' and store the value (unless 'name' is empty).

Definition at line 36 of file YEnvVar.cc.

3.51.3 Member Function Documentation

3.51.3.1 bool YEnvVar::contains (const std::string & *str*, bool *caseSensitive* = `false`) const

Return 'true' if the environment variable is set and the value contains 'str'.

Definition at line 66 of file YEnvVar.cc.

3.51.3.2 bool YEnvVar::isEqual (const std::string & *str*, bool *caseSensitive* = `false`) const

Return 'true' if the environment variable is set and the value is 'str'.

Definition at line 54 of file YEnvVar.cc.

3.51.3.3 bool YEnvVar::isSet ( ) const `[inline]`

Return 'true' if the environment variable is set.

Definition at line 54 of file YEnvVar.h.

3.51.3.4 std::string YEnvVar::name ( ) const `[inline]`

Return the name of the environment variable.

Definition at line 49 of file YEnvVar.h.

3.51.3.5 bool YEnvVar::operator== (const std::string & *str*) const [inline]

Case-insensitive comparison (shortcut for isEqual()): Return 'true' if the environment variable is set and the value is 'str'.

Definition at line 72 of file YEnvVar.h.

Here is the call graph for this function:

3.51.3.6 std::string YEnvVar::value ( ) const [inline]

Return the value of the environment variable.

Definition at line 59 of file YEnvVar.h.

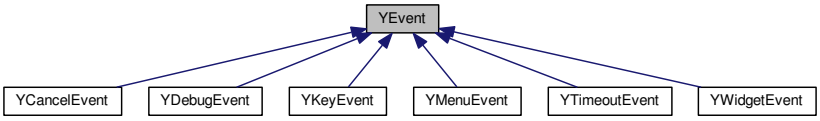
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEnvVar.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEnvVar.cc

3.52 YEvent Class Reference

```
#include <YEvent.h>
```

Inheritance diagram for YEvent:



Public Types

- enum **EventType** {
  NoEvent = 0, **UnknownEvent**, **WidgetEvent**, **MenuEvent**,
  KeyEvent, **CancelEvent**, **TimeoutEvent**, **DebugEvent**,
  InvalidEvent = 0x4242 }

- enum **EventReason** {
  UnknownReason = 0, **Activated**, **SelectionChanged**, **ValueChanged**,
  ContextMenuActivated }

Public Member Functions

- YEvent (EventType eventType=UnknownEvent)
- EventType eventType () const
- unsigned long serial () const
- virtual YWidget * widget () const
- virtual YItem * item () const
- YDialog * dialog () const
- bool isValid () const

Static Public Member Functions

- static const char * toString (EventType eventType)
- static const char * toString (EventReason reason)

Protected Member Functions

- void setDialog (YDialog *dia)
- virtual ~YEvent ()
- void invalidate ()

Friends

- void **YDialog::deleteEvent** (YEvent *event)
- void **YSimpleEventHandler::deleteEvent** (YEvent *event)

3.52.1 Detailed Description

Abstract base class for events to be returned upon UI::UserInput() and related functions.

Definition at line 43 of file YEvent.h.

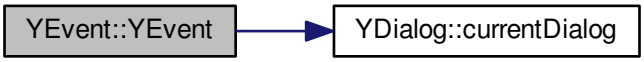
3.52.2 Constructor & Destructor Documentation

3.52.2.1 YEvent::YEvent (EventType *eventType* = `UnknownEvent`)

Constructor.

Definition at line 38 of file YEvent.cc.

Here is the call graph for this function:



3.52.2.2 YEvent::~YEvent ( ) [protected],[virtual]

Protected destructor - events can only be deleted via YDialog::deleteEvent(). The associated dialog will take care of this event and delete it when appropriate.

This desctructor is virtual to force a polymorph object so dynamic_cast<> can be used.

Definition at line 46 of file YEvent.cc.

Here is the call graph for this function:

3.52.3 Member Function Documentation

3.52.3.1 YDialog* YEvent::dialog ( ) const [inline]

Return the dialog this event belongs to or 0 if no dialog was set yet.

Definition at line 106 of file YEvent.h.

3.52.3.2 EventType YEvent::eventType ( ) const [inline]

Returns the event type.

Definition at line 79 of file YEvent.h.

3.52.3.3 void YEvent::invalidate ( ) [protected]

Mark this event as invalid. This cannot be undone.

Definition at line 60 of file YEvent.cc.

3.52.3.4 bool YEvent::isValid ( ) const

Check if this event is valid. Events become invalid in the destructor.

Definition at line 53 of file YEvent.cc.

3.52.3.5 virtual YItem* YEvent::item ( ) const [inline], [virtual]

Return the YItem that corresponds to this event or 0 if there is none.

This default implementation always returns 0. Subclasses that actually return items should overwrite this method.

Reimplemented in YMenuEvent.

Definition at line 101 of file YEvent.h.

3.52.3.6 unsigned long YEvent::serial ( ) const [inline]

Returns the unique serial no. of this event. This is mainly useful for debugging.

Definition at line 85 of file YEvent.h.

3.52.3.7 void YEvent::setDialog (YDialog * *dia*) [inline], [protected]

Set the dialog this event belongs to.

Definition at line 129 of file YEvent.h.

3.52.3.8 const char * YEvent::toString (EventType *eventType*) [static]

Returns the character representation of an event type.

Definition at line 67 of file YEvent.cc.

3.52.3.9 const char * YEvent::toString (EventReason *reason*) [static]

Returns the character representation of an event reason.

Definition at line 90 of file YEvent.cc.

3.52.3.10 virtual YWidget* YEvent::widget ( ) const [inline], [virtual]

Returns the widget that caused this event or 0 if there is none.

This default implementation always returns 0. Subclasses that actually return widgets should overwrite this method.

Reimplemented in YWidgetEvent.

Definition at line 93 of file YEvent.h.

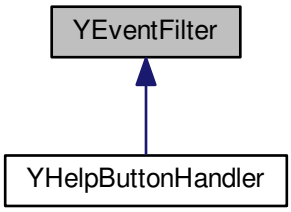
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEvent.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEvent.cc

3.53 YEventFilter Class Reference

```
#include <YEventFilter.h>
```

Inheritance diagram for YEventFilter:



Public Member Functions

- virtual ~YEventFilter ()
- virtual YEvent * filter (YEvent *event)=0
- YDialog * dialog () const

Protected Member Functions

- YEventFilter (YDialog *dialog=0)

3.53.1 Detailed Description

Abstract base class to filter events.

This class can be used to examine events just before they are delivered to the application. This is most useful for higher-level widgets or for libraries that need to react to certain events and either consume them, have them delivered unchanged to the application, or exchange an event with another one.

A YEventFilter belongs to one specific dialog. Each dialog can have any number of event filters. Each of those event filters is called (its YEventFilter::filter() method) for each event inside YDialog::waitForEvent(). The order in which event filters are called is undefined.

YEventFilter objects should be created with 'new' (on the heap). Since an YEventFilter registers itself with its dialog, the dialog will delete it in its destructor if it still exists after all child widgets are deleted.

Thus, it is safe to store a pointer to an YEventFilter until the corresponding dialog is deleted. After that, the pointer becomes invalid.

See YHelpButtonHandler in YDialog.cc for an example.

Definition at line 62 of file YEventFilter.h.

3.53.2 Constructor & Destructor Documentation

3.53.2.1 YEventFilter::YEventFilter (YDialog * *dialog* = 0) [protected]

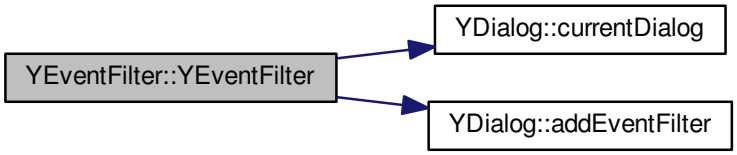
Constructor.

This registers the event filter with the specified dialog. The dialog assumes ownership of this object and will delete it in its destructor (unless this object is destroyed before that time).

If 'dialog' is 0, YDialog::currentDialog() is used (which can throw a YUINoDialogException if there is no dialog).

Definition at line 44 of file YEventFilter.cc.

Here is the call graph for this function:



3.53.2.2 YEventFilter::~YEventFilter ( ) [virtual]

Destructor.

This will unregister this object with its dialog.

Definition at line 56 of file YEventFilter.cc.

Here is the call graph for this function:

3.53.3 Member Function Documentation

3.53.3.1 YDialog * YEventFilter::dialog ( ) const

Return the dialog this event filter belongs to.

Definition at line 63 of file YEventFilter.cc.

3.53.3.2 virtual YEvent* YEventFilter::filter (YEvent * *event*) [pure virtual]

The heart of the matter: The event filter function. Derived classes are required to implement this.

This method can inspect the event it receives. Hint: event->widget() is typically the most interesting information.

This method can react on individual events and

- consume the event (i.e., return 0)
- pass the event through unchanged (simply return the event)
- create a new event (typically based on data in the received event).

If 0 or a new event (another value than 'event') is returned, the old event is deleted. If a value different from 'event' or 0 is returned, that value is assumed to be a pointer to a newly created event. The dialog will assume ownership of that event and delete it when appropriate.

Note: Never delete 'event' in this method! Return 0 or a new event instead; the caller will take care of deleting the old event.

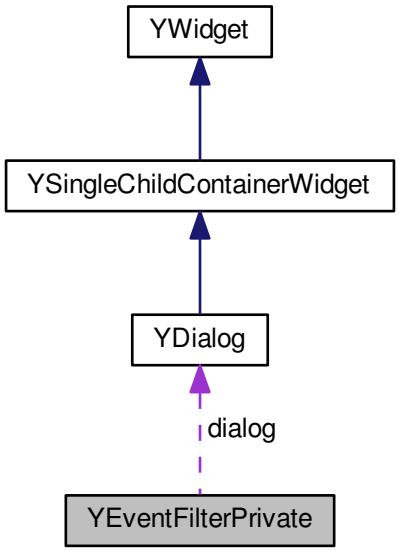
Implemented in YHelpButtonHandler.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEventFilter.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEventFilter.cc

3.54 YEventFilterPrivate Struct Reference

Collaboration diagram for YEventFilterPrivate:



Public Member Functions

- **YEventFilterPrivate** (YDialog *dialog)

Public Attributes

- YDialog * **dialog**

3.54.1 Detailed Description

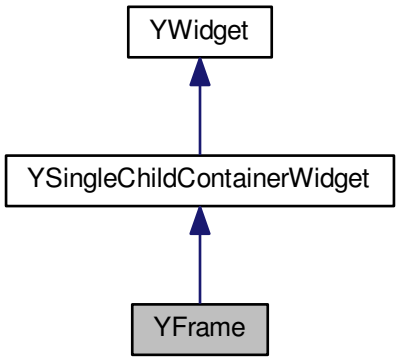
Definition at line 32 of file YEventFilter.cc.

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

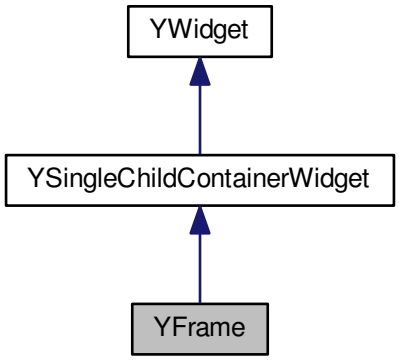
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEventFilter.cc

3.55 YFrame Class Reference

Inheritance diagram for YFrame:



Collaboration diagram for YFrame:



Public Member Functions

- virtual ~YFrame ()
- virtual const char * widgetClass () const
- virtual void setLabel (const std::string &newLabel)
- std::string label () const
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Protected Member Functions

- YFrame (YWidget *parent, const std::string &label)

3.55.1 Detailed Description

Definition at line 35 of file YFrame.h.

3.55.2 Constructor & Destructor Documentation

3.55.2.1 YFrame::YFrame (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

Definition at line 45 of file YFrame.cc.

3.55.2.2 YFrame::~YFrame ( ) [virtual]

Destructor.

Definition at line 53 of file YFrame.cc.

3.55.3 Member Function Documentation

3.55.3.1 YPropertyValue YFrame::getProperty (const std::string & *propertyName*) [virtual]

Get a property. Reimplemented from YWidget.

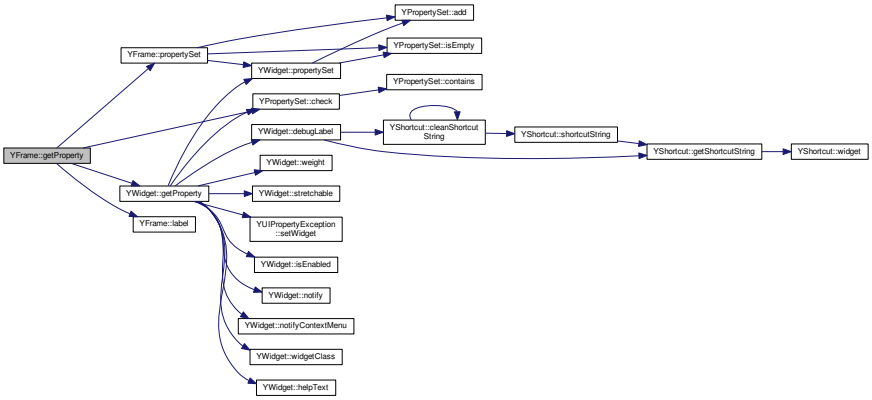
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from YWidget.

Definition at line 106 of file YFrame.cc.

Here is the call graph for this function:



3.55.3.2 std::string YFrame::label ( ) const

Get the current frame label.

Definition at line 65 of file YFrame.cc.

3.55.3.3 const YPropertySet & YFrame::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property set upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 72 of file YFrame.cc.

Here is the call graph for this function:

3.55.3.4 void YFrame::setLabel (const std::string & *newLabel*) [virtual]

Change the frame label.

Derived classes should overwrite this, but call this base class function in the overwritten function.

Definition at line 59 of file YFrame.cc.

Here is the call graph for this function:

3.55.3.5 bool YFrame::setProperty (const std::string & *propertyName,* const **YPropertyValue** & *val*) [virtual]

Set a property. Reimplemented from YWidget.

This method may throw exceptions, for example

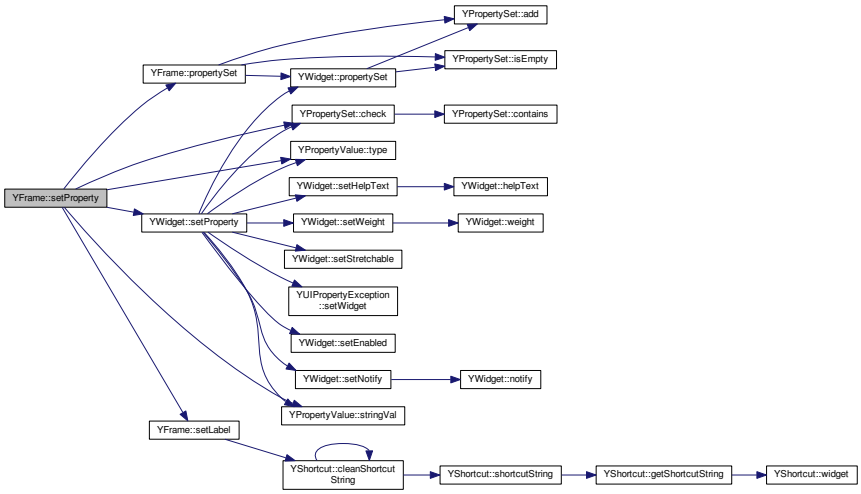
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 91 of file YFrame.cc.

Here is the call graph for this function:



3.55.3.6 virtual const char* YFrame::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 53 of file YFrame.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YFrame.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YFrame.cc

3.56 YFramePrivate Struct Reference

Public Member Functions

- **YFramePrivate** (const std::string &frameLabel)

Public Attributes

- std::string **label**

3.56.1 Detailed Description

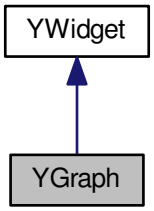
Definition at line 33 of file YFrame.cc.

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

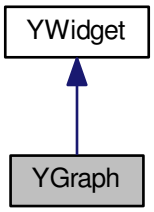
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YFrame.cc

3.57 YGraph Class Reference

Inheritance diagram for YGraph:



Collaboration diagram for YGraph:



Public Member Functions

- virtual ~YGraph ()
- virtual const char * widgetClass () const
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- std::string filename () const
- virtual void setFilename (const std::string &filename)
- std::string layoutAlgorithm () const
- virtual void setLayoutAlgorithm (const std::string &filename)
- virtual void setGraph (void *graph)
- virtual std::string activatedNode () const

Protected Member Functions

- YGraph (YWidget *parent, const std::string &filename, const std::string &layoutAlgorithm)
- YGraph (YWidget *parent, void *graph)
- virtual void renderGraph (const std::string &filename, const std::string &layoutAlgorithm)=0
- virtual void renderGraph (void *graph)=0

3.57.1 Detailed Description

Definition at line 43 of file YGraph.h.

3.57.2 Constructor & Destructor Documentation

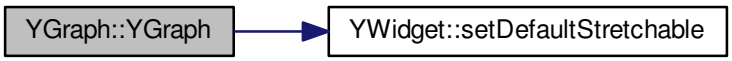
3.57.2.1 YGraph::YGraph (YWidget * *parent*, const std::string & *filename*, const std::string & *layoutAlgorithm*) [protected]

Constructor.

Loads a graph in DOT format from filename and uses the layout algorithm layoutAlgorithm to layout and then render the graph. The layout algorithm can be any string accepted by the function gvLayout from graphviz, e.g. "dot" or "neato".

Definition at line 44 of file YGraph.cc.

Here is the call graph for this function:



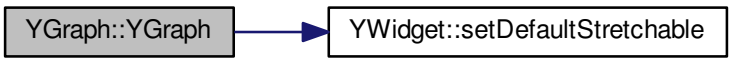
3.57.2.2 YGraph::YGraph (YWidget * *parent*, void * *graph*) [protected]

Constructor.

Renders the graph. The graph must already contain layout information.

Definition at line 53 of file YGraph.cc.

Here is the call graph for this function:



3.57.2.3 YGraph::~YGraph ( ) [virtual]

Destructor.

Definition at line 62 of file YGraph.cc.

3.57.3 Member Function Documentation

3.57.3.1 std::string YGraph::activatedNode ( ) const [virtual]

Return name of activated node. Activation can happen due to e.g. single right mouse click (context menu) or double left mouse click.

Definition at line 106 of file YGraph.cc.

3.57.3.2 std::string YGraph::filename ( ) const

Return the filename that describes the graph.

Definition at line 69 of file YGraph.cc.

3.57.3.3 YPropertyValue YGraph::getProperty (const std::string & *propertyName*) [virtual]

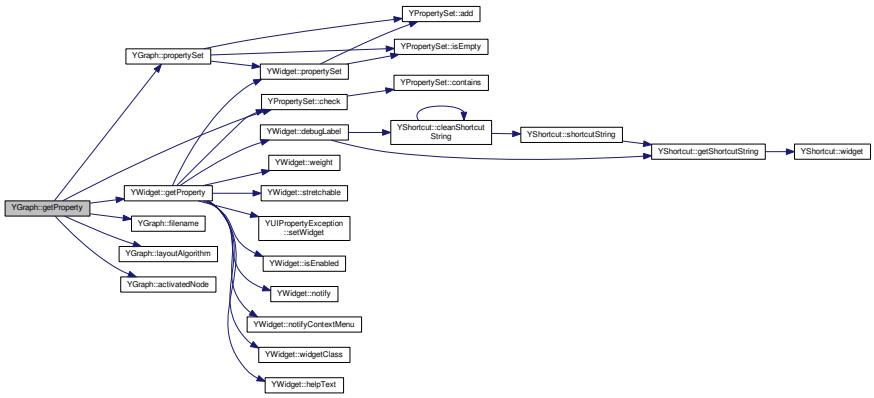
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 151 of file YGraph.cc.

Here is the call graph for this function:



3.57.3.4 std::string YGraph::layoutAlgorithm ( ) const

Return the layout-algorithm used for the graph.

Definition at line 84 of file YGraph.cc.

3.57.3.5 const YPropertySet & YGraph::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 113 of file YGraph.cc.

Here is the call graph for this function:

3.57.3.6 virtual void YGraph::renderGraph (const std::string & *filename,* const std::string & *layoutAlgorithm*) [protected], [pure virtual]

Render the graph from the filename. Derived classes are required to implement this.

3.57.3.7 virtual void YGraph::renderGraph (void * *graph*) [protected], [pure virtual]

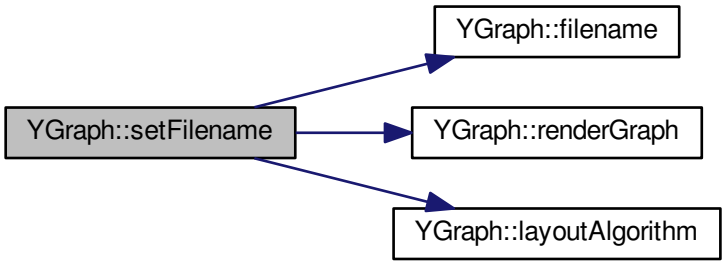
Render the graph. Derived classes are required to implement this.

3.57.3.8 void YGraph::setFilename (const std::string & *filename*) [virtual]

Set the filename that describes the graph and render the graph. Derived classes can reimplent this, but they should call this base class method in the new implementation. Most derived classes only need to implement renderGraph().

Definition at line 76 of file YGraph.cc.

Here is the call graph for this function:

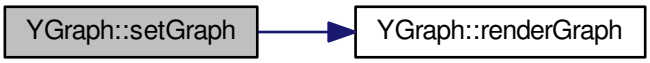


3.57.3.9 void YGraph::setGraph (void * *graph*) [virtual]

Render the graph. Derived classes can reimplent this, but they should call this base class method in the new implementation. Most derived classes only need to implement renderGraph().

Definition at line 91 of file YGraph.cc.

Here is the call graph for this function:



3.57.3.10 void YGraph::setLayoutAlgorithm (const std::string & *filename*) [virtual]

Set the layout-algorithm used for the graph. Derived classes can reimplent this, but they should call this base class method in the new implementation.

Definition at line 99 of file YGraph.cc.

Here is the call graph for this function:

3.57.3.11 bool YGraph::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

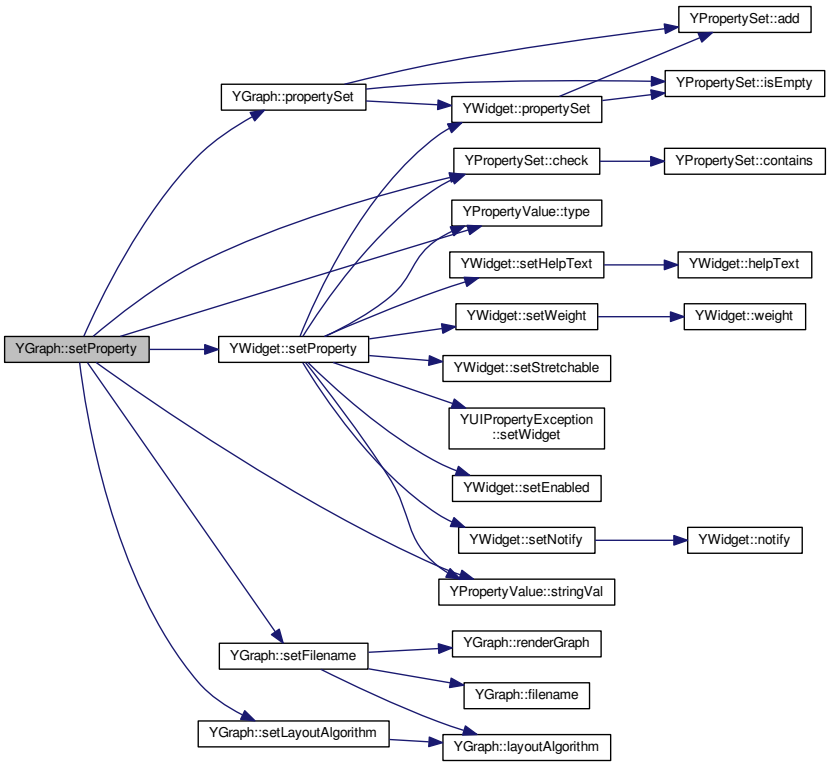
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 135 of file YGraph.cc.

Here is the call graph for this function:



3.57.3.12 virtual const char* YGraph::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 75 of file YGraph.h.

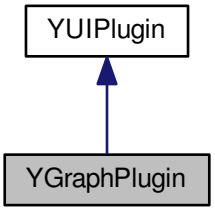
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YGraph.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YGraph.cc

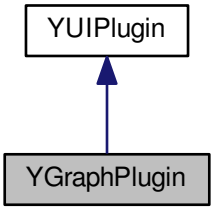
3.58 YGraphPlugin Class Reference

```
#include <YGraphPlugin.h>
```

Inheritance diagram for YGraphPlugin:



Collaboration diagram for YGraphPlugin:



Public Member Functions

- virtual YGraph * createGraph (YWidget *parent, const std::string &filename, const std::string &layoutAlgorithm)=0

Protected Member Functions

- YGraphPlugin (const char *pluginLibBaseName)
- virtual ~YGraphPlugin ()

3.58.1 Detailed Description

Abstract base class for simplified access to UI plugins for graph widget.

Definition at line 37 of file YGraphPlugin.h.

3.58.2 Constructor & Destructor Documentation

3.58.2.1 YGraphPlugin::YGraphPlugin (const char * *pluginLibBaseName*) [inline],[protected]

Constructor: Load the specified plugin library from the standard UI plugin directory (/usr/lib/yui/).

Definition at line 44 of file YGraphPlugin.h.

3.58.2.2 virtual YGraphPlugin::~YGraphPlugin ( ) [inline],[protected],[virtual]

Destructor. Calls dlclose() which will unload the plugin library if it is no longer used, i.e. if the reference count dlopen() uses reaches 0.

Definition at line 51 of file YGraphPlugin.h.

3.58.3 Member Function Documentation

3.58.3.1 virtual YGraph* YGraphPlugin::createGraph (YWidget * *parent*, const std::string & *filename*, const std::string & *layoutAlgorithm*) [pure virtual]

Create a graph widget. Derived classes need to implement this.

This might return 0 if the plugin lib could not be loaded or if the appropriate symbol could not be located in the plugin lib.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YGraphPlugin.h

3.59 YGraphPrivate Struct Reference

Public Member Functions

- **YGraphPrivate** (std::string filename, std::string layoutAlgorithm)

Public Attributes

- std::string **filename**
- std::string **layoutAlgorithm**

3.59.1 Detailed Description

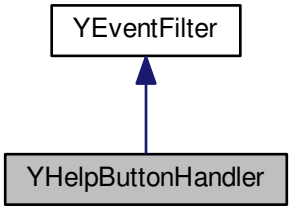
Definition at line 32 of file YGraph.cc.

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

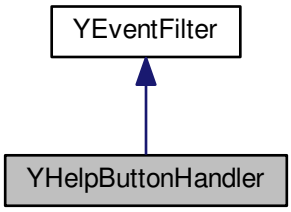
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YGraph.cc

3.60 YHelpButtonHandler Class Reference

Inheritance diagram for YHelpButtonHandler:



Collaboration diagram for YHelpButtonHandler:



Public Member Functions

- **YHelpButtonHandler** (YDialog *dialog)
- YEvent * filter (YEvent *event)

Additional Inherited Members

3.60.1 Detailed Description

Helper class: Event filter that handles "Help" buttons.

Definition at line 80 of file YDialog.cc.

3.60.2 Member Function Documentation

3.60.2.1 YEvent* YHelpButtonHandler::filter (YEvent * *event*) [inline], [virtual]

The heart of the matter: The event filter function. Derived classes are required to implement this.

This method can inspect the event it receives. Hint: event->widget() is typically the most interesting information.

This method can react on individual events and

- consume the event (i.e., return 0)
- pass the event through unchanged (simply return the event)
- create a new event (typically based on data in the received event).

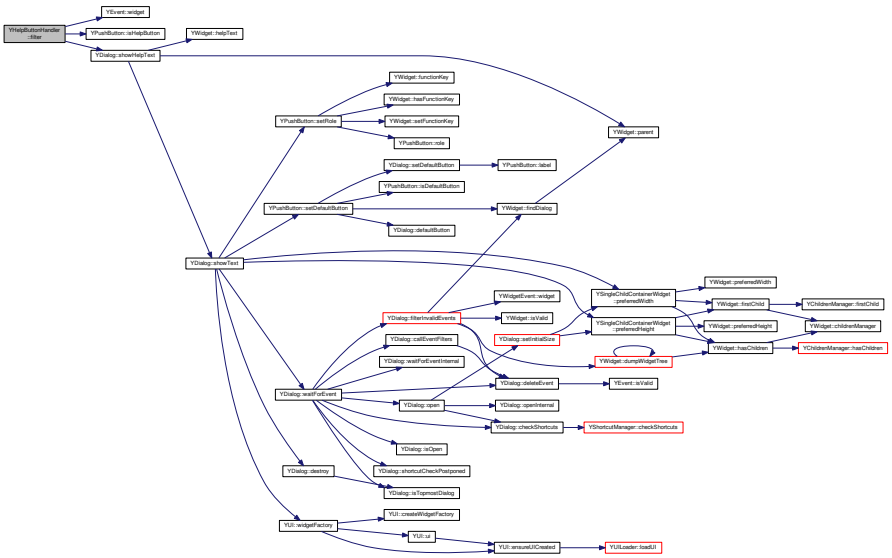
If 0 or a new event (another value than 'event') is returned, the old event is deleted. If a value different from 'event' or 0 is returned, that value is assumed to be a pointer to a newly created event. The dialog will assume ownership of that event and delete it when appropriate.

Note: Never delete 'event' in this method! Return 0 or a new event instead; the caller will take care of deleting the old event.

Implements YEventFilter.

Definition at line 89 of file YDialog.cc.

Here is the call graph for this function:



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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDialog.cc

3.61 YIconLoader Class Reference

Public Member Functions

- std::string **findIcon** (std::string name)
- void **setIconBasePath** (std::string path)
- std::string **iconBasePath** () const
- void **addIconSearchPath** (std::string path)

3.61.1 Detailed Description

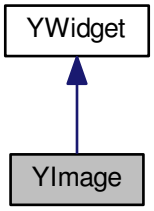
Definition at line 32 of file YIconLoader.h.

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

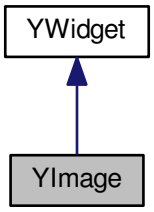
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YIconLoader.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YIconLoader.cc

3.62 YImage Class Reference

Inheritance diagram for YImage:



Collaboration diagram for YImage:



Public Member Functions

- YImage (YWidget *parent, const std::string &imageFileName, bool animated=false)
- virtual ~YImage ()
- virtual const char * widgetClass () const
- std::string imageFileName () const
- bool animated () const
- virtual void setImage (const std::string &imageFileName, bool animated=false)
- void setMovie (const std::string &movieFileName)
- bool hasZeroSize (YUIDimension dim) const
- void setZeroSize (YUIDimension dim, bool zeroSize=true)
- bool autoScale () const
- virtual void setAutoScale (bool autoScale=true)

Additional Inherited Members

3.62.1 Detailed Description

Definition at line 35 of file YImage.h.

3.62.2 Constructor & Destructor Documentation

3.62.2.1 YImage::YImage (YWidget * *parent*, const std::string & *imageFileName*, bool *animated* = `false`)

Constructor.

'animated' indicates if 'imageFileName' is an animated image format (e.g., MNG).

Definition at line 54 of file YImage.cc.

3.62.2.2 YImage::~YImage ( ) `[virtual]`

Destructor.

Definition at line 64 of file YImage.cc.

3.62.3 Member Function Documentation

3.62.3.1 bool YImage::animated ( ) const

Returns 'true' if the current image is an animated image format (e.g., MNG).

Definition at line 76 of file YImage.cc.

3.62.3.2 bool YImage::autoScale ( ) const

Return 'true' if the image should be scaled to fit into the available space.

Definition at line 102 of file YImage.cc.

3.62.3.3 bool YImage::hasZeroSize (YUIDimension *dim*) const

Return 'true' if the image widget should be stretchable with a default width of 0 in the specified dimension. This is useful if the widget width is determined by outside constraints, like the width of a neighbouring widget.

Definition at line 89 of file YImage.cc.

3.62.3.4 std::string YImage::imageFileName ( ) const

Return the file name of this widget's image.

Definition at line 70 of file YImage.cc.

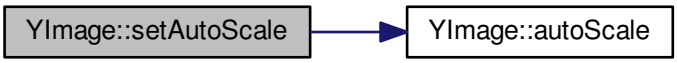
3.62.3.5 void YImage::setAutoScale (bool *autoScale* = `true`) `[virtual]`

Make the image fit into the available space.

Derived classes should overwrite this, but call this base class function in the new function.

Definition at line 108 of file YImage.cc.

Here is the call graph for this function:



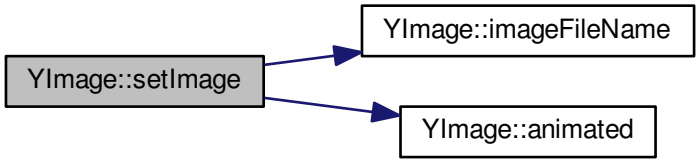
3.62.3.6 void YImage::setImage (const std::string & *imageFileName*, bool *animated* = `false`) `[virtual]`

Set and display a new image (or movie if animated is 'true').

Derived classes should overwrite this, but call this base class function in the new function.

Definition at line 82 of file YImage.cc.

Here is the call graph for this function:



3.62.3.7 void YImage::setMovie (const std::string & *movieFileName*) [inline]

Set and display a movie (an animated image).

Definition at line 81 of file YImage.h.

Here is the call graph for this function:

3.62.3.8 void YImage::setZeroSize (YUIDimension *dim*, bool *zeroSize* = true)

Make the image widget stretchable with a default size of 0 in the specified dimension. This is useful if the widget width is determined by outside constraints, like the width of a neighbouring widget.

This function is intentionally not virtual because it is only relevant during the next geometry update, in which case the derived class has to check this value anyway.

Definition at line 95 of file YImage.cc.

Here is the call graph for this function:

3.62.3.9 virtual const char* YImage::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

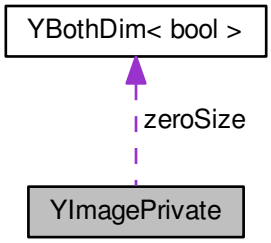
Definition at line 57 of file YImage.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YImage.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YImage.cc

3.63 YImagePrivate Struct Reference

Collaboration diagram for YImagePrivate:



Public Member Functions

- YImagePrivate (const std::string &imageFileName, bool animated)

Public Attributes

- std::string **imageFileName**
- bool **animated**
- YBothDim< bool > **zeroSize**
- bool **autoScale**

3.63.1 Detailed Description

Definition at line 30 of file YImage.cc.

3.63.2 Constructor & Destructor Documentation

3.63.2.1 YImagePrivate::YImagePrivate (const std::string & *imageFileName*, bool *animated*) [inline]

Constructor.

Definition at line 35 of file YImage.cc.

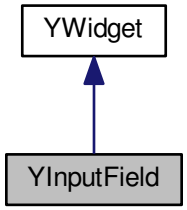
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YImage.cc

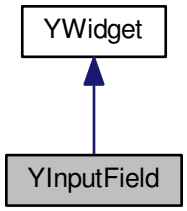
3.64 YInputField Class Reference

```
#include <YInputField.h>
```

Inheritance diagram for YInputField:



Collaboration diagram for YInputField:



Public Member Functions

- virtual ~YInputField ()
- virtual const char * widgetClass () const
- virtual std::string value ()=0
- virtual void setValue (const std::string &text)=0
- std::string label () const
- virtual void setLabel (const std::string &label)
- bool passwordMode () const
- std::string validChars ()
- virtual void setValidChars (const std::string &validChars)
- int inputMaxLength () const
- virtual void setInputMaxLength (int numberOfChars)
- bool shrinkable () const
- virtual void setShrinkable (bool shrinkable=true)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)
- const char * userInputProperty ()
- virtual void saveUserInput (YMacroRecorder *macroRecorder)

Protected Member Functions

- YInputField (YWidget *parent, const std::string &label, bool passwordMode=false)

3.64.1 Detailed Description

InputField: General purpose one line input field for entering text and other data. Can be used for entering passwords with a "*" echoed for every character typed.

Like most widgets, the InputField has a label (a caption) above the input field itself. The label can and should get a keyboard shortcut (specified with '&') that will make the input field receive the keyboard focus with a special key combination ("&Name" -> Alt-N or Ctrl-N will make the keyboard focus jump to the corresponding input field).

Definition at line 46 of file YInputField.h.

3.64.2 Constructor & Destructor Documentation

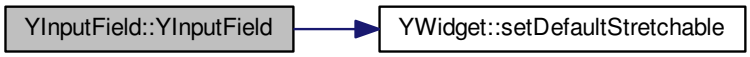
3.64.2.1 YInputField::YInputField (YWidget * *parent*, const std::string & *label*, bool *passwordMode* = `false`) `[protected]`

Constructor.

Create an input field with 'label' as the caption. If 'passwordMode' is set, the input will be not be echoed as clear text.

Definition at line 53 of file YInputField.cc.

Here is the call graph for this function:



3.64.2.2 YInputField::~YInputField ( ) `[virtual]`

Destructor.

Definition at line 64 of file YInputField.cc.

3.64.3 Member Function Documentation

3.64.3.1 YPropertyValue YInputField::getProperty (const std::string & *propertyName*) `[virtual]`

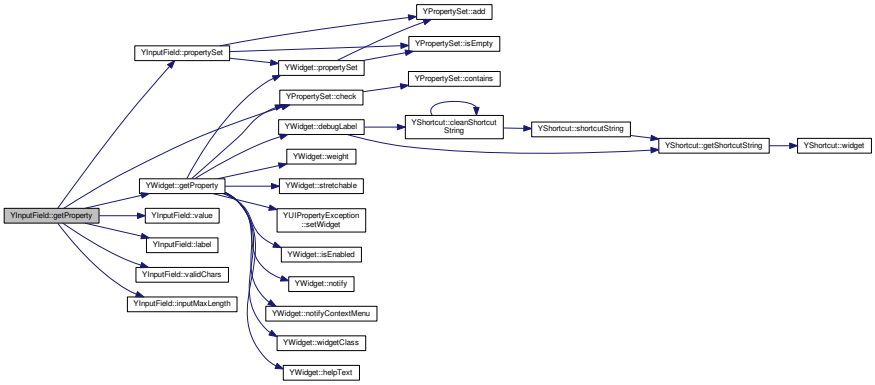
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 168 of file YInputField.cc.

Here is the call graph for this function:



3.64.3.2 int YInputField::inputMaxLength ( ) const

The maximum input length, i.e., the maximum number of characters the user can enter. -1 means no limit.

Definition at line 113 of file YInputField.cc.

3.64.3.3 std::string YInputField::label ( ) const

Get the label (the caption above the input field).

Definition at line 70 of file YInputField.cc.

3.64.3.4 bool YInputField::passwordMode ( ) const

Returns 'true' if this input field is in password mode, i.e. if there should be no on-screen echo or only a '*' for each character typed.

Notice that this can only be set in the constructor.

Definition at line 82 of file YInputField.cc.

3.64.3.5 const YPropertySet & YInputField::propertySet ( ) [virtual]

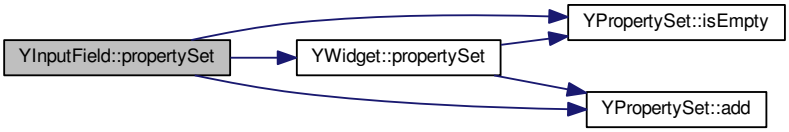
Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 126 of file YInputField.cc.

Here is the call graph for this function:



3.64.3.6 void YInputField::saveUserInput (YMacroRecorder * *macroRecorder*) [virtual]

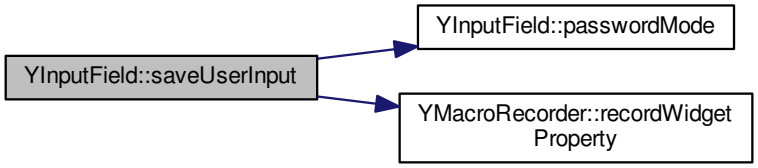
Save the widget's user input to a macro recorder.

Reimplemented from YWidget to avoid recording passwords.

Reimplemented from YWidget.

Definition at line 184 of file YInputField.cc.

Here is the call graph for this function:



3.64.3.7 void YInputField::setInputMaxLength (int *numberOfChars*) [virtual]

Set the maximum input length, i.e., the maximum number of characters the user can enter. -1 means no limit.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 119 of file YInputField.cc.

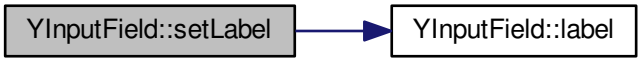
3.64.3.8 void YInputField::setLabel (const std::string & *label*) [virtual]

Set the label (the caption above the input field).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 76 of file YInputField.cc.

Here is the call graph for this function:



3.64.3.9 bool YInputField::setProperty (const std::string & *propertyName,* const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

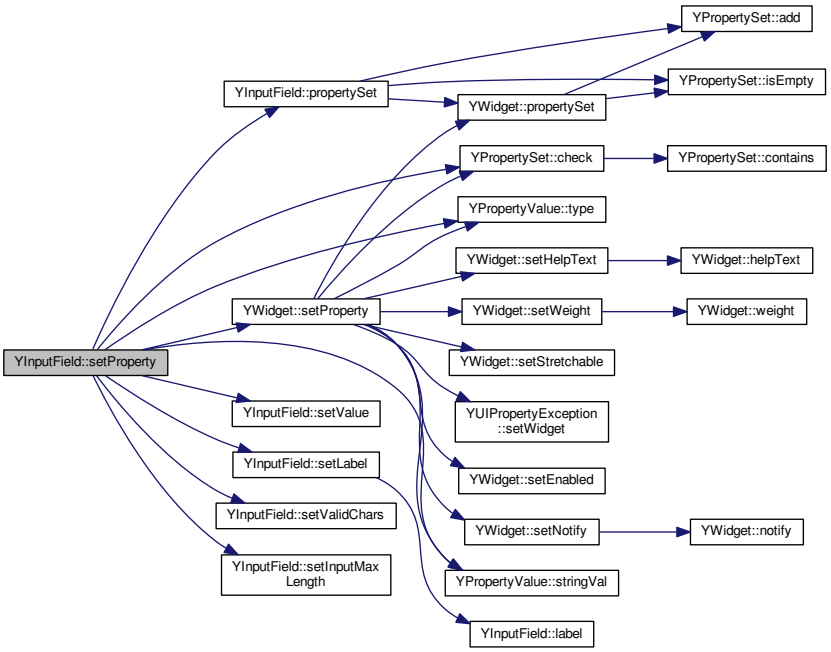
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 150 of file YInputField.cc.

Here is the call graph for this function:



3.64.3.10 virtual void YInputField::setShortcutString (const std::string & *str*) [inline], [virtual]

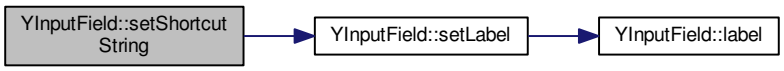
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 193 of file YInputField.h.

Here is the call graph for this function:



3.64.3.11 void YInputField::setShrinkable (bool *shrinkable* = true) [virtual]

Make this InputField very small. This will take effect only upon the next geometry management run.

Derived classes can overwrite this, but should call this base class function in the new function.

Definition at line 94 of file YInputField.cc.

Here is the call graph for this function:

3.64.3.12 void YInputField::setValidChars (const std::string & *validChars*) [virtual]

Set the valid input characters. No input validation is performed (i.e., the user can enter anything) if this is empty.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 107 of file YInputField.cc.

3.64.3.13 virtual void YInputField::setValue (const std::string & *text*) [pure virtual]

Set the current value (the text entered by the user or set from the outside) of this input field.

Derived classes are required to implement this.

3.64.3.14 virtual std::string YInputField::shortcutString ( ) const [inline], [virtual]

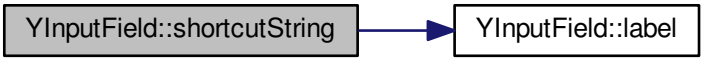
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 186 of file YInputField.h.

Here is the call graph for this function:



3.64.3.15 bool YInputField::shrinkable ( ) const

Return 'true' if this InputField should be very small.

Definition at line 88 of file YInputField.cc.

3.64.3.16 const char* YInputField::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 200 of file YInputField.h.

3.64.3.17 std::string YInputField::validChars ( )

Get the valid input characters. No input validation is performed (i.e., the user can enter anything) if this is empty.

Definition at line 101 of file YInputField.cc.

3.64.3.18 virtual std::string YInputField::value ( ) [pure virtual]

Get the current value (the text entered by the user or set from the outside) of this input field.

Derived classes are required to implement this.

3.64.3.19 const char * YInputField::widgetClass ( ) const [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 194 of file YInputField.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YInputField.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YInputField.cc

3.65 YInputFieldPrivate Struct Reference

Public Member Functions

- **YInputFieldPrivate** (std::string label, bool passwordMode)

Public Attributes

- std::string **label**
- bool **passwordMode**
- bool **shrinkable**
- std::string **validChars**
- int **inputMaxLength**

3.65.1 Detailed Description

Definition at line 35 of file YInputField.cc.

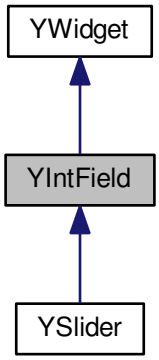
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YInputField.cc

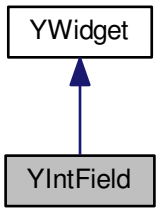
3.66 YIntField Class Reference

```
#include <YIntField.h>
```

Inheritance diagram for YIntField:



Collaboration diagram for YIntField:



Public Member Functions

- virtual ~YIntField ()
- virtual const char * widgetClass () const
- virtual int value ()=0
- void setValue (int val)
- int minValue () const
- void setMinValue (int val)
- int maxValue () const
- void setMaxValue (int val)
- std::string label () const
- virtual void setLabel (const std::string &label)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)

- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)
- const char * userInputProperty ()

Protected Member Functions

- YIntField (YWidget *parent, const std::string &label, int minValue, int maxValue)
- virtual void setValueInternal (int val)=0
- int enforceRange (int val) const

3.66.1 Detailed Description

IntField: Input field for integer values. Enforces input range between a specified minimum and maximum value.

Definition at line 38 of file YIntField.h.

3.66.2 Constructor & Destructor Documentation

3.66.2.1 YIntField::YIntField (YWidget * *parent*, const std::string & *label*, int *minValue*, int *maxValue*) [protected]

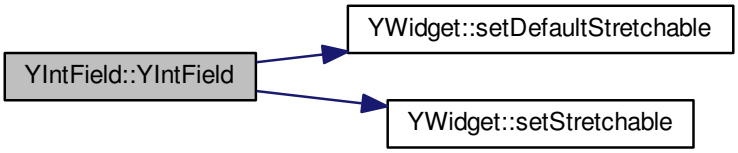
Constructor.

Create an IntField with 'label' as the caption, and the specified minimum and maximum values.

Note that YWidgetFactory::createIntField() also has an 'initialValue' parameter that is not used here (because the current value is not stored in this base class, but in the derived class).

Definition at line 50 of file YIntField.cc.

Here is the call graph for this function:



3.66.2.2 YIntField::~YIntField ( ) [virtual]

Destructor.

Definition at line 64 of file YIntField.cc.

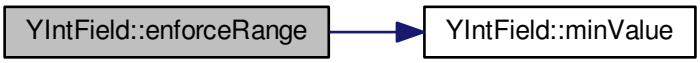
3.66.3 Member Function Documentation

3.66.3.1 int YIntField::enforceRange (int *val*) const [protected]

Enforce 'val' to be between minValue and maxValue. Return a value that is in range. This does not change the internally stored value of this IntField in any way.

Definition at line 71 of file YIntField.cc.

Here is the call graph for this function:



3.66.3.2 YPropertyValue YIntField::getProperty (const std::string & *propertyName*) [virtual]

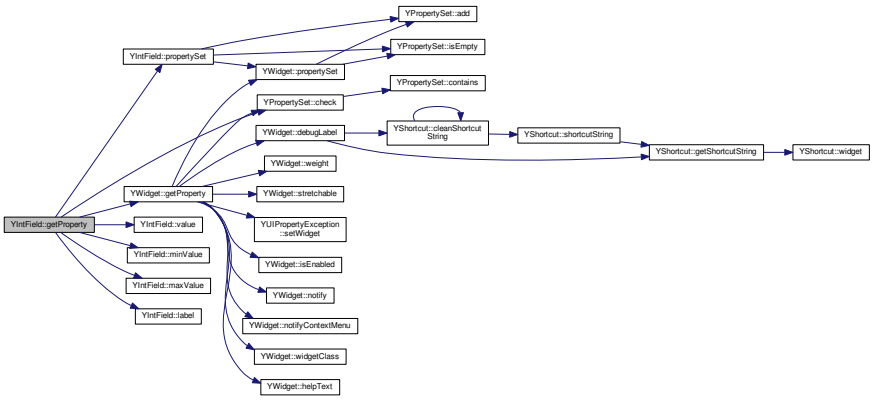
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 181 of file YIntField.cc.

Here is the call graph for this function:



3.66.3.3 std::string YIntField::label ( ) const

Get the label (the caption above the input field).

Definition at line 124 of file YIntField.cc.

3.66.3.4 int YIntField::maxValue ( ) const

Return the maximum value.

Definition at line 104 of file YIntField.cc.

3.66.3.5 int YIntField::minValue ( ) const

Return the minimum value.

Definition at line 84 of file YIntField.cc.

3.66.3.6 const YPropertySet & YIntField::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 139 of file YIntField.cc.

Here is the call graph for this function:

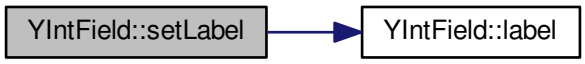
3.66.3.7 void YIntField::setLabel (const std::string & *label*) [virtual]

Set the label (the caption above the input field).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 131 of file YIntField.cc.

Here is the call graph for this function:

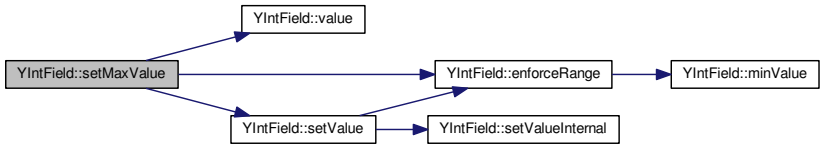


3.66.3.8 void YIntField::setMaxValue (int *val*)

Set a new maximum value. If the current value is greater than that, it will be set to the new maximum.

Definition at line 111 of file YIntField.cc.

Here is the call graph for this function:

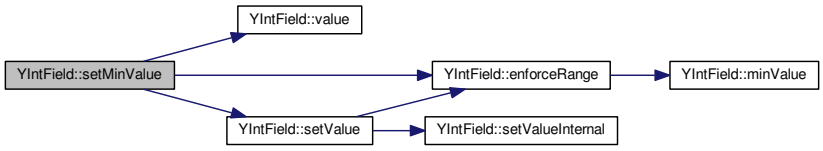


3.66.3.9 void YIntField::setMinValue (int *val*)

Set a new minimum value. If the current value is less than that, it will be set to the new minimum.

Definition at line 91 of file YIntField.cc.

Here is the call graph for this function:



3.66.3.10 bool YIntField::setProperty (const std::string & *propertyName,* const **YPropertyValue** & *val*) [virtual]

Set a property. Reimplemented from YWidget.

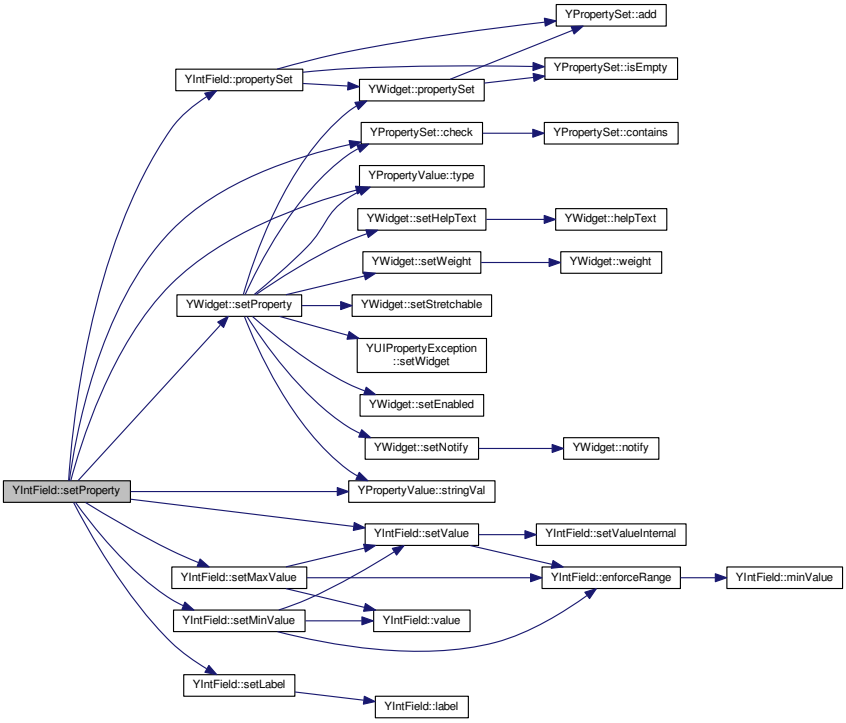
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 163 of file YIntField.cc.

Here is the call graph for this function:



3.66.3.11 virtual void YIntField::setShortcutString (const std::string & *str*) [inline],[virtual]

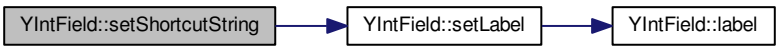
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 179 of file YIntField.h.

Here is the call graph for this function:

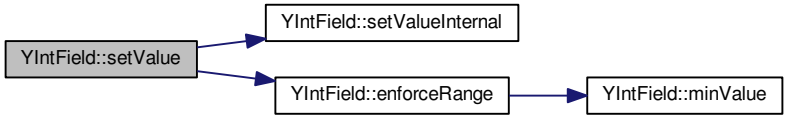


3.66.3.12 void YIntField::setValue (int *val*) [inline]

Set the current value (the number entered by the user or set from the outside) of this IntField. This method enforces 'val to be between minValue and maxValue.

Definition at line 81 of file YIntField.h.

Here is the call graph for this function:



3.66.3.13 virtual void YIntField::setValueInternal (int *val*) [protected], [pure virtual]

Set the current value (the number entered by the user or set from the outside) of this IntField. 'val' is guaranteed to be between minValue and maxValue; no further checks are required.

Derived classes are required to implement this method.

3.66.3.14 virtual std::string YIntField::shortcutString ( ) const [inline], [virtual]

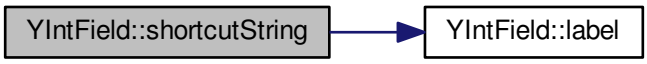
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 172 of file YIntField.h.

Here is the call graph for this function:



3.66.3.15 const char* YIntField::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 186 of file YIntField.h.

3.66.3.16 virtual int YIntField::value ( ) [pure virtual]

Get the current value (the number entered by the user or set from the outside) of this IntField.

Derived classes are required to implement this.

3.66.3.17 virtual const char* YIntField::widgetClass ( ) const [inline], [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Reimplemented in YSlider.

Definition at line 66 of file YIntField.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YIntField.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YIntField.cc

3.67 YIntFieldPrivate Struct Reference

Public Member Functions

- **YIntFieldPrivate** (const std::string &label, int minValue, int maxValue)

Public Attributes

- std::string **label**
- int **minValue**
- int **maxValue**

3.67.1 Detailed Description

Definition at line 32 of file YIntField.cc.

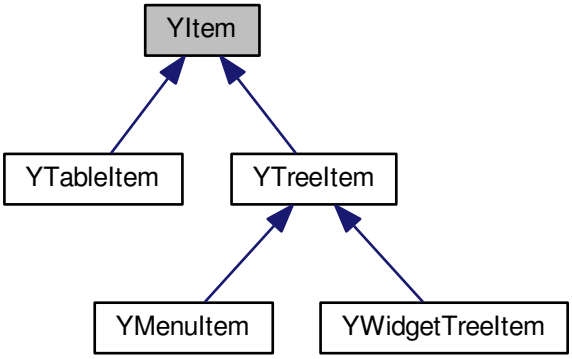
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YIntField.cc

3.68 YItem Class Reference

```
#include <YItem.h>
```

Inheritance diagram for YItem:



Public Member Functions

- YItem (const std::string &label, bool selected=false)
- YItem (const std::string &label, const std::string &iconName, bool selected=false)
- virtual ~YItem ()
- std::string label () const
- void setLabel (const std::string &newLabel)
- std::string iconName () const
- bool hasIconName () const
- void setIconName (const std::string &newIconName)
- bool selected () const
- void setSelected (bool sel=true)
- void setIndex (int index)
- int index () const
- void setData (void *newData)
- void * data () const
- virtual bool hasChildren () const
- virtual YItemIterator childrenBegin ()
- virtual YItemConstIterator **childrenBegin** () const
- virtual YItemIterator childrenEnd ()
- virtual YItemConstIterator **childrenEnd** () const
- virtual YItem * parent () const

3.68.1 Detailed Description

Simple item class for SelectionBox, ComboBox, MultiSelectionBox etc. items. This class provides stubs for children management.

Definition at line 43 of file YItem.h.

3.68.2 Constructor & Destructor Documentation

3.68.2.1 YItem::YItem (const std::string & *label*, bool *selected* = `false`) [inline]

Constructor with just the label and optionally the selected state.

Definition at line 49 of file YItem.h.

3.68.2.2 YItem::YItem (const std::string & *label*, const std::string & *iconName*, bool *selected* = `false`) [inline]

Constructor with label and icon name and optionally the selected state.

Definition at line 59 of file YItem.h.

3.68.2.3 virtual YItem::~YItem ( ) [inline], [virtual]

Destructor.

Definition at line 70 of file YItem.h.

3.68.3 Member Function Documentation

3.68.3.1 virtual YItemIterator YItem::childrenBegin ( ) [inline], [virtual]

Return an iterator that points to the first child item of this item.

This default implementation returns the 'end' iterator of the class-static always empty _noChildren YItemCollection. It is safe to use this iterator in classic iterator loops:

for (YItemIterator it = myItem->childrenBegin(); it != myItem->childrenEnd(); ++it) { ... }

The loop body will only ever be executed if this item is a derived class that actually manages child items.

Reimplemented in YTreeItem.

Definition at line 166 of file YItem.h.

3.68.3.2 virtual YItemIterator YItem::childrenEnd ( ) [inline], [virtual]

Return an iterator that points after the last child item of this item.

This default implementation returns the 'end' iterator of the class-static always empty _noChildren YItemCollection.

Reimplemented in YTreeItem.

Definition at line 175 of file YItem.h.

3.68.3.3 void* YItem::data ( ) const [inline]

Return the opaque data pointer.

Definition at line 133 of file YItem.h.

3.68.3.4 virtual bool YItem::hasChildren ( ) const [inline], [virtual]

Return 'true' if this item has any child items.

Reimplemented in YTreeItem.

Definition at line 147 of file YItem.h.

3.68.3.5 bool YItem::hasIconName ( ) const [inline]

Return 'true' if this item has an icon name.

Definition at line 91 of file YItem.h.

3.68.3.6 std::string YItem::iconName ( ) const [inline]

Return this item's icon name.

Definition at line 86 of file YItem.h.

3.68.3.7 int YItem::index ( ) const [inline]

Return the index of this item (as set with setIndex()).

Definition at line 118 of file YItem.h.

3.68.3.8 std::string YItem::label ( ) const [inline]

Return this item's label. This is what the user sees in a dialog, so this will usually be a translated text.

Definition at line 76 of file YItem.h.

3.68.3.9 virtual YItem* YItem::parent ( ) const [inline], [virtual]

Returns this item's parent item or 0 if it is a toplevel item. This default implementation always returns 0. Derived classes that handle children should reimplement this.

Reimplemented in YTreeItem, and YMenuItem.

Definition at line 183 of file YItem.h.

3.68.3.10 bool YItem::selected ( ) const [inline]

Return 'true' if this item is currently selected.

Definition at line 101 of file YItem.h.

3.68.3.11 void YItem::setData (void * *newData*) [inline]

Set the opaque data pointer for application use.

Applications can use this to store the pointer to a counterpart of this tree item. It is the application's responsibility to watch for dangling pointers and possibliy deleting the data. All this class ever does with this pointer is to store it.

Definition at line 128 of file YItem.h.

3.68.3.12 void YItem::setIconName (const std::string & *newIconName*) [inline]

Set this item's icon name.

Definition at line 96 of file YItem.h.

3.68.3.13 void YItem::setIndex (int *index*) [inline]

Set this item's index.

Definition at line 113 of file YItem.h.

Here is the call graph for this function:

3.68.3.14 void YItem::setLabel (const std::string & *newLabel*) [inline]

Set this item's label.

Definition at line 81 of file YItem.h.

3.68.3.15 void YItem::setSelected (bool *sel* = true) [inline]

Select or unselect this item. This does not have any effect on any other item; if it is desired that only one item is selected at any time, the caller has to take care of that.

Definition at line 108 of file YItem.h.

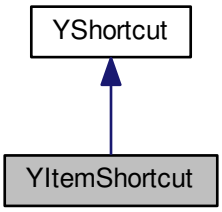
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YItem.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YItem.cc

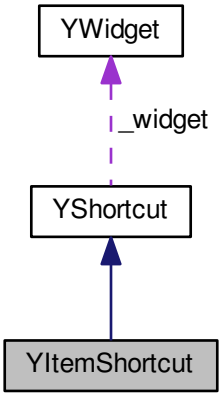
3.69 YItemShortcut Class Reference

```
#include <YShortcut.h>
```

Inheritance diagram for YItemShortcut:



Collaboration diagram for YItemShortcut:



Public Member Functions

- YItemShortcut (YWidget *widget, YItem *item)
- virtual ~YItemShortcut ()
- YItem * item () const
- virtual void setShortcut (char newShortcut)

Protected Member Functions

- virtual std::string getShortcutString ()

Additional Inherited Members

3.69.1 Detailed Description

Special case for widgets that can have multiple shortcuts based on items (like YDumbTab)

Definition at line 225 of file YShortcut.h.

3.69.2 Constructor & Destructor Documentation

3.69.2.1 YItemShortcut::YItemShortcut (YWidget * *widget,* YItem * *item*) [inline]

Constructor.

Definition at line 231 of file YShortcut.h.

3.69.2.2 virtual YItemShortcut::~YItemShortcut ( ) [inline], [virtual]

Destructor.

Definition at line 239 of file YShortcut.h.

3.69.3 Member Function Documentation

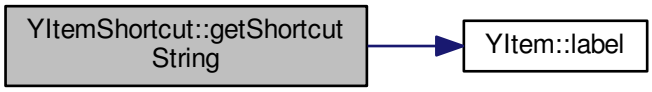
3.69.3.1 std::string YItemShortcut::getShortcutString ( ) [protected], [virtual]

Obtain the the shortcut property of this shortcut's widget - the string that contains "&" to designate a shortcut.

Reimplemented from YShortcut.

Definition at line 310 of file YShortcut.cc.

Here is the call graph for this function:



3.69.3.2 YItem* YItemShortcut::item ( ) const [inline]

Return the associated item.

Definition at line 244 of file YShortcut.h.

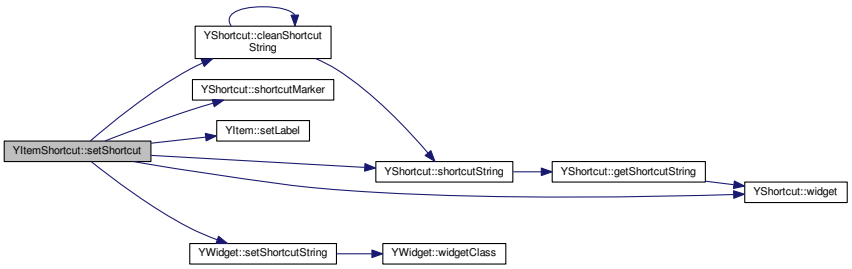
3.69.3.3 void YItemShortcut::setShortcut (char *newShortcut*) [virtual]

Set (override) the shortcut character. In this subclass, it will change the internally stored item.

Reimplemented from YShortcut.

Definition at line 320 of file YShortcut.cc.

Here is the call graph for this function:

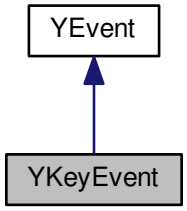


The documentation for this class was generated from the following files:

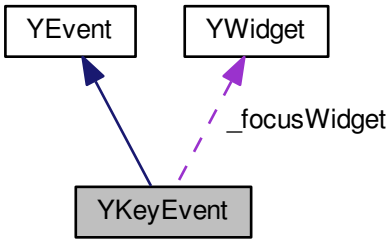
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YShortcut.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YShortcut.cc

3.70 YKeyEvent Class Reference

Inheritance diagram for YKeyEvent:



Collaboration diagram for YKeyEvent:



Public Member Functions

- YKeyEvent (const std::string &keySymbol, YWidget *focusWidget=0)
- std::string keySymbol () const
- YWidget * focusWidget () const

Protected Member Functions

- virtual ~YKeyEvent ()

Protected Attributes

- std::string **_keySymbol**
- YWidget * **_focusWidget**

Additional Inherited Members

3.70.1 Detailed Description

Definition at line 206 of file YEvent.h.

3.70.2 Constructor & Destructor Documentation

3.70.2.1 YKeyEvent::YKeyEvent (const std::string & *keySymbol*, YWidget * *focusWidget* = 0)

Constructor.

Create a key event with a specified key symbol (a text describing the key, such as "CursorLeft", "F1", etc.) and optionally the widget that currently has the keyboard focus.

Definition at line 123 of file YEvent.cc.

3.70.2.2 virtual YKeyEvent::~YKeyEvent ( ) [inline],[protected],[virtual]

Protected destructor - events can only be deleted via YDialog::deleteEvent(). The associated dialog will take care of this event and delete it when appropriate.

Definition at line 241 of file YEvent.h.

3.70.3 Member Function Documentation

3.70.3.1 YWidget* YKeyEvent::focusWidget ( ) const [inline]

Returns the widget that currently has the keyboard focus.

This might be 0 if no widget has the focus or if the creator of this event could not obtain that information.

Definition at line 232 of file YEvent.h.

3.70.3.2 std::string YKeyEvent::keySymbol ( ) const [inline]

Returns the key symbol - a text describing the key, such as "CursorLeft", "F1", "a", "A", etc.

Definition at line 224 of file YEvent.h.

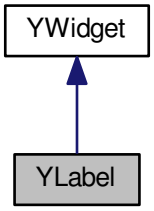
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEvent.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEvent.cc

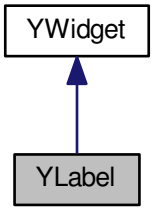
3.71 YLabel Class Reference

```
#include <YLabel.h>
```

Inheritance diagram for YLabel:



Collaboration diagram for YLabel:



Public Member Functions

- YLabel (YWidget *parent, const std::string &text, bool isHeading=false, bool isOutputField=false)
- virtual ~YLabel ()
- virtual const char * widgetClass () const
- std::string text () const
- std::string value () const
- std::string **label** () const
- virtual void setText (const std::string &newText)
- void setValue (const std::string &newValue)
- void **setLabel** (const std::string &newLabel)
- bool isHeading () const
- bool isOutputField () const
- bool useBoldFont () const
- virtual void setUseBoldFont (bool bold=true)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- virtual std::string debugLabel () const

Additional Inherited Members

3.71.1 Detailed Description

Implementation of the Label, Heading and OutputField widgets

Definition at line 38 of file YLabel.h.

3.71.2 Constructor & Destructor Documentation

3.71.2.1 YLabel::YLabel (YWidget * *parent*, const std::string & *text*, bool *isHeading* = `false`, bool *isOutputField* = `false`)

Constructor.

'isHeading' indicates if this should be displayed as a Heading widget, i.e. with a bold and/or larger font. This cannot be changed after creating the widget.

'isOutputField' indicates if this should be displayed as an OutputField widget, i.e. similar to an InputField the user can't change. This cannot be changed after creating the widget.

Definition at line 56 of file YLabel.cc.

3.71.2.2 YLabel::~YLabel ( ) [virtual]

Destructor.

Definition at line 67 of file YLabel.cc.

3.71.3 Member Function Documentation

3.71.3.1 std::string YLabel::debugLabel ( ) const [virtual]

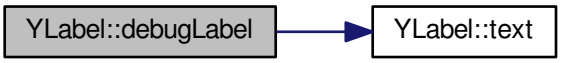
Returns a descriptive label of this widget instance for debugging.

Reimplemented from YWidget since a YLabel doesn't have a shortcut property.

Reimplemented from YWidget.

Definition at line 164 of file YLabel.cc.

Here is the call graph for this function:



3.71.3.2 YPropertyValue YLabel::getProperty (const std::string & *propertyName*) [virtual]

Get a property. Reimplemented from YWidget.

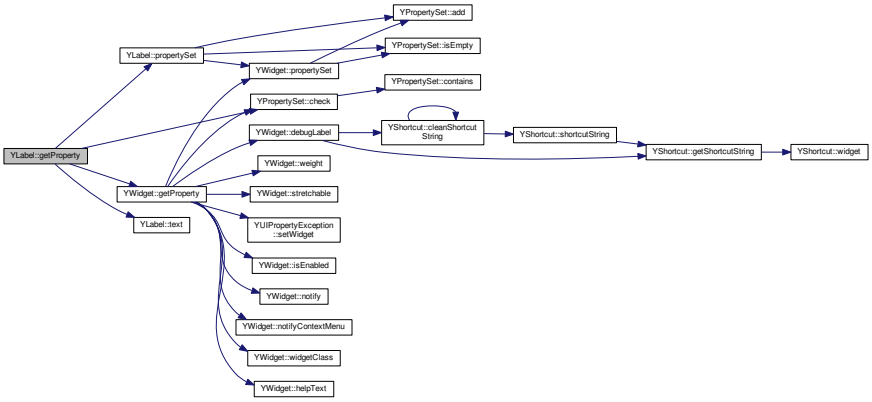
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from YWidget.

Definition at line 150 of file YLabel.cc.

Here is the call graph for this function:



3.71.3.3 bool YLabel::isHeading ( ) const

Return 'true' if this is a Heading widget, i.e., it should display its text in a bold and/or larger font.

This cannot be changed after creating the widget.

Definition at line 85 of file YLabel.cc.

3.71.3.4 bool YLabel::isOutputField ( ) const

Return 'true' if this is an OutputField widget, i.e., it should display its text similar to an InputField the user can't change.

This cannot be changed after creating the widget.

Definition at line 91 of file YLabel.cc.

3.71.3.5 const YPropertySet & YLabel::propertySet ( ) [virtual]

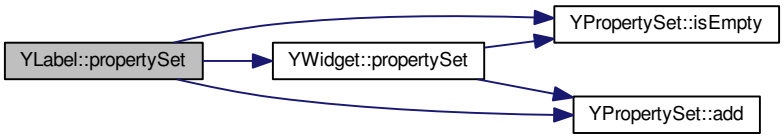
Return this class's property set. This also initializes the property set upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 110 of file YLabel.cc.

Here is the call graph for this function:



3.71.3.6 bool YLabel::setProperty (const std::string & *propertyName,* const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

This method may throw exceptions, for example

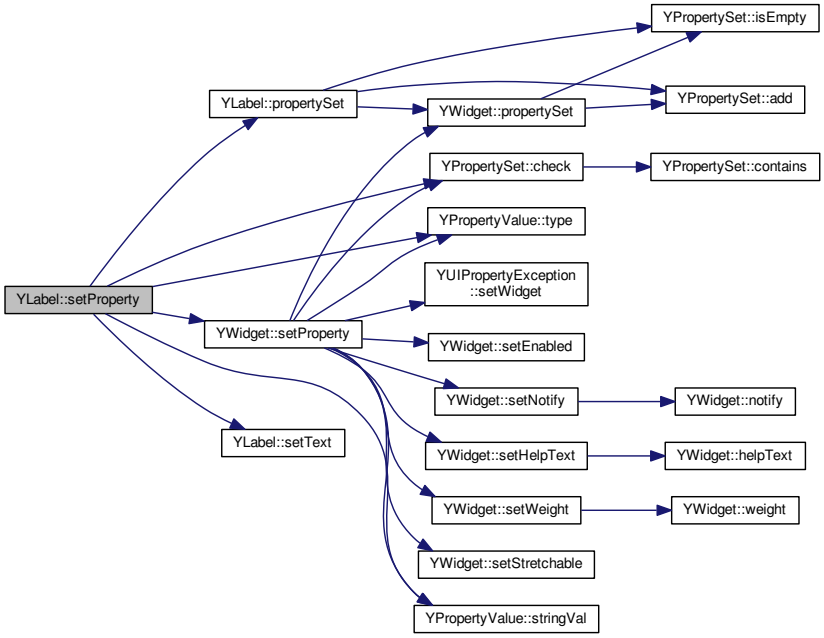
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 133 of file YLabel.cc.

Here is the call graph for this function:



3.71.3.7 void YLabel::setText (const std::string & *newText*) [virtual]

Set the text the widget displays.

Derived classes should overwrite this, but call this base class function in the overwritten function.

Definition at line 79 of file YLabel.cc.

3.71.3.8 void YLabel::setUseBoldFont (bool *bold* = true) [virtual]

Switch bold font on or off.

Derived classes should overwrite this, but call this base class function in the overwritten function.

Definition at line 103 of file YLabel.cc.

3.71.3.9 void YLabel::setValue (const std::string & *newValue*) [inline]

Aliases for setText().

Definition at line 93 of file YLabel.h.

Here is the call graph for this function:

3.71.3.10 std::string YLabel::text ( ) const

Return the text the widget displays.

Definition at line 73 of file YLabel.cc.

3.71.3.11 bool YLabel::useBoldFont ( ) const

Return 'true' if a bold font should be used.

Definition at line 97 of file YLabel.cc.

3.71.3.12 std::string YLabel::value ( ) const [inline]

Aliases for text().

Definition at line 79 of file YLabel.h.

Here is the call graph for this function:

3.71.3.13 const char * YLabel::widgetClass ( ) const [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 186 of file YLabel.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YLabel.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YLabel.cc

3.72 YLabelPrivate Struct Reference

Public Member Functions

- YLabelPrivate (const std::string &text, bool isHeading, bool isOutputField)

Public Attributes

- std::string **text**
- bool **isHeading**
- bool **isOutputField**
- bool **useBoldFont**

3.72.1 Detailed Description

Definition at line 35 of file YLabel.cc.

3.72.2 Constructor & Destructor Documentation

3.72.2.1 YLabelPrivate::YLabelPrivate (const std::string & *text*, bool *isHeading*, bool *isOutputField*) [inline]

Constructor

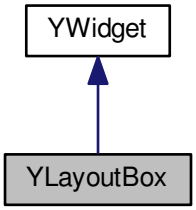
Definition at line 40 of file YLabel.cc.

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

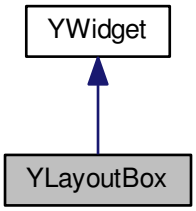
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YLabel.cc

3.73 YLayoutBox Class Reference

Inheritance diagram for YLayoutBox:



Collaboration diagram for YLayoutBox:



Public Types

- typedef std::vector< int > **sizeVector**
- typedef std::vector< int > **posVector**

Public Member Functions

- virtual ~YLayoutBox ()
- virtual const char * widgetClass () const
- YUIDimension primary () const
- YUIDimension secondary () const
- bool debugLayout () const
- void setDebugLayout (bool deb=true)
- virtual int preferredSize (YUIDimension dim)
- virtual int preferredWidth ()
- virtual int preferredHeight ()
- virtual void setSize (int newWidth, int newHeight)
- virtual bool stretchable (YUIDimension dimension) const
- virtual void moveChild (YWidget *child, int newX, int newY)=0

Static Public Member Functions

- static bool isLayoutStretch (YWidget *child, YUIDimension dimension)

Protected Member Functions

- YLayoutBox (YWidget *parent, YUIDimension dim)
- int childrenTotalWeight (YUIDimension dimension)
- int childrenMaxPreferredSize (YUIDimension dimension)
- int totalNonWeightedChildrenPreferredSize (YUIDimension dimension)
- int countNonWeightedChildren (YUIDimension dimension)
- int countStretchableChildren (YUIDimension dimension)
- int countLayoutStretchChildren (YUIDimension dimension)

- YWidget * findDominatingChild ()
- void calcPrimaryGeometry (int newSize, sizeVector &childSize, posVector &childPos)
- void calcSecondaryGeometry (int newSize, sizeVector &childSize, posVector &childPos)
- void doResize (sizeVector &width, sizeVector &height, posVector &x_pos, posVector &y_pos)

3.73.1 Detailed Description

Definition at line 35 of file YLayoutBox.h.

3.73.2 Constructor & Destructor Documentation

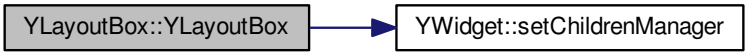
3.73.2.1 YLayoutBox::YLayoutBox (YWidget * *parent*, YUIDimension *dim*) [protected]

Constructor.

Creates a VBox for dim == YD_VERT or a HBox for YD_HORIZ.

Definition at line 66 of file YLayoutBox.cc.

Here is the call graph for this function:



3.73.2.2 YLayoutBox::~YLayoutBox ( ) [virtual]

Destructor.

Definition at line 75 of file YLayoutBox.cc.

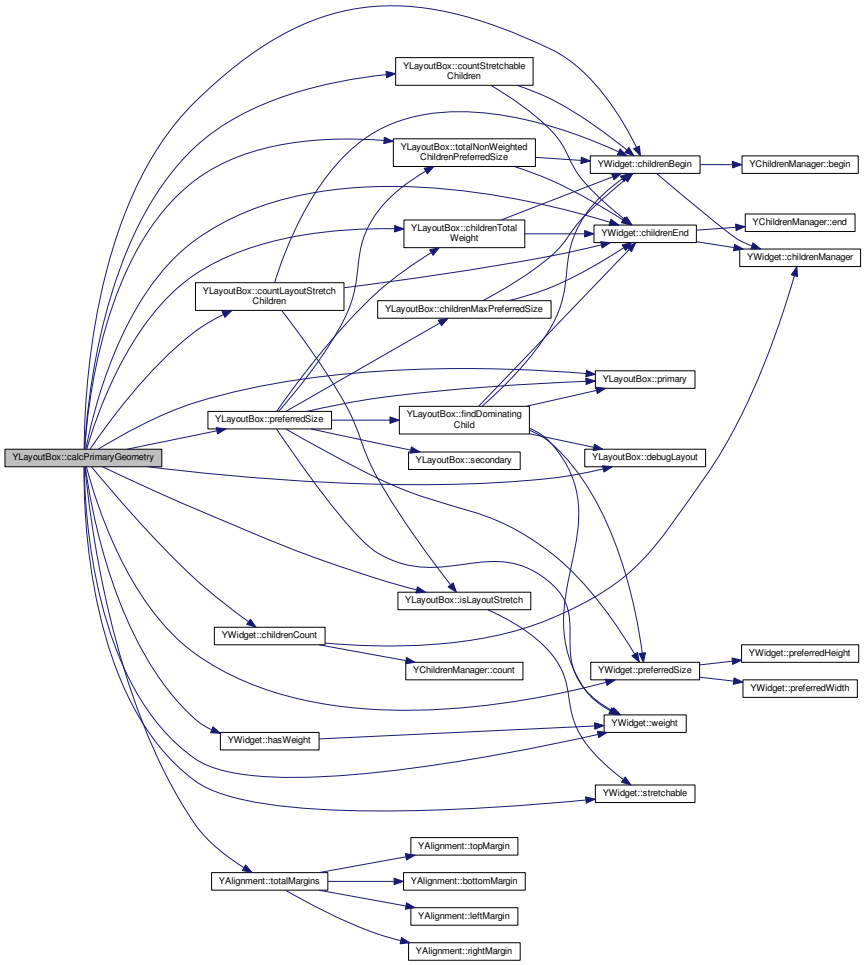
3.73.3 Member Function Documentation

3.73.3.1 void YLayoutBox::calcPrimaryGeometry (int *newSize*, sizeVector & *childSize*, posVector & *childPos*) [protected]

Calculate the sizes and positions of all children in the primary dimension and store them in "childSize" and "childPos".

Definition at line 398 of file YLayoutBox.cc.

Here is the call graph for this function:

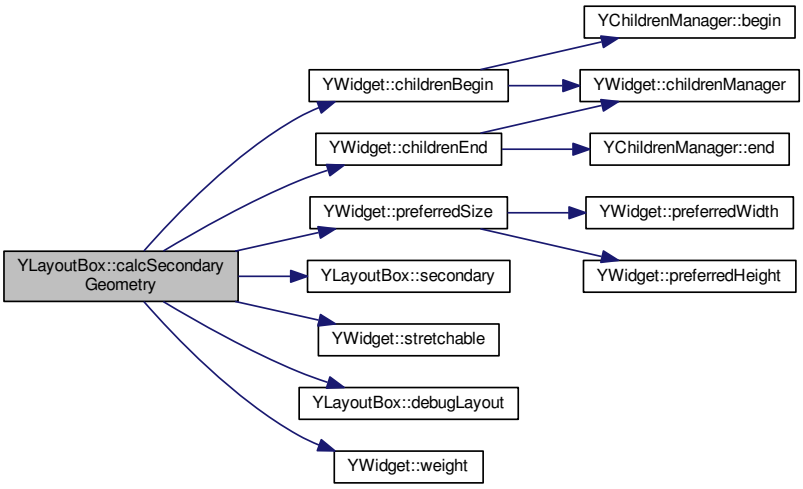


3.73.3.2 void YLayoutBox::calcSecondaryGeometry (int *newSize*, sizeVector & *childSize*, posVector & *childPos*) [protected]

Calculate the sizes and positions of all children in the secondary dimension and store them in "childSize" and "childPos".

Definition at line 694 of file YLayoutBox.cc.

Here is the call graph for this function:

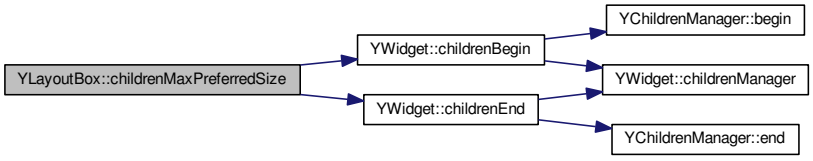


3.73.3.3 int YLayoutBox::childrenMaxPreferredSize (YUIDimension *dimension*) [protected]

Return the maximum preferred size of all children in the specified dimension.

Definition at line 231 of file YLayoutBox.cc.

Here is the call graph for this function:

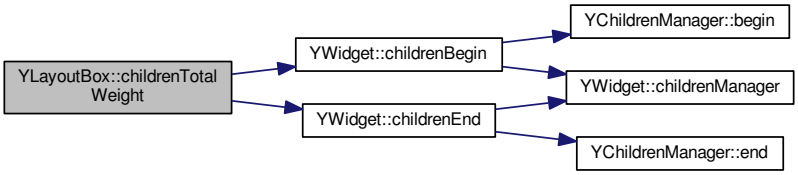


3.73.3.4 int YLayoutBox::childrenTotalWeight (YUIDimension *dimension*) [protected]

Add up all the children's weights.

Definition at line 247 of file YLayoutBox.cc.

Here is the call graph for this function:

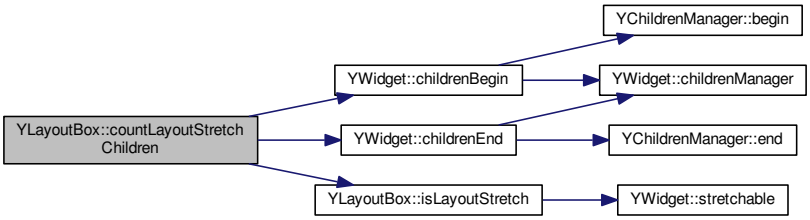


3.73.3.5 int YLayoutBox::countLayoutStretchChildren (YUIDimension *dimension*) [protected]

Count the number of "rubber bands", i.e. the number of stretchable layout spacings (e.g. {H|V}Weight, {H|V}Spacing). Only those without a weight are counted.

Definition at line 315 of file YLayoutBox.cc.

Here is the call graph for this function:



3.73.3.6 int YLayoutBox::countNonWeightedChildren (YUIDimension *dimension*) [protected]

Count the number of non-weighted children.

Definition at line 280 of file YLayoutBox.cc.

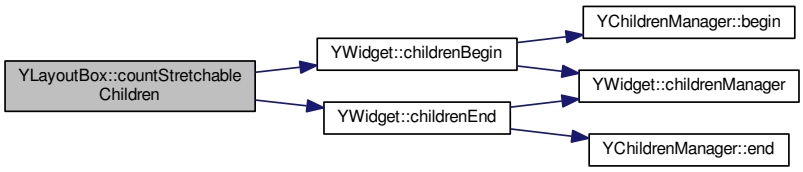
Here is the call graph for this function:

3.73.3.7 int YLayoutBox::countStretchableChildren (YUIDimension *dimension*) [protected]

Count the number of stretchable (non-weighted) children. Note: Weighted children are *always* considered stretchable.

Definition at line 297 of file YLayoutBox.cc.

Here is the call graph for this function:



3.73.3.8 bool YLayoutBox::debugLayout ( ) const

Returns 'true' if layout debugging (verbose logging during layout) is on.

Definition at line 96 of file YLayoutBox.cc.

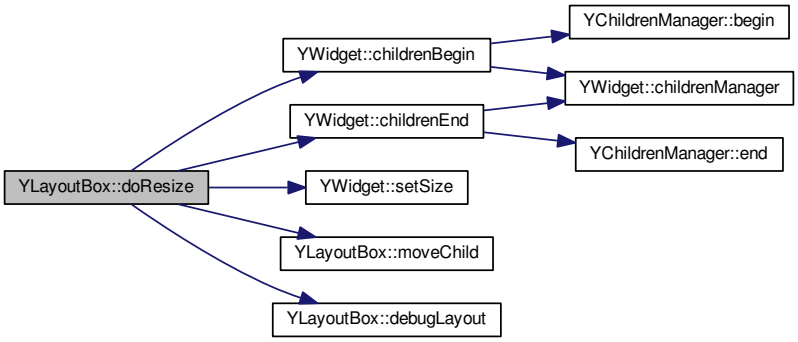
3.73.3.9 void YLayoutBox::doResize (sizeVector & *width,* sizeVector & *height,* posVector & *x_pos,* posVector & *y_pos*) [protected]

Actually perform resizing and moving the child widgets to the appropriate position.

The vectors passed are the sizes previously calculated by calcPrimaryGeometry() and calcSecondaryGeometry().

Definition at line 746 of file YLayoutBox.cc.

Here is the call graph for this function:



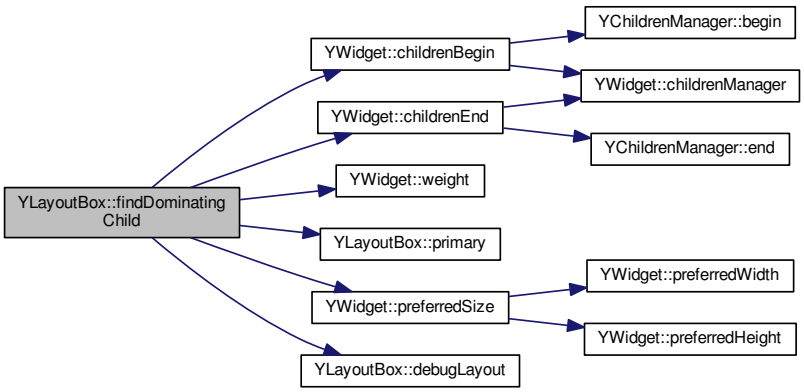
3.73.3.10 YWidget * YLayoutBox::findDominatingChild ( ) [protected]

Determine the number of the "dominating child" - the child widget that determines the overall size with respect to its weight.

Return 0 if there is no dominating child, i.e. none of the children has a weight specified.

Definition at line 185 of file YLayoutBox.cc.

Here is the call graph for this function:

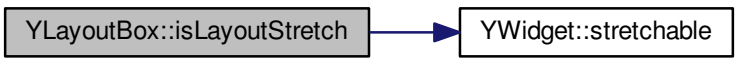


3.73.3.11 bool YLayoutBox::isLayoutStretch (YWidget * *child*, YUIDimension *dimension*) [static]

Check if this is a layout stretch widget in the specfied dimension, i.e. an empty widget that is stretchable.

Definition at line 333 of file YLayoutBox.cc.

Here is the call graph for this function:



3.73.3.12 virtual void YLayoutBox::moveChild (YWidget * *child*, int *newX*, int *newY*) [pure virtual]

Move a child to a new position.

Derived classes are required to implement this.

3.73.3.13 int YLayoutBox::preferredHeight ( ) [virtual]

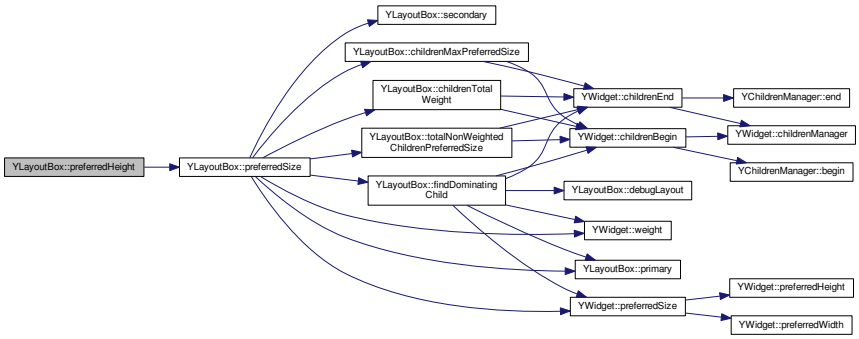
Preferred height of the widget.

Reimplemented from YWidget.

Implements YWidget.

Definition at line 165 of file YLayoutBox.cc.

Here is the call graph for this function:



3.73.3.14 int YLayoutBox::preferredSize (YUIDimension *dim*) [virtual]

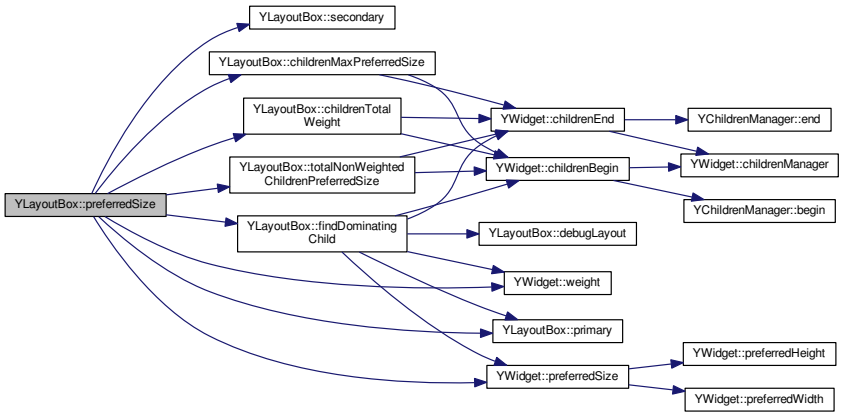
Preferred size of the widget in the specified dimension.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 111 of file YLayoutBox.cc.

Here is the call graph for this function:



3.73.3.15 int YLayoutBox::preferredWidth ( ) [virtual]

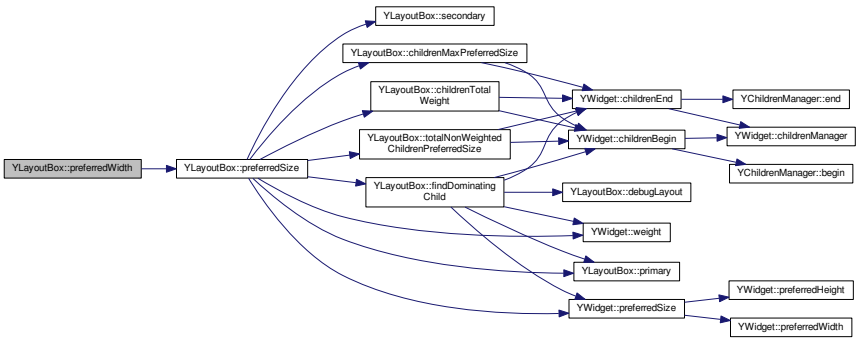
Preferred width of the widget.

Reimplemented from YWidget.

Implements YWidget.

Definition at line 159 of file YLayoutBox.cc.

Here is the call graph for this function:



3.73.3.16 YUIDimension YLayoutBox::primary ( ) const

Return the primary dimension, i.e., the dimension this LayoutBox lays out its children in: YD_VERT for a VBox, YD_HORIZ for a HBox.

Definition at line 82 of file YLayoutBox.cc.

3.73.3.17 YUIDimension YLayoutBox::secondary ( ) const

Return the secondary dimension.

Definition at line 89 of file YLayoutBox.cc.

3.73.3.18 void YLayoutBox::setDebugLayout (bool *deb* = `true`)

Enable or disable layout debugging.

Definition at line 102 of file YLayoutBox.cc.

3.73.3.19 void YLayoutBox::setSize (int *newWidth*, int *newHeight*) [virtual]

Sets the size of the layout box. This is where the layout policy is implemented.

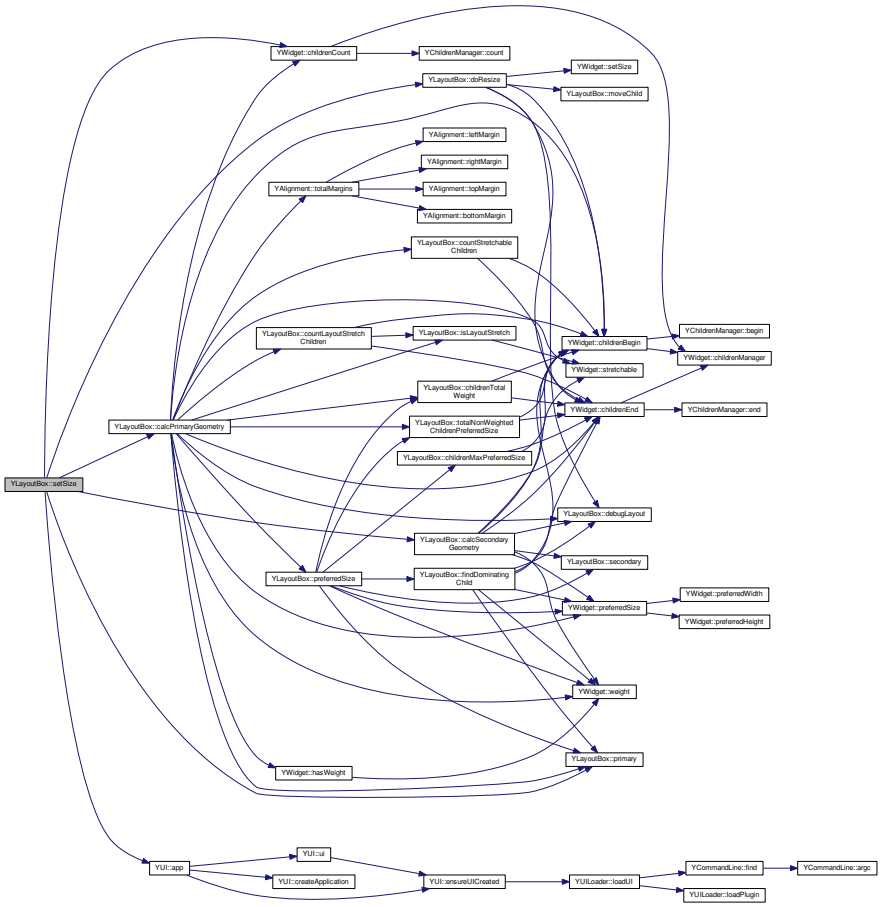
Derived classes can reimplement this, but this base class method should be called in the reimplemented function.

Reimplemented from YWidget.

Implements YWidget.

Definition at line 365 of file YLayoutBox.cc.

Here is the call graph for this function:



3.73.3.20 bool YLayoutBox::stretchable (YUIDimension *dimension*) const [virtual]

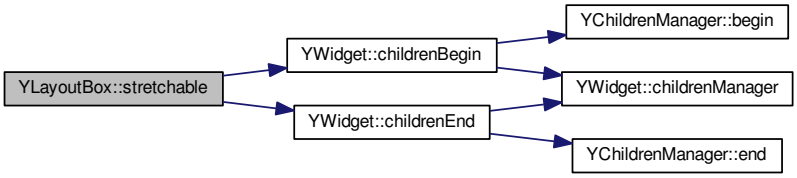
Returns the stretchability of the layout box: The layout box is stretchable if one of the children is stretchable in this dimension or if one of the child widgets has a layout weight in this dimension.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 349 of file YLayoutBox.cc.

Here is the call graph for this function:

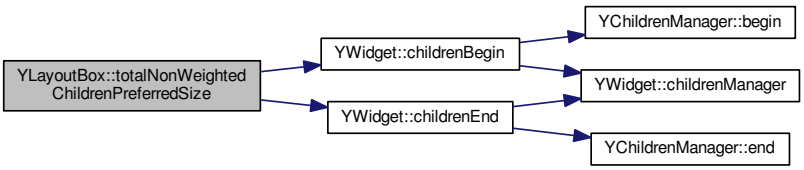


3.73.3.21 int YLayoutBox::totalNonWeightedChildrenPreferredSize (YUIDimension *dimension*) [protected]

Add up all the non-weighted children's preferred sizes in the specified dimension.

Definition at line 263 of file YLayoutBox.cc.

Here is the call graph for this function:



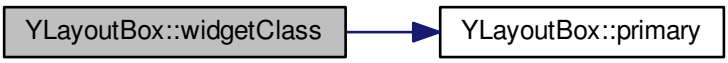
3.73.3.22 const char * YLayoutBox::widgetClass ( ) const [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 775 of file YLayoutBox.cc.

Here is the call graph for this function:



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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YLayoutBox.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YLayoutBox.cc

3.74 YLayoutBoxPrivate Struct Reference

Public Member Functions

- YLayoutBoxPrivate (YUIDimension prim)

Public Attributes

- YUIDimension **primary**
- YUIDimension **secondary**
- bool **debugLayout**

3.74.1 Detailed Description

Definition at line 43 of file YLayoutBox.cc.

3.74.2 Constructor & Destructor Documentation

3.74.2.1 YLayoutBoxPrivate::YLayoutBoxPrivate (YUIDimension *prim*) [inline]

Constructor

Definition at line 48 of file YLayoutBox.cc.

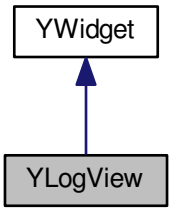
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YLayoutBox.cc

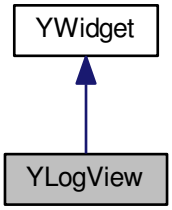
3.75 YLogView Class Reference

```
#include <YLogView.h>
```

Inheritance diagram for YLogView:



Collaboration diagram for YLogView:



Public Member Functions

- virtual ~YLogView ()
- virtual const char * widgetClass () const
- std::string label () const
- virtual void setLabel (const std::string &label)
- int visibleLines () const
- void setVisibleLines (int newVisibleLines)
- int maxLines () const
- void setMaxLines (int newMaxLines)
- std::string logText () const
- void setLogText (const std::string &text)
- std::string lastLine () const
- void appendLines (const std::string &text)
- void clearText ()
- int lines () const
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)

- virtual const YPropertySet & propertySet ()
- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)

Protected Member Functions

- YLogView (YWidget *parent, const std::string &label, int visibleLines, int maxLines)
- virtual void displayLogText (const std::string &text)=0

3.75.1 Detailed Description

LogView: A scrollable (output-only) text to display a growing log, very much like the "tail -f" shell command.

Definition at line 37 of file YLogView.h.

3.75.2 Constructor & Destructor Documentation

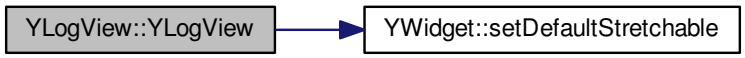
3.75.2.1 YLogView::YLogView (YWidget * *parent*, const std::string & *label*, int *visibleLines*, int *maxLines*) [protected]

Constructor.

'label' is the caption above the log. 'visibleLines' indicates how many lines should be visible by default (unless changed by other layout constraints), 'maxLines' specifies how many lines (always the last ones) to keep in the log. 0 for 'maxLines' means "keep all lines".

Definition at line 58 of file YLogView.cc.

Here is the call graph for this function:



3.75.2.2 YLogView::~YLogView ( ) [virtual]

Destructor.

Definition at line 69 of file YLogView.cc.

3.75.3 Member Function Documentation

3.75.3.1 void YLogView::appendLines (const std::string & *text*)

Append one or more lines to the log text and trigger a display update.

Definition at line 161 of file YLogView.cc.

3.75.3.2 void YLogView::clearText ( )

Clear the log text and trigger a display update.

Definition at line 206 of file YLogView.cc.

3.75.3.3 virtual void YLogView::displayLogText (const std::string & *text*) [protected],[pure virtual]

Display the part of the log text that should be displayed. 'text' contains the last 'visibleLines()' lines. This is called whenever the log text changes. Note that the text might also be empty, in which case the displayed log text should be cleared.

Derived classes are required to implement this.

3.75.3.4 YPropertyValue YLogView::getProperty (const std::string & *propertyName*) [virtual]

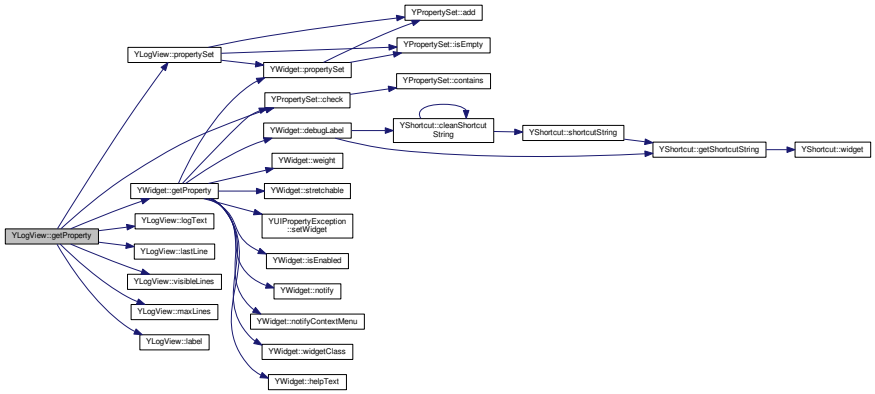
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 273 of file YLogView.cc.

Here is the call graph for this function:



3.75.3.5 std::string YLogView::label ( ) const

Return the label (the caption above the log text).

Definition at line 76 of file YLogView.cc.

3.75.3.6 std::string YLogView::lastLine ( ) const

Return the last log line.

Definition at line 151 of file YLogView.cc.

3.75.3.7 int YLogView::lines ( ) const

Return the current number of lines.

Definition at line 213 of file YLogView.cc.

3.75.3.8 std::string YLogView::logText ( ) const

Return the entire log text as one large string of concatenated lines delimited with newlines.

Definition at line 125 of file YLogView.cc.

3.75.3.9 int YLogView::maxLines ( ) const

Return the maximum number of lines to store. The last maxLines() lines of the log text will be kept.

Definition at line 104 of file YLogView.cc.

3.75.3.10 const YPropertySet & YLogView::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 228 of file YLogView.cc.

Here is the call graph for this function:

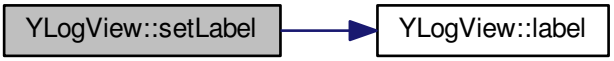
3.75.3.11 void YLogView::setLabel (const std::string & *label*) [virtual]

Set the label (the caption above the log text).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 83 of file YLogView.cc.

Here is the call graph for this function:

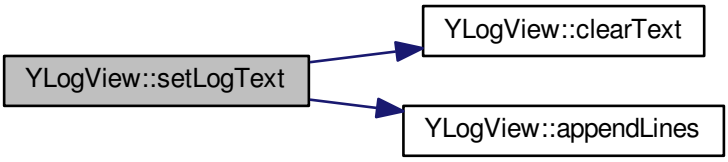


3.75.3.12 void YLogView::setLogText (const std::string & *text*) [inline]

Set (replace) the entire log text and trigger a display update.

Definition at line 122 of file YLogView.h.

Here is the call graph for this function:



3.75.3.13 void YLogView::setMaxLines (int *newMaxLines*)

Set the maximum number of lines to store. "0" means "keep all lines" (beware of memory overflow!).

If the new value is lower than the old value, any (now) excess lines before the last 'newMaxLines' lines of the log text is cut off and a display update is triggered.

This method is intentionally not virtual since a display update is triggered when appropriate.

Definition at line 111 of file YLogView.cc.

3.75.3.14 bool YLogView::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

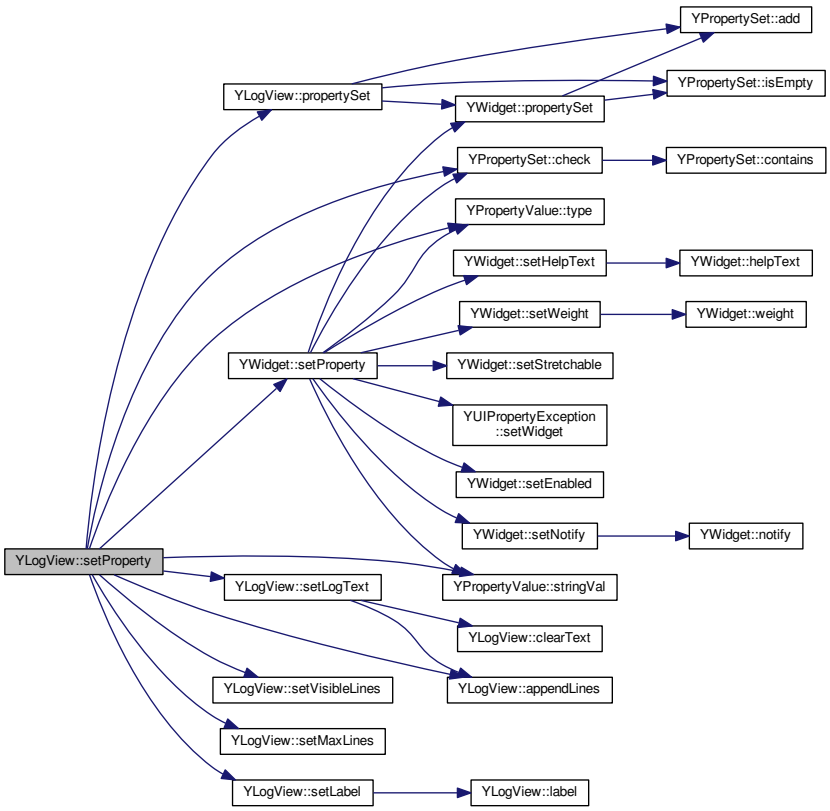
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 254 of file YLogView.cc.

Here is the call graph for this function:



3.75.3.15 virtual void YLogView::setShortcutString (const std::string & *str*) [inline],[virtual]

Set the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 186 of file YLogView.h.

Here is the call graph for this function:

3.75.3.16 void YLogView::setVisibleLines (int *newVisibleLines*)

Set the number of visible lines. Changing this has only effect upon the next geometry call, so applications calling this function might want to trigger a re-layout afterwards.

This method is intentionally not virtual: visibleLines() should be queried in the preferredHeight() implementation.

Definition at line 97 of file YLogView.cc.

3.75.3.17 virtual std::string YLogView::shortcutString ( ) const [inline], [virtual]

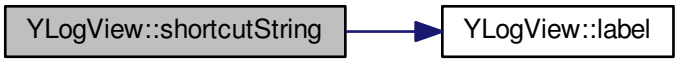
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 179 of file YLogView.h.

Here is the call graph for this function:



3.75.3.18 int YLogView::visibleLines ( ) const

Return the number of visible lines.

Definition at line 90 of file YLogView.cc.

3.75.3.19 virtual const char* YLogView::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 64 of file YLogView.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YLogView.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YLogView.cc

3.76 YLogViewPrivate Struct Reference

Public Member Functions

- **YLogViewPrivate** (const std::string &label, int visibleLines, int maxLines)

Public Attributes

- std::string **label**
- int **visibleLines**
- int **maxLines**
- StringDeque **logText**

3.76.1 Detailed Description

Definition at line 40 of file YLogView.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YLogView.cc

3.77 YMacro Class Reference

```
#include <YMacro.h>
```

Static Public Member Functions

- static void setRecorder (YMacroRecorder *recorder)
- static void setPlayer (YMacroPlayer *player)
- static void record (const std::string &macroFile)
- static void endRecording ()
- static bool recording ()
- static void play (const std::string &macroFile)
- static void playNextBlock ()
- static bool playing ()
- static YMacroRecorder * recorder ()
- static YMacroPlayer * player ()
- static void deleteRecorder ()
- static void deletePlayer ()

3.77.1 Detailed Description

Simple access to macro recording and playing.

This class stores an instance of a macro recorder and a macro player. Since both YMacroRecorder and YMacroPlayer are abstract base classes, derived classes from either of them have to be instantiated and set (setRecorder(), setPlayer()) from the outside for anything to happen. Until that point, none of the macro operations here do anything (but also don't throw any error or exception).

Definition at line 44 of file YMacro.h.

3.77.2 Member Function Documentation

3.77.2.1 void YMacro::deletePlayer ( ) [static]

Delete the current macro player if there is one.

Definition at line 105 of file YMacro.cc.

3.77.2.2 void YMacro::deleteRecorder ( ) [static]

Delete the current macro recorder if there is one.

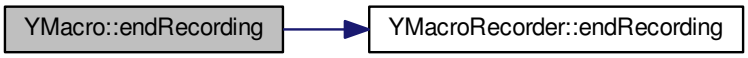
Definition at line 98 of file YMacro.cc.

3.77.2.3 void YMacro::endRecording ( ) [static]

End macro recording.

Definition at line 59 of file YMacro.cc.

Here is the call graph for this function:

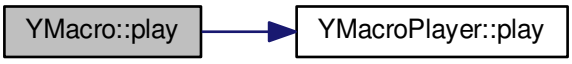


3.77.2.4 void YMacro::play (const std::string & *macroFile*) [static]

Play a macro from the specified macro file.

Definition at line 75 of file YMacro.cc.

Here is the call graph for this function:



3.77.2.5 static YMacroPlayer* YMacro::player ( ) [inline], [static]

Return the current macro player or 0 if there is none.

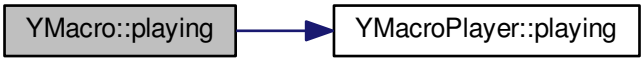
Definition at line 106 of file YMacro.h.

3.77.2.6 bool YMacro::playing ( ) [static]

Return 'true' if a macro is currently being played.

Definition at line 89 of file YMacro.cc.

Here is the call graph for this function:

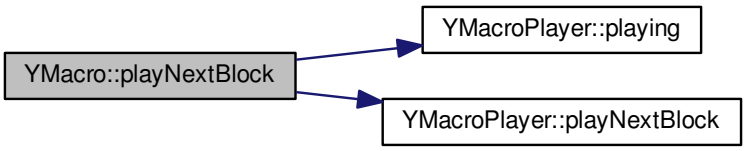


3.77.2.7 void YMacro::playNextBlock ( ) [static]

Play the next block from the current macro, if there is one playing.

Definition at line 82 of file YMacro.cc.

Here is the call graph for this function:

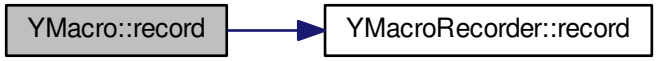


3.77.2.8 void YMacro::record (const std::string & *macroFile*) [static]

Record a macro to the specified macro file.

Definition at line 52 of file YMacro.cc.

Here is the call graph for this function:



3.77.2.9 static YMacroRecorder* YMacro::recorder ( ) [inline],[static]

Return the current macro recorder or 0 if there is none.

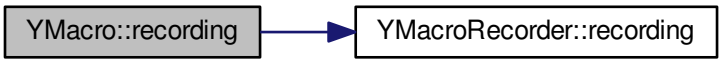
Definition at line 101 of file YMacro.h.

3.77.2.10 bool YMacro::recording ( ) [static]

Return 'true' if a macro is currently being recorded.

Definition at line 66 of file YMacro.cc.

Here is the call graph for this function:



3.77.2.11 void YMacro::setPlayer (YMacroPlayer * *player*) [static]

Set a macro player.

This needs to be done from the outside since YMacroRecorder is an abstract base class, i.e., it needs to be derived to be instantiated.

Definition at line 43 of file YMacro.cc.

Here is the call graph for this function:

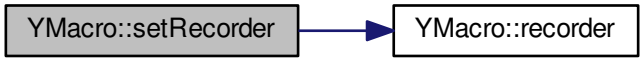
3.77.2.12 void YMacro::setRecorder (YMacroRecorder * *recorder*) [static]

Set a macro recorder.

This needs to be done from the outside since YMacroRecorder is an abstract base class, i.e., it needs to be derived to be instantiated.

Definition at line 34 of file YMacro.cc.

Here is the call graph for this function:



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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMacro.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMacro.cc

3.78 YMacroPlayer Class Reference

```
#include <YMacroPlayer.h>
```

Public Member Functions

- virtual ~YMacroPlayer ()
- virtual void play (const std::string &macroFile)=0
- virtual void playNextBlock ()=0
- virtual bool playing () const =0

Protected Member Functions

- YMacroPlayer ()

Friends

- class **YMacro**

3.78.1 Detailed Description

Abstract base class for macro player.

Applications should not use this directly, but the static methods in YMacro.

Definition at line 35 of file YMacroPlayer.h.

3.78.2 Constructor & Destructor Documentation

3.78.2.1 YMacroPlayer::YMacroPlayer ( ) [inline],[protected]

Constructor

Definition at line 43 of file YMacroPlayer.h.

3.78.2.2 virtual YMacroPlayer::~YMacroPlayer ( ) [inline],[virtual]

Destructor

Definition at line 49 of file YMacroPlayer.h.

3.78.3 Member Function Documentation

3.78.3.1 virtual void YMacroPlayer::play (const std::string & *macroFile*) [pure virtual]

Play a macro from the specified macro file.

3.78.3.2 virtual bool YMacroPlayer::playing ( ) const [pure virtual]

Return 'true' if a macro is currently being played.

3.78.3.3 virtual void YMacroPlayer::playNextBlock ( ) [pure virtual]

Play the next block from the current macro, if there is one playing.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMacroPlayer.h

3.79 YMacroRecorder Class Reference

```
#include <YMacroRecorder.h>
```

Public Member Functions

- virtual ~YMacroRecorder ()
- virtual void record (const std::string &macroFileName)=0
- virtual void endRecording ()=0
- virtual bool recording () const =0
- virtual void recordWidgetProperty (YWidget *widget, const char *propertyName)=0
- virtual void recordMakeScreenShot (bool enabled=false, const std::string &filename=std::string())=0

Protected Member Functions

- YMacroRecorder ()

Friends

- class **YMacro**

3.79.1 Detailed Description

Abstract base class for macro recorders.

Applications should not use this directly, but the static methods in YMacro.

Definition at line 38 of file YMacroRecorder.h.

3.79.2 Constructor & Destructor Documentation

3.79.2.1 YMacroRecorder::YMacroRecorder ( ) [inline],[protected]

Constructor

Definition at line 47 of file YMacroRecorder.h.

3.79.2.2 virtual YMacroRecorder::~YMacroRecorder ( ) [inline],[virtual]

Destructor

Definition at line 53 of file YMacroRecorder.h.

3.79.3 Member Function Documentation

3.79.3.1 virtual void YMacroRecorder::endRecording ( ) [pure virtual]

End recording and close the current macro file (if there is any).

3.79.3.2 virtual void YMacroRecorder::record (const std::string & *macroFileName*) [pure virtual]

Start recording a macro to the specified file.

3.79.3.3 virtual bool YMacroRecorder::recording ( ) const [pure virtual]

Return 'true' if a macro is currently being recorded.

3.79.3.4 virtual void YMacroRecorder::recordMakeScreenShot (bool *enabled* = false, const std::string & *filename* = std::string()) [pure virtual]

Record a "UI::MakeScreenShot()" statement.

If 'enabled' is 'false', this statement will be commented out. If no file name is given, a default file name (with auto-increment) will be used.

3.79.3.5 virtual void YMacroRecorder::recordWidgetProperty (YWidget * *widget*, const char * *propertyName*) [pure virtual]

Record one widget property.

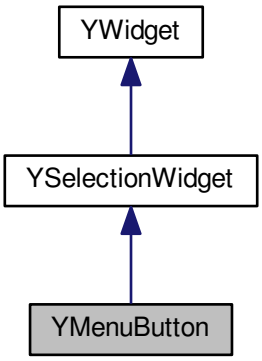
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMacroRecorder.h

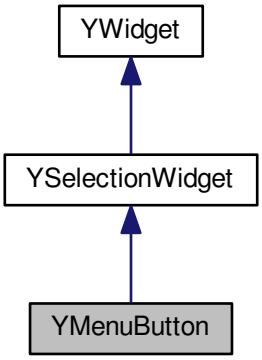
3.80 YMenuButton Class Reference

```
#include <YMenuButton.h>
```

Inheritance diagram for YMenuButton:



Collaboration diagram for YMenuButton:



Public Member Functions

- virtual ~YMenuButton ()
- virtual const char * widgetClass () const
- virtual void rebuildMenuTree ()=0
- virtual void addItems (const YItemCollection &itemCollection)
- virtual void addItem (YItem *item_disown)
- virtual void deleteAllItems ()
- void resolveShortcutConflicts ()
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Protected Member Functions

- YMenuButton (YWidget *parent, const std::string &label)
- YMenuItem * findMenuItem (int index)
- YMenuItem * findMenuItem (int index, YItemConstIterator begin, YItemConstIterator end)
- YMenuItem * itemAt (int index)

3.80.1 Detailed Description

MenuButton: Similar to PushButton, but with several actions: Upon clicking on a MenuButton (or activating it with the keyboard), a pop-up menu opens where the user can activate an action. Menu items in that pop-up menu can have submenus (that will pop up in separate pop-up menus).

Internally, this widget is more similar to the Tree widget. The difference is that it does not keep a "selected" status, but triggers an action right away, just like a PushButton. Like PushButton, MenuButton sends an event right away when the user selects an item (clicks on a menu item or activates it with the keyboard). Items that have a submenu never send an event, they simply open their submenu when activated.

Definition at line 48 of file YMenuButton.h.

3.80.2 Constructor & Destructor Documentation

3.80.2.1 YMenuButton::YMenuButton (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

'label' is the user-visible text on the button (not above it like all other SelectionWidgets).

Definition at line 46 of file YMenuButton.cc.

3.80.2.2 YMenuButton::~YMenuButton ( ) [virtual]

Destructor.

Definition at line 55 of file YMenuButton.cc.

3.80.3 Member Function Documentation

3.80.3.1 void YMenuButton::addItem (YItem * *item_disown*) [virtual]

Add one item. This widget assumes ownership of the item object and will delete it in its destructor.

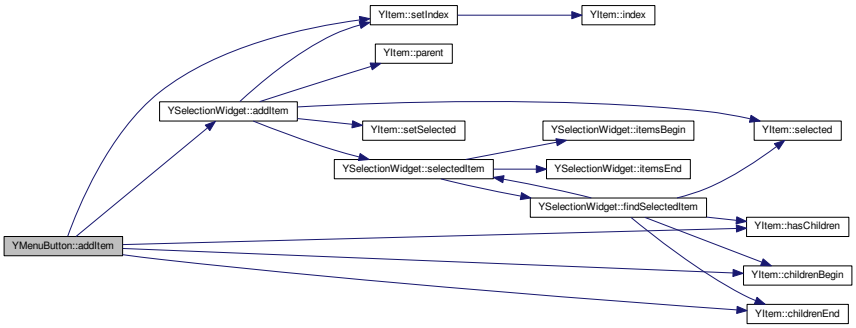
This reimplementation will an index to the item that is unique for all items in this MenuButton. That index can be used later with findMenuItem() to find the item by that index.

Reimplemented from YSelectionWidget.

Reimplemented from YSelectionWidget.

Definition at line 71 of file YMenuButton.cc.

Here is the call graph for this function:



3.80.3.2 void YMenuButton::addItems (const YItemCollection & *itemCollection*) [virtual]

Add multiple items. For some UIs, this can be more efficient than calling addItem() multiple times. This function also automatically calls resolveShortcutConflicts() and rebuildMenuTree() at the end.

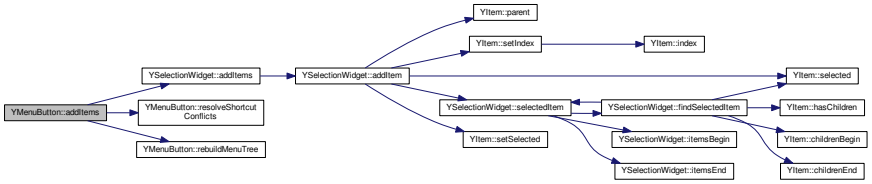
Derived classes can overwrite this function, but they should call this base class function at the end of the new implementation.

Reimplemented from YSelectionWidget.

Reimplemented from YSelectionWidget.

Definition at line 62 of file YMenuButton.cc.

Here is the call graph for this function:



3.80.3.3 void YMenuButton::deleteAllItems ( ) [virtual]

Delete all items.

Reimplemented from YSelectionWidget.

Reimplemented from YSelectionWidget.

Definition at line 97 of file YMenuButton.cc.

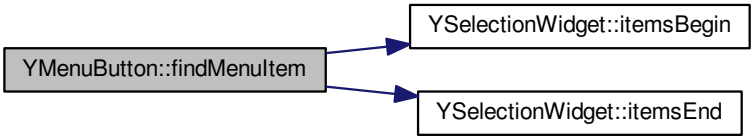
Here is the call graph for this function:

3.80.3.4 YMenuItem * YMenuButton::findMenuItem (int *index*) [protected]

Recursively find the first menu item with the specified index. Returns 0 if there is no such item.

Definition at line 105 of file YMenuButton.cc.

Here is the call graph for this function:



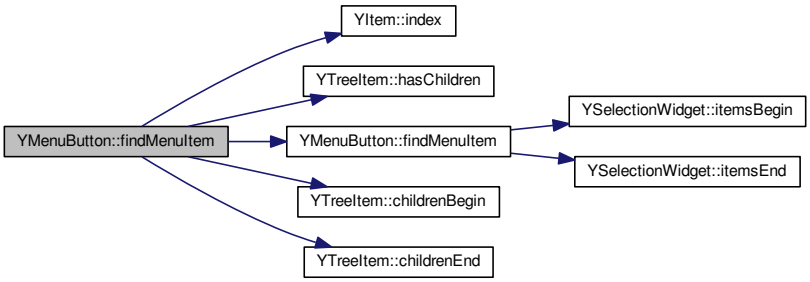
3.80.3.5 YMenuItem * YMenuButton::findMenuItem (int *index*, YItemConstIterator *begin*, YItemConstIterator *end*) [protected]

Recursively find the first menu item with the specified index from iterator 'begin' to iterator 'end'.

Returns 0 if there is no such item.

Definition at line 112 of file YMenuButton.cc.

Here is the call graph for this function:



3.80.3.6 YPropertyValue YMenuButton::getProperty (const std::string & *propertyName*) [virtual]

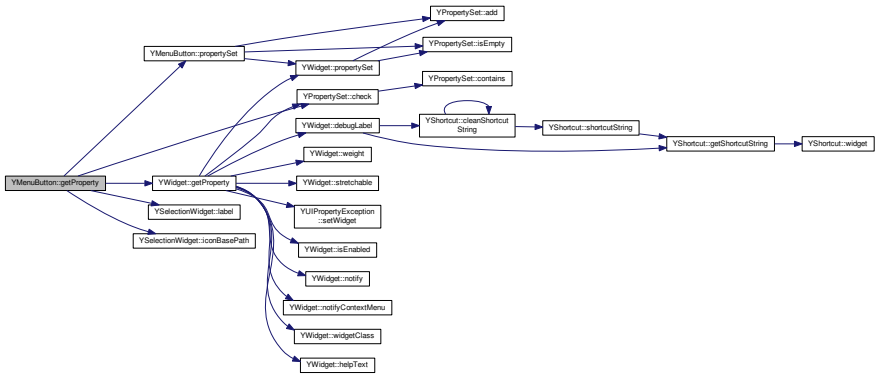
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 192 of file YMenuButton.cc.

Here is the call graph for this function:



3.80.3.7 YMenuItem* YMenuButton::itemAt (int *index*) [inline], [protected]

Alias for findMenuItem(). Reimplemented to ensure consistent behaviour with YSelectionWidget::itemAt().

Definition at line 170 of file YMenuButton.h.

Here is the call graph for this function:

3.80.3.8 const YPropertySet & YMenuButton::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 153 of file YMenuButton.cc.

Here is the call graph for this function:

3.80.3.9 virtual void YMenuButton::rebuildMenuTree ( ) [pure virtual]

Rebuild the displayed menu tree from the internally stored YMenuItems.

The application should call this (once) after all items have been added with addItem(). YMenuButton::addItems() calls this automatically.

Derived classes are required to implement this.

3.80.3.10 void YMenuButton::resolveShortcutConflicts ( )

Resolve keyboard shortcut conflicts: Change shortcuts of menu items if there are duplicates in the respective menu level.

This has to be called after all items are added, but before rebuildMenuTree() (see above). YMenuButton::addItems() calls this automatically.

Definition at line 138 of file YMenuButton.cc.

3.80.3.11 bool YMenuButton::setProperty (const std::string & *propertyName,* const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

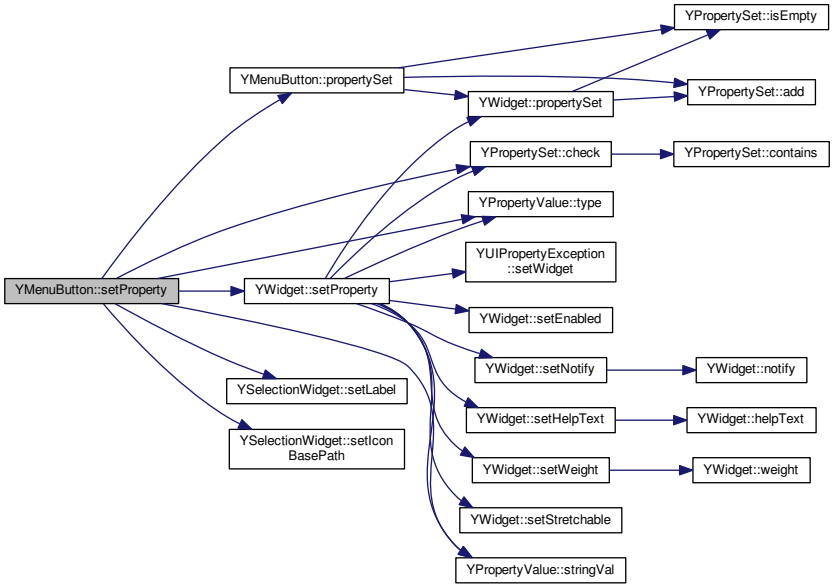
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 175 of file YMenuButton.cc.

Here is the call graph for this function:



3.80.3.12 virtual const char* YMenuButton::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YSelectionWidget.

Definition at line 69 of file YMenuButton.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMenuButton.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMenuButton.cc

3.81 YMenuButtonPrivate Struct Reference

Public Attributes

- int **nextSerialNo**

3.81.1 Detailed Description

Definition at line 34 of file YMenuButton.cc.

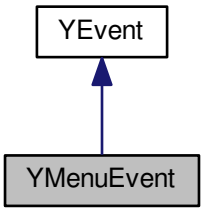
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMenuButton.cc

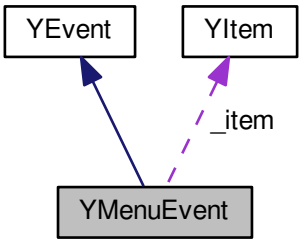
3.82 YMenuEvent Class Reference

```
#include <YEvent.h>
```

Inheritance diagram for YMenuEvent:



Collaboration diagram for YMenuEvent:



Public Member Functions

- **YMenuEvent** (YItem *item)
- **YMenuEvent** (const char *id)
- **YMenuEvent** (const std::string &id)
- virtual YItem * item () const
- std::string id () const

Protected Member Functions

- virtual ~YMenuEvent ()

Protected Attributes

- YItem * **_item**
- std::string **_id**

Additional Inherited Members

3.82.1 Detailed Description

Event to be returned upon menu selection.

Definition at line 256 of file YEvent.h.

3.82.2 Constructor & Destructor Documentation

3.82.2.1 virtual YMenuEvent::~YMenuEvent ( ) [inline], [protected], [virtual]

Protected destructor - events can only be deleted via YDialog::deleteEvent(). The associated dialog will take care of this event and delete it when appropriate.

Definition at line 289 of file YEvent.h.

3.82.3 Member Function Documentation

3.82.3.1 std::string YMenuEvent::id ( ) const [inline]

Return the string ID of this event. This will be an empty string if the event was constructed with a YItem.

Definition at line 280 of file YEvent.h.

3.82.3.2 virtual YItem* YMenuEvent::item ( ) const [inline], [virtual]

Return the YItem that corresponds to this event or 0 if the event was constructed with a string ID.

Reimplemented from YEvent.

Reimplemented from YEvent.

Definition at line 274 of file YEvent.h.

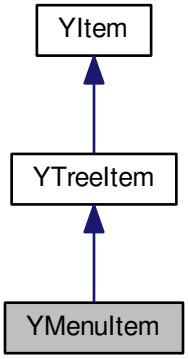
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEvent.h

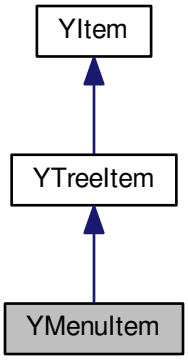
3.83 YMenuItem Class Reference

```
#include <YMenuItem.h>
```

Inheritance diagram for YMenuItem:



Collaboration diagram for YMenuItem:



Public Member Functions

- YMenuItem (const std::string &label)
- **YMenuItem** (const std::string &label, const std::string &iconName)
- YMenuItem (YMenuItem *parent, const std::string &label)
- **YMenuItem** (YMenuItem *parent, const std::string &label, const std::string &iconName)
- virtual ~YMenuItem ()
- YMenuItem * parent () const

3.83.1 Detailed Description

Item class for menu items.

Definition at line 35 of file YMenuItem.h.

3.83.2 Constructor & Destructor Documentation

3.83.2.1 YMenuItem::YMenuItem (const std::string & *label*) [inline]

Constructors for toplevel items.

Definition at line 41 of file YMenuItem.h.

3.83.2.2 YMenuItem::YMenuItem (YMenuItem * *parent*, const std::string & *label*) [inline]

Constructors for items that have a parent item.

They will automatically register this item with the parent item. The parent assumes ownership of this item and will delete it in its (the parent's) destructor.

Definition at line 57 of file YMenuItem.h.

3.83.2.3 virtual YMenuItem::~YMenuItem ( ) [inline],[virtual]

Destructor.

This will delete all children.

Definition at line 73 of file YMenuItem.h.

3.83.3 Member Function Documentation

3.83.3.1 YMenuItem* YMenuItem::parent ( ) const [inline],[virtual]

Returns this item's parent item or 0 if it is a toplevel item.

Reimplemented from YTreeItem.

Definition at line 79 of file YMenuItem.h.

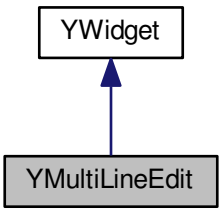
Here is the call graph for this function:

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

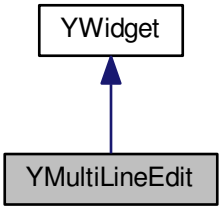
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMenuItem.h

3.84 YMultiLineEdit Class Reference

Inheritance diagram for YMultiLineEdit:



Collaboration diagram for YMultiLineEdit:



Public Member Functions

- virtual ~YMultiLineEdit ()
- virtual const char * widgetClass () const
- virtual std::string value ()=0
- virtual void setValue (const std::string &text)=0
- std::string label () const
- virtual void setLabel (const std::string &label)
- int inputMaxLength () const
- virtual void setInputMaxLength (int numberOfChars)
- int defaultVisibleLines () const
- virtual void setDefaultVisibleLines (int newVisibleLines)

- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)
- const char * userInputProperty ()

Protected Member Functions

- YMultiLineEdit (YWidget *parent, const std::string &label)

3.84.1 Detailed Description

Definition at line 33 of file YMultiLineEdit.h.

3.84.2 Constructor & Destructor Documentation

3.84.2.1 YMultiLineEdit::YMultiLineEdit (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

Definition at line 52 of file YMultiLineEdit.cc.

Here is the call graph for this function:

3.84.2.2 YMultiLineEdit::~YMultiLineEdit ( ) [virtual]

Destructor.

Definition at line 63 of file YMultiLineEdit.cc.

3.84.3 Member Function Documentation

3.84.3.1 int YMultiLineEdit::defaultVisibleLines ( ) const

Return the number of input lines that are visible by default.

This is what the widget would like to get (which will be reflected by preferredHeight()), not what it currently actually has due to layout constraints.

Definition at line 93 of file YMultiLineEdit.cc.

3.84.3.2 YPropertyValue YMultiLineEdit::getProperty (const std::string & *propertyName*) [virtual]

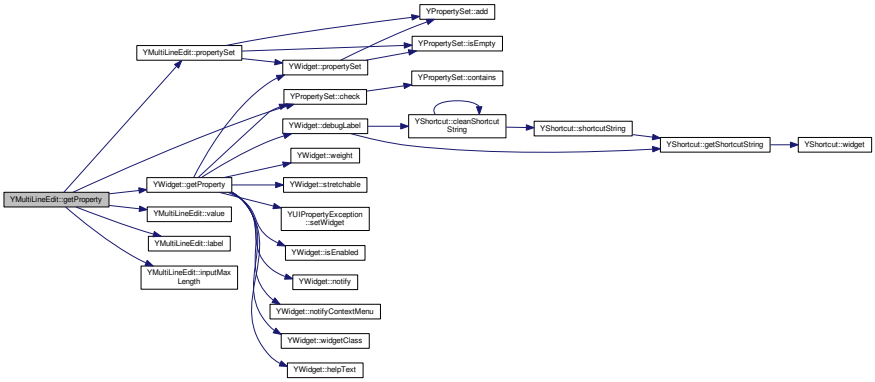
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 145 of file YMultiLineEdit.cc.

Here is the call graph for this function:



3.84.3.3 int YMultiLineEdit::inputMaxLength ( ) const

The maximum input length, i.e., the maximum number of characters the user can enter. -1 means no limit.

Definition at line 81 of file YMultiLineEdit.cc.

3.84.3.4 std::string YMultiLineEdit::label ( ) const

Get the label (the caption above the MultiLineEdit).

Definition at line 69 of file YMultiLineEdit.cc.

3.84.3.5 const YPropertySet & YMultiLineEdit::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 106 of file YMultiLineEdit.cc.

Here is the call graph for this function:

3.84.3.6 void YMultiLineEdit::setDefaultVisibleLines (int *newVisibleLines*) [virtual]

Set the number of input lines that are visible by default.

This is what the widget would like to get (which will be reflected by preferredHeight()), not what it currently actually has due to layout constraints.

Notice that since a MultiLineEdit is stretchable in both dimensions, it might get more or less screen space, depending on the layout. This value is only meaningful if there are no other layout constraints.

Changing this value will not trigger a re-layout.

Derived classes can overwrite this function (but should call this base class function in the new function implementation), but it will normally be sufficient to query defaultVisibleLines() in preferredHeight().

Definition at line 99 of file YMultiLineEdit.cc.

3.84.3.7 void YMultiLineEdit::setInputMaxLength (int *numberOfChars*) [virtual]

Set the maximum input length, i.e., the maximum number of characters the user can enter. -1 means no limit.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 87 of file YMultiLineEdit.cc.

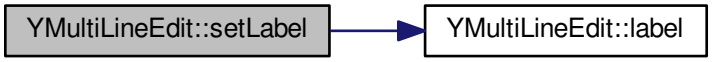
3.84.3.8 void YMultiLineEdit::setLabel (const std::string & *label*) [virtual]

Set the label (the caption above the MultiLineEdit).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 75 of file YMultiLineEdit.cc.

Here is the call graph for this function:



3.84.3.9 bool YMultiLineEdit::setProperty (const std::string & *propertyName,* const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

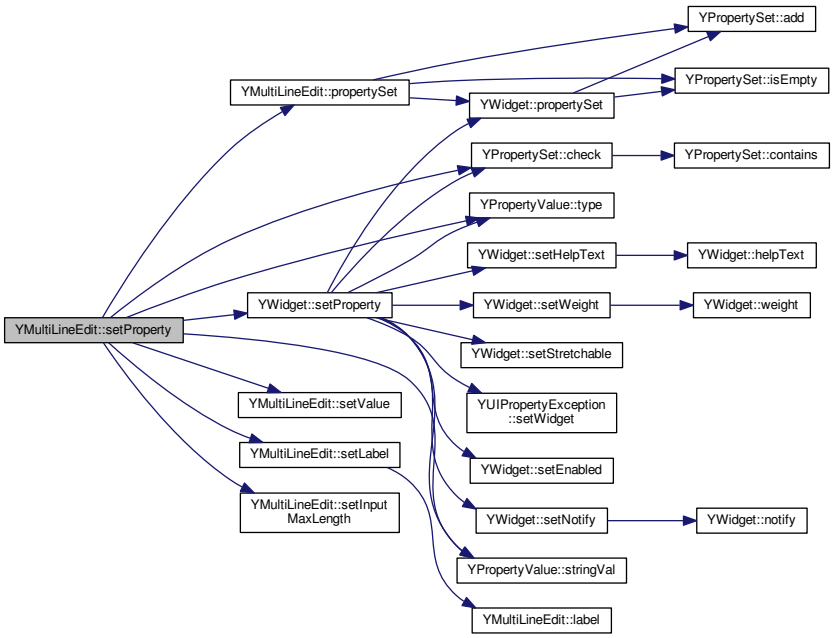
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 128 of file YMultiLineEdit.cc.

Here is the call graph for this function:



3.84.3.10 virtual void YMultiLineEdit::setShortcutString (const std::string & *str*) [inline], [virtual]

Set the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 166 of file YMultiLineEdit.h.

Here is the call graph for this function:

3.84.3.11 virtual void YMultiLineEdit::setValue (const std::string & *text*) [pure virtual]

Set the current value (the text entered by the user or set from the outside) of this MultiLineEdit.

Derived classes are required to implement this.

3.84.3.12 virtual std::string YMultiLineEdit::shortcutString ( ) const [inline], [virtual]

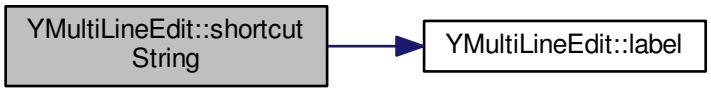
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 159 of file YMultiLineEdit.h.

Here is the call graph for this function:



3.84.3.13 const char* YMultiLineEdit::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 173 of file YMultiLineEdit.h.

3.84.3.14 virtual std::string YMultiLineEdit::value ( ) [pure virtual]

Get the current value (the text entered by the user or set from the outside) of this MultiLineEdit.

Derived classes are required to implement this.

3.84.3.15 virtual const char* YMultiLineEdit::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 51 of file YMultiLineEdit.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMultiLineEdit.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMultiLineEdit.cc

3.85 YMultiLineEditPrivate Struct Reference

Public Member Functions

- **YMultiLineEditPrivate** (const std::string &label)

Public Attributes

- std::string **label**
- int **inputMaxLength**
- int **defaultVisibleLines**

3.85.1 Detailed Description

Definition at line 36 of file YMultiLineEdit.cc.

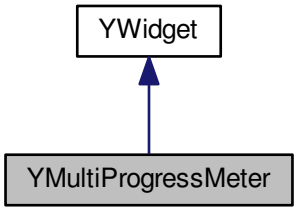
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMultiLineEdit.cc

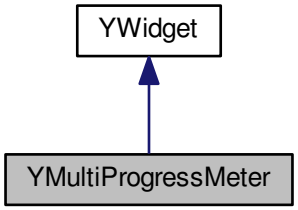
3.86 YMultiProgressMeter Class Reference

```
#include <YMultiProgressMeter.h>
```

Inheritance diagram for YMultiProgressMeter:



Collaboration diagram for YMultiProgressMeter:



Public Member Functions

- virtual ~YMultiProgressMeter ()
- virtual const char * widgetClass () const
- YUIDimension dimension () const
- bool horizontal () const
- bool vertical () const
- int segments () const
- float maxValue (int segment) const
- float currentValue (int segment) const
- void setCurrentValue (int segment, float value)
- void setCurrentValues (const std::vector< float > &values)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- virtual void doUpdate ()=0

Protected Member Functions

- YMultiProgressMeter (YWidget *parent, YUIDimension dim, const std::vector< float > &maxValues)

3.86.1 Detailed Description

MultiProgressMeter: Progress bar with several segments that can indicate progress individually. This is useful to display progress of several activities that might not necessarily all be done in sequence.

A common example is installing packages from several CDs: Each CD would get a separate segment. Each segment's size would be proportional to the amount of data to be installed from that CD. This visualizes at the same time (a) how many CDs are involved (b) how much in proportion is to be expected from each CD (c) whether or not a specific CD is finished.

Visual example (horizontal MultiProgressMeter):

```
[=============...] [===] [......] [.]
```

This corresponds to 4 CDs:

CD #1: A lot of packages are to be installed from this CD, and a fair amount of those are already installed, but some are still missing. CD #2: Some packages were installed from this, but this CD is finished. CD #3: Quite some packages are to be installed from this CD. CD #4: Very few packages are to be installed from this CD.

As can be seen from this simple example, this widget can visualize a lot of complex information at the same time in a very natural way.

This is an optional widget, i.e. not all UIs support it.

Definition at line 64 of file YMultiProgressMeter.h.

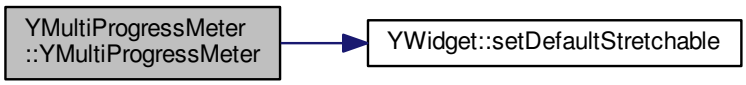
3.86.2 Constructor & Destructor Documentation

3.86.2.1 YMultiProgressMeter::YMultiProgressMeter (YWidget * *parent*, YUIDimension *dim*, const std::vector< float > & *maxValues*) [protected]

Constructor

Definition at line 54 of file YMultiProgressMeter.cc.

Here is the call graph for this function:



3.86.2.2 YMultiProgressMeter::~YMultiProgressMeter ( ) [virtual]

Destructor.

Definition at line 67 of file YMultiProgressMeter.cc.

3.86.3 Member Function Documentation

3.86.3.1 float YMultiProgressMeter::currentValue (int *segment*) const

Return the current value for the specified segment (counting from 0). If no value has been set yet, -1 is returned.

Definition at line 106 of file YMultiProgressMeter.cc.

3.86.3.2 YUIDimension YMultiProgressMeter::dimension ( ) const

Return the orientation of the MultiProgressBar.

Definition at line 74 of file YMultiProgressMeter.cc.

3.86.3.3 virtual void YMultiProgressMeter::doUpdate ( ) [pure virtual]

Notification that values have been updated and the widget needs to be redisplayed. Derived classes need to reimplement this.

3.86.3.4 YPropertyValue YMultiProgressMeter::getProperty (const std::string & *propertyName*) [virtual]

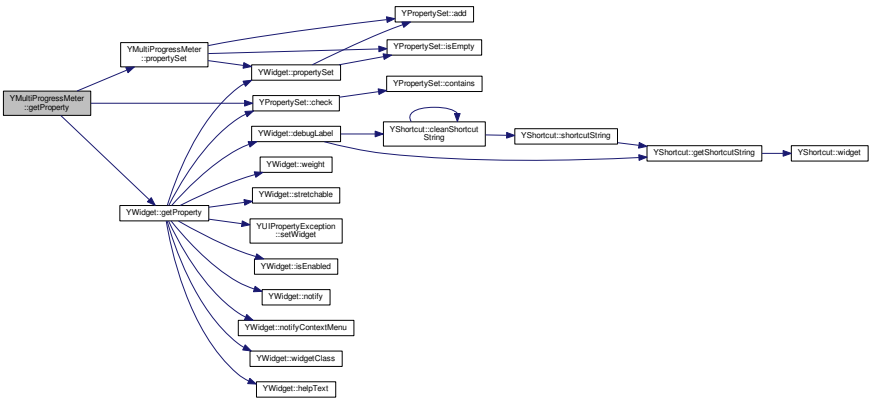
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 173 of file YMultiProgressMeter.cc.

Here is the call graph for this function:



3.86.3.5 bool YMultiProgressMeter::horizontal ( ) const

Return 'true' if the orientation is horizontal.

Definition at line 80 of file YMultiProgressMeter.cc.

3.86.3.6 float YMultiProgressMeter::maxValue (int *segment*) const

Return the maximum value for the specified segment (counting from 0).

Definition at line 98 of file YMultiProgressMeter.cc.

3.86.3.7 const YPropertySet & YMultiProgressMeter::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 140 of file YMultiProgressMeter.cc.

Here is the call graph for this function:

3.86.3.8 int YMultiProgressMeter::segments ( ) const

Return the number of segments.

Definition at line 92 of file YMultiProgressMeter.cc.

3.86.3.9 void YMultiProgressMeter::setCurrentValue (int *segment*, float *value*)

Set the current value for the specified segment. This must be in the range 0..maxValue(segment).

Remember to call doUpdate() after all changed values are set!

Definition at line 114 of file YMultiProgressMeter.cc.

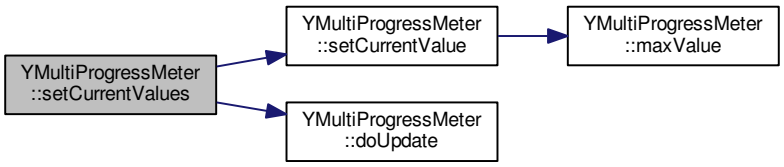
Here is the call graph for this function:

3.86.3.10 void YMultiProgressMeter::setCurrentValues (const std::vector< float > & *values*)

Set all current values and call doUpdate().

Definition at line 128 of file YMultiProgressMeter.cc.

Here is the call graph for this function:



3.86.3.11 bool YMultiProgressMeter::setProperty (const std::string & *propertyName,* const **YPropertyValue** & *val*) [virtual]

Set a property. Reimplemented from YWidget.

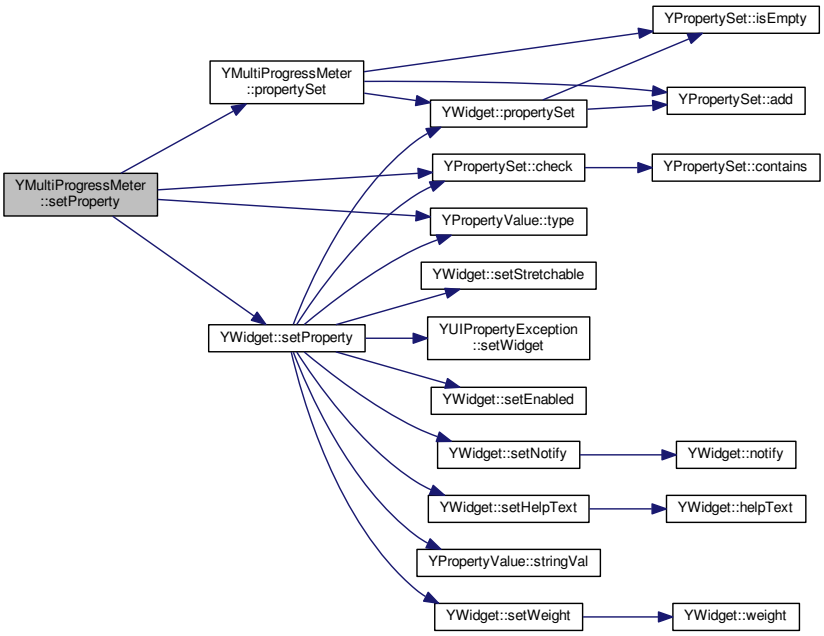
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 158 of file YMultiProgressMeter.cc.

Here is the call graph for this function:



3.86.3.12 bool YMultiProgressMeter::vertical ( ) const

Return 'true' if the orientation is vertical.

Definition at line 86 of file YMultiProgressMeter.cc.

3.86.3.13 virtual const char* YMultiProgressMeter::widgetClass ( ) const [inline], [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 84 of file YMultiProgressMeter.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMultiProgressMeter.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMultiProgressMeter.cc

3.87 YMultiProgressMeterPrivate Struct Reference

Public Member Functions

- **YMultiProgressMeterPrivate** (YUIDimension dim, const std::vector< float > &maxValues)

Public Attributes

- YUIDimension **dim**
- std::vector< float > **maxValues**
- std::vector< float > **currentValues**

3.87.1 Detailed Description

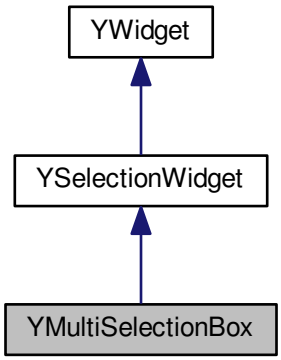
Definition at line 33 of file YMultiProgressMeter.cc.

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

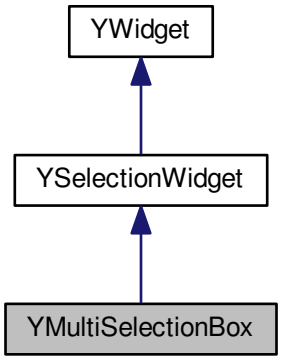
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMultiProgressMeter.cc

3.88 YMultiSelectionBox Class Reference

Inheritance diagram for YMultiSelectionBox:



Collaboration diagram for YMultiSelectionBox:



Public Member Functions

- virtual ~YMultiSelectionBox ()
- virtual const char * widgetClass () const
- bool shrinkable () const
- virtual void setShrinkable (bool shrinkable=true)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- const char * userInputProperty ()
- virtual YItem * currentItem ()=0
- virtual void setCurrentItem (YItem *item)=0
- virtual void saveUserInput (YMacroRecorder *macroRecorder)

Protected Member Functions

- YMultiSelectionBox (YWidget *parent, const std::string &label)

3.88.1 Detailed Description

Definition at line 33 of file YMultiSelectionBox.h.

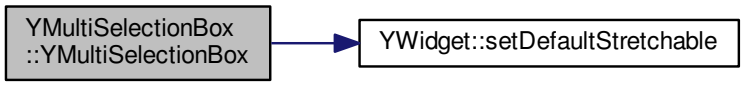
3.88.2 Constructor & Destructor Documentation

3.88.2.1 YMultiSelectionBox::YMultiSelectionBox (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

Definition at line 46 of file YMultiSelectionBox.cc.

Here is the call graph for this function:



3.88.2.2 YMultiSelectionBox::~YMultiSelectionBox ( ) [virtual]

Destructor.

Definition at line 59 of file YMultiSelectionBox.cc.

3.88.3 Member Function Documentation

3.88.3.1 virtual YItem* YMultiSelectionBox::currentItem ( ) [pure virtual]

Return the the item that currently has the keyboard focus or 0 if no item currently has the keyboard focus.

Notice that for a MultiSelectionBox the current item is not necessarily selected, i.e., its check box may or may not be checked.

Derived classes are required to implement this function.

3.88.3.2 YPropertyValue YMultiSelectionBox::getProperty (const std::string & *propertyName*) [virtual]

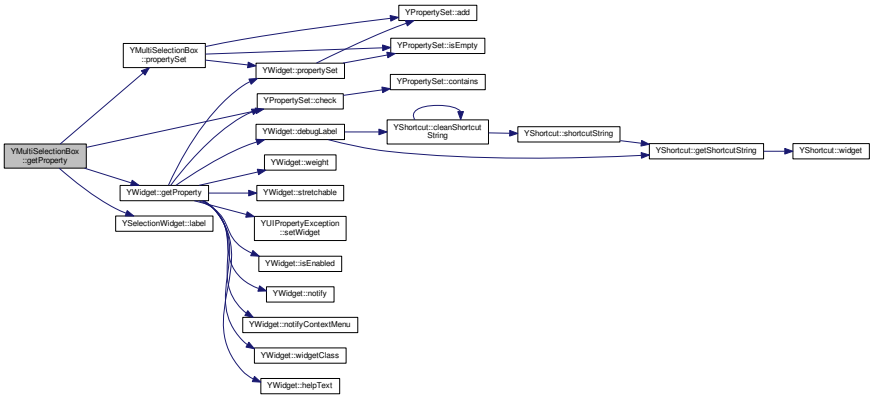
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 122 of file YMultiSelectionBox.cc.

Here is the call graph for this function:



3.88.3.3 const YPropertySet & YMultiSelectionBox::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 78 of file YMultiSelectionBox.cc.

Here is the call graph for this function:

3.88.3.4 void YMultiSelectionBox::saveUserInput (YMacroRecorder * *macroRecorder*) [virtual]

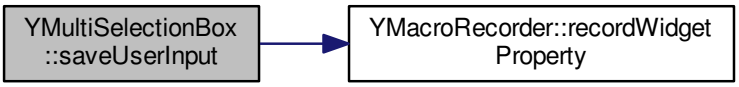
Save the widget's user input to a macro recorder.

Reimplemented from YWidget because two properties need to be recorded.

Reimplemented from YWidget.

Definition at line 139 of file YMultiSelectionBox.cc.

Here is the call graph for this function:



3.88.3.5 virtual void YMultiSelectionBox::setCurrentItem (YItem * *item*) [pure virtual]

Set the keyboard focus to the specified item. 0 means clear the keyboard focus.

Notice that for a MultiSelectionBox the current item is not necessarily selected, i.e., its check box may or may not be checked. Use selectItem() for that.

Also notice that selectItem() does not make that newly selected item the current item.

Derived classes are required to implement this function.

3.88.3.6 bool YMultiSelectionBox::setProperty (const std::string & *propertyName,* const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

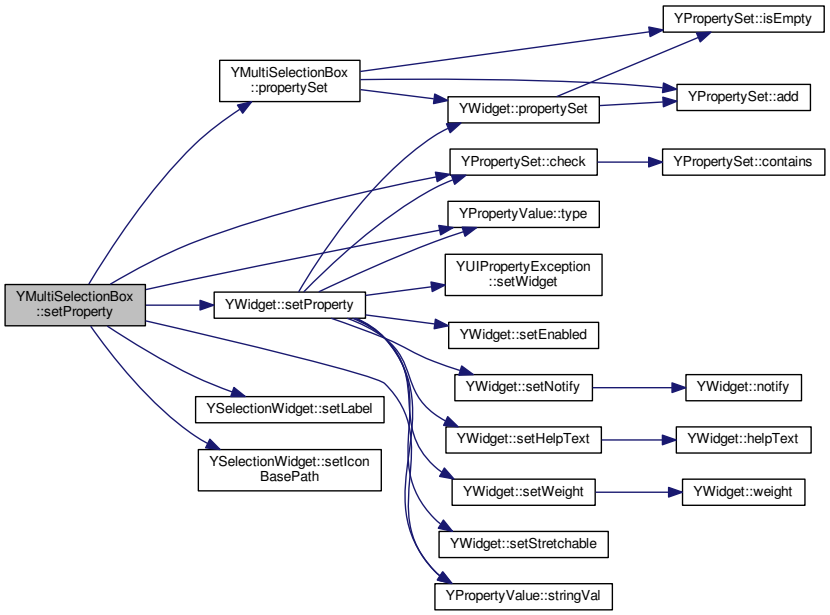
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 103 of file YMultiSelectionBox.cc.

Here is the call graph for this function:



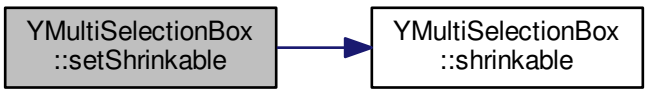
3.88.3.7 void YMultiSelectionBox::setShrinkable (bool *shrinkable* = `true`) [virtual]

Make this MultiSelectionBox very small. This will take effect only upon the next geometry management run.

Derived classes can overwrite this, but should call this base class function in the new function.

Definition at line 71 of file YMultiSelectionBox.cc.

Here is the call graph for this function:



3.88.3.8 bool YMultiSelectionBox::shrinkable ( ) const

Return 'true' if this MultiSelectionBox should be very small.

Definition at line 65 of file YMultiSelectionBox.cc.

3.88.3.9 const char* YMultiSelectionBox::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 100 of file YMultiSelectionBox.h.

3.88.3.10 virtual const char* YMultiSelectionBox::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YSelectionWidget.

Definition at line 51 of file YMultiSelectionBox.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMultiSelectionBox.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMultiSelectionBox.cc

3.89 YMultiSelectionBoxPrivate Struct Reference

Public Attributes

- bool **shrinkable**

3.89.1 Detailed Description

Definition at line 35 of file YMultiSelectionBox.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YMultiSelectionBox.cc

3.90 YOptionalWidgetFactory Class Reference

```
#include <YOptionalWidgetFactory.h>
```

Public Member Functions

- virtual bool **hasWizard** ()
- virtual YWizard * **createWizard** (YWidget *parent, const std::string &backButtonLabel, const std::string &abortButtonLabel, const std::string &nextButtonLabel, YWizardMode wizardMode=YWizardMode_Standard)
- virtual bool **hasDumbTab** ()
- virtual YDumbTab * **createDumbTab** (YWidget *parent)
- virtual bool **hasSlider** ()
- virtual YSlider * **createSlider** (YWidget *parent, const std::string &label, int minVal, int maxVal, int initialVal)
- virtual bool **hasDateField** ()
- virtual YDateField * **createDateField** (YWidget *parent, const std::string &label)

- virtual bool **hasTimeField** ()
- virtual YTimeField ∗ **createTimeField** (YWidget ∗parent, const std::string &label)
- virtual bool **hasBarGraph** ()
- virtual YBarGraph ∗ **createBarGraph** (YWidget ∗parent)
- virtual bool **hasPatternSelector** ()
- virtual YWidget ∗ **createPatternSelector** (YWidget ∗parent, long modeFlags=0)
- virtual bool **hasSimplePatchSelector** ()
- virtual YWidget ∗ **createSimplePatchSelector** (YWidget ∗parent, long modeFlags=0)
- virtual bool **hasMultiProgressMeter** ()
- YMultiProgressMeter ∗ **createHMultiProgressMeter** (YWidget ∗parent, const std::vector< float > &maxValues)
- YMultiProgressMeter ∗ **createVMultiProgressMeter** (YWidget ∗parent, const std::vector< float > &maxValues)
- virtual YMultiProgressMeter ∗ **createMultiProgressMeter** (YWidget ∗parent, YUIDimension dim, const std::vector< float > &maxValues)
- virtual bool **hasPartitionSplitter** ()
- virtual YPartitionSplitter ∗ **createPartitionSplitter** (YWidget ∗parent, int usedSize, int totalFreeSize, int newPartSize, int minNewPartSize, int minFreeSize, const std::string &usedLabel, const std::string &freeLabel, const std::string &newPartLabel, const std::string &freeFieldLabel, const std::string &newPartFieldLabel)
- virtual bool **hasDownloadProgress** ()
- virtual YDownloadProgress ∗ **createDownloadProgress** (YWidget ∗parent, const std::string &label, const std::string &filename, YFileSize_t expectedFileSize)
- bool **hasDummySpecialWidget** ()
- YWidget ∗ **createDummySpecialWidget** (YWidget ∗parent)
- virtual bool **hasTimezoneSelector** ()
- virtual YTimezoneSelector ∗ **createTimezoneSelector** (YWidget ∗parent, const std::string &pixmap, const std::map< std::string, std::string > &timezones)
- virtual bool **hasGraph** ()
- virtual YGraph ∗ **createGraph** (YWidget ∗parent, const std::string &filename, const std::string &layoutAlgorithm)
- virtual YGraph ∗ **createGraph** (YWidget ∗parent, void ∗graph)
- virtual bool **hasContextMenu** ()

Protected Member Functions

- YOptionalWidgetFactory ()
- virtual ∼YOptionalWidgetFactory ()

Friends

- class **YUI**

3.90.1 Detailed Description

Abstract widget factory for optional ("special") widgets.

Remember to always check with the corresponding "has..()" method if the current UI actually provides the requested widget. Otherwise the "create...()" method will throw an exception.

Definition at line 56 of file YOptionalWidgetFactory.h.

3.90.2 Constructor & Destructor Documentation

3.90.2.1 YOptionalWidgetFactory::YOptionalWidgetFactory ( ) [protected]

Constructor.

Use YUI::optionalWidgetFactory() to get the singleton for this class.

Definition at line 38 of file YOptionalWidgetFactory.cc.

3.90.2.2 YOptionalWidgetFactory::~YOptionalWidgetFactory ( ) [protected], [virtual]

Destructor.

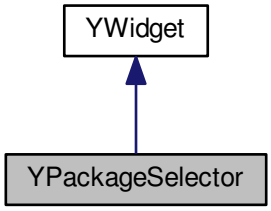
Definition at line 43 of file YOptionalWidgetFactory.cc.

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

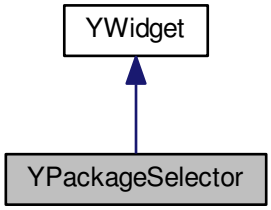
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YOptionalWidgetFactory.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YOptionalWidgetFactory.cc

3.91 YPackageSelector Class Reference

Inheritance diagram for YPackageSelector:



Collaboration diagram for YPackageSelector:



Public Member Functions

- virtual const char ∗ widgetClass () const
- bool testMode () const
- bool **onlineUpdateMode** () const
- bool **updateMode** () const
- bool **searchMode** () const
- bool **summaryMode** () const
- bool **repoMode** () const
- bool **repoMgrEnabled** () const
- bool **confirmUnsupported** () const

Protected Member Functions

- YPackageSelector (YWidget ∗parent, long modeFlags=0)

Protected Attributes

- long **_modeFlags**

3.91.1 Detailed Description

Definition at line 40 of file YPackageSelector.h.

3.91.2 Constructor & Destructor Documentation

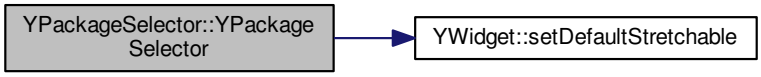
3.91.2.1 YPackageSelector::YPackageSelector (YWidget ∗ *parent*, long *modeFlags* = 0) [protected]

Constructor.

'modeFlags' are flags determining which modes to use, ORed together: YPkg_OnlineUpdateMode | YPkg_TestMode

Definition at line 32 of file YPackageSelector.cc.

Here is the call graph for this function:



3.91.3 Member Function Documentation

3.91.3.1 bool YPackageSelector::testMode ( ) const [inline]

Check for the various modes.

Definition at line 61 of file YPackageSelector.h.

3.91.3.2 virtual const char* YPackageSelector::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 56 of file YPackageSelector.h.

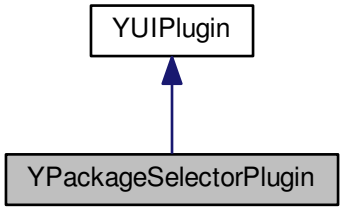
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPackageSelector.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPackageSelector.cc

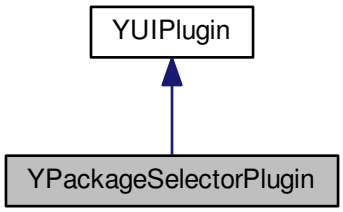
3.92 YPackageSelectorPlugin Class Reference

```
#include <YPackageSelectorPlugin.h>
```

Inheritance diagram for YPackageSelectorPlugin:



Collaboration diagram for YPackageSelectorPlugin:



Public Member Functions

- virtual YPackageSelector ∗ createPackageSelector (YWidget ∗parent, long modeFlags=0)=0

Protected Member Functions

- YPackageSelectorPlugin (const char ∗pluginLibBaseName)
- virtual ∼YPackageSelectorPlugin ()

3.92.1 Detailed Description

Abstract base class for simplified access to UI plugins for package selection.

Definition at line 38 of file YPackageSelectorPlugin.h.

3.92.2 Constructor & Destructor Documentation

3.92.2.1 YPackageSelectorPlugin::YPackageSelectorPlugin (const char ∗ *pluginLibBaseName*) [inline],[protected]

Constructor: Load the specified plugin library from the standard UI plugin directory (/usr/lib/yui/).

Definition at line 45 of file YPackageSelectorPlugin.h.

3.92.2.2 virtual YPackageSelectorPlugin::∼YPackageSelectorPlugin ( ) [inline],[protected],[virtual]

Destructor. Calls dlclose() which will unload the plugin library if it is no longer used, i.e. if the reference count dlopen() uses reaches 0.

Definition at line 52 of file YPackageSelectorPlugin.h.

3.92.3 Member Function Documentation

3.92.3.1 virtual YPackageSelector* YPackageSelectorPlugin::createPackageSelector (YWidget * *parent*, long *modeFlags* = 0) [pure virtual]

Create a package selector. Derived classes need to implement this.

This might return 0 if the plugin lib could not be loaded or if the appropriate symbol could not be located in the plugin lib.

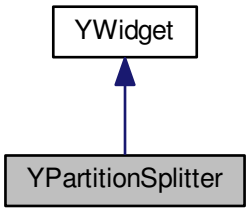
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPackageSelectorPlugin.h

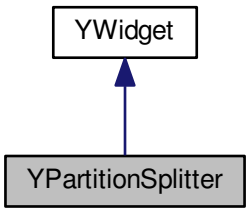
3.93 YPartitionSplitter Class Reference

`#include <YPartitionSplitter.h>`

Inheritance diagram for YPartitionSplitter:



Collaboration diagram for YPartitionSplitter:



Public Member Functions

- virtual ~YPartitionSplitter ()
- virtual const char * widgetClass () const

- virtual int value ()=0
- virtual void setValue (int newValue)=0
- int **usedSize** () const
- int **totalFreeSize** () const
- int **minFreeSize** () const
- int **maxFreeSize** () const
- int **freeSize** ()
- int **newPartSize** ()
- int **minNewPartSize** () const
- int **maxNewPartSize** () const
- std::string **usedLabel** () const
- std::string **freeLabel** () const
- std::string **newPartLabel** () const
- std::string **freeFieldLabel** () const
- std::string **newPartFieldLabel** () const
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- const char * userInputProperty ()

Protected Member Functions

- YPartitionSplitter (YWidget *parent, int usedSize, int totalFreeSize, int newPartSize, int minNewPartSize, int minFreeSize, const std::string &usedLabel, const std::string &freeLabel, const std::string &newPartLabel, const std::string &freeFieldLabel, const std::string &newPartFieldLabel)

3.93.1 Detailed Description

PartitionSplitter: A (very custom) widget for easily splitting one existing partition into two.

Layout:

```
+--------------------+---------------+--------------------------+
| Old Partition      | Old Partition | New Partition            |
| used               | free          |                          |
+--------------------+---------------+--------------------------+

Old Partition free                                  New Partition
[ 123 ] ================O================================ [ 123 ]
```

At the top, there is a BarGraph that dynamicylla displays the sizes in graphical form. Below are an IntField to the left and an IntField to the right, each with its respective label. Between the two IntFields there is a Slider.

The user can enter a value in either IntField or drag the slider. The other sub-widgets (including the BarGraph) will automatically be adjusted. Visually (in the BarGraph), the border between "old partition free" and "new partition" will move left and right. The border between "old partition used" and "old partition free" is static.

There are built-in (configurable) limits for the minimum sizes of "old partition free" and "new partition".

Definition at line 63 of file YPartitionSplitter.h.

3.93.2 Constructor & Destructor Documentation

3.93.2.1 YPartitionSplitter::YPartitionSplitter (YWidget * *parent*, int *usedSize*, int *totalFreeSize*, int *newPartSize*, int *minNewPartSize*, int *minFreeSize*, const std::string & *usedLabel*, const std::string & *freeLabel*, const std::string & *newPartLabel*, const std::string & *freeFieldLabel*, const std::string & *newPartFieldLabel*) [protected]

Constructor.

usedSize: Used size of the old partition (constant)

totalFreeSize: Total free size of the old partition before the split: OldPartitionFree + NewPartition

newPartSize': Initial size of the new partition

minNewPartSize: Miminum size of the new partition

minFreeSize: Minimum free size of the old partition

usedLabel: BarGraph label for the used part of the old partition

freeLabel: BarGraph label for the free part of the old partition

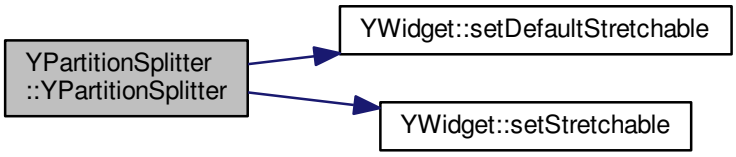
newPartLabel: BarGraph label for the new partition

freeFieldLabel: IntField label for the free part of the old partition

newPartFieldLabel: IntField label for the size of the new partition

Definition at line 69 of file YPartitionSplitter.cc.

Here is the call graph for this function:



3.93.2.2 YPartitionSplitter::~YPartitionSplitter ( ) [virtual]

Destructor.

Definition at line 99 of file YPartitionSplitter.cc.

3.93.3 Member Function Documentation

3.93.3.1 YPropertyValue YPartitionSplitter::getProperty (const std::string & *propertyName*) [virtual]

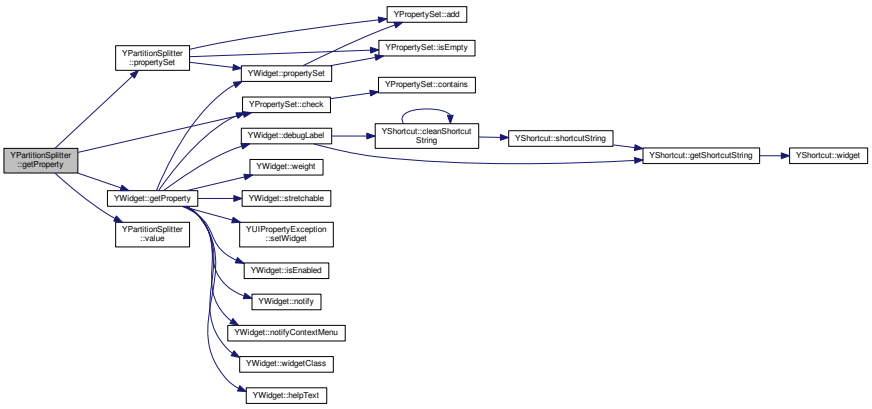
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 193 of file YPartitionSplitter.cc.

Here is the call graph for this function:



3.93.3.2 const YPropertySet & YPartitionSplitter::propertySet ( ) [virtual]

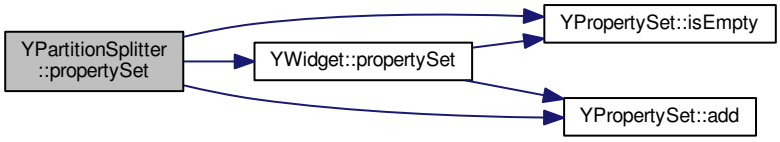
Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 160 of file YPartitionSplitter.cc.

Here is the call graph for this function:



3.93.3.3 bool YPartitionSplitter::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

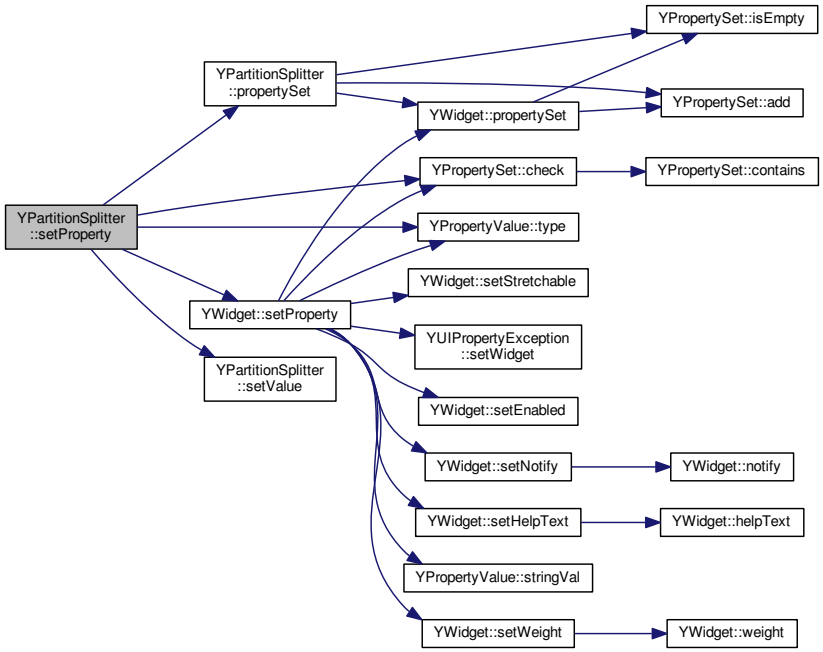
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 178 of file YPartitionSplitter.cc.

Here is the call graph for this function:



3.93.3.4 virtual void YPartitionSplitter::setValue (int *newValue*) [pure virtual]

Set the value (the size of the new partition).

Derived classes are required to implement this.

3.93.3.5 const char* YPartitionSplitter::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 181 of file YPartitionSplitter.h.

3.93.3.6 virtual int YPartitionSplitter::value ( ) [pure virtual]

The value of this PartitionSplitter: The size of the new partition.

Derived classes are required to implement this.

3.93.3.7 virtual const char* YPartitionSplitter::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 113 of file YPartitionSplitter.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPartitionSplitter.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPartitionSplitter.cc

3.94 YPartitionSplitterPrivate Struct Reference

Public Member Functions

- **YPartitionSplitterPrivate** (int usedSize, int totalFreeSize, int minNewPartSize, int minFreeSize, const std::string &usedLabel, const std::string &freeLabel, const std::string &newPartLabel, const std::string &freeFieldLabel, const std::string &newPartFieldLabel)

Public Attributes

- int **usedSize**
- int **totalFreeSize**
- int **minNewPartSize**
- int **minFreeSize**
- std::string **usedLabel**
- std::string **freeLabel**
- std::string **newPartLabel**
- std::string **freeFieldLabel**
- std::string **newPartFieldLabel**

3.94.1 Detailed Description

Definition at line 33 of file YPartitionSplitter.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPartitionSplitter.cc

3.95 YPath Class Reference

```
#include <YPath.h>
```

Public Member Functions

- YPath (const std::string &directory, const std::string &filename)
- ~YPath ()
- std::string path ()
- std::string dir ()

3.95.1 Detailed Description

Finds files (e.g. plugins or theme pixmaps) recursively inside a directory.

Definition at line 43 of file YPath.h.

3.95.2 Constructor & Destructor Documentation

3.95.2.1 YPath::YPath (const std::string & *directory*, const std::string & *filename*)

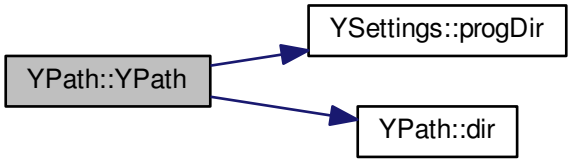
Constructor

to be called with the directory where to look inside and filename which to lookup.

YSettings::progSubDir will be preferred by the lookup.

Definition at line 47 of file YPath.cc.

Here is the call graph for this function:



3.95.2.2 YPath::~YPath ( )

Destructor

Definition at line 126 of file YPath.cc.

3.95.3 Member Function Documentation

3.95.3.1 std::string YPath::dir ( )

Returns the directory where the file is found; if not found just the subdir part (if there's any) of the filename given in constructor.

Definition at line 176 of file YPath.cc.

3.95.3.2 std::string YPath::path ( )

Returns the full path of the file if found; if not found just the filename given in constructor.

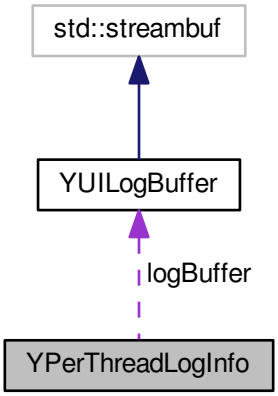
Definition at line 171 of file YPath.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPath.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPath.cc

3.96 YPerThreadLogInfo Struct Reference

Collaboration diagram for YPerThreadLogInfo:



Public Member Functions

- YPerThreadLogInfo ()
- ~YPerThreadLogInfo ()
- bool isThread (pthread_t otherThreadHandle)

Public Attributes

- pthread_t **threadHandle**
- YUILogBuffer **logBuffer**
- std::ostream **logStream**

3.96.1 Detailed Description

Helper class: Per-thread logging information.

Multiple threads can easily clobber each others' half-done logging. A naive approach to prevent this would be to lock a mutex when a thread starts logging and unlock it when it's done logging. But that "when it's done" condition might never come true. std::endl or a newline in the output stream would be one indication, but there is no way to make sure there

always is such a delimiter. If it is forgotten and that thread (that still has the mutex locked) runs into a waiting condition itself (e.g., UI thread synchronization with pipes), there would be a deadlock.

So this much safer approach was chosen: Give each thread its own logging infrastructure, i.e., its own log stream and its own log buffer.

Sure, in bad cases the logger function might still be executed in parallel and thus clobber a line or two of log output. But that's merely bad output formatting, not writing another thread's data structures without control - which can easily happen if multiple threads are working on the same output buffer, i.e. manipulate the same string.

Definition at line 208 of file YUILog.cc.

3.96.2 Constructor & Destructor Documentation

3.96.2.1 YPerThreadLogInfo::YPerThreadLogInfo ( ) [inline]

Constructor

Definition at line 213 of file YUILog.cc.

3.96.2.2 YPerThreadLogInfo::~YPerThreadLogInfo ( ) [inline]

Destructor

Definition at line 224 of file YUILog.cc.

Here is the call graph for this function:

3.96.3 Member Function Documentation

3.96.3.1 bool YPerThreadLogInfo::isThread (pthread_t *otherThreadHandle*) [inline]

Check if this per-thread logging information belongs to the specified thread.

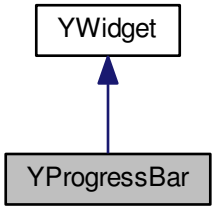
Definition at line 232 of file YUILog.cc.

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

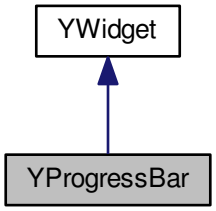
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUILog.cc

3.97 YProgressBar Class Reference

Inheritance diagram for YProgressBar:



Collaboration diagram for YProgressBar:



Public Member Functions

- virtual ~YProgressBar ()
- virtual const char * widgetClass () const
- std::string label ()
- virtual void setLabel (const std::string &label)
- int maxValue () const
- int value () const
- virtual void setValue (int newValue)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Protected Member Functions

- YProgressBar (YWidget *parent, const std::string &label, int maxValue=100)

3.97.1 Detailed Description

Definition at line 33 of file YProgressBar.h.

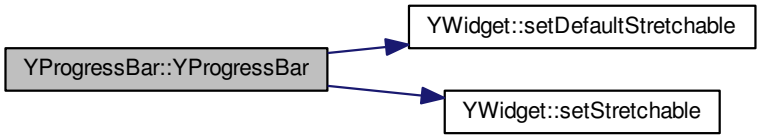
3.97.2 Constructor & Destructor Documentation

3.97.2.1 YProgressBar::YProgressBar (YWidget * *parent*, const std::string & *label*, int *maxValue* = 100) [protected]

Constructor.

Definition at line 52 of file YProgressBar.cc.

Here is the call graph for this function:



3.97.2.2 YProgressBar::~YProgressBar ( ) [virtual]

Destructor.

Definition at line 65 of file YProgressBar.cc.

3.97.3 Member Function Documentation

3.97.3.1 YPropertyValue YProgressBar::getProperty (const std::string & *propertyName*) [virtual]

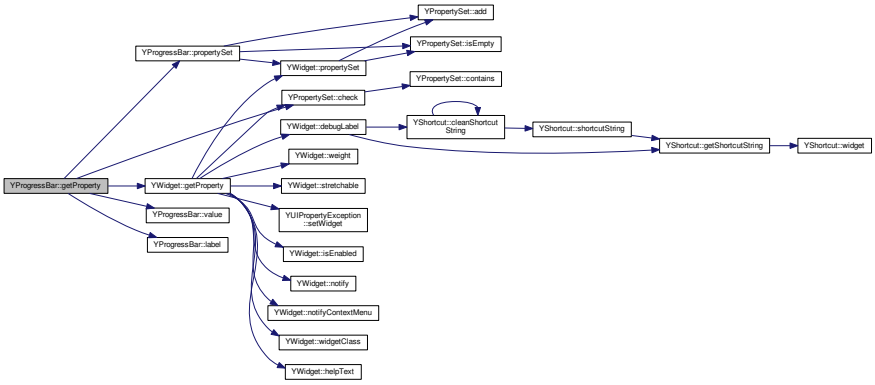
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 144 of file YProgressBar.cc.

Here is the call graph for this function:



3.97.3.2 std::string YProgressBar::label ( )

Get the label (the caption above the progress bar).

Definition at line 71 of file YProgressBar.cc.

3.97.3.3 int YProgressBar::maxValue ( ) const

Return the maximum progress value. Notice that this value can only be set in the constructor.

Definition at line 83 of file YProgressBar.cc.

3.97.3.4 const YPropertySet & YProgressBar::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 108 of file YProgressBar.cc.

Here is the call graph for this function:

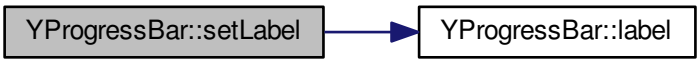
3.97.3.5 void YProgressBar::setLabel (const std::string & *label*) [virtual]

Set the label (the caption above the progress bar).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 77 of file YProgressBar.cc.

Here is the call graph for this function:



3.97.3.6 bool YProgressBar::setProperty (const std::string & *propertyName,* const **YPropertyValue** & *val*) [virtual]

Set a property. Reimplemented from YWidget.

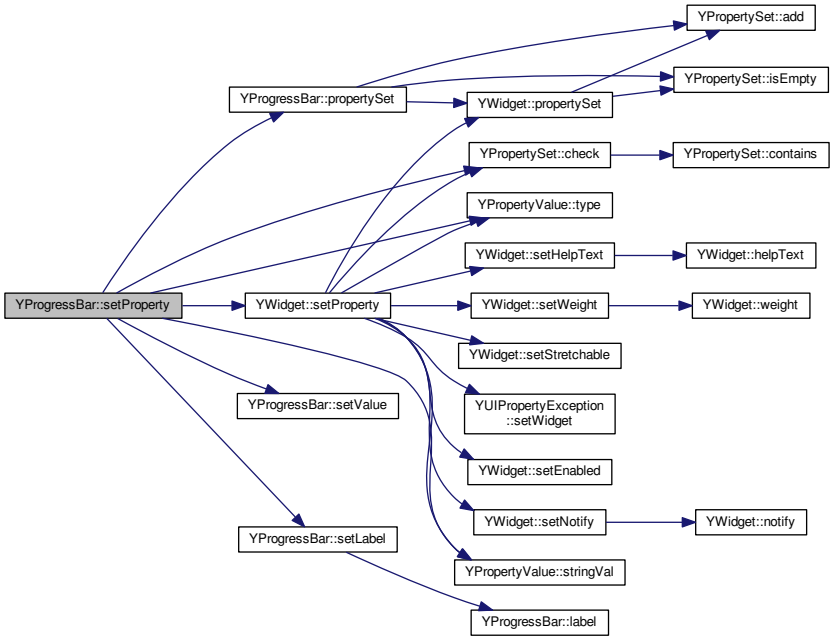
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 128 of file YProgressBar.cc.

Here is the call graph for this function:



3.97.3.7 void YProgressBar::setValue (int *newValue*) [virtual]

Set the current progress value (<= maxValue()).

Derived classes should reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 95 of file YProgressBar.cc.

3.97.3.8 int YProgressBar::value ( ) const

Return the current progress value.

Definition at line 89 of file YProgressBar.cc.

3.97.3.9 virtual const char* YProgressBar::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 53 of file YProgressBar.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YProgressBar.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YProgressBar.cc

3.98 YProgressBarPrivate Struct Reference

Public Member Functions

- **YProgressBarPrivate** (const std::string &label, int maxValue)

Public Attributes

- std::string **label**
- int **maxValue**
- int **value**

3.98.1 Detailed Description

Definition at line 32 of file YProgressBar.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YProgressBar.cc

3.99 YProperty Class Reference

```
#include <YProperty.h>
```

Public Member Functions

- YProperty (const std::string &name, YPropertyType type, bool isReadOnly=false)
- std::string name () const
- YPropertyType type () const
- bool isReadOnly () const
- std::string typeAsStr () const

Static Public Member Functions

- static std::string typeAsStr (YPropertyType type)

3.99.1 Detailed Description

Class for widget properties.

Definition at line 51 of file YProperty.h.

3.99.2 Constructor & Destructor Documentation

3.99.2.1 YProperty::YProperty (const std::string & *name*, YPropertyType *type*, bool *isReadOnly* = `false`) `[inline]`

Constructor: Create a property with the specified name and type. 'isReadOnly' is for properties that cannot be set, only retrieved.

Definition at line 58 of file YProperty.h.

3.99.3 Member Function Documentation

3.99.3.1 bool YProperty::isReadOnly ( ) const [inline]

Returns 'true' if this property cannot be changed, only retrieved.

Definition at line 77 of file YProperty.h.

3.99.3.2 std::string YProperty::name ( ) const [inline]

Returns the name of this property.

Definition at line 67 of file YProperty.h.

3.99.3.3 YPropertyType YProperty::type ( ) const [inline]

Returns the type of this property.

Definition at line 72 of file YProperty.h.

3.99.3.4 std::string YProperty::typeAsStr ( ) const [inline]

Returns the type of this property as string.

Definition at line 82 of file YProperty.h.

3.99.3.5 std::string YProperty::typeAsStr (YPropertyType *type*) [static]

Returns a string description of a property type.

Definition at line 31 of file YProperty.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YProperty.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YProperty.cc

3.100 YPropertySet Class Reference

```
#include <YProperty.h>
```

Public Types

- typedef std::vector< YProperty >
  ::const_iterator **const_iterator**

Public Member Functions

- YPropertySet ()
- void check (const std::string &propertyName) const
- void check (const std::string &propertyName, YPropertyType type) const
- void check (const YProperty &prop) const
- bool contains (const std::string &propertyName) const throw ()
- bool contains (const std::string &propertyName, YPropertyType type) const
- bool contains (const YProperty &prop) const
- bool isEmpty () const
- int size () const
- void add (const YProperty &prop)
- void add (const YPropertySet &otherSet)
- const_iterator propertiesBegin () const
- const_iterator propertiesEnd () const

3.100.1 Detailed Description

A set of properties to check names and types against.

Definition at line 184 of file YProperty.h.

3.100.2 Constructor & Destructor Documentation

3.100.2.1 YPropertySet::YPropertySet ( )

Constructor.

Definition at line 55 of file YProperty.cc.

3.100.3 Member Function Documentation

3.100.3.1 void YPropertySet::add (const YProperty & *prop*)

Add a property to this property set.

Definition at line 120 of file YProperty.cc.

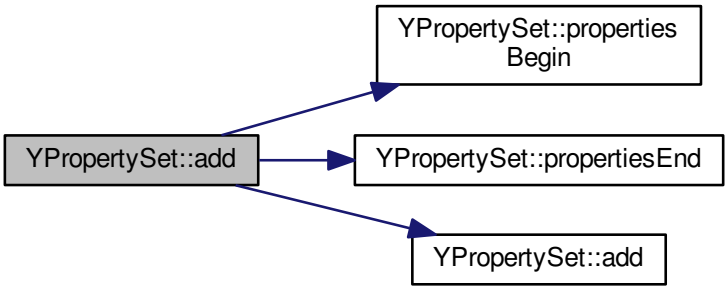
3.100.3.2 void YPropertySet::add (const YPropertySet & *otherSet*)

Adds all properties of another property set.

If that other set contains duplicates (properties that are already in this set), those others will never be found with lookup().

Definition at line 127 of file YProperty.cc.

Here is the call graph for this function:



3.100.3.3 void YPropertySet::check (const std::string & *propertyName*) const

Check if a property 'propertyName' exists in this property set. Throw a YUIUnknownPropertyException if it does not exist. Use YPropertySet::contains() for a check that simply returns 'false' if it does not exist.

Definition at line 62 of file YProperty.cc.

Here is the call graph for this function:

3.100.3.4 void YPropertySet::check (const std::string & *propertyName,* YPropertyType *type*) const

Check if a property 'propertyName' exists in this property set. Throw a YUIUnknownPropertyException if it does not exist.

If there is a property with that name, check also the expected type against 'type'. If the types don't match, throw a YUIPropertyTypeMismatchException. If the property is read-only, throw a YUISetReadOnlyPropertyException.

Definition at line 70 of file YProperty.cc.

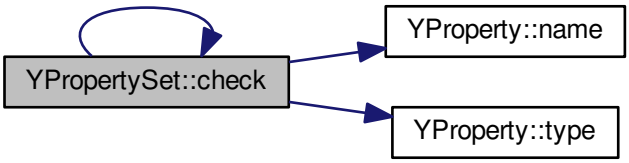
Here is the call graph for this function:

3.100.3.5 void YPropertySet::check (const YProperty & *prop*) const [inline]

Same as above, overloaded for convenience.

Definition at line 214 of file YProperty.h.

Here is the call graph for this function:



3.100.3.6 bool YPropertySet::contains (const std::string & *propertyName*) const throw ()

Check if a property 'propertyName' exists in this property set. Returns 'true' if it exists, 'false' if not.

Use YPropertySet::check() for a check that throws exceptions if there is no such property.

Definition at line 81 of file YProperty.cc.

3.100.3.7 bool YPropertySet::contains (const std::string & *propertyName,* YPropertyType *type*) const

Check if a property 'propertyName' exists in this property set. Returns 'true' if it exists, 'false' if not.

If there is a property with that name, check also the expected type against 'type'. If the types don't match, throw a YUIPropertyTypeMismatchException.

If the property is read-only, throw a YUISetReadOnlyPropertyException.

Use YPropertySet::check() for a check that throws exceptions if there is no such property.

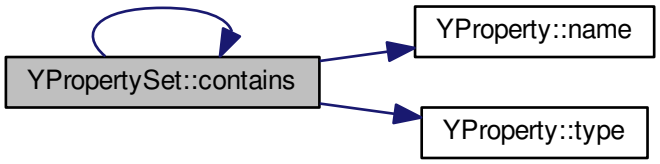
Definition at line 96 of file YProperty.cc.

3.100.3.8 bool YPropertySet::contains (const YProperty & *prop*) const [inline]

Same as above, overloaded for convenience.

Definition at line 244 of file YProperty.h.

Here is the call graph for this function:



3.100.3.9 bool YPropertySet::isEmpty ( ) const [inline]

Returns 'true' if this property set does not contain anything.

Definition at line 250 of file YProperty.h.

3.100.3.10 YPropertySet::const_iterator YPropertySet::propertiesBegin ( ) const

Returns an iterator that points to the first property in this set.

Definition at line 139 of file YProperty.cc.

3.100.3.11 YPropertySet::const_iterator YPropertySet::propertiesEnd ( ) const

Returns an iterator that points after the last property in this set.

Definition at line 145 of file YProperty.cc.

3.100.3.12 int YPropertySet::size ( ) const [inline]

Returns the number of properties in this set.

Definition at line 255 of file YProperty.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YProperty.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YProperty.cc

3.101 YPropertyValue Class Reference

```
#include <YProperty.h>
```

Public Member Functions

- YPropertyValue (const std::string &str)
- YPropertyValue (const char ∗str)
- YPropertyValue (bool b)
- YPropertyValue (YInteger num)
- YPropertyValue (int num)
- **YPropertyValue** (YPropertyType type)
- YPropertyValue ()
- ∼YPropertyValue ()
- YPropertyType type () const
- std::string typeAsStr () const
- std::string stringVal () const
- bool **boolVal** () const
- YInteger **integerVal** () const

3.101.1 Detailed Description

Transport class for the value of simple properties.

More complex properties (lists of items, tree descriptions, ...) have to be handled specifically someplace else, but most properties are of simple types and can be treated in similar ways.

Definition at line 104 of file YProperty.h.

3.101.2 Constructor & Destructor Documentation

3.101.2.1 YPropertyValue::YPropertyValue (const std::string & *str*) [inline]

Constructor for string properties.

Definition at line 111 of file YProperty.h.

3.101.2.2 YPropertyValue::YPropertyValue (const char ∗ *str*) [inline]

Constructor for const char ∗ (string) properties.

Definition at line 117 of file YProperty.h.

3.101.2.3 YPropertyValue::YPropertyValue (bool *b*) [inline], [explicit]

Constructor for bool properties.

Definition at line 123 of file YProperty.h.

3.101.2.4 YPropertyValue::YPropertyValue (YInteger *num*) [inline],[explicit]

Constructor for numerical (YCP integer) properties.

Definition at line 129 of file YProperty.h.

3.101.2.5 YPropertyValue::YPropertyValue (int *num*) [inline],[explicit]

Constructor for numerical (YCP integer) properties.

Definition at line 135 of file YProperty.h.

3.101.2.6 YPropertyValue::YPropertyValue ( ) [inline]

Default constructor

Definition at line 144 of file YProperty.h.

3.101.2.7 YPropertyValue::~YPropertyValue ( )

Destructor.

Definition at line 49 of file YProperty.cc.

3.101.3 Member Function Documentation

3.101.3.1 std::string YPropertyValue::stringVal ( ) const [inline]

Methods to get the value of this property. Check with type() which one to use.

Definition at line 167 of file YProperty.h.

3.101.3.2 YPropertyType YPropertyValue::type ( ) const [inline]

Returns the type of this property value. Use this to determine which xyVal() method to use.

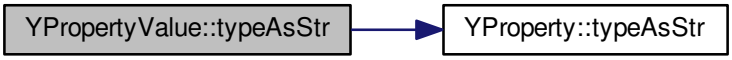
Definition at line 156 of file YProperty.h.

3.101.3.3 std::string YPropertyValue::typeAsStr ( ) const [inline]

Returns the type of this property value as string.

Definition at line 161 of file YProperty.h.

Here is the call graph for this function:

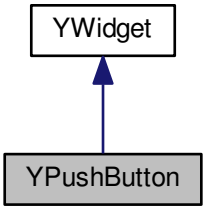


The documentation for this class was generated from the following files:

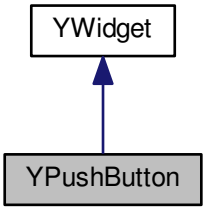
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YProperty.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YProperty.cc

3.102 YPushButton Class Reference

Inheritance diagram for YPushButton:



Collaboration diagram for YPushButton:



Public Member Functions

- virtual ~YPushButton ()
- virtual const char * widgetClass () const
- std::string label () const
- virtual void setLabel (const std::string &label)
- virtual void setIcon (const std::string &iconName)
- bool isDefaultButton () const
- virtual void setDefaultButton (bool def=true)
- virtual void setRole (YButtonRole role)
- YButtonRole role () const
- virtual void setFunctionKey (int fkey_no)
- bool isHelpButton () const
- virtual void setHelpButton (bool helpButton=true)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)

Protected Member Functions

- YPushButton (YWidget *parent, const std::string &label)

3.102.1 Detailed Description

Definition at line 34 of file YPushButton.h.

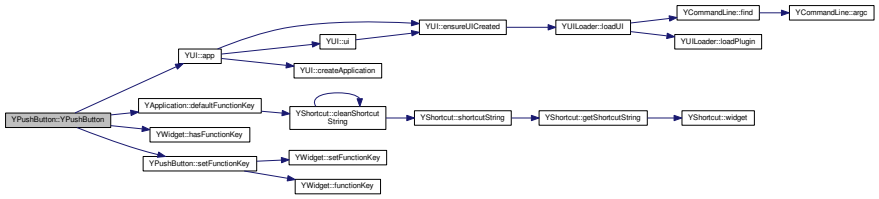
3.102.2 Constructor & Destructor Documentation

3.102.2.1 YPushButton::YPushButton (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

Definition at line 56 of file YPushButton.cc.

Here is the call graph for this function:

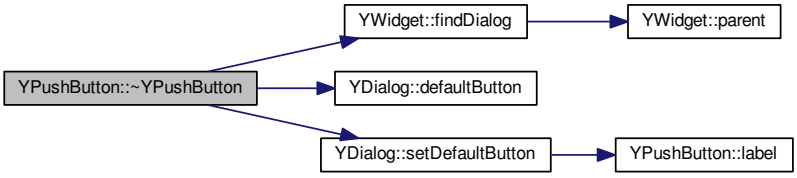


3.102.2.2 YPushButton::~YPushButton ( ) [virtual]

Destructor.

Definition at line 67 of file YPushButton.cc.

Here is the call graph for this function:



3.102.3 Member Function Documentation

3.102.3.1 YPropertyValue YPushButton::getProperty (const std::string & *propertyName*) [virtual]

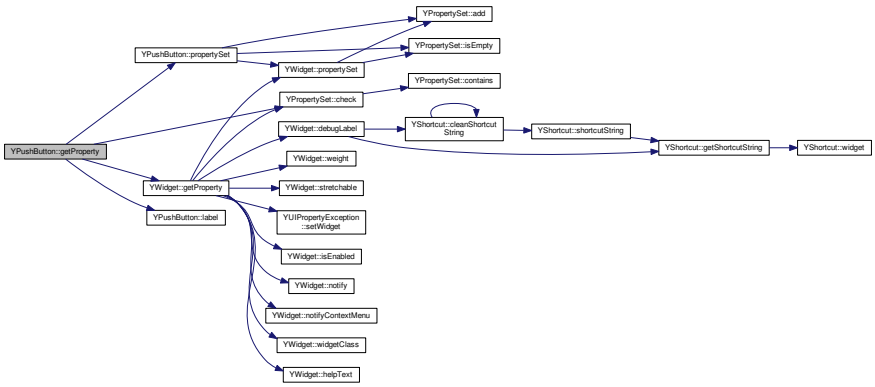
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 231 of file YPushButton.cc.

Here is the call graph for this function:



3.102.3.2 bool YPushButton::isDefaultButton ( ) const

Returns 'true' if this is the dialog's default button, i.e. the one button that gets activated if the user hits the [Return] key anywhere in the dialog.

Definition at line 90 of file YPushButton.cc.

3.102.3.3 bool YPushButton::isHelpButton ( ) const

Returns 'true' if this is a "Help" button.

When activated, a help button will traverse up its widget hierarchy and search for the topmost widget with a helpText() set and display that help text in a pop-up dialog (with a local event loop).

NOTE that this is only done during YDialog::waitForEvent() (i.e. in YCP UI::WaitForEvent(), UI::UserInput(), UI::TimeoutUserInput()) and not during YDialog::pollEvent() (i.e. YCP UI::PollInput()) since displaying the help text will block the application until the user closes the help text.

Definition at line 125 of file YPushButton.cc.

3.102.3.4 std::string YPushButton::label ( ) const

Get the label (the text on the button).

Definition at line 84 of file YPushButton.cc.

3.102.3.5 const YPropertySet & YPushButton::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 198 of file YPushButton.cc.

Here is the call graph for this function:

3.102.3.6 YButtonRole YPushButton::role ( ) const

Return the role of this button.

Definition at line 164 of file YPushButton.cc.

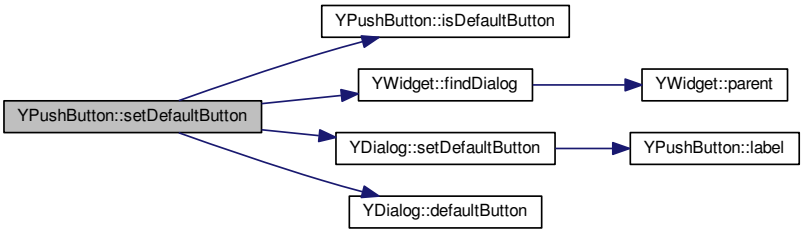
3.102.3.7 void YPushButton::setDefaultButton (bool *def* = true) [virtual]

Make this button the default button.

Derived classes should reimplement this, but call this base class function in the overwritten function.

Definition at line 96 of file YPushButton.cc.

Here is the call graph for this function:



3.102.3.8 void YPushButton::setFunctionKey (int *fkey_no*) [virtual]

Assign a function key to this widget (1 for F1, 2 for F2, etc.; 0 for none)

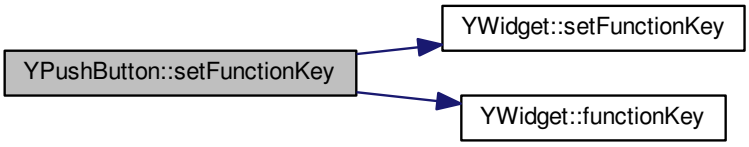
Reimplemented from YWidget to map function keys to button roles.

Derived classes may want to overwrite this function, but they should call this base class function in the new function.

Reimplemented from YWidget.

Definition at line 172 of file YPushButton.cc.

Here is the call graph for this function:



3.102.3.9 void YPushButton::setHelpButton (bool *helpButton* = true) [virtual]

Make this button a help button.

Derived classes are free to reimplement this, but they should call this base class method in the overloaded function.

Definition at line 131 of file YPushButton.cc.

3.102.3.10 virtual void YPushButton::setIcon (const std::string & *iconName*) [inline], [virtual]

Set this button's icon from an icon file in the UI's default icon directory. Clear the icon if the name is empty.

This default implementation does nothing. UIs that can handle icons can choose to overwrite this method.

Definition at line 74 of file YPushButton.h.

3.102.3.11 void YPushButton::setLabel (const std::string & *label*) [virtual]

Set the label (the text on the button).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 78 of file YPushButton.cc.

Here is the call graph for this function:

3.102.3.12 bool YPushButton::setProperty (const std::string & *propertyName,* const **YPropertyValue** & *val*) [virtual]

Set a property. Reimplemented from YWidget.

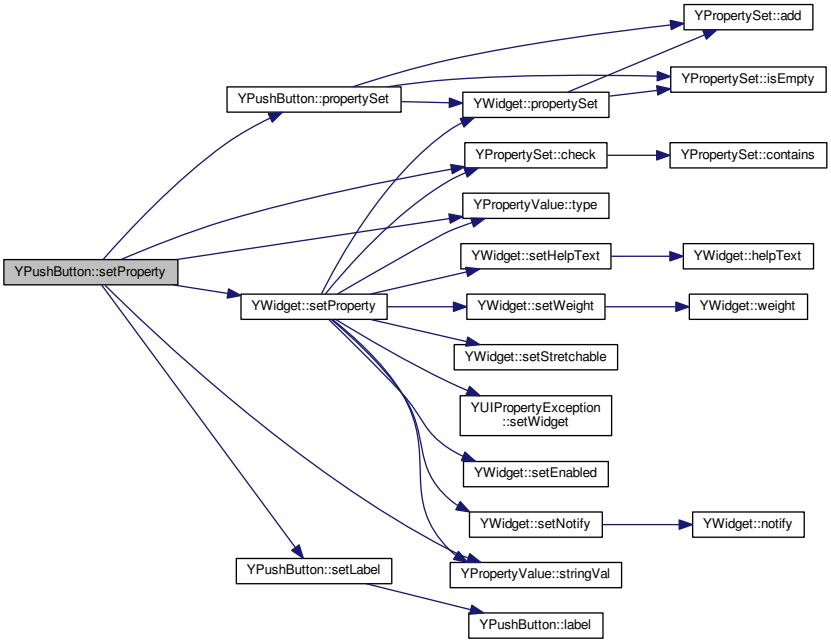
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 216 of file YPushButton.cc.

Here is the call graph for this function:



3.102.3.13 void YPushButton::setRole (YButtonRole *role*) [virtual]

Set a predefined role for this button.

This is important when the button is a child of a YButtonBox so the layout can be arranged according to the conventions of the current UI or desktop environment.

See YButtonBox.h for more details. YButtonRole is defined in YTypes.h

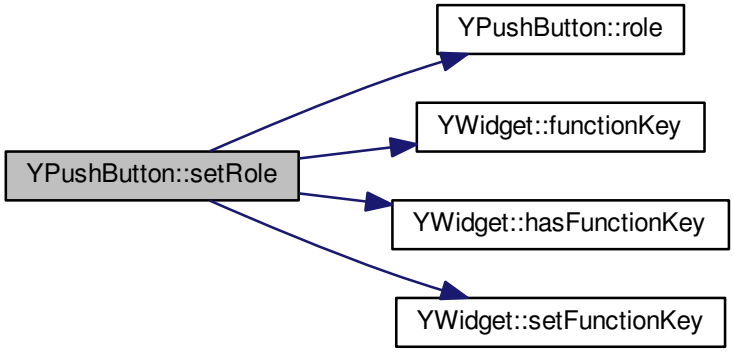
The default is YCustomButton, i.e., no predefined role. setFunctionKey() uses some heuristics to map function keys to buttons:

```
F10 -> YOkButton
F9  -> YCancelButton
F1  -> YHelpButton
```

Derived classes are free to reimplement this, but they should call this base class function in the overwritten function.

Definition at line 140 of file YPushButton.cc.

Here is the call graph for this function:



3.102.3.14 virtual void YPushButton::setShortcutString (const std::string & *str*) [inline], [virtual]

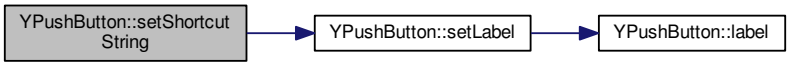
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 193 of file YPushButton.h.

Here is the call graph for this function:



3.102.3.15 virtual std::string YPushButton::shortcutString ( ) const [inline], [virtual]

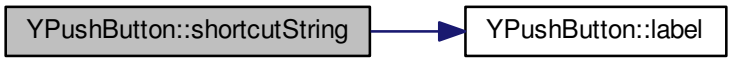
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 186 of file YPushButton.h.

Here is the call graph for this function:



3.102.3.16 virtual const char* YPushButton::widgetClass ( ) const [inline], [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 52 of file YPushButton.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPushButton.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPushButton.cc

3.103 YPushButtonPrivate Struct Reference

Public Member Functions

- **YPushButtonPrivate** (const std::string &label)

Public Attributes

- std::string **label**
- bool **isDefaultButton**
- bool **setDefaultButtonRecursive**
- bool **isHelpButton**
- YButtonRole **role**

3.103.1 Detailed Description

Definition at line 38 of file YPushButton.cc.

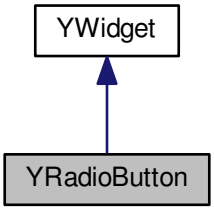
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YPushButton.cc

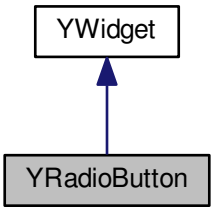
3.104 YRadioButton Class Reference

```
#include <YRadioButton.h>
```

Inheritance diagram for YRadioButton:



Collaboration diagram for YRadioButton:



Public Member Functions

- virtual ~YRadioButton ()
- virtual const char * widgetClass () const
- virtual bool value ()=0
- virtual void setValue (bool checked)=0
- std::string label () const
- virtual void setLabel (const std::string &label)
- bool useBoldFont () const
- virtual void setUseBoldFont (bool bold=true)
- YRadioButtonGroup * buttonGroup ()
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)
- const char * userInputProperty ()

Protected Member Functions

- YRadioButton (YWidget *parent, const std::string &label)
- YRadioButtonGroup * findRadioButtonGroup () const
- virtual void saveUserInput (YMacroRecorder *macroRecorder)

3.104.1 Detailed Description

RadioButton: Widget for one-out-of-many selection.

Only one RadioButton in a RadioBox (in a RadioButtonGroup) can be set to "on" at the same time. Setting any RadioButton of a RadioButtonGroup to "on" automatically sets all others in the same RadioButtonGroup to "off".

RadioButtons customarily have a distinct visual appearance from CheckBoxes:

```
( ) RadioButton 1
(*) RadioButton 2
( ) RadioButton 3

[ ] CheckBox 1
[*] CheckBox 2
[*] CheckBox 3
```

Definition at line 51 of file YRadioButton.h.

3.104.2 Constructor & Destructor Documentation

3.104.2.1 YRadioButton::YRadioButton (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

Creates a new RadioButton with user-visible text 'label'. 'label' can and should contain a keyboard shortcut (designated with '&').

The caller has to take care to add this RadioButton to its RadioButtonGroup:

if (radioButton->buttonGroup()) radioButton->buttonGroup()->addRadioButton(radioButton);

This can't be done in the constructor because it would involve calling a virtual function, which doesn't work yet within the constructor.

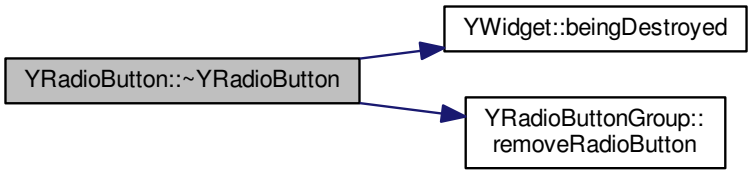
Definition at line 60 of file YRadioButton.cc.

3.104.2.2 YRadioButton::~YRadioButton ( ) [virtual]

Destructor: Removes the button from the radio button group.

Definition at line 77 of file YRadioButton.cc.

Here is the call graph for this function:



3.104.3 Member Function Documentation

3.104.3.1 YRadioButtonGroup * YRadioButton::buttonGroup ( )

Get a pointer to the radio button group this button belongs to.

Definition at line 163 of file YRadioButton.cc.

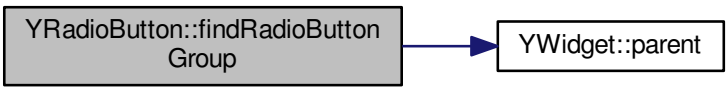
Here is the call graph for this function:

3.104.3.2 YRadioButtonGroup * YRadioButton::findRadioButtonGroup ( ) const [protected]

Traverse the widget hierarchy upwards to find the corresponding YRadioButtonGroup, i.e. the class that controls the radio box behaviour (i.e. that makes sure that no more than one RadioButton is set to "on" at the same time).

Definition at line 175 of file YRadioButton.cc.

Here is the call graph for this function:



3.104.3.3 YPropertyValue YRadioButton::getProperty (const std::string & *propertyName*) [virtual]

Get a property. Reimplemented from YWidget.

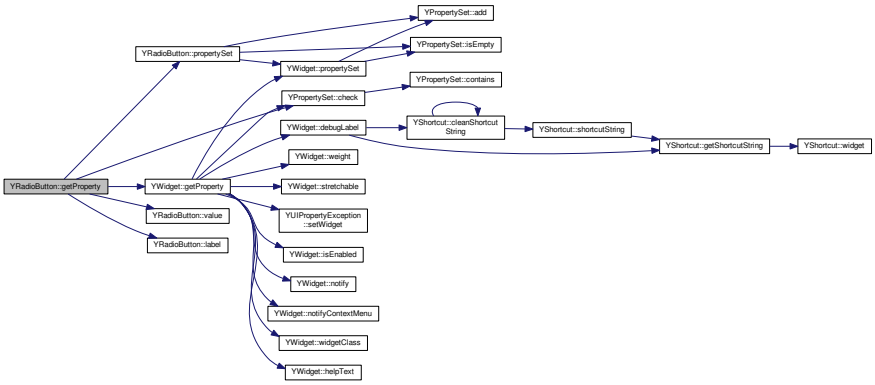
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from YWidget.

Definition at line 149 of file YRadioButton.cc.

Here is the call graph for this function:



3.104.3.4 std::string YRadioButton::label ( ) const

Get the label (the text on the RadioButton).

Definition at line 93 of file YRadioButton.cc.

3.104.3.5 const YPropertySet & YRadioButton::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property set upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 112 of file YRadioButton.cc.

Here is the call graph for this function:

3.104.3.6 void YRadioButton::saveUserInput (YMacroRecorder * *macroRecorder*) [protected],[virtual]

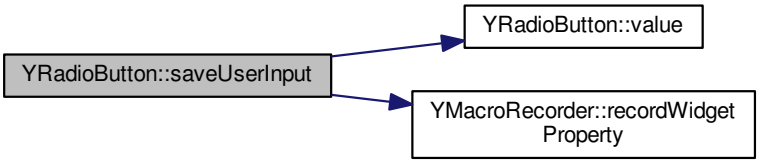
Save the widget's user input to a macro recorder.

Reimplemented from YWidget because only radio buttons that are on (no more than one per radio box) are recorded.

Reimplemented from YWidget.

Definition at line 194 of file YRadioButton.cc.

Here is the call graph for this function:



3.104.3.7 void YRadioButton::setLabel (const std::string & *label*) [virtual]

Set the label (the text on the RadioButton).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 87 of file YRadioButton.cc.

3.104.3.8 bool YRadioButton::setProperty (const std::string & *propertyName,* const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

This method may throw exceptions, for example

- if there is no property with that name

- if the expected type and the type mismatch

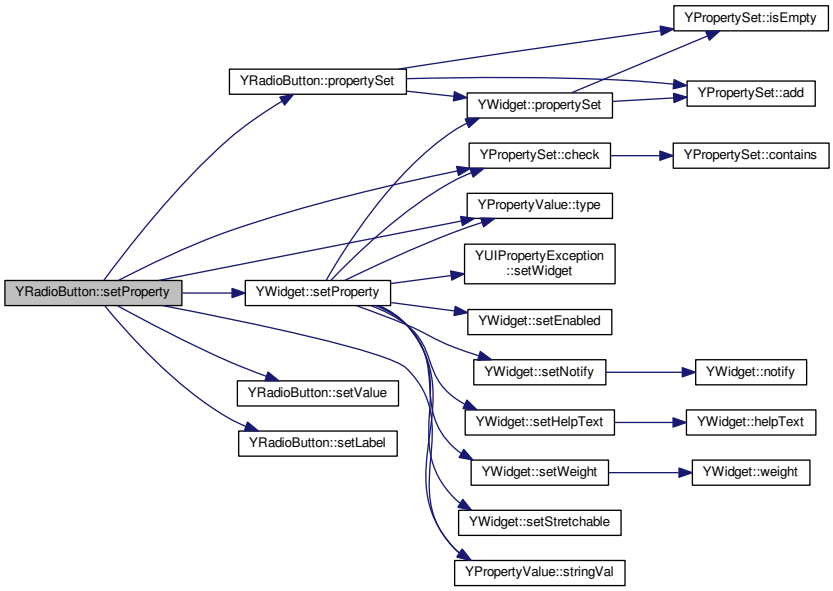
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 133 of file YRadioButton.cc.

Here is the call graph for this function:



3.104.3.9 virtual void YRadioButton::setShortcutString (const std::string & *str*) [inline], [virtual]

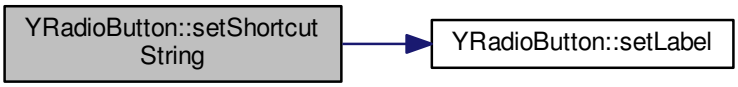
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 177 of file YRadioButton.h.

Here is the call graph for this function:



3.104.3.10 void YRadioButton::setUseBoldFont (bool *bold* = `true`) `[virtual]`

Indicate whether or not a bold font should be used.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 105 of file YRadioButton.cc.

3.104.3.11 virtual void YRadioButton::setValue (bool *checked*) `[pure virtual]`

Set the radio button value (on/off).

Derived classes are required to implement this.

3.104.3.12 virtual std::string YRadioButton::shortcutString ( ) const `[inline],[virtual]`

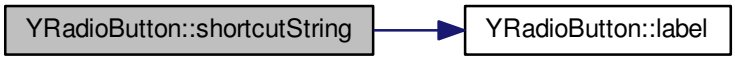
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 170 of file YRadioButton.h.

Here is the call graph for this function:



3.104.3.13 bool YRadioButton::useBoldFont ( ) const

Returns 'true' if a bold font should be used.

Definition at line 99 of file YRadioButton.cc.

3.104.3.14 const char* YRadioButton::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 184 of file YRadioButton.h.

3.104.3.15 virtual bool YRadioButton::value ( ) [pure virtual]

Get the current on/off value: 'true' if checked, 'false' if unchecked.

Derived classes are required to implement this.

3.104.3.16 virtual const char* YRadioButton::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Reimplemented from YWidget.

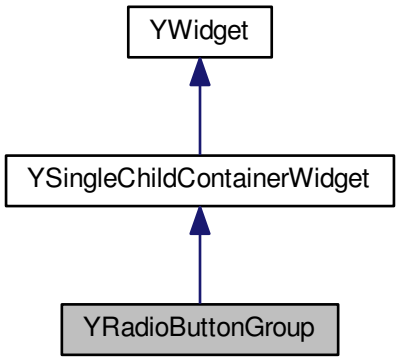
Definition at line 84 of file YRadioButton.h.

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

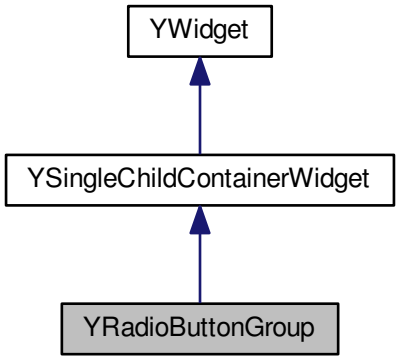
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRadioButton.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRadioButton.cc

3.105 YRadioButtonGroup Class Reference

Inheritance diagram for YRadioButtonGroup:



Collaboration diagram for YRadioButtonGroup:



Public Member Functions

- virtual ~YRadioButtonGroup ()
- virtual const char * widgetClass () const
- YRadioButton * currentButton () const
- YRadioButton * value () const
- virtual void addRadioButton (YRadioButton *radioButton)
- virtual void removeRadioButton (YRadioButton *radioButton)
- void uncheckOtherButtons (YRadioButton *radioButton)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Protected Member Functions

- YRadioButtonGroup (YWidget *parent)
- YRadioButtonListConstIterator radioButtonsBegin () const
- YRadioButtonListConstIterator radioButtonsEnd () const
- int radioButtonsCount () const

3.105.1 Detailed Description

Definition at line 38 of file YRadioButtonGroup.h.

3.105.2 Constructor & Destructor Documentation

3.105.2.1 YRadioButtonGroup::YRadioButtonGroup (YWidget * *parent*) [protected]

Constructor.

Definition at line 46 of file YRadioButtonGroup.cc.

3.105.2.2 YRadioButtonGroup::~YRadioButtonGroup ( ) [virtual]

Destructor.

Definition at line 54 of file YRadioButtonGroup.cc.

3.105.3 Member Function Documentation

3.105.3.1 void YRadioButtonGroup::addRadioButton (YRadioButton * *radioButton*) [virtual]

Add a RadioButton to this button group. RadioButtons are required to call this in their constructor.

Derived classes are free to overload this, but they should call this base class function in the overloaded function.

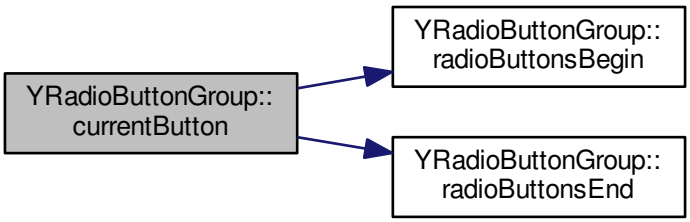
Definition at line 81 of file YRadioButtonGroup.cc.

3.105.3.2 YRadioButton * YRadioButtonGroup::currentButton ( ) const

Find the currently selected button.

Definition at line 108 of file YRadioButtonGroup.cc.

Here is the call graph for this function:



3.105.3.3 YPropertyValue YRadioButtonGroup::getProperty (const std::string & *propertyName*) [virtual]

Get a property. Reimplemented from YWidget.

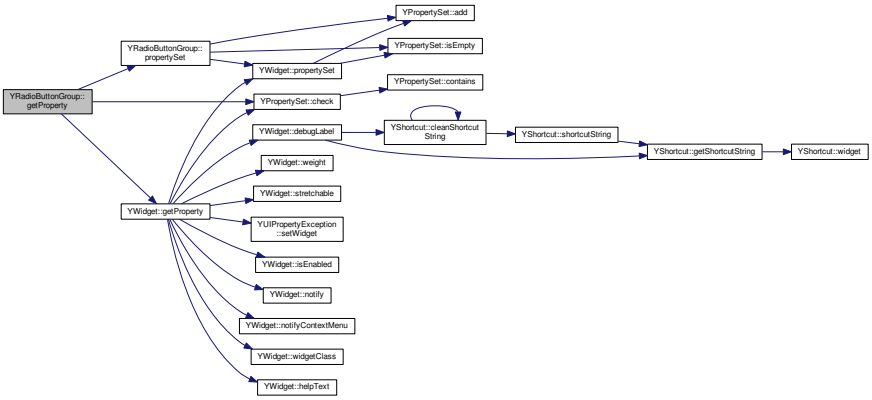
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from YWidget.

Definition at line 159 of file YRadioButtonGroup.cc.

Here is the call graph for this function:



3.105.3.4 const YPropertySet & YRadioButtonGroup::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property set upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 123 of file YRadioButtonGroup.cc.

Here is the call graph for this function:

3.105.3.5 YRadioButtonListConstIterator YRadioButtonGroup::radioButtonsBegin ( ) const [protected]

Return an iterator that points to the first RadioButton of this button group.

Note that RadioButtons in this group may be direct or indirect children of the group, so don't confuse this with YWidget::widgetsBegin().

Definition at line 60 of file YRadioButtonGroup.cc.

3.105.3.6 int YRadioButtonGroup::radioButtonsCount ( ) const [protected]

Return the number of RadioButtons in this button group.

Definition at line 74 of file YRadioButtonGroup.cc.

3.105.3.7 YRadioButtonListConstIterator YRadioButtonGroup::radioButtonsEnd ( ) const [protected]

Return an iterator that points behind the last RadioButton of this button group.

Definition at line 67 of file YRadioButtonGroup.cc.

3.105.3.8 void YRadioButtonGroup::removeRadioButton (YRadioButton * *radioButton*) [virtual]

Remove a RadioButton from this button group. RadioButtons are required to call this in their destructor, but only if the button group is not also in the process of being destroyed (otherwise there may be race conditions with child widgets already destroyed):

```
if ( ! buttonGroup()->beingDestroyed )
    buttonGroup()->removeRadioButton( this );
```

Definition at line 88 of file YRadioButtonGroup.cc.

3.105.3.9 bool YRadioButtonGroup::setProperty (const std::string & *propertyName,* const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

This method may throw exceptions, for example

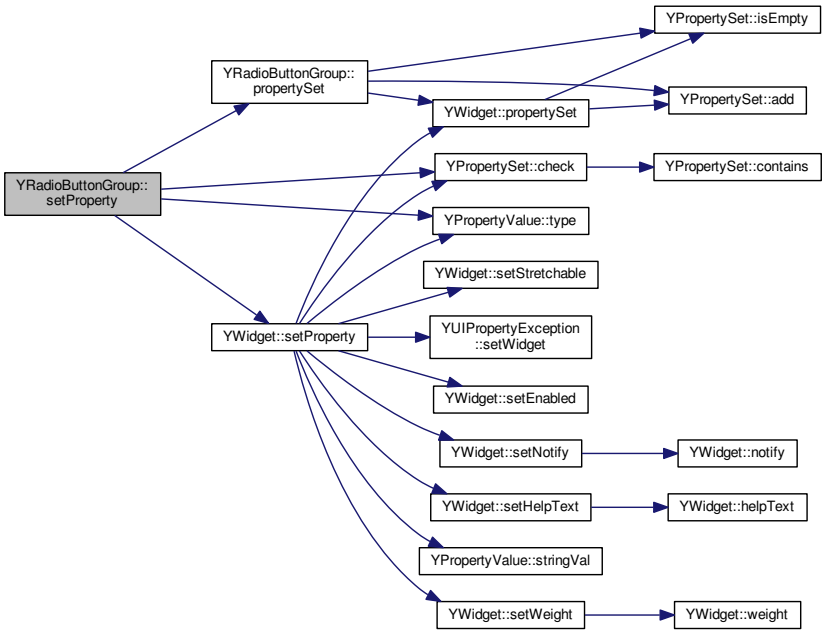
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 143 of file YRadioButtonGroup.cc.

Here is the call graph for this function:

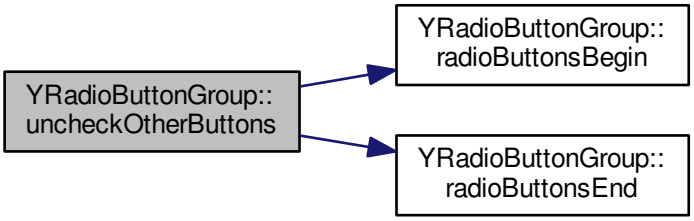


3.105.3.10 void YRadioButtonGroup::uncheckOtherButtons (YRadioButton * *radioButton*)

Unchecks all radio buttons except one. This method can be used by a concrete UI (the Qt UI or the NCurses UI) in the implementation of YRadioButton::setValue().

Definition at line 95 of file YRadioButtonGroup.cc.

Here is the call graph for this function:

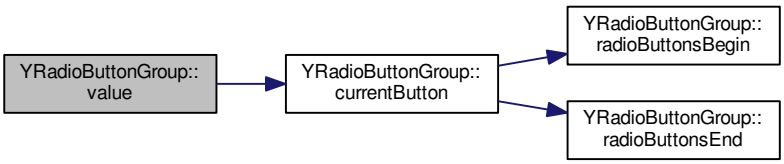


3.105.3.11 YRadioButton* YRadioButtonGroup::value ( ) const [inline]

The same as currentButton() above for convenience.

Definition at line 66 of file YRadioButtonGroup.h.

Here is the call graph for this function:



3.105.3.12 virtual const char* YRadioButtonGroup::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 56 of file YRadioButtonGroup.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRadioButtonGroup.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRadioButtonGroup.cc

3.106 YRadioButtonGroupPrivate Struct Reference

Public Attributes

- YRadioButtonList **buttonList**

3.106.1 Detailed Description

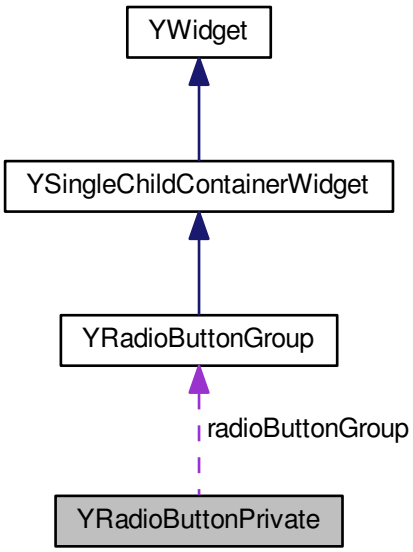
Definition at line 34 of file YRadioButtonGroup.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRadioButtonGroup.cc

3.107 YRadioButtonPrivate Struct Reference

Collaboration diagram for YRadioButtonPrivate:



Public Member Functions

- YRadioButtonPrivate (const std::string &label)

Public Attributes

- std::string **label**
- YRadioButtonGroup * **radioButtonGroup**
- bool **useBoldFont**

3.107.1 Detailed Description

Definition at line 39 of file YRadioButton.cc.

3.107.2 Constructor & Destructor Documentation

3.107.2.1 YRadioButtonPrivate::YRadioButtonPrivate (const std::string & *label*) [inline]

Constructor

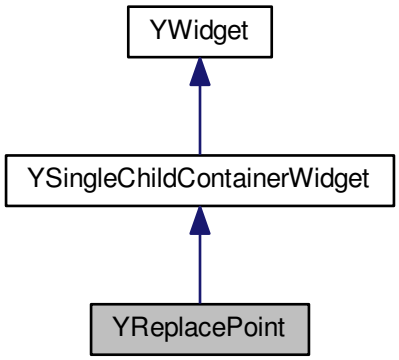
Definition at line 44 of file YRadioButton.cc.

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

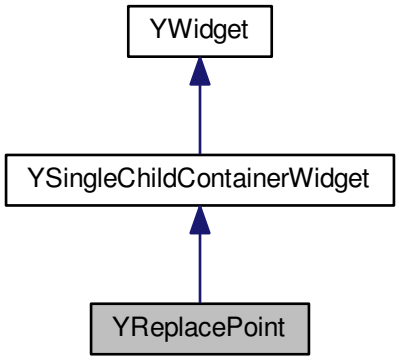
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRadioButton.cc

3.108 YReplacePoint Class Reference

Inheritance diagram for YReplacePoint:



Collaboration diagram for YReplacePoint:



Public Member Functions

- virtual void showChild ()
- virtual const char * widgetClass () const

Protected Member Functions

- YReplacePoint (YWidget *parent)

3.108.1 Detailed Description

Definition at line 30 of file YReplacePoint.h.

3.108.2 Constructor & Destructor Documentation

3.108.2.1 YReplacePoint::YReplacePoint (YWidget * *parent*) [protected]

Constructor

Definition at line 28 of file YReplacePoint.cc.

3.108.3 Member Function Documentation

3.108.3.1 void YReplacePoint::showChild ( ) [virtual]

Show a newly added child. The application using the ReplacePoint is required to call this after the new child is created.

This cannot be done in the child widget's constructor (e.g., by overwriting YWidget::addChild()) since at that point YWidget::widgetRep() may or may not be initialized yet.

This default implementation does nothing. Derived classes should reimplement this to make new child widgets visible.

Definition at line 35 of file YReplacePoint.cc.

3.108.3.2 virtual const char* YReplacePoint::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

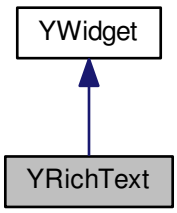
Definition at line 56 of file YReplacePoint.h.

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

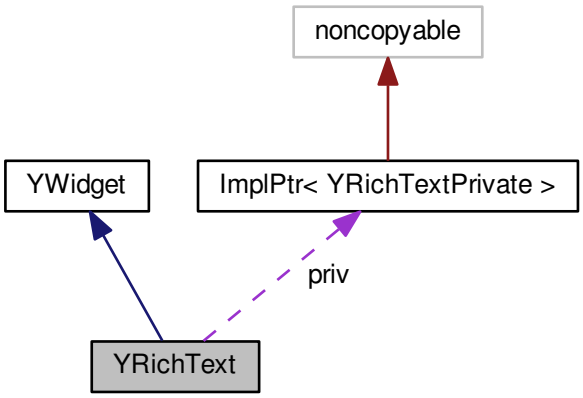
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YReplacePoint.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YReplacePoint.cc

3.109 YRichText Class Reference

Inheritance diagram for YRichText:



Collaboration diagram for YRichText:



Public Member Functions

- YRichText (YWidget *parent, const std::string &text, bool plainTextMode=false)
- virtual ~YRichText ()
- virtual const char * widgetClass () const
- virtual void setValue (const std::string &newValue)
- std::string value () const
- void setText (const std::string &newText)
- std::string text () const
- bool plainTextMode () const

- virtual void setPlainTextMode (bool on=true)
- bool autoScrollDown () const
- virtual void setAutoScrollDown (bool on=true)
- bool shrinkable () const
- void setShrinkable (bool shrinkable=true)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Protected Attributes

- ImplPtr< YRichTextPrivate > **priv**

Additional Inherited Members

3.109.1 Detailed Description

Definition at line 36 of file YRichText.h.

3.109.2 Constructor & Destructor Documentation

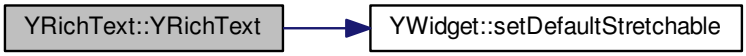
3.109.2.1 YRichText::YRichText (YWidget * *parent,* const std::string & *text,* bool *plainTextMode* = `false`)

Constructor.

'plainTextMode' indicates that the text should be treated as plain text, i.e. any HTML-like tags in the text should not be interpreted in any way.

Definition at line 54 of file YRichText.cc.

Here is the call graph for this function:



3.109.2.2 YRichText::~YRichText ( ) [virtual]

Destructor.

Definition at line 65 of file YRichText.cc.

3.109.3 Member Function Documentation

3.109.3.1 bool YRichText::autoScrollDown ( ) const

Return 'true' if this RichText widget should automatically scroll down when the text content is changed. This is useful for progress displays and log files.

Definition at line 95 of file YRichText.cc.

3.109.3.2 YPropertyValue YRichText::getProperty (const std::string & *propertyName*) [virtual]

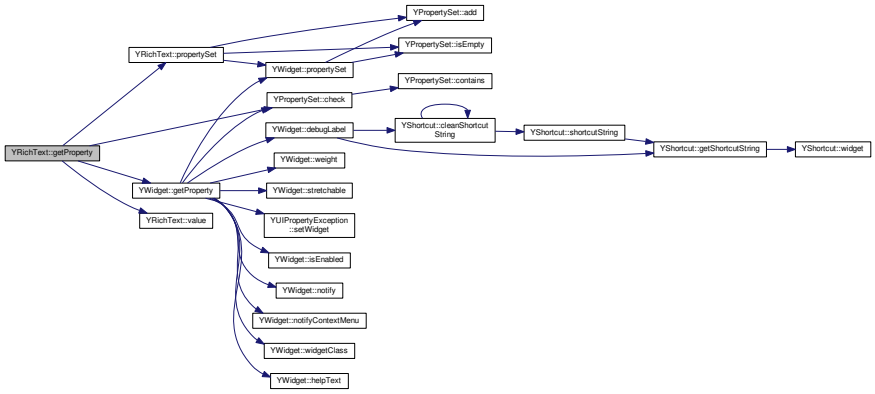
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 156 of file YRichText.cc.

Here is the call graph for this function:



3.109.3.3 bool YRichText::plainTextMode ( ) const

Return 'true' if this RichText widget is in "plain text" mode, i.e. does not try to interpret RichText/HTML tags.

Definition at line 83 of file YRichText.cc.

3.109.3.4 const YPropertySet & YRichText::propertySet ( ) [virtual]

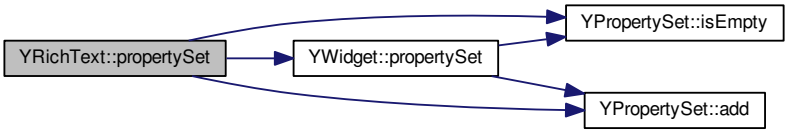
Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 120 of file YRichText.cc.

Here is the call graph for this function:



3.109.3.5 void YRichText::setAutoScrollDown (bool *on* = `true`) [virtual]

Set this RichText widget's "auto scroll down" mode on or off.

Derived classes may want to reimplement this, but they should call this base class function in the new function.

Definition at line 101 of file YRichText.cc.

Here is the call graph for this function:

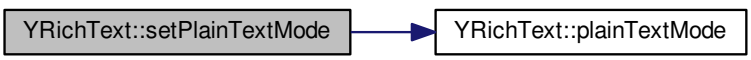
3.109.3.6 void YRichText::setPlainTextMode (bool *on* = `true`) [virtual]

Set this RichText widget's "plain text" mode on or off.

Derived classes may want to reimplement this, but they should call this base class function in the new function.

Definition at line 89 of file YRichText.cc.

Here is the call graph for this function:



3.109.3.7 bool YRichText::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

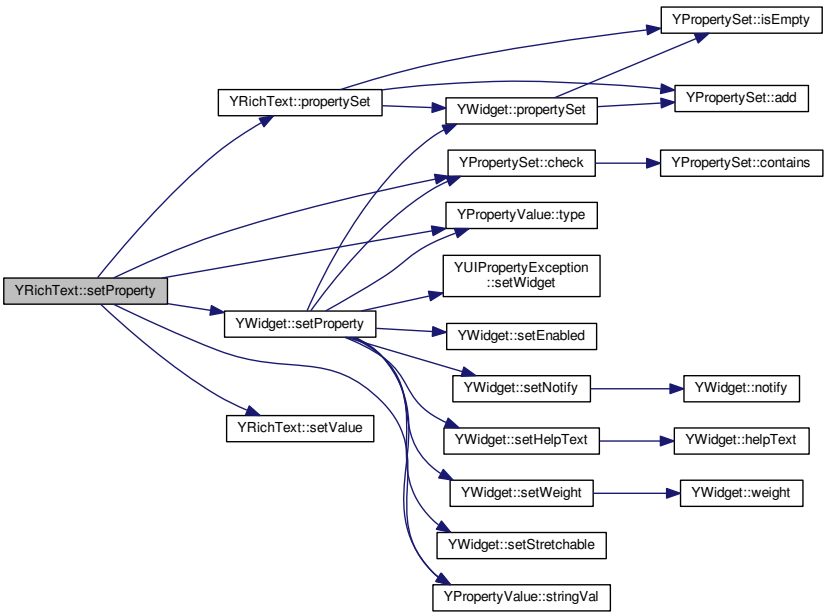
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 140 of file YRichText.cc.

Here is the call graph for this function:



3.109.3.8 void YRichText::setShrinkable (bool *shrinkable* = true)

Make this widget shrinkable, i.e. very small in layouts.

This method is intentionally not virtual because it doesn't have any immediate effect; it is only needed in preferredWidth() / preferredHeight().

Definition at line 113 of file YRichText.cc.

Here is the call graph for this function:

3.109.3.9 void YRichText::setText (const std::string & *newText*) [inline]

Alias for setValue().

Definition at line 78 of file YRichText.h.

Here is the call graph for this function:

3.109.3.10 void YRichText::setValue (const std::string & *newValue*) [virtual]

Change the text content of the RichText widget.

Derived classes should overwrite this function, but call this base class function in the new function.

Definition at line 71 of file YRichText.cc.

3.109.3.11 bool YRichText::shrinkable ( ) const

Returns 'true' if this widget is "shrinkable", i.e. it should be very small by default.

Definition at line 107 of file YRichText.cc.

3.109.3.12 std::string YRichText::text ( ) const [inline]

Alias for value().

Definition at line 83 of file YRichText.h.

Here is the call graph for this function:

3.109.3.13 std::string YRichText::value ( ) const

Return the text content of the RichText widget.

Definition at line 77 of file YRichText.cc.

3.109.3.14 virtual const char* YRichText::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 60 of file YRichText.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRichText.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRichText.cc

3.110 YRichTextPrivate Struct Reference

Public Member Functions

- YRichTextPrivate (const std::string &text, bool plainTextMode)

Public Attributes

- std::string **text**
- bool **plainTextMode**
- bool **autoScrollDown**
- bool **shrinkable**

3.110.1 Detailed Description

Definition at line 33 of file YRichText.cc.

3.110.2 Constructor & Destructor Documentation

3.110.2.1 YRichTextPrivate::YRichTextPrivate (const std::string & *text*, bool *plainTextMode*) [inline]

Constructor.

Definition at line 38 of file YRichText.cc.

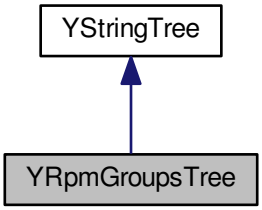
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRichText.cc

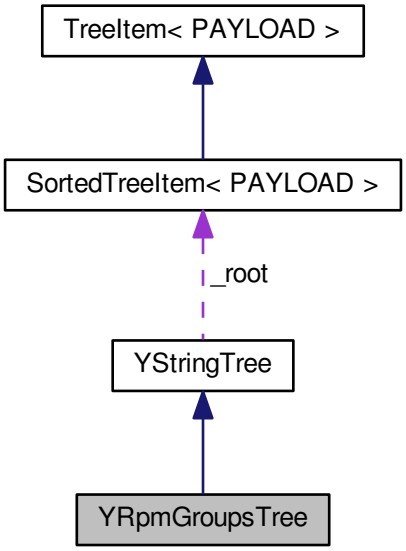
3.111 YRpmGroupsTree Class Reference

```
#include <YRpmGroupsTree.h>
```

Inheritance diagram for YRpmGroupsTree:



Collaboration diagram for YRpmGroupsTree:



Public Member Functions

- YRpmGroupsTree ()
- virtual ~YRpmGroupsTree ()
- YStringTreeItem * addRpmGroup (const std::string &rpmGroup)
- std::string rpmGroup (const YStringTreeItem *node)
- std::string translatedRpmGroup (const YStringTreeItem *node)
- void addFallbackRpmGroups ()

Additional Inherited Members

3.111.1 Detailed Description

Efficient storage for RPM group tags

Definition at line 35 of file YRpmGroupsTree.h.

3.111.2 Constructor & Destructor Documentation

3.111.2.1 YRpmGroupsTree::YRpmGroupsTree ( )

Constructor.

Definition at line 33 of file YRpmGroupsTree.cc.

3.111.2.2 YRpmGroupsTree::~YRpmGroupsTree ( ) [virtual]

Destructor.

Definition at line 41 of file YRpmGroupsTree.cc.

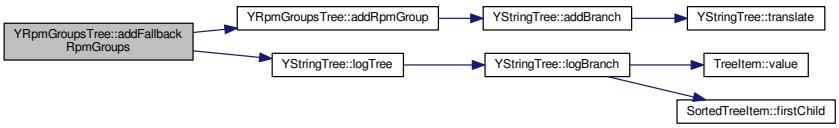
3.111.3 Member Function Documentation

3.111.3.1 void YRpmGroupsTree::addFallbackRpmGroups ( )

Add a predefined set of RPM groups

Definition at line 273 of file YRpmGroupsTree.cc.

Here is the call graph for this function:

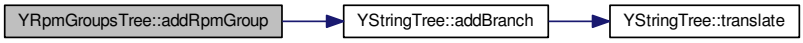


3.111.3.2 YStringTreeItem* YRpmGroupsTree::addRpmGroup (const std::string & *rpmGroup*) [inline]

Insert an RPM group into this tree if not already present. Splits the RPM group string ("abc/def/ghi") and creates tree items for each level as required. Returns the tree entry for this RPM group.

Definition at line 56 of file YRpmGroupsTree.h.

Here is the call graph for this function:

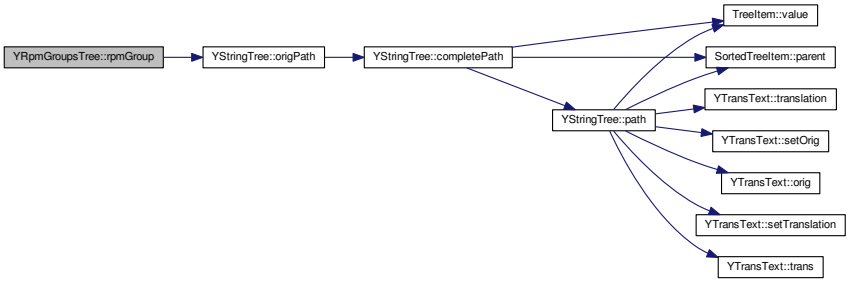


3.111.3.3 std::string YRpmGroupsTree::rpmGroup (const YStringTreeItem * *node*) [inline]

Returns the complete (untranslated) RPM group tag string for 'node'.

Definition at line 62 of file YRpmGroupsTree.h.

Here is the call graph for this function:

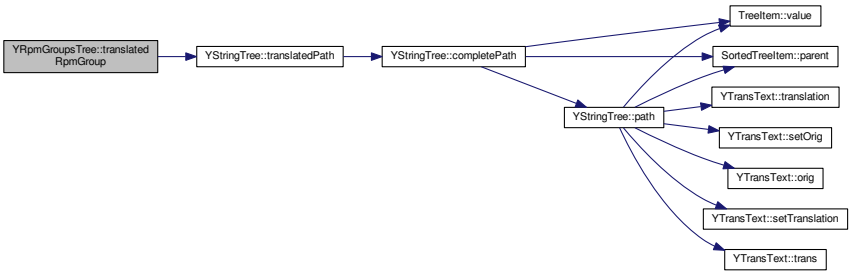


3.111.3.4 std::string YRpmGroupsTree::translatedRpmGroup (const YStringTreeItem * *node*) [inline]

Returns the complete translated RPM group tag string for 'node'.

Definition at line 68 of file YRpmGroupsTree.h.

Here is the call graph for this function:



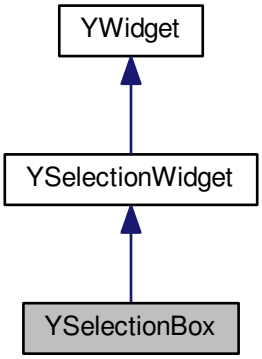
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRpmGroupsTree.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YRpmGroupsTree.cc

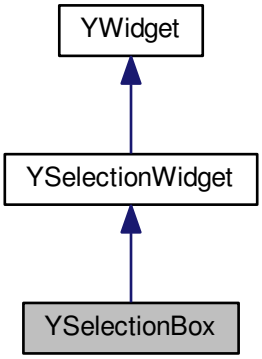
3.112 YSelectionBox Class Reference

```
#include <YSelectionBox.h>
```

Inheritance diagram for YSelectionBox:



Collaboration diagram for YSelectionBox:



Public Member Functions

- virtual ~YSelectionBox ()
- virtual const char * widgetClass () const
- bool shrinkable () const
- virtual void setShrinkable (bool shrinkable=true)
- bool immediateMode () const
- void setImmediateMode (bool on=true)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)

- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- const char * userInputProperty ()

Protected Member Functions

- YSelectionBox (YWidget *parent, const std::string &label)

3.112.1 Detailed Description

Selection box: List box that displays a (scrollable) list of items from which the user can select exactly one. Each item has a label text and an optional icon (*).

This widget displays a number of items at once (as screen space permits). If there is little screen space, you might consider using a ComboBox instead which (in non-editable mode which is the default) displays just one item (the selected item) right away and the others in a pop-up dialog upon mouse click or keypress.

The selection box also has a caption label that is displayed above the list. The hotkey displayed in that caption label will move the keyboard focus into the list.

If multiple columns are needed, use the table widget instead. For tree-like structures, use the tree widget.

(*) Not all UIs (in particular not text-based UIs) support displaying icons, so an icon should never be an exclusive means to display any kind of information.

Definition at line 56 of file YSelectionBox.h.

3.112.2 Constructor & Destructor Documentation

3.112.2.1 YSelectionBox::YSelectionBox (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

Definition at line 48 of file YSelectionBox.cc.

Here is the call graph for this function:

3.112.2.2 YSelectionBox::~YSelectionBox ( ) [virtual]

Destructor.

Definition at line 61 of file YSelectionBox.cc.

3.112.3 Member Function Documentation

3.112.3.1 YPropertyValue YSelectionBox::getProperty (const std::string & *propertyName*) [virtual]

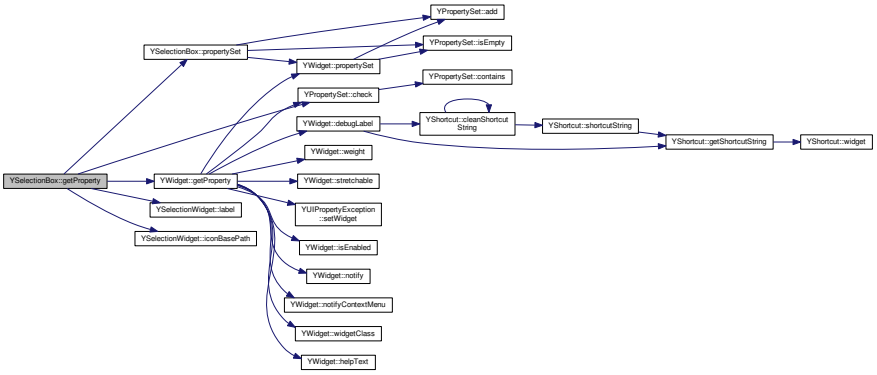
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 140 of file YSelectionBox.cc.

Here is the call graph for this function:



3.112.3.2 bool YSelectionBox::immediateMode ( ) const

Deliver even more events than with notify() set.

For YSelectionBox, this is relevant mostly for the NCurses UI:

In graphical UIs like the Qt UI, the user can use the mouse to select an item in a selection box. With notify() set, this will send an event right away (i.e., it will make UserInput and related return, while normally it would only return when the user clicks a PushButton).

In the NCurses UI, there is no mouse, so the user has to use the cursor keys to move to the item he wants to select. In immediateMode(), every cursor key press will make the selection box send an event. Without immediateMode(), the NCSelectionBox will wait until the user hits the [Return] key until an event is sent. Depending on what the application does upon each selection box event, immediateMode() might make the application less responsive.

Definition at line 79 of file YSelectionBox.cc.

3.112.3.3 const YPropertySet & YSelectionBox::propertySet ( ) [virtual]

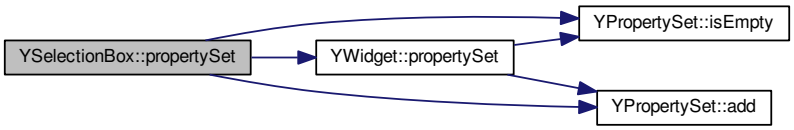
Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 95 of file YSelectionBox.cc.

Here is the call graph for this function:

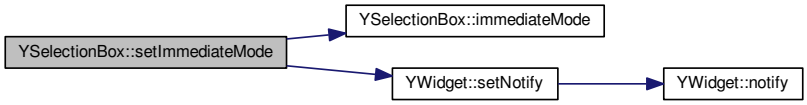


3.112.3.4 void YSelectionBox::setImmediateMode (bool *on* = `true`)

Set immediateMode() on or off.

Definition at line 85 of file YSelectionBox.cc.

Here is the call graph for this function:



3.112.3.5 bool YSelectionBox::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) `[virtual]`

Set a property. Reimplemented from YWidget.

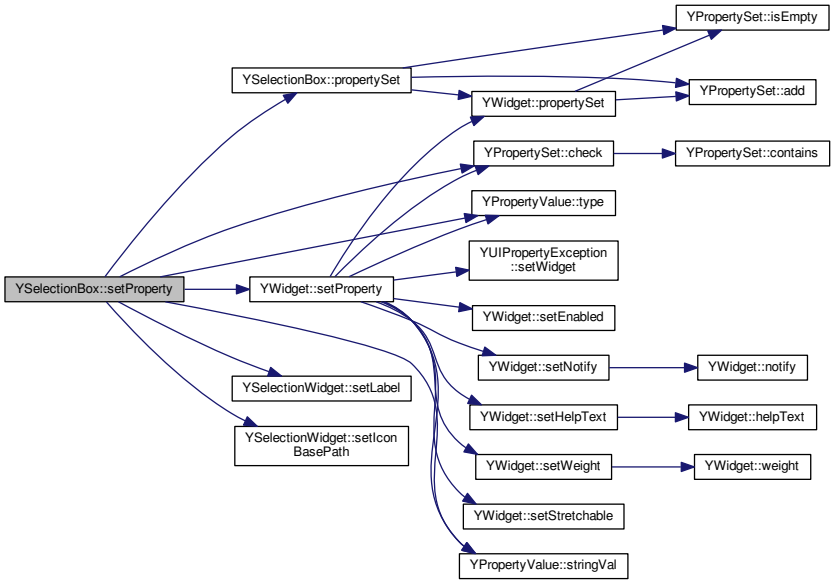
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 121 of file YSelectionBox.cc.

Here is the call graph for this function:



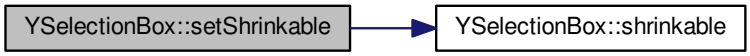
3.112.3.6 void YSelectionBox::setShrinkable (bool *shrinkable* = `true`) [virtual]

Make this SelectionBox very small. This will take effect only upon the next geometry management run.

Derived classes can overwrite this, but should call this base class function in the new function.

Definition at line 73 of file YSelectionBox.cc.

Here is the call graph for this function:



3.112.3.7 bool YSelectionBox::shrinkable ( ) const

Return 'true' if this SelectionBox should be very small.

Definition at line 67 of file YSelectionBox.cc.

3.112.3.8 const char* YSelectionBox::userInputProperty ( ) [inline],[virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 149 of file YSelectionBox.h.

3.112.3.9 virtual const char* YSelectionBox::widgetClass ( ) const [inline],[virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YSelectionWidget.

Definition at line 75 of file YSelectionBox.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSelectionBox.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSelectionBox.cc

3.113 YSelectionBoxPrivate Struct Reference

Public Attributes

- bool **shrinkable**
- bool **immediateMode**

3.113.1 Detailed Description

Definition at line 34 of file YSelectionBox.cc.

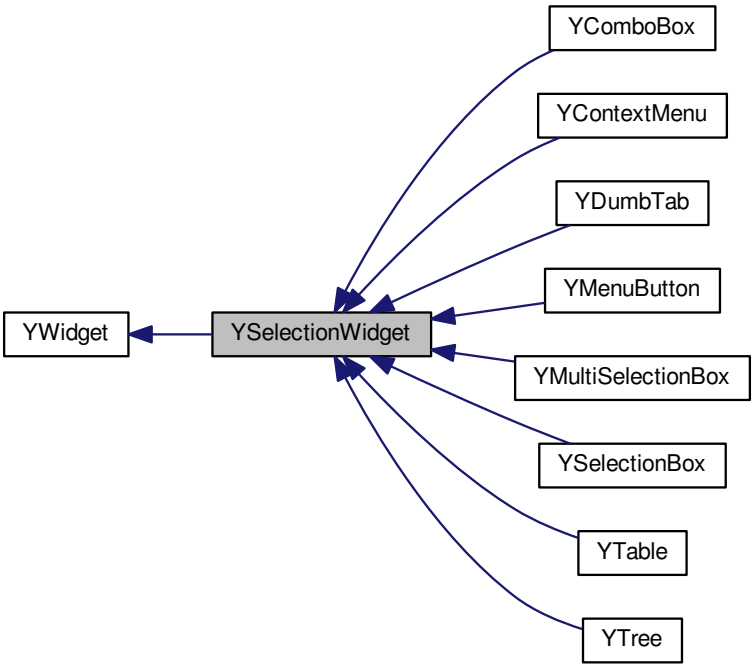
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSelectionBox.cc

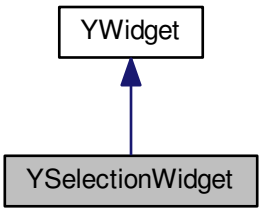
3.114 YSelectionWidget Class Reference

```
#include <YSelectionWidget.h>
```

Inheritance diagram for YSelectionWidget:



Collaboration diagram for YSelectionWidget:



Public Member Functions

- virtual ~YSelectionWidget ()
- virtual const char * widgetClass () const
- std::string label () const

- virtual void setLabel (const std::string &newLabel)
- virtual void addItem (YItem *item_disown)
- void addItem (const std::string &itemLabel, bool selected=false)
- void addItem (const std::string &itemLabel, const std::string &iconName, bool selected=false)
- virtual void addItems (const YItemCollection &itemCollection)
- virtual void deleteAllItems ()
- void setItems (const YItemCollection &itemCollection)
- YItemIterator itemsBegin ()
- YItemConstIterator **itemsBegin** () const
- YItemIterator itemsEnd ()
- YItemConstIterator **itemsEnd** () const
- bool hasItems () const
- int itemsCount () const
- YItem * firstItem () const
- virtual YItem * selectedItem ()
- virtual YItemCollection selectedItems ()
- bool hasSelectedItem ()
- virtual void selectItem (YItem *item, bool selected=true)
- virtual void deselectAllItems ()
- void setIconBasePath (const std::string &basePath)
- std::string iconBasePath () const
- std::string iconFullPath (const std::string &iconName) const
- std::string iconFullPath (YItem *item) const
- bool itemsContain (YItem *item) const
- YItem * findItem (const std::string &itemLabel) const
- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)

Protected Member Functions

- YSelectionWidget (YWidget *parent, const std::string &label, bool enforceSingleSelection, bool recurisve-Selection=false)
- void setEnforceSingleSelection (bool on)
- bool enforceSingleSelection () const
- bool recursiveSelection () const
- YItem * findSelectedItem (YItemConstIterator begin, YItemConstIterator end)
- void findSelectedItems (YItemCollection &selectedItems, YItemConstIterator begin, YItemConstIterator end)
- void deselectAllItems (YItemIterator begin, YItemIterator end)
- YItem * findItem (const std::string &wantedItemLabel, YItemConstIterator begin, YItemConstIterator end) const
- bool itemsContain (YItem *wantedItem, YItemConstIterator begin, YItemConstIterator end) const
- YItem * itemAt (int index) const

3.114.1 Detailed Description

Base class for selection widgets:

- YSelectionBox
- MultiselectionBox
- YCombobox

- YTree
- YDumbTab

Definition at line 42 of file YSelectionWidget.h.

3.114.2 Constructor & Destructor Documentation

3.114.2.1 YSelectionWidget::YSelectionWidget (YWidget * *parent*, const std::string & *label*, bool *enforceSingleSelection*, bool *recurisveSelection* = `false`) `[protected]`

Constructor.

'singleSelectionMode' indicates if this base class should enforce single selection when items are added or when items are selected from the application. Note that single selection can also mean that no item is selected.

Definition at line 55 of file YSelectionWidget.cc.

3.114.2.2 YSelectionWidget::~YSelectionWidget ( ) `[virtual]`

Destructor.

Definition at line 70 of file YSelectionWidget.cc.

Here is the call graph for this function:

3.114.3 Member Function Documentation

3.114.3.1 void YSelectionWidget::addItem (YItem * *item_disown*) `[virtual]`

Add one item. This widget assumes ownership of the item object and will delete it in its destructor.

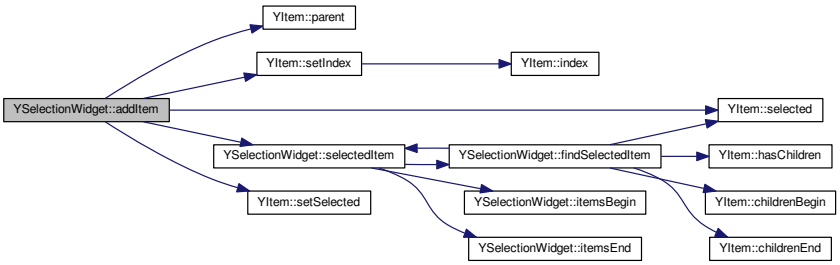
NOTE: For tree items, call this only for the toplevel items; all non-toplevel items are already owned by their respective parent items. Adding them to the parent widget will clash with this ownership.

Derived classes can overwrite this function, but they should call this base class function in the new implementation.

Reimplemented in YContextMenu, YMenuButton, and YDumbTab.

Definition at line 168 of file YSelectionWidget.cc.

Here is the call graph for this function:

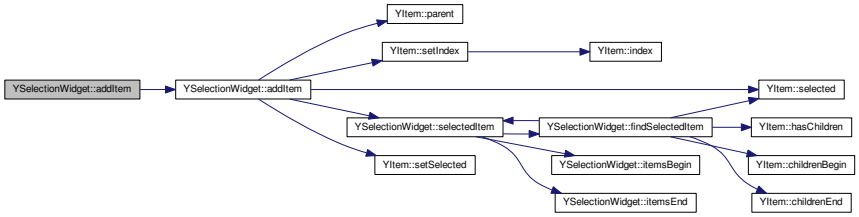


3.114.3.2 void YSelectionWidget::addItem (const std::string & *itemLabel,* bool *selected* = `false`)

Overloaded for convenience: Add an item by string.

Definition at line 235 of file YSelectionWidget.cc.

Here is the call graph for this function:

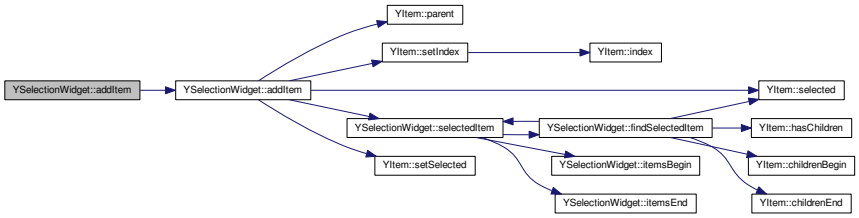


3.114.3.3 void YSelectionWidget::addItem (const std::string & *itemLabel,* const std::string & *iconName,* bool *selected* = `false`)

Overloaded for convenience: Add an item with a text and an icon. Note that not all UIs can display icons.

Definition at line 225 of file YSelectionWidget.cc.

Here is the call graph for this function:



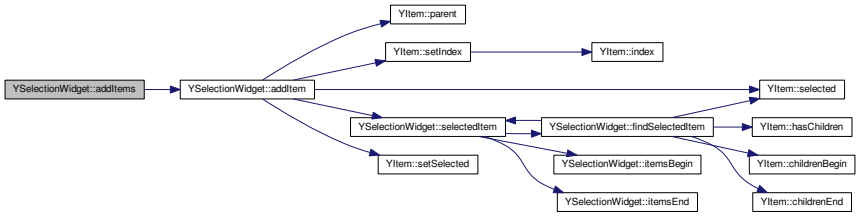
3.114.3.4 void YSelectionWidget::addItems (const YItemCollection & *itemCollection*) [virtual]

Add multiple items. For some UIs, this can be more efficient than calling addItem() multiple times.

Reimplemented in YTree, YContextMenu, and YMenuButton.

Definition at line 241 of file YSelectionWidget.cc.

Here is the call graph for this function:



3.114.3.5 void YSelectionWidget::deleteAllItems ( ) [virtual]

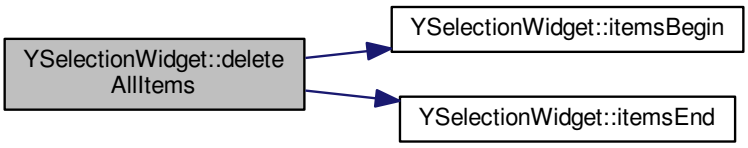
Delete all items.

Derived classes can overwrite this function, but they should call this base class function in the new implementation.

Reimplemented in YContextMenu, and YMenuButton.

Definition at line 76 of file YSelectionWidget.cc.

Here is the call graph for this function:



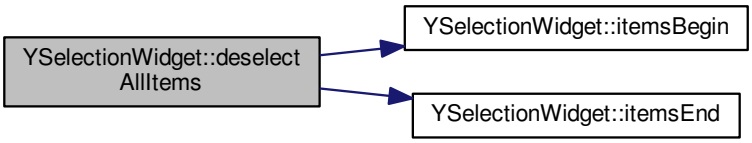
3.114.3.6 void YSelectionWidget::deselectAllItems ( ) [virtual]

Deselect all items.

Derived classes can overwrite this function, but they should call this base class function in the new implementation.

Definition at line 454 of file YSelectionWidget.cc.

Here is the call graph for this function:

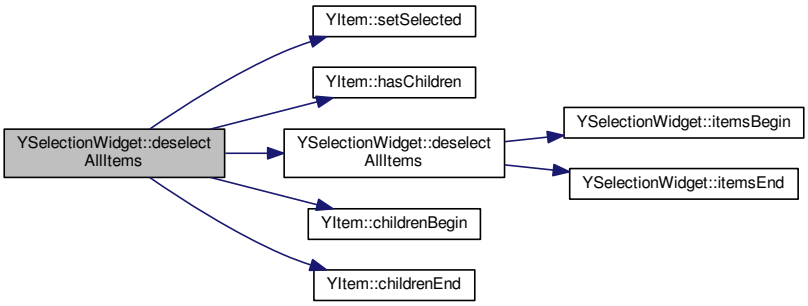


3.114.3.7 void YSelectionWidget::deselectAllItems (YItemIterator *begin*, YItemIterator *end*) [protected]

Recursively deselect all items between iterators 'begin' and 'end'.

Definition at line 460 of file YSelectionWidget.cc.

Here is the call graph for this function:



3.114.3.8 bool YSelectionWidget::enforceSingleSelection ( ) const [protected]

Return 'true' if this base class should enforce single selection.

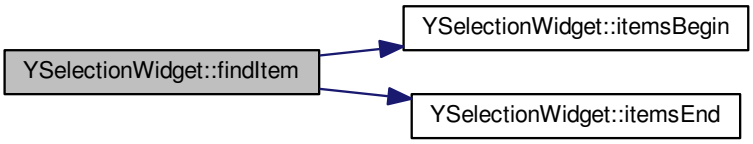
Definition at line 107 of file YSelectionWidget.cc.

3.114.3.9 YItem * YSelectionWidget::findItem (const std::string & *itemLabel*) const

Find the (first) item with the specified label. Return 0 if there is no item with that label.

Definition at line 476 of file YSelectionWidget.cc.

Here is the call graph for this function:

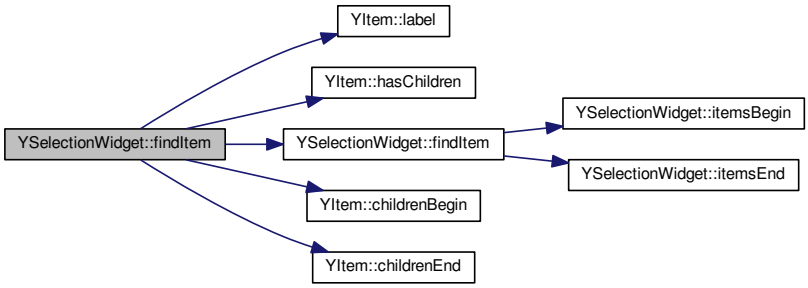


3.114.3.10 YItem * YSelectionWidget::findItem (const std::string & *wantedItemLabel,* YItemConstIterator *begin,* YItemConstIterator *end*) const [protected]

Recursively try to find an item with label 'wantedItemLabel' between iterators 'begin' and 'end'. Return that item or 0 if there is none.

Definition at line 483 of file YSelectionWidget.cc.

Here is the call graph for this function:

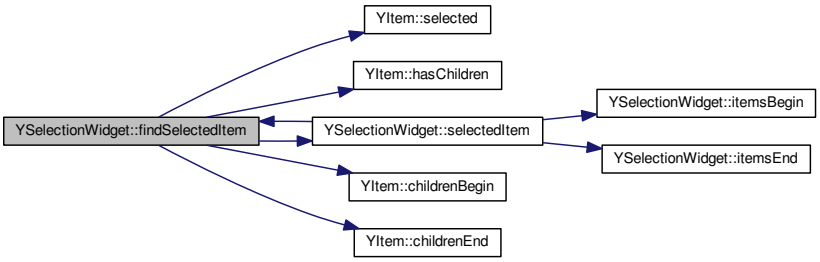


3.114.3.11 YItem * YSelectionWidget::findSelectedItem (YItemConstIterator *begin,* YItemConstIterator *end*) [protected]

Recursively try to find the first selected item between iterators 'begin' and 'end'. Return that item or 0 if there is none.

Definition at line 326 of file YSelectionWidget.cc.

Here is the call graph for this function:

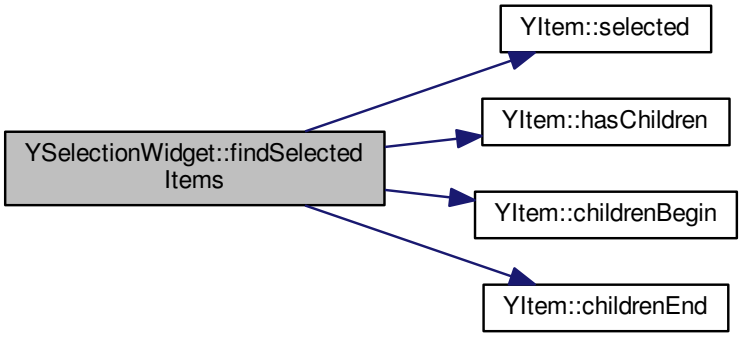


3.114.3.12 void YSelectionWidget::findSelectedItems (YItemCollection & *selectedItems*, YItemConstIterator *begin*, YItemConstIterator *end*) [protected]

Recursively find all selected items between iterators 'begin' and 'end' and add each of them to the 'selectedItems' YItemCollection.

Definition at line 363 of file YSelectionWidget.cc.

Here is the call graph for this function:



3.114.3.13 YItem * YSelectionWidget::firstItem ( ) const

Return the first item or 0 if there is none.

Definition at line 299 of file YSelectionWidget.cc.

3.114.3.14 bool YSelectionWidget::hasItems ( ) const

Return 'true' if this widget has any items.

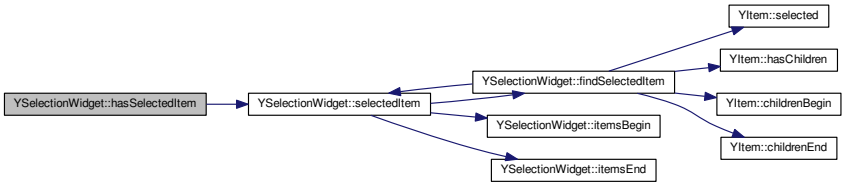
Definition at line 286 of file YSelectionWidget.cc.

3.114.3.15 bool YSelectionWidget::hasSelectedItem ( )

Return 'true' if any item is selected.

Definition at line 384 of file YSelectionWidget.cc.

Here is the call graph for this function:



3.114.3.16 std::string YSelectionWidget::iconBasePath ( ) const

Return this widget's base path where to look up icons as set with setIconBasePath().

Definition at line 131 of file YSelectionWidget.cc.

3.114.3.17 std::string YSelectionWidget::iconFullPath (const std::string & *iconName*) const

Return the full path + file name for the specified icon name. If iconBasePath is non-empty, it is prepended to the icon name. Otherwise, YUI::yApp()->iconLoader() and its icon search paths is used find the icon in one of them

If 'iconName' is empty, this will return an empty string.

Definition at line 137 of file YSelectionWidget.cc.

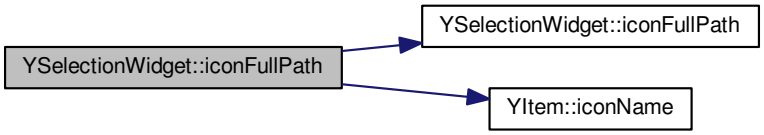
3.114.3.18 std::string YSelectionWidget::iconFullPath (YItem * *item*) const

Return the full path + file name for the icon of the specified item. If iconBasePath is non-empty, it is prepended to the item's iconName. Otherwise, YUI::yApp()->iconLoader() and its icon search paths is used find the icon in one of them

If 'item' does not have an iconName specified, this will return an empty string.

Definition at line 159 of file YSelectionWidget.cc.

Here is the call graph for this function:



3.114.3.19 YItem * YSelectionWidget::itemAt (int *index*) const [protected]

Return the item at index 'index' (from 0) or 0 if there is no such item.

Definition at line 309 of file YSelectionWidget.cc.

3.114.3.20 YItemIterator YSelectionWidget::itemsBegin ( )

Return an iterator that points to the first item.

For YSelectionWidgets that can have tree structures, this iterator will iterate over the toplevel items.

Important: Don't use this iterator to iterate over all items and check their "selected" state; that information might not always be up to date. Use the dedicated functions for that.

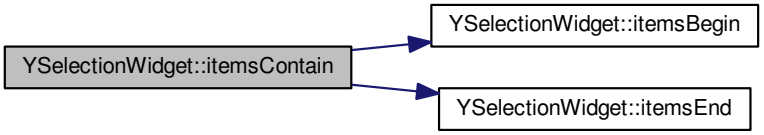
Definition at line 260 of file YSelectionWidget.cc.

3.114.3.21 bool YSelectionWidget::itemsContain (YItem * *item*) const

Return 'true' if this widget's items contain the specified item.

Definition at line 420 of file YSelectionWidget.cc.

Here is the call graph for this function:

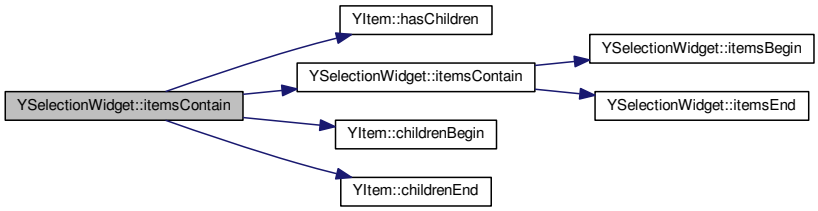


3.114.3.22 bool YSelectionWidget::itemsContain (YItem * *wantedItem,* YItemConstIterator *begin,* YItemConstIterator *end*) const [protected]

Recursively check if 'wantedItem' is between iterators 'begin' and 'end'.

Definition at line 428 of file YSelectionWidget.cc.

Here is the call graph for this function:



3.114.3.23 int YSelectionWidget::itemsCount ( ) const

Return the number of items.

For YSelectionWidgets that can have tree structures, this returns the number of toplevel items.

Definition at line 292 of file YSelectionWidget.cc.

3.114.3.24 YItemIterator YSelectionWidget::itemsEnd ( )

Return an iterator that points behind the last item.

Definition at line 273 of file YSelectionWidget.cc.

3.114.3.25 std::string YSelectionWidget::label ( ) const

Return this widget's label (the caption above the item list).

Definition at line 95 of file YSelectionWidget.cc.

3.114.3.26 bool YSelectionWidget::recursiveSelection ( ) const [protected]

Return 'true' if this base class should select children recursively.

Definition at line 112 of file YSelectionWidget.cc.

3.114.3.27 YItem * YSelectionWidget::selectedItem ( ) [virtual]

Return the (first) selected item or 0 if none is selected.

Reimplemented in YComboBox.

Definition at line 319 of file YSelectionWidget.cc.

Here is the call graph for this function:

3.114.3.28 YItemCollection YSelectionWidget::selectedItems ( ) [virtual]

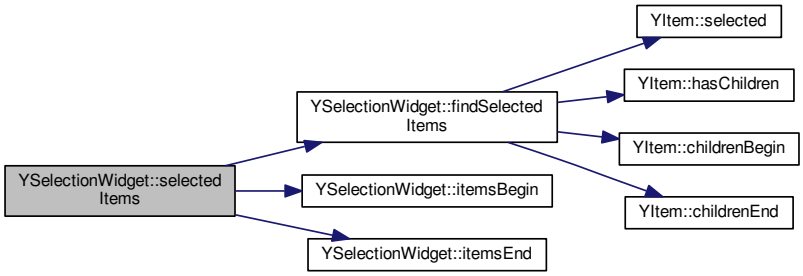
Return all selected items. This is mostly useful for derived classes that allow selecting multiple items.

This function does not transfer ownership of those items to the caller, so don't try to delete them!

Reimplemented in YComboBox.

Definition at line 353 of file YSelectionWidget.cc.

Here is the call graph for this function:



3.114.3.29 void YSelectionWidget::selectItem (YItem * *item*, bool *selected* = true) [virtual]

Select or deselect an item.

Notice that this is different from YItem::setSelected() because unlike the latter function, this function informs the parent widget of the selection change.

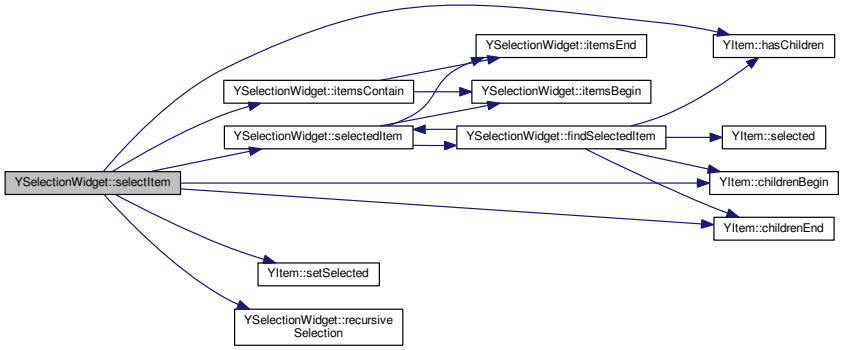
If only one item can be selected at any time (single selection), the derived class will make sure to deselect any previous selection, if applicable.

Derived classes should overwrite this function, but they should call this base class function at the new function's start (this will also check if the item really belongs to this widget and throw an exception if not).

Reimplemented in YComboBox.

Definition at line 390 of file YSelectionWidget.cc.

Here is the call graph for this function:



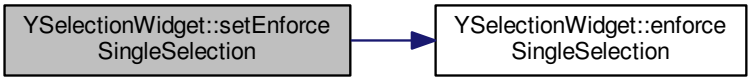
3.114.3.30 void YSelectionWidget::setEnforceSingleSelection (bool *on*) [protected]

Set single selection mode on or off. In single selection mode, only one item can be selected at any time.

If set, this base class enforces this when items are added or when items are selected from the application. Note that single selection can also mean that no item is selected.

Definition at line 119 of file YSelectionWidget.cc.

Here is the call graph for this function:



3.114.3.31 void YSelectionWidget::setIconBasePath (const std::string & *basePath*)

Set this widget's base path where to look up icons. If this is a relative path, YUI::qApp()->iconBasePath() is prepended.

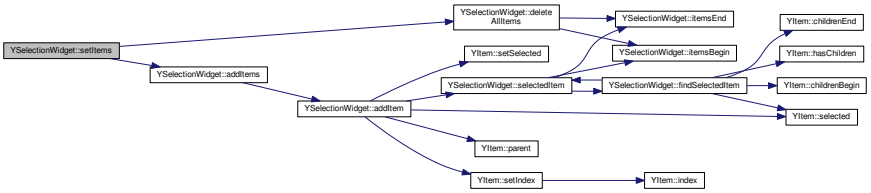
Definition at line 125 of file YSelectionWidget.cc.

3.114.3.32 void YSelectionWidget::setItems (const YItemCollection & *itemCollection*) [inline]

Delete all items and add new items.

Definition at line 127 of file YSelectionWidget.h.

Here is the call graph for this function:



3.114.3.33 void YSelectionWidget::setLabel (const std::string & *newLabel*) [virtual]

Change this widget's label (the caption above the item list).

Derived classes should overwrite this function, but they should call this base class function in the new implementation.

Definition at line 101 of file YSelectionWidget.cc.

3.114.3.34 virtual void YSelectionWidget::setShortcutString (const std::string & *str*) [inline], [virtual]

Set the string of this widget that holds the keyboard shortcut.

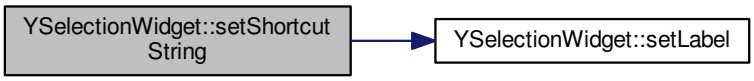
Reimplemented from YWidget.

Reimplemented from YWidget.

Reimplemented in YDumbTab.

Definition at line 268 of file YSelectionWidget.h.

Here is the call graph for this function:



3.114.3.35 virtual std::string YSelectionWidget::shortcutString ( ) const [inline], [virtual]

Get the string of this widget that holds the keyboard shortcut.

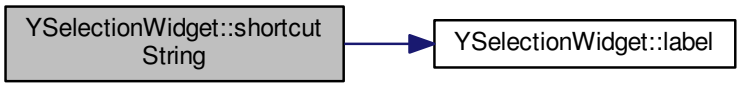
Reimplemented from YWidget.

Reimplemented from YWidget.

Reimplemented in YDumbTab.

Definition at line 261 of file YSelectionWidget.h.

Here is the call graph for this function:



3.114.3.36 virtual const char* YSelectionWidget::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Reimplemented in YTable, YSelectionBox, YComboBox, YTree, YContextMenu, YMenuButton, YDumbTab, and YMultiSelectionBox.

Definition at line 69 of file YSelectionWidget.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSelectionWidget.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSelectionWidget.cc

3.115 YSelectionWidgetPrivate Struct Reference

Public Member Functions

- **YSelectionWidgetPrivate** (const std::string &label, bool enforceSingleSelection, bool recursiveSelection)

Public Attributes

- std::string **label**
- bool **enforceSingleSelection**
- bool **recursiveSelection**
- std::string **iconBasePath**
- YItemCollection **itemCollection**

3.115.1 Detailed Description

Definition at line 35 of file YSelectionWidget.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSelectionWidget.cc

3.116 YSettings Class Reference

```
#include <YSettings.h>
```

Static Public Member Functions

- static void setProgDir (std::string directory)
- static std::string progDir ()
- static void setIconDir (std::string directory)
- static std::string iconDir ()
- static void setThemeDir (std::string directory)
- static std::string themeDir ()
- static void setLocaleDir (std::string directory)
- static std::string localeDir ()

3.116.1 Detailed Description

Settings for libyui

This singleton-object hold some presets for libyui.

Definition at line 43 of file YSettings.h.

3.116.2 Member Function Documentation

3.116.2.1 std::string YSettings::iconDir ( ) [static]

Returns the value of your program's icons subdir.

Definition at line 95 of file YSettings.cc.

3.116.2.2 std::string YSettings::localeDir ( ) [static]

Returns the value of your program's locale subdir.

Definition at line 157 of file YSettings.cc.

3.116.2.3 std::string YSettings::progDir ( ) [static]

Returns the value of your program's subdir.

Definition at line 71 of file YSettings.cc.

3.116.2.4 void YSettings::setIconDir (std::string *directory*) [static]

This can be used to set a subdir ICONDIR, where your program stores a custom icons.

Once this is set, it can't be altered. If you do so although an exception will be thrown.

Definition at line 79 of file YSettings.cc.

3.116.2.5 void YSettings::setLocaleDir (std::string *directory*) [static]

This can be used to set a subdir LOCALEDIR, where your program stores translations

Once this is set, it can't be altered. If you do so although an exception will be thrown.

Definition at line 141 of file YSettings.cc.

3.116.2.6 void YSettings::setProgDir (std::string *directory*) [static]

This can be used to set a subdir beneath PLUGINDIR or THEMEDIR, where your program stores a custom plugin or theme.

Once this is set, it can't be altered. If you do so although an exception will be thrown.

Definition at line 55 of file YSettings.cc.

3.116.2.7 void YSettings::setThemeDir (std::string *directory*) [static]

This can be used to set a subdir THEMEDIR, where your program stores a custom icons.

Once this is set, it can't be altered. If you do so although an exception will be thrown.

Definition at line 108 of file YSettings.cc.

3.116.2.8 std::string YSettings::themeDir ( ) [static]

Returns the value of your program's theme subdir.

Definition at line 124 of file YSettings.cc.

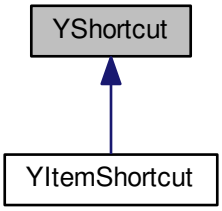
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSettings.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSettings.cc

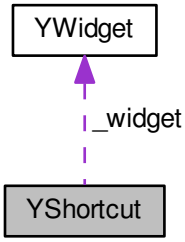
3.117 YShortcut Class Reference

```
#include <YShortcut.h>
```

Inheritance diagram for YShortcut:



Collaboration diagram for YShortcut:



Public Types

- enum { **None** = 0 }

Public Member Functions

- YShortcut (YWidget *shortcut_widget)
- virtual ~YShortcut ()
- YWidget * widget () const
- const char * widgetClass () const
- bool isButton () const
- bool isWizardButton () const
- std::string shortcutString ()
- std::string cleanShortcutString ()
- char preferred ()
- char shortcut ()

- virtual void setShortcut (char newShortcut)
- void clearShortcut ()
- bool conflict ()
- void setConflict (bool newConflictState=true)
- int distinctShortcutChars ()
- bool hasValidShortcutChar ()

Static Public Member Functions

- static std::string cleanShortcutString (std::string shortcutString)
- static char shortcutMarker ()
- static std::string::size_type findShortcutPos (const std::string &str, std::string::size_type start_pos=0)
- static char findShortcut (const std::string &str, std::string::size_type start_pos=0)
- static bool isValid (char c)
- static char normalized (char c)
- static std::string getShortcutString (const YWidget *widget)

Protected Member Functions

- virtual std::string getShortcutString ()

Protected Attributes

- YWidget * **_widget**
- std::string **_shortcutString**
- bool **_shortcutStringCached**
- std::string **_cleanShortcutString**
- bool **_cleanShortcutStringCached**
- int **_preferred**
- int **_shortcut**
- bool **_conflict**
- bool **_isButton**
- bool **_isWizardButton**
- int **_distinctShortcutChars**

3.117.1 Detailed Description

Helper class for shortcut management: This class holds data about the shortcut for one single widget.

Definition at line 40 of file YShortcut.h.

3.117.2 Member Enumeration Documentation

3.117.2.1 anonymous enum

Marker for "no shortcut"

Definition at line 56 of file YShortcut.h.

3.117.3 Constructor & Destructor Documentation

3.117.3.1 YShortcut::YShortcut (YWidget * *shortcut_widget*)

Constructor

Definition at line 41 of file YShortcut.cc.

Here is the call graph for this function:

3.117.3.2 YShortcut::~YShortcut ( ) [virtual]

Destructor

Definition at line 69 of file YShortcut.cc.

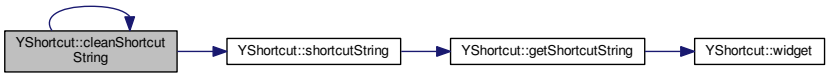
3.117.4 Member Function Documentation

3.117.4.1 std::string YShortcut::cleanShortcutString ( )

Returns the shortcut string (from the widget's shortcut property) without any "&" markers.

Definition at line 91 of file YShortcut.cc.

Here is the call graph for this function:

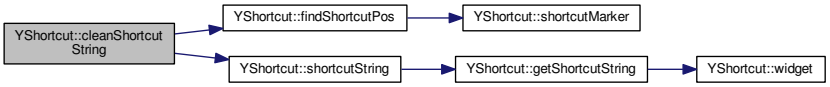


3.117.4.2 std::string YShortcut::cleanShortcutString (std::string *shortcutString*) [static]

Static version of the above for general use: Returns the specified string without any "&" markers.

Definition at line 103 of file YShortcut.cc.

Here is the call graph for this function:

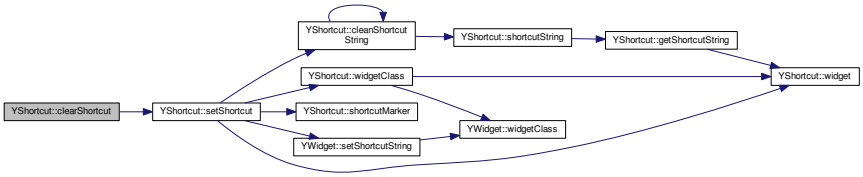


3.117.4.3 void YShortcut::clearShortcut ( )

Clear the shortcut: Override the shortcut character with nothing. This may happen if a conflict cannot be resolved.

Definition at line 173 of file YShortcut.cc.

Here is the call graph for this function:



3.117.4.4 bool YShortcut::conflict ( ) [inline]

Query the internal 'conflict' marker. This class doesn't care about that flag, it just stores it for the convenience of higher-level classes.

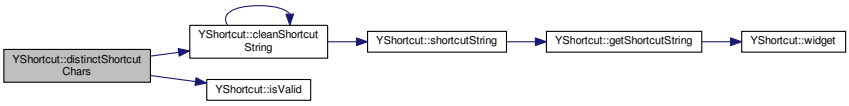
Definition at line 131 of file YShortcut.h.

3.117.4.5 int YShortcut::distinctShortcutChars ( )

Obtain the number of distinct valid shortcut characters in the shortcut string, i.e. how many different shortcuts that widget could get.

Definition at line 180 of file YShortcut.cc.

Here is the call graph for this function:



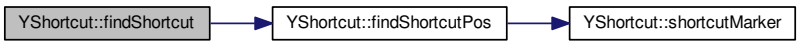
3.117.4.6 char YShortcut::findShortcut (const std::string & *str*, std::string::size_type *start_pos* = 0) [static]

Static function: Find the next shortcut marker in a string, beginning at starting position start_pos.

Returns the shortcut character or 0 if none found.

Definition at line 280 of file YShortcut.cc.

Here is the call graph for this function:



3.117.4.7 std::string::size_type YShortcut::findShortcutPos (const std::string & *str*, std::string::size_type *start_pos* = 0) [static]

Static function: Find the next occurrence of the shortcut marker ('&') in a string, beginning at starting position start_pos.

Returns string::npos if not found or the position of the shortcut marker (not the shortcut character!) if found.

Definition at line 254 of file YShortcut.cc.

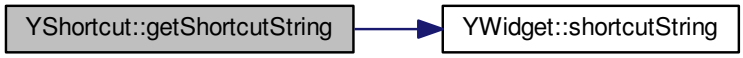
Here is the call graph for this function:

3.117.4.8 std::string YShortcut::getShortcutString (const YWidget * *widget*) [static]

Obtain a widget's shortcut property - the string that contains "&" to designate a shortcut.

Definition at line 244 of file YShortcut.cc.

Here is the call graph for this function:



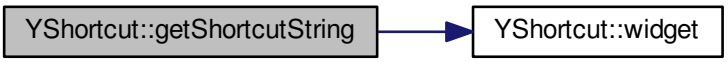
3.117.4.9 std::string YShortcut::getShortcutString ( ) [protected],[virtual]

Obtain the the shortcut property of this shortcut's widget - the string that contains "&" to designate a shortcut.

Reimplemented in YItemShortcut.

Definition at line 237 of file YShortcut.cc.

Here is the call graph for this function:

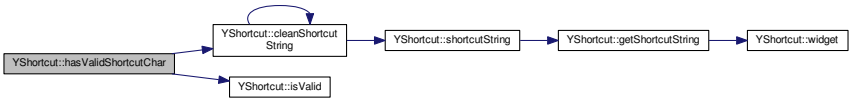


3.117.4.10 bool YShortcut::hasValidShortcutChar ( )

Return true if this shortcut contains any character that would be valid as a shortcut character.

Definition at line 222 of file YShortcut.cc.

Here is the call graph for this function:



3.117.4.11 bool YShortcut::isButton ( ) const [inline]

Returns 'true' if the widget that is associated with this shortcut is a button (derived from YPushButton).

Definition at line 73 of file YShortcut.h.

3.117.4.12 bool YShortcut::isValid (char *c*) [static]

Returns 'true' if 'c' is a valid shortcut character, i.e. [a-zA-Z0-9], 'false' otherwise.

Definition at line 289 of file YShortcut.cc.

3.117.4.13 bool YShortcut::isWizardButton ( ) const [inline]

Returns 'true' if the widget that is associated with this shortcut is a wizard button (one of the navigation buttons of a wizard).

Definition at line 79 of file YShortcut.h.

3.117.4.14 char YShortcut::normalized (char *c*) [static]

Return the normalized version of shortcut character 'c', i.e. a lowercase letter or a digit [a-z0-9]. Returns 0 if 'c' is invalid.

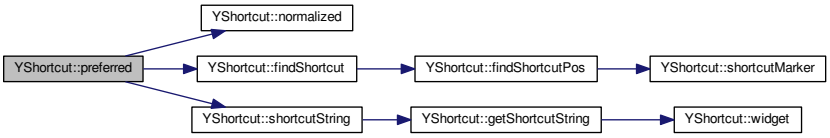
Definition at line 299 of file YShortcut.cc.

3.117.4.15 char YShortcut::preferred ( )

The preferred shortcut character, i.e. the character that had been preceded by "&" before checking / resolving conflicts began.

Definition at line 117 of file YShortcut.cc.

Here is the call graph for this function:



3.117.4.16 void YShortcut::setConflict (bool *newConflictState* = true) [inline]

Set or unset the internal 'conflict' marker.

Definition at line 136 of file YShortcut.h.

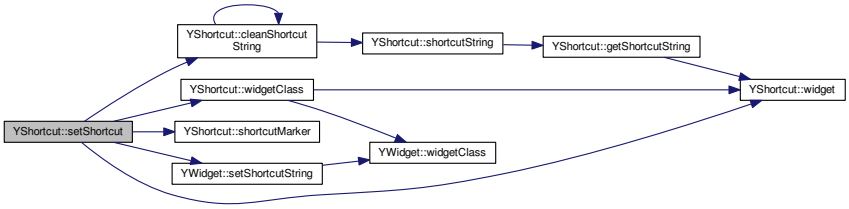
3.117.4.17 void YShortcut::setShortcut (char *newShortcut*) [virtual]

Set (override) the shortcut character.

Reimplemented in YItemShortcut.

Definition at line 141 of file YShortcut.cc.

Here is the call graph for this function:



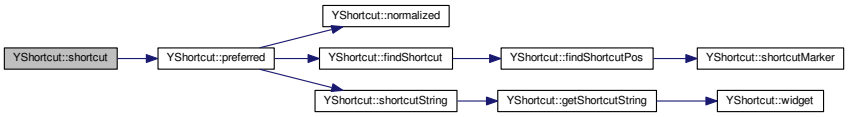
3.117.4.18 char YShortcut::shortcut ( )

The actual shortcut character.

This may be different from preferred() if it is overridden.

Definition at line 129 of file YShortcut.cc.

Here is the call graph for this function:



3.117.4.19 static char YShortcut::shortcutMarker ( ) [inline], [static]

Static function: Returns the character used for marking keyboard shortcuts.

Definition at line 154 of file YShortcut.h.

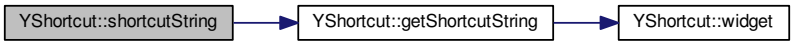
3.117.4.20 std::string YShortcut::shortcutString ( )

Returns the complete shortcut string (which may or may not contain "&"), i.e. the value of the widget's shortcut property. For PushButtons, this is the label on the button (e.g., "&Details..."), for other widgets usually the caption above it.

This value is chached, i.e. this isn't a too expensive operation.

Definition at line 75 of file YShortcut.cc.

Here is the call graph for this function:



3.117.4.21 YWidget* YShortcut::widget ( ) const [inline]

Returns the YWidget this shortcut data belong to.

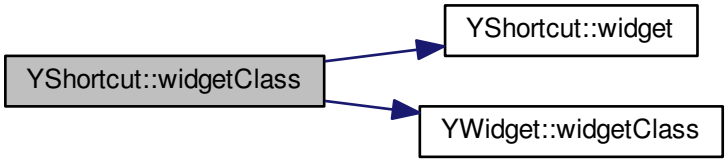
Definition at line 61 of file YShortcut.h.

3.117.4.22 const char* YShortcut::widgetClass ( ) const [inline]

Returns the textual representation of the widget class of the widget this shortcut data belongs to.

Definition at line 67 of file YShortcut.h.

Here is the call graph for this function:



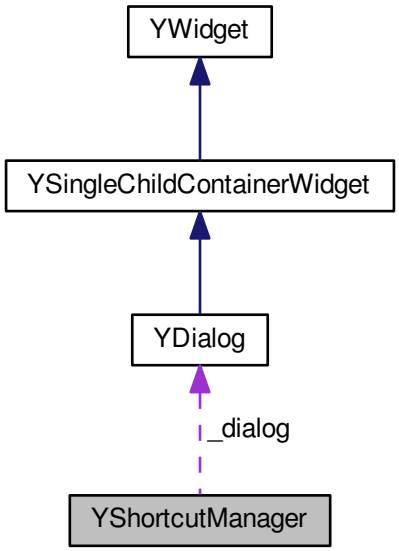
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YShortcut.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YShortcut.cc

3.118 YShortcutManager Class Reference

```
#include <YShortcutManager.h>
```

Collaboration diagram for YShortcutManager:



Public Member Functions

- YShortcutManager (YDialog *dialog)
- virtual ~YShortcutManager ()
- void checkShortcuts (bool autoResolve=true)
- int conflictCount ()
- void resolveAllConflicts ()
- YDialog * dialog ()

Protected Member Functions

- void clearShortcutList ()
- void findShortcutWidgets (YWidgetListConstIterator begin, YWidgetListConstIterator end)
- void resolveConflict (YShortcut *shortcut)
- int findShortestWizardButton (const YShortcutList &conflictList)
- unsigned findShortestWidget (const YShortcutList &conflictList)

Protected Attributes

- YDialog * _dialog
- YShortcutList _shortcutList
- int _wanted [sizeof(char)<< 8]
- bool _used [sizeof(char)<< 8]
- int _conflictCount

3.118.1 Detailed Description

Helper class to manage keyboard shortcuts within one dialog and resolve keyboard shortcut conflicts.

Definition at line 38 of file YShortcutManager.h.

3.118.2 Constructor & Destructor Documentation

3.118.2.1 YShortcutManager::YShortcutManager (YDialog * *dialog*)

Constructor.

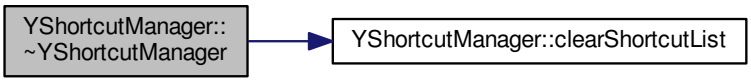
Definition at line 44 of file YShortcutManager.cc.

3.118.2.2 YShortcutManager::~YShortcutManager ( ) [virtual]

Destructor

Definition at line 53 of file YShortcutManager.cc.

Here is the call graph for this function:



3.118.3 Member Function Documentation

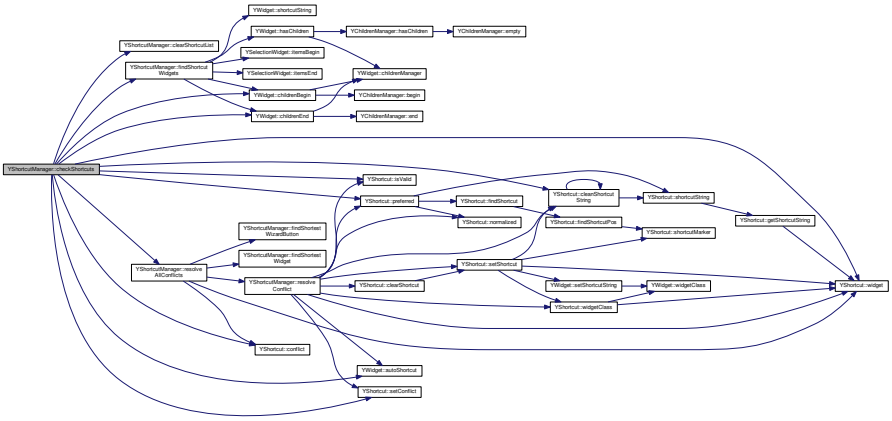
3.118.3.1 void YShortcutManager::checkShortcuts (bool *autoResolve* = `true`)

Check the keyboard shortcuts of all children of this dialog (not for sub-dialogs!).

Call resolveAllConflicts() if 'autoResolve' is 'true'.

Definition at line 60 of file YShortcutManager.cc.

Here is the call graph for this function:



3.118.3.2 void YShortcutManager::clearShortcutList ( ) `[protected]`

Delete all members of the internal shortcut list, then empty the list.

Definition at line 367 of file YShortcutManager.cc.

3.118.3.3 int YShortcutManager::conflictCount ( ) `[inline]`

Returns the number of shortcut conflicts. Valid only after checkShortcuts() or resolveAllConflicts().

Definition at line 63 of file YShortcutManager.h.

3.118.3.4 YDialog* YShortcutManager::dialog ( ) [inline]

Returns the dialog this shortcut manager works on.

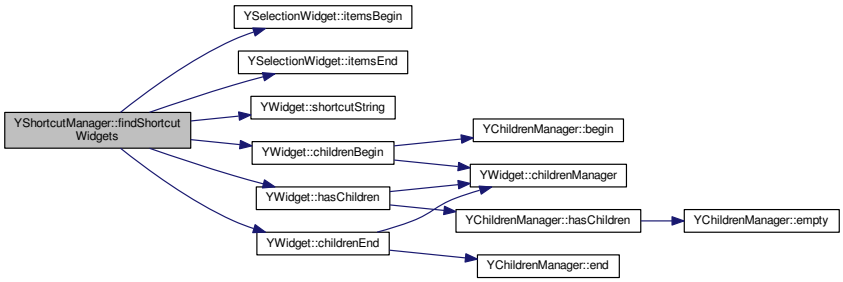
Definition at line 104 of file YShortcutManager.h.

3.118.3.5 void YShortcutManager::findShortcutWidgets (YWidgetListConstIterator *begin*, YWidgetListConstIterator *end*) [protected]

Recursively search all widgets between iterators 'begin' and 'end' (not those of any sub-dialogs!) for child widgets that could accept a keyboard shortcut and add these to _shortcutList.

Definition at line 379 of file YShortcutManager.cc.

Here is the call graph for this function:



3.118.3.6 unsigned YShortcutManager::findShortestWidget (const YShortcutList & *conflictList*) [protected]

Find the shortest widget in 'conflictList'. Buttons get priority if they have the same number of eligible shortcut characters as another widget.

Returns the index of the shortest widget.

Definition at line 332 of file YShortcutManager.cc.

3.118.3.7 int YShortcutManager::findShortestWizardButton (const YShortcutList & *conflictList*) [protected]

Find the shortest wizard button in 'conflictList', if there is any. Returns the index of that shortest wizard button or -1 if there is none.

Definition at line 307 of file YShortcutManager.cc.

3.118.3.8 void YShortcutManager::resolveAllConflicts ( )

Resolve shortcut conflicts. Requires checkShortcuts() to be called first.

Note: This may or may not work. There is no general solution to that problem. This method tries its best, but you may end up with widgets that don't have any (more) shortcut.

Why? Just picture the following (admittedly pathologic) situation:

[& OK] [& OK] [& OK]

This will result in something like this:

[& OK] [O& K] [OK]

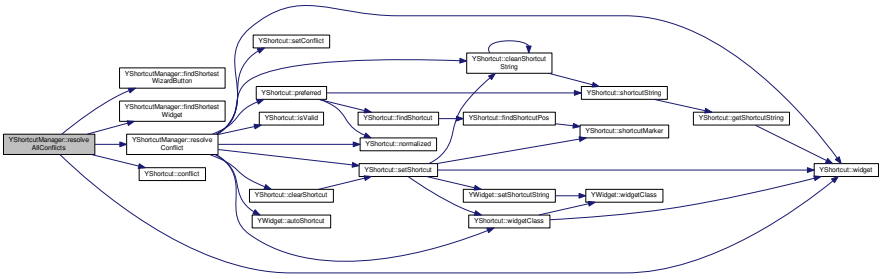
I.e. the first OK button will retain its preferred shortcut ('O'), the second OK button's shortcut will be reassigned to 'K' and the third won't get any - there are simply not enough eligible shortcut characters.

This may even fail in much less pathological situations. This example is only supposed to give you a general idea why not to blindly rely on automatic shortcut resolving.

It's always best to resolve conflicts manually. This will generally result in much better shortcuts: Easier to memorize, less chance of picking characters that cannot really do a good job showing their shortcut like very narrow characters (.e.g., 'i') or descender characters (e.g., 'g', 'p', 'q' - imagine those underlined!).

Definition at line 161 of file YShortcutManager.cc.

Here is the call graph for this function:

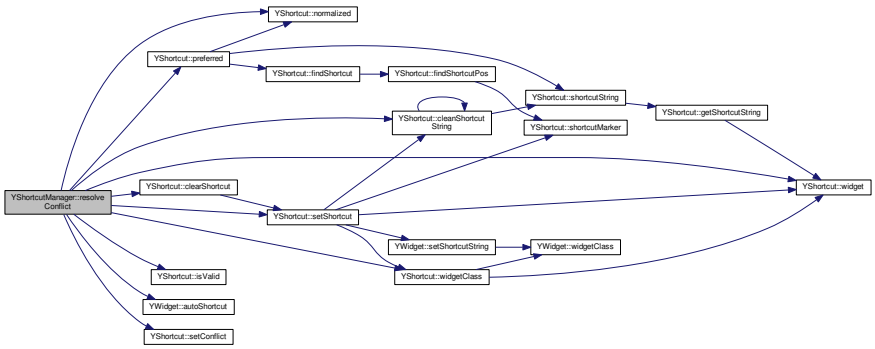


3.118.3.9 void YShortcutManager::resolveConflict (YShortcut * *shortcut*) [protected]

Pick a new shortcut character for 'shortcut' - one that isn't marked as used in the '_used' array. Unset the conflict marker if that succeeded.

Definition at line 229 of file YShortcutManager.cc.

Here is the call graph for this function:



3.118.4 Member Data Documentation

3.118.4.1 int YShortcutManager::_conflictCount [protected]

Counter for shortcut conflicts

Definition at line 166 of file YShortcutManager.h.

3.118.4.2 YDialog* YShortcutManager::_dialog [protected]

The dialog this shortcut manager works on.

Definition at line 144 of file YShortcutManager.h.

3.118.4.3 YShortcutList YShortcutManager::_shortcutList [protected]

List of all the shortcuts in this dialog.

Definition at line 149 of file YShortcutManager.h.

3.118.4.4 bool YShortcutManager::_used[sizeof(char)<< 8] [protected]

Flags for used shortcut characters.

Definition at line 160 of file YShortcutManager.h.

3.118.4.5 int YShortcutManager::_wanted[sizeof(char)<< 8] [protected]

Counters for wanted shortcut characters.

Definition at line 154 of file YShortcutManager.h.

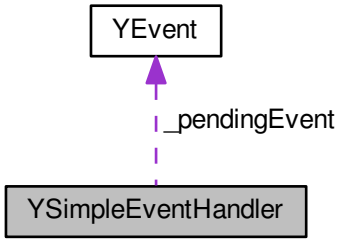
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YShortcutManager.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YShortcutManager.cc

3.119 YSimpleEventHandler Class Reference

```
#include <YSimpleEventHandler.h>
```

Collaboration diagram for YSimpleEventHandler:



Public Member Functions

- YSimpleEventHandler ()
- virtual ~YSimpleEventHandler ()
- void sendEvent (YEvent *event_disown)
- bool eventPendingFor (YWidget *widget) const
- YEvent * pendingEvent () const
- YEvent * consumePendingEvent ()
- void deletePendingEventsFor (YWidget *widget)
- void clear ()
- void blockEvents (bool block=true)
- void unblockEvents ()
- bool eventsBlocked () const
- void deleteEvent (YEvent *event)

Protected Attributes

- YEvent * **_pendingEvent**
- bool **_eventsBlocked**

3.119.1 Detailed Description

Simple event handler suitable for most UIs.

This event handler keeps track of one single event that gets overwritten when a new one arrives.

Definition at line 39 of file YSimpleEventHandler.h.

3.119.2 Constructor & Destructor Documentation

3.119.2.1 YSimpleEventHandler::YSimpleEventHandler ( )

Constructor.

Definition at line 38 of file YSimpleEventHandler.cc.

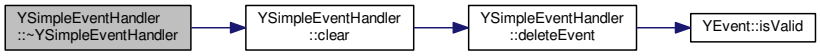
3.119.2.2 YSimpleEventHandler::~YSimpleEventHandler ( ) [virtual]

Destructor.

If there is a pending event, it is deleted here.

Definition at line 45 of file YSimpleEventHandler.cc.

Here is the call graph for this function:



3.119.3 Member Function Documentation

3.119.3.1 void YSimpleEventHandler::blockEvents (bool *block* = true)

Block (or unblock) events. If events are blocked, any event sent with sendEvent() from now on is ignored (and will get lost) until events are unblocked again.

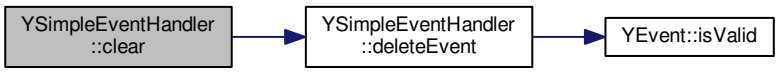
Definition at line 146 of file YSimpleEventHandler.cc.

3.119.3.2 void YSimpleEventHandler::clear ( )

Clears any pending event (deletes the corresponding object).

Definition at line 51 of file YSimpleEventHandler.cc.

Here is the call graph for this function:



3.119.3.3 YEvent * YSimpleEventHandler::consumePendingEvent ( )

Consumes the pending event. Sets the internal pending event to 0. Does NOT delete the internal consuming event.

The caller assumes ownership of the object this pending event points to. In particular, he has to take care to delete that object when he is done processing it.

Returns the pending event or 0 if there is none.

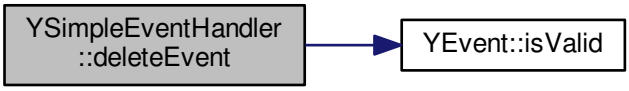
Definition at line 63 of file YSimpleEventHandler.cc.

3.119.3.4 void YSimpleEventHandler::deleteEvent (YEvent * *event*)

Delete an event. Don't call this from the outside; this is public only because of limitations of C++ .

Definition at line 157 of file YSimpleEventHandler.cc.

Here is the call graph for this function:

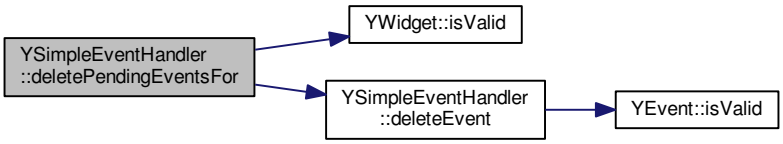


3.119.3.5 void YSimpleEventHandler::deletePendingEventsFor (YWidget * *widget*)

Delete any pending events for the specified widget. This is useful mostly if the widget is about to be destroyed.

Definition at line 131 of file YSimpleEventHandler.cc.

Here is the call graph for this function:

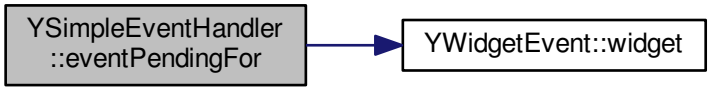


3.119.3.6 bool YSimpleEventHandler::eventPendingFor (YWidget * *widget*) const

Returns 'true' if there is any event pending for the specified widget.

Definition at line 120 of file YSimpleEventHandler.cc.

Here is the call graph for this function:



3.119.3.7 bool YSimpleEventHandler::eventsBlocked ( ) const [inline]

Returns 'true' if events are currently blocked.

Definition at line 121 of file YSimpleEventHandler.h.

3.119.3.8 YEvent* YSimpleEventHandler::pendingEvent ( ) const [inline]

Returns the last event that isn't processed yet or 0 if there is none.

This event handler keeps track of only one single (the last one) event.

Definition at line 80 of file YSimpleEventHandler.h.

3.119.3.9 void YSimpleEventHandler::sendEvent (YEvent * *event_disown*)

Widget event handlers call this when an event occured that should be the answer to a UserInput() / PollInput() (etc.) call.

The UI assumes ownership of the event object that 'event' points to, so the event MUST be created with new(). The UI is to take care to delete the event after it has been processed.

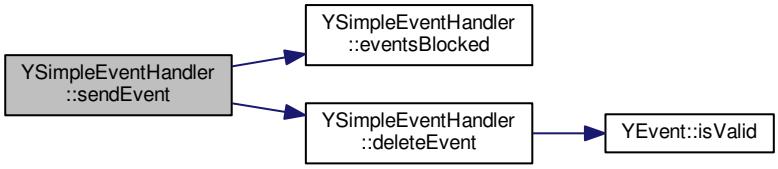
If events are blocked (see blockEvents()), the event sent with this function will be ignored (but safely deleted - no memory leak).

It is an error to pass 0 for 'event'. This simple event handler keeps track of only the latest user event. If there is more than one, older events are automatically discarded. Since Events are created on the heap with the "new" operator, discarded events need to be deleted.

Events that are not discarded are deleted later (after they are processed) by the generic UI.

Definition at line 76 of file YSimpleEventHandler.cc.

Here is the call graph for this function:



3.119.3.10 void YSimpleEventHandler::unblockEvents ( ) [inline]

Unblock events previously blocked. This is just an alias for blockEvents(false) for better readability.

Definition at line 116 of file YSimpleEventHandler.h.

Here is the call graph for this function:

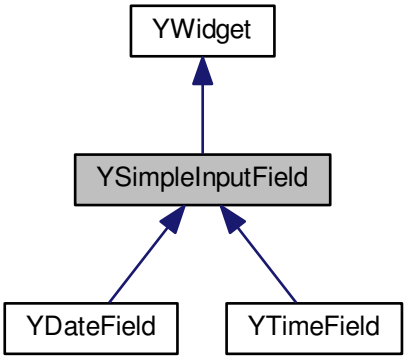
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSimpleEventHandler.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSimpleEventHandler.cc

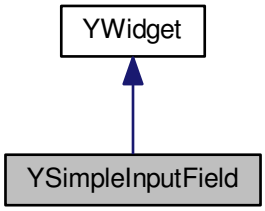
3.120 YSimpleInputField Class Reference

```
#include <YSimpleInputField.h>
```

Inheritance diagram for YSimpleInputField:



Collaboration diagram for YSimpleInputField:



Public Member Functions

- virtual ~YSimpleInputField ()
- virtual std::string value ()=0
- virtual void setValue (const std::string &text)=0
- std::string label () const
- virtual void setLabel (const std::string &label)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)
- const char * userInputProperty ()

Protected Member Functions

- YSimpleInputField (YWidget *parent, const std::string &label)

3.120.1 Detailed Description

Abstract base class for simple input fields with a label above the field and a text value.

Definition at line 37 of file YSimpleInputField.h.

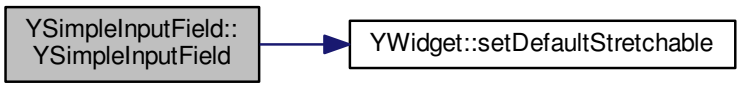
3.120.2 Constructor & Destructor Documentation

3.120.2.1 YSimpleInputField::YSimpleInputField (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

Definition at line 45 of file YSimpleInputField.cc.

Here is the call graph for this function:



3.120.2.2 YSimpleInputField::~YSimpleInputField ( ) [virtual]

Destructor.

Definition at line 56 of file YSimpleInputField.cc.

3.120.3 Member Function Documentation

3.120.3.1 YPropertyValue YSimpleInputField::getProperty (const std::string & *propertyName*) [virtual]

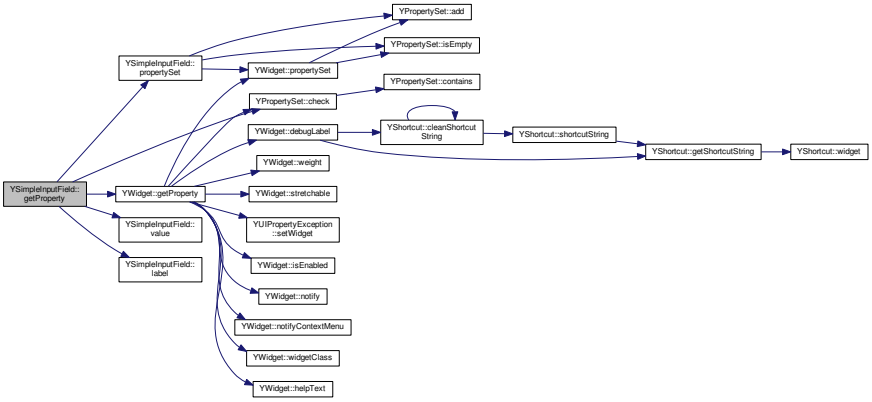
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 112 of file YSimpleInputField.cc.

Here is the call graph for this function:



3.120.3.2 std::string YSimpleInputField::label ( ) const

Get the label (the caption above the input field).

Definition at line 62 of file YSimpleInputField.cc.

3.120.3.3 const YPropertySet & YSimpleInputField::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 76 of file YSimpleInputField.cc.

Here is the call graph for this function:

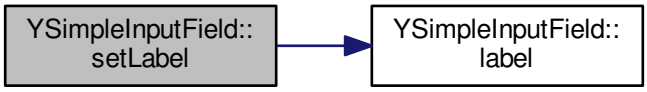
3.120.3.4 void YSimpleInputField::setLabel (const std::string & *label*) [virtual]

Set the label (the caption above the input field).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 68 of file YSimpleInputField.cc.

Here is the call graph for this function:



3.120.3.5 bool YSimpleInputField::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

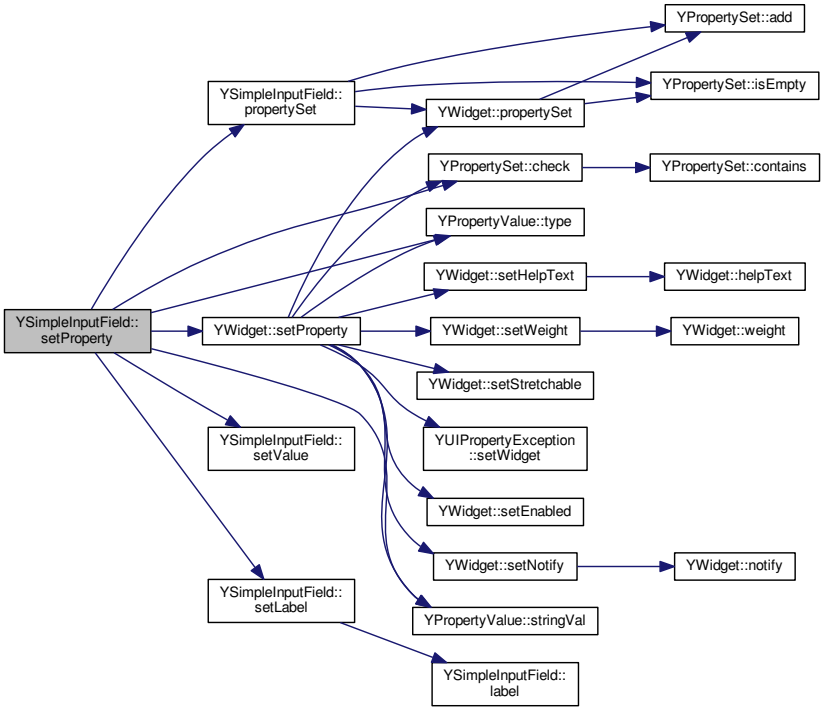
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 96 of file YSimpleInputField.cc.

Here is the call graph for this function:



3.120.3.6 virtual void YSimpleInputField::setShortcutString (const std::string & *str*) [inline],[virtual]

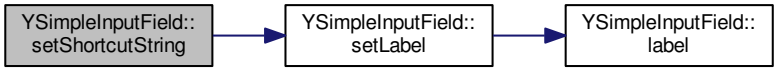
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 121 of file YSimpleInputField.h.

Here is the call graph for this function:



3.120.3.7 virtual void YSimpleInputField::setValue (const std::string & *text*) [pure virtual]

Set the current value (the text entered by the user or set from the outside) of this input field.

Derived classes are required to implement this.

3.120.3.8 virtual std::string YSimpleInputField::shortcutString ( ) const [inline], [virtual]

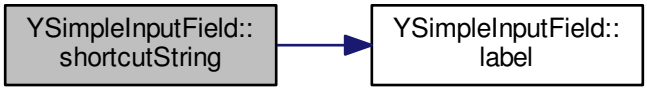
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 114 of file YSimpleInputField.h.

Here is the call graph for this function:



3.120.3.9 const char* YSimpleInputField::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 128 of file YSimpleInputField.h.

3.120.3.10 virtual std::string YSimpleInputField::value ( ) [pure virtual]

Get the current value (the text entered by the user or set from the outside) of this input field.

Derived classes are required to implement this.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSimpleInputField.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSimpleInputField.cc

3.121 YSimpleInputFieldPrivate Struct Reference

Public Member Functions

- **YSimpleInputFieldPrivate** (const std::string &label)

Public Attributes

- std::string **label**

3.121.1 Detailed Description

Definition at line 33 of file YSimpleInputField.cc.

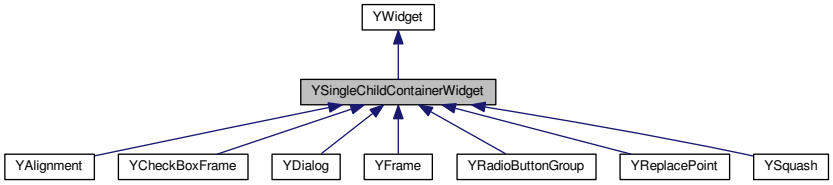
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSimpleInputField.cc

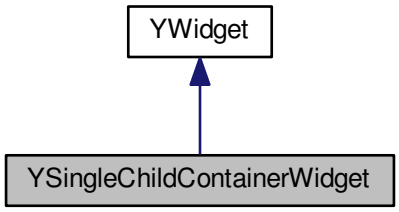
3.122 YSingleChildContainerWidget Class Reference

```
#include <YSingleChildContainerWidget.h>
```

Inheritance diagram for YSingleChildContainerWidget:



Collaboration diagram for YSingleChildContainerWidget:



Public Member Functions

- virtual ~YSingleChildContainerWidget ()
- virtual int preferredWidth ()
- virtual int preferredHeight ()
- virtual void setSize (int newWidth, int newHeight)
- virtual bool stretchable (YUIDimension dim) const

Protected Member Functions

- YSingleChildContainerWidget (YWidget *parent)

3.122.1 Detailed Description

Container widget class that manages one child.

Definition at line 34 of file YSingleChildContainerWidget.h.

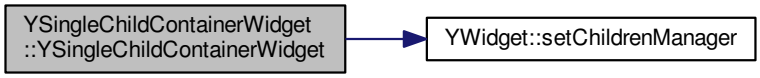
3.122.2 Constructor & Destructor Documentation

3.122.2.1 YSingleChildContainerWidget::YSingleChildContainerWidget (YWidget * *parent*) [protected]

Constructor.

Definition at line 29 of file YSingleChildContainerWidget.cc.

Here is the call graph for this function:



3.122.2.2 YSingleChildContainerWidget::~YSingleChildContainerWidget ( ) [virtual]

Destructor.

Definition at line 36 of file YSingleChildContainerWidget.cc.

3.122.3 Member Function Documentation

3.122.3.1 int YSingleChildContainerWidget::preferredHeight ( ) [virtual]

Preferred height of the widget.

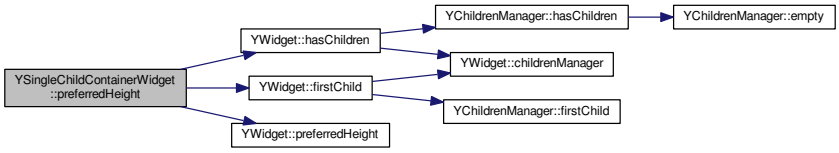
Reimplemented from YWidget.

Implements YWidget.

Reimplemented in YAlignment.

Definition at line 51 of file YSingleChildContainerWidget.cc.

Here is the call graph for this function:



3.122.3.2 int YSingleChildContainerWidget::preferredWidth ( ) [virtual]

Preferred width of the widget.

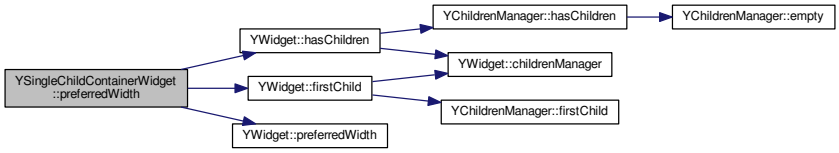
Reimplemented from YWidget.

Implements YWidget.

Reimplemented in YAlignment.

Definition at line 42 of file YSingleChildContainerWidget.cc.

Here is the call graph for this function:



3.122.3.3 void YSingleChildContainerWidget::setSize (int *newWidth*, int *newHeight*) [virtual]

Set the new size of the widget. In this case, the size of the single child is set.

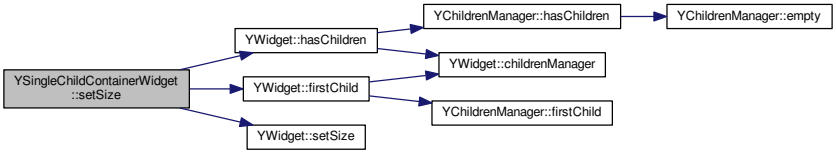
Reimplemented from YWidget.

Implements YWidget.

Reimplemented in YAlignment.

Definition at line 60 of file YSingleChildContainerWidget.cc.

Here is the call graph for this function:



3.122.3.4 bool YSingleChildContainerWidget::stretchable (YUIDimension *dim*) const [virtual]

Returns 'true' if this widget is stretchable in the specified dimension. In this case, the stretchability of the single child is returned.

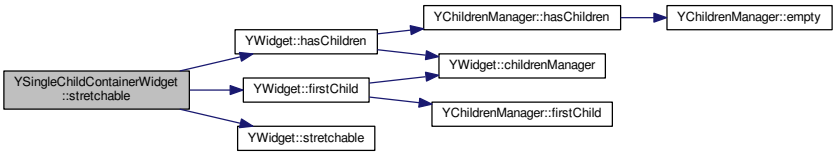
Reimplemented from YWidget.

Reimplemented from YWidget.

Reimplemented in YAlignment, and YSquash.

Definition at line 68 of file YSingleChildContainerWidget.cc.

Here is the call graph for this function:



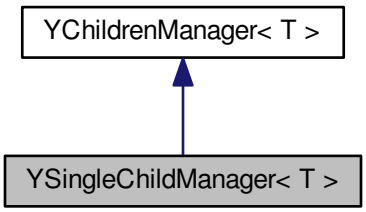
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSingleChildContainerWidget.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSingleChildContainerWidget.cc

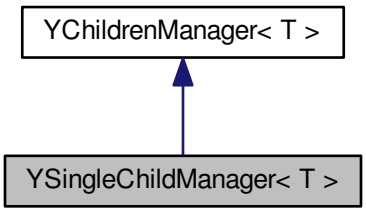
3.123 YSingleChildManager< T > Class Template Reference

```
#include <YChildrenManager.h>
```

Inheritance diagram for YSingleChildManager< T >:



Collaboration diagram for YSingleChildManager< T >:



Public Member Functions

- **YSingleChildManager** (T *containerParent)
- virtual void add (T *child)
- void replace (T *newChild)

Additional Inherited Members

3.123.1 Detailed Description

template<class T>class YSingleChildManager< T >

Children manager that can handle one single child (rejecting any more). Useful for YAlignment, YFrame etc.

Definition at line 161 of file YChildrenManager.h.

3.123.2 Member Function Documentation

3.123.2.1 template<class T > virtual void YSingleChildManager< T >::add (T * *child*) [inline], [virtual]

Add a new child.

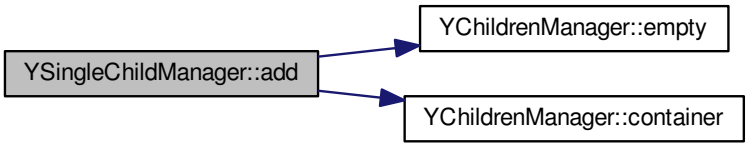
Reimplemented from YChildrenManager.

This will throw a YUITooManyChildrenException if there already is a child.

Reimplemented from YChildrenManager< T >.

Definition at line 177 of file YChildrenManager.h.

Here is the call graph for this function:



3.123.2.2 template<class T > void YSingleChildManager< T >::replace (T * *newChild*) [inline]

Replace the previous child (if any) with a new one.

Definition at line 188 of file YChildrenManager.h.

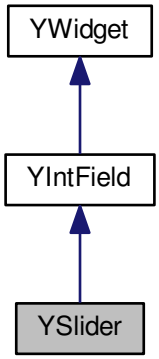
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YChildrenManager.h

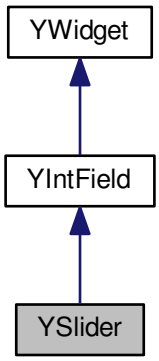
3.124 YSlider Class Reference

```
#include <YSlider.h>
```

Inheritance diagram for YSlider:



Collaboration diagram for YSlider:



Public Member Functions

- virtual ~YSlider ()
- virtual const char * widgetClass () const

Protected Member Functions

- YSlider (YWidget *parent, const std::string &label, int minValue, int maxValue)

3.124.1 Detailed Description

Slider: Input widget for an integer value between a minimum and a maximum value. Very similar to IntField in semantics, but with a graphical slider that can be dragged to the desired value. It also contains an IntField to allow entering the value directly.

Don't confuse this widget with ProgressBar: ProgressBar is output-only.

This is an optional widget, i.e. not all UIs support it.

Definition at line 44 of file YSlider.h.

3.124.2 Constructor & Destructor Documentation

3.124.2.1 YSlider::YSlider (YWidget * *parent*, const std::string & *label*, int *minValue*, int *maxValue*) [protected]

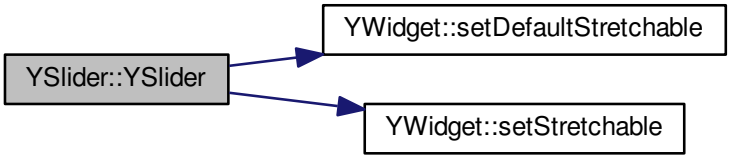
Constructor.

Create a Slider with 'label' as the caption, and the specified minimum and maximum values.

Note that YWidgetFactory::createSlider() also has an 'initialValue' parameter that is not used here (because the current value is not stored in this base class, but in the derived class).

Definition at line 43 of file YSlider.cc.

Here is the call graph for this function:



3.124.2.2 YSlider::~YSlider ( ) [virtual]

Destructor.

Definition at line 57 of file YSlider.cc.

3.124.3 Member Function Documentation

3.124.3.1 virtual const char* YSlider::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YIntField.

Definition at line 72 of file YSlider.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSlider.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSlider.cc

3.125 YSliderPrivate Struct Reference

Public Attributes

- bool **dummy**

3.125.1 Detailed Description

Definition at line 32 of file YSlider.cc.

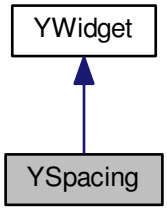
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSlider.cc

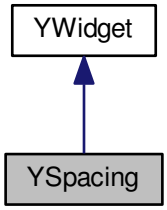
3.126 YSpacing Class Reference

```
#include <YSpacing.h>
```

Inheritance diagram for YSpacing:



Collaboration diagram for YSpacing:



Public Member Functions

- YSpacing (YWidget *parent, YUIDimension dim, bool stretchable=false, YLayoutSize_t layoutUnits=0.0)
- virtual ~YSpacing ()
- virtual const char * widgetClass () const
- YUIDimension dimension () const
- int size () const
- int size (YUIDimension dim) const
- virtual int preferredWidth ()
- virtual int preferredHeight ()

Additional Inherited Members

3.126.1 Detailed Description

HSpacing, VSpacing, HStretch, VStretch

Definition at line 37 of file YSpacing.h.

3.126.2 Constructor & Destructor Documentation

3.126.2.1 YSpacing::YSpacing (YWidget * *parent*, YUIDimension *dim*, bool *stretchable* = `false`, YLayoutSize_t *layoutUnits* = `0.0`)

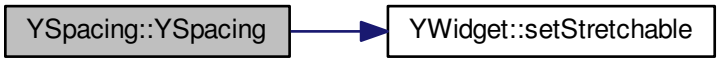
Constructor.

A Spacing/Stretch widget works only in one dimension ('dim') at the same time. But it can be stretchable and have a size at the same time, in which case the specified size acts very much like a minimal size - but not exactly, since YLayoutBox will reduce Spacings first before other widgets have to be resized below their preferred size.

'layoutUnits' is specified in abstract UI units where a main window (800x600 pixels in the Qt UI) corresponds to a 80x25 window.

Definition at line 45 of file YSpacing.cc.

Here is the call graph for this function:



3.126.2.2 YSpacing::~YSpacing ( ) [virtual]

Destructor.

Definition at line 55 of file YSpacing.cc.

3.126.3 Member Function Documentation

3.126.3.1 YUIDimension YSpacing::dimension ( ) const

Return the primary dimension of this Spacing/Stretch, i.e. the dimension in which it uses space or stretches.

Definition at line 62 of file YSpacing.cc.

3.126.3.2 int YSpacing::preferredHeight ( ) [virtual]

Preferred height of the widget.

Reimplemented from YWidget.

Implements YWidget.

Definition at line 90 of file YSpacing.cc.

3.126.3.3 int YSpacing::preferredWidth ( ) [virtual]

Preferred width of the widget.

Reimplemented from YWidget.

Implements YWidget.

Definition at line 81 of file YSpacing.cc.

3.126.3.4 int YSpacing::size ( ) const

Return the size in the primary dimension.

This is the device dependent size (pixels or character cells), not the abstract UI layout unit from the constructor.

Definition at line 68 of file YSpacing.cc.

3.126.3.5 int YSpacing::size (YUIDimension *dim*) const

Return the size in the specified dimension.

This is the device dependent size (pixels or character cells), not the abstract UI layout unit from the constructor.

Definition at line 74 of file YSpacing.cc.

3.126.3.6 const char * YSpacing::widgetClass ( ) const [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 100 of file YSpacing.cc.

Here is the call graph for this function:

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSpacing.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSpacing.cc

3.127 YSpacingPrivate Struct Reference

Public Member Functions

- **YSpacingPrivate** (YUIDimension dim, int size)

Public Attributes

- YUIDimension **dim**
- int **size**

3.127.1 Detailed Description

Definition at line 31 of file YSpacing.cc.

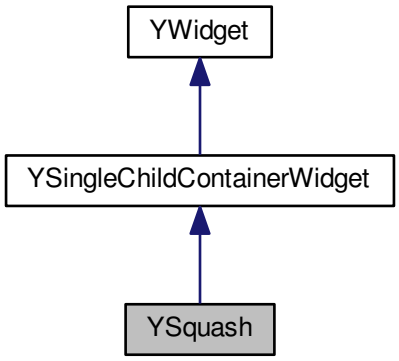
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSpacing.cc

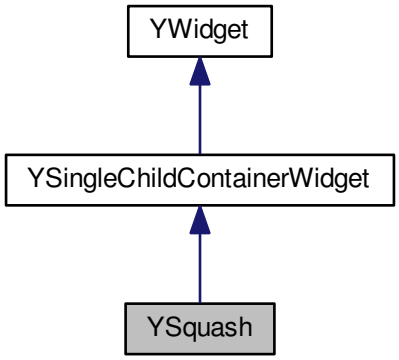
3.128 YSquash Class Reference

```
#include <YSquash.h>
```

Inheritance diagram for YSquash:



Collaboration diagram for YSquash:



Public Member Functions

- virtual ~YSquash ()
- virtual const char * widgetClass () const
- bool horSquash () const
- bool vertSquash () const

- bool stretchable (YUIDimension dim) const

Protected Member Functions

- YSquash (YWidget *parent, bool horSquash, bool vertSquash)

3.128.1 Detailed Description

HSquash, VSquash HVSquash:

Squash is a widget that "squashes" its one child during layout, i.e., it reduces it in size down to its preferred size. It may squash vertically, horizontally or in both dimensions.

Definition at line 41 of file YSquash.h.

3.128.2 Constructor & Destructor Documentation

3.128.2.1 YSquash::YSquash (YWidget * *parent*, bool *horSquash*, bool *vertSquash*) [protected]

Constructor.

Squashes horizontally if 'horSquash' is 'true', vertically if 'vertSquash' is 'true'.

Definition at line 44 of file YSquash.cc.

3.128.2.2 YSquash::~YSquash ( ) [virtual]

Destructor.

Definition at line 52 of file YSquash.cc.

3.128.3 Member Function Documentation

3.128.3.1 bool YSquash::horSquash ( ) const

Returns 'true' if this widget squashes horizontally.

Definition at line 58 of file YSquash.cc.

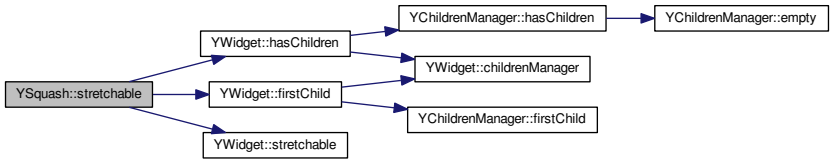
3.128.3.2 bool YSquash::stretchable (YUIDimension *dim*) const [virtual]

In a squashed dimension the widget NOT stretchable. In an unsquashed dimension the widget is stretchable if the child is stretchable.

Reimplemented from YSingleChildContainerWidget.

Definition at line 70 of file YSquash.cc.

Here is the call graph for this function:



3.128.3.3 bool YSquash::vertSquash ( ) const

Returns 'true' if this widget squashes vertically.

Definition at line 64 of file YSquash.cc.

3.128.3.4 const char * YSquash::widgetClass ( ) const [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

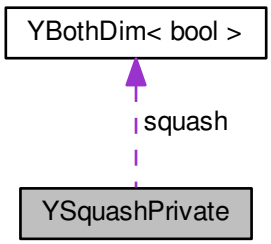
Definition at line 80 of file YSquash.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSquash.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSquash.cc

3.129 YSquashPrivate Struct Reference

Collaboration diagram for YSquashPrivate:



Public Member Functions

- YSquashPrivate (bool horSquash, bool vertSquash)

Public Attributes

- YBothDim< bool > **squash**

3.129.1 Detailed Description

Definition at line 29 of file YSquash.cc.

3.129.2 Constructor & Destructor Documentation

3.129.2.1 YSquashPrivate::YSquashPrivate (bool *horSquash,* bool *vertSquash*) [inline]

Constructor.

Definition at line 34 of file YSquash.cc.

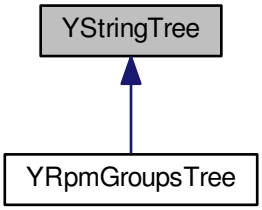
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YSquash.cc

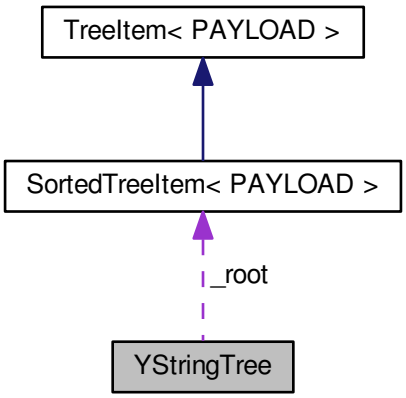
3.130 YStringTree Class Reference

```
#include <YStringTree.h>
```

Inheritance diagram for YStringTree:



Collaboration diagram for YStringTree:



Public Member Functions

- YStringTree (const char ∗textdomain)
- virtual ∼YStringTree ()
- YStringTreeItem ∗ addBranch (const std::string &content, char delimiter=0, YStringTreeItem ∗parent=0)
- std::string origPath (const YStringTreeItem ∗item, char delimiter, bool startWithDelimiter=true)
- std::string translatedPath (const YStringTreeItem ∗item, char delimiter, bool startWithDelimiter=true)
- YTransText path (const YStringTreeItem ∗item, char delimiter, bool startWithDelimiter=true)
- void logTree ()
- YStringTreeItem ∗ root () const
- const char ∗ textdomain () const
- void setTextdomain (const char ∗domain)
- std::string translate (const std::string &orig)

Protected Member Functions

- std::string completePath (const YStringTreeItem ∗item, bool translated, char delimiter, bool startWithDelimiter)
- void logBranch (YStringTreeItem ∗branch, std::string indentation)

Protected Attributes

- YStringTreeItem ∗ **_root**
- std::string **_textdomain**

3.130.1 Detailed Description

Abstract base class for filter views with hierarchical filter criteria - e.g., RPM group tags, MIME types.

Definition at line 41 of file YStringTree.h.

3.130.2 Constructor & Destructor Documentation

3.130.2.1 YStringTree::YStringTree (const char * *textdomain*)

Constructor.

'textdomain' specifies the gettext textdomain to use to translate pathname components as new branches are added.

NOTE: This will NOT change the gettext environment in any way - the tree uses dgettext() internally. The caller is responsible to bind that textdomain to a message catalog (bindtextdomain() etc.).

Definition at line 32 of file YStringTree.cc.

Here is the call graph for this function:

3.130.2.2 YStringTree::~YStringTree ( ) [virtual]

Destructor.

Definition at line 40 of file YStringTree.cc.

3.130.3 Member Function Documentation

3.130.3.1 YStringTreeItem * YStringTree::addBranch (const std::string & *content*, char *delimiter* = 0, YStringTreeItem * *parent* = 0)

Add a unique new branch with text content 'content' to the tree, beginning at 'parent' (root if parent == 0). This content can be a path specification delimited with character 'delimiter' (if not 0), i.e. this method will split 'content' up into path components and insert tree items for each level as appropriate. Leading delimiters will be ignored. If 'delimiter' is 0, 'content' is not split but used 'as is'. Items are automatically sorted alphabetically. Pathname components are automatically translated using the textdomain specified in the constructor.

Returns the tree node for this branch - either newly created or the existing one.

Example: addBranch("/usr/local/bin", '/') addBranch("/usr/lib", '/')

"usr" "lib" "local" "bin"

Definition at line 48 of file YStringTree.cc.

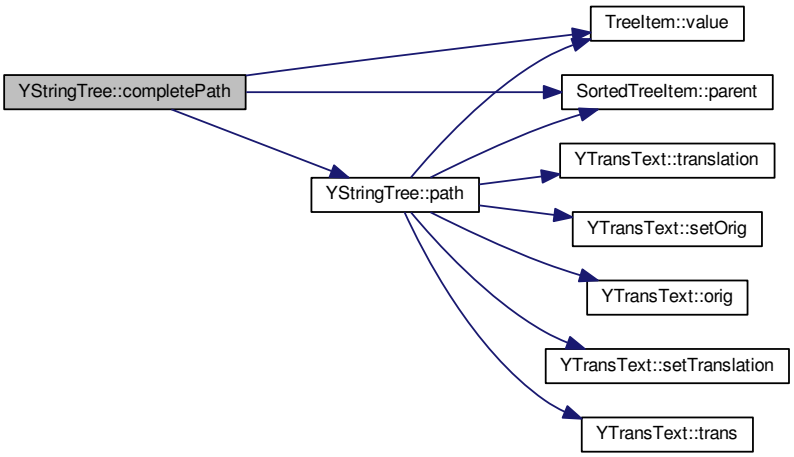
Here is the call graph for this function:

3.130.3.2 std::string YStringTree::completePath (const YStringTreeItem * *item,* bool *translated,* char *delimiter,* bool *startWithDelimiter*) [protected]

Construct a complete original or translated path for the specified tree item. 'startWithDelimiter' specifies whether or not the complete path should start with the delimiter character.

Definition at line 127 of file YStringTree.cc.

Here is the call graph for this function:

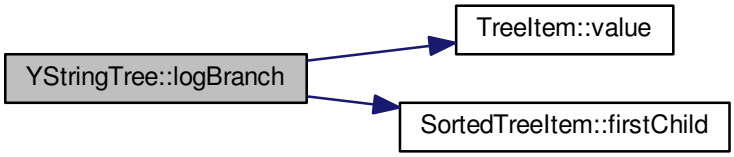


3.130.3.3 void YStringTree::logBranch (YStringTreeItem * *branch,* std::string *indentation*) [protected]

Debugging - dump one branch of the tree into the log file.

Definition at line 195 of file YStringTree.cc.

Here is the call graph for this function:



3.130.3.4 void YStringTree::logTree ( )

Debugging - dump the tree into the log file.

Definition at line 186 of file YStringTree.cc.

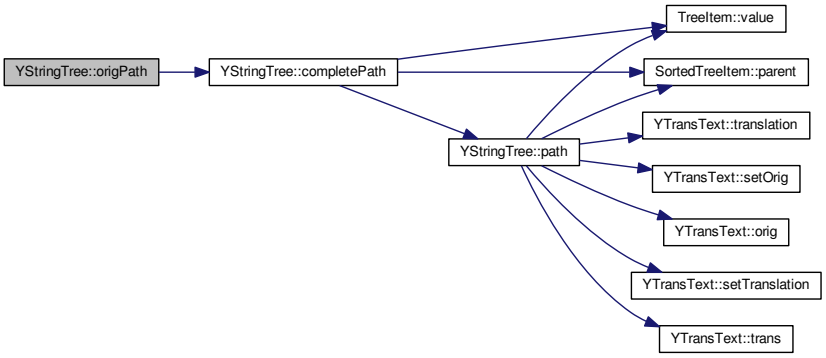
Here is the call graph for this function:

3.130.3.5 std::string YStringTree::origPath (const YStringTreeItem * *item*, char *delimiter*, bool *startWithDelimiter* = `true`) `[inline]`

Construct a complete original path for the specified tree item. 'startWithDelimiter' specifies whether or not the complete path should start with the delimiter character.

Definition at line 97 of file YStringTree.h.

Here is the call graph for this function:



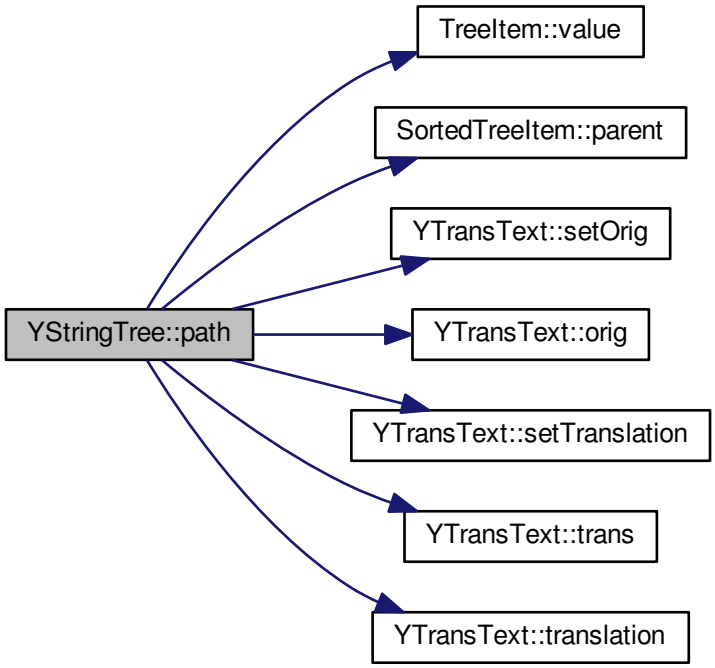
3.130.3.6 YTransText YStringTree::path (const YStringTreeItem * *item,* char *delimiter,* bool *startWithDelimiter* = `true`)

Construct a complete path (both original and translated) for the specified tree item. 'startWithDelimiter' specifies whether or not the complete path should start with the delimiter character.

Note: origPath() or translatedPath() are much cheaper if only one version (original or translated) is required.

Definition at line 158 of file YStringTree.cc.

Here is the call graph for this function:



3.130.3.7 YStringTreeItem* YStringTree::root ( ) const [inline]

Returns the root of the filter view tree. Note: In most cases, the root item itself will not contain any useful information. Consider it the handle for the entire tree, not an actual data element.

Definition at line 139 of file YStringTree.h.

3.130.3.8 void YStringTree::setTextdomain (const char * *domain*) [inline]

Set the textdomain used internally for translation of pathname components.

NOTE: This will NOT change the gettext environment in any way - the tree uses dgettext() internally. The caller is responsible to bind that textdomain to a message catalog (bindtextdomain() etc.).

Definition at line 157 of file YStringTree.h.

3.130.3.9 const char* YStringTree::textdomain ( ) const [inline]

Returns the textdomain used internally for translation of pathname components.

Definition at line 146 of file YStringTree.h.

3.130.3.10 std::string YStringTree::translate (const std::string & *orig*)

Translate message 'orig' using the internal textdomain. Returns the translated text or the original if there is no translation.

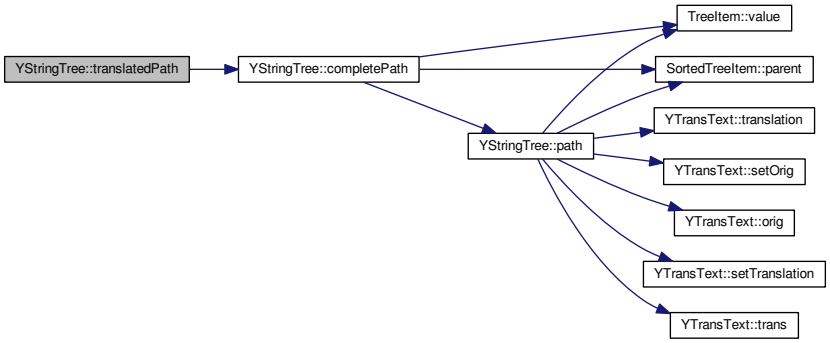
Definition at line 118 of file YStringTree.cc.

3.130.3.11 std::string YStringTree::translatedPath (const YStringTreeItem * *item*, char *delimiter*, bool *startWithDelimiter* = `true`) `[inline]`

Construct a complete original path for the specified tree item. 'startWithDelimiter' specifies whether or not the complete path should start with the delimiter character.

Definition at line 108 of file YStringTree.h.

Here is the call graph for this function:



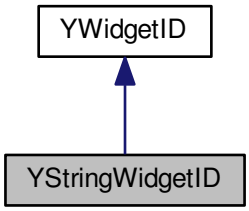
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YStringTree.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YStringTree.cc

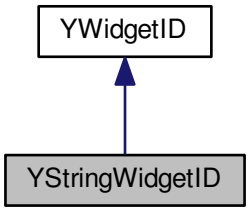
3.131 YStringWidgetID Class Reference

```
#include <YWidgetID.h>
```

Inheritance diagram for YStringWidgetID:



Collaboration diagram for YStringWidgetID:



Public Member Functions

- YStringWidgetID (const std::string &value)
- virtual ~YStringWidgetID ()
- virtual bool isEqual (YWidgetID *otherID) const
- virtual std::string toString () const
- std::string value () const
- const std::string & valueConstRef () const

Additional Inherited Members

3.131.1 Detailed Description

Simple widget ID class based on strings.

Definition at line 72 of file YWidgetID.h.

3.131.2 Constructor & Destructor Documentation

3.131.2.1 YStringWidgetID::YStringWidgetID (const std::string & *value*)

Constructor.

Definition at line 31 of file YWidgetID.cc.

3.131.2.2 YStringWidgetID::~YStringWidgetID ( ) [virtual]

Destructor.

Definition at line 38 of file YWidgetID.cc.

3.131.3 Member Function Documentation

3.131.3.1 bool YStringWidgetID::isEqual (YWidgetID * *otherID*) const [virtual]

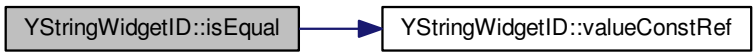
Check if this ID is equal to another.

Reimplemented from YWidgetID.

Implements YWidgetID.

Definition at line 45 of file YWidgetID.cc.

Here is the call graph for this function:



3.131.3.2 std::string YStringWidgetID::toString ( ) const [virtual]

Convert the ID value to string. Used for logging and debugging.

Reimplemented from YWidgetID.

Implements YWidgetID.

Definition at line 58 of file YWidgetID.cc.

3.131.3.3 std::string YStringWidgetID::value ( ) const

Return the ID value.

Definition at line 65 of file YWidgetID.cc.

3.131.3.4 const std::string & YStringWidgetID::valueConstRef ( ) const

Return the ID value as a const ref.

Definition at line 72 of file YWidgetID.cc.

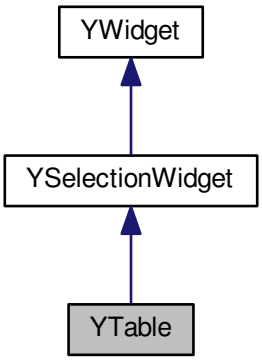
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWidgetID.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWidgetID.cc

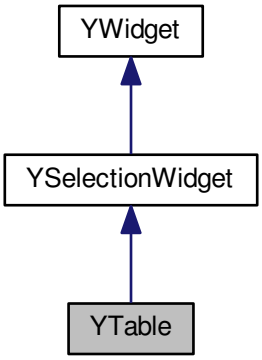
3.132 YTable Class Reference

```
#include <YTable.h>
```

Inheritance diagram for YTable:



Collaboration diagram for YTable:



Public Member Functions

- virtual ~YTable ()
- virtual const char * widgetClass () const
- int columns () const
- bool hasColumn (int column) const
- std::string header (int column) const
- YAlignmentType alignment (int column) const
- bool immediateMode () const
- void setImmediateMode (bool immediateMode=true)
- bool keepSorting () const
- virtual void setKeepSorting (bool keepSorting)
- bool hasMultiSelection () const
- virtual void cellChanged (const YTableCell *cell)=0
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- const char * userInputProperty ()

Protected Member Functions

- YTable (YWidget *parent, YTableHeader *header, bool multiSelection)
- void setTableHeader (YTableHeader *newHeader)

3.132.1 Detailed Description

Table: Selection list with multiple columns. The user can select exactly one row (with all its columns) from that list. Each cell (each column within each row) has a label text and an optional icon (*).

This widget is similar to SelectionBox, but it has several columns for each item (each row). If just one column is desired, consider using SelectionBox instead.

Note: This is not something like a spread sheet, and it doesn't pretend or want to be. Actions are performed on rows, not on individual cells (columns within one row).

(∗) Not all UIs (in particular not text-based UIs) support displaying icons, so an icon should never be an exclusive means to display any kind of information.

Definition at line 55 of file YTable.h.

3.132.2 Constructor & Destructor Documentation

3.132.2.1 YTable::YTable (YWidget ∗ *parent,* YTableHeader ∗ *header,* bool *multiSelection*) [protected]

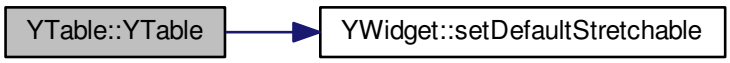
Constructor.

'header' describes the table's headers: Number of columns, column headings, and column alignment. The widget assumes ownership of this object and will delete it when appropriate. The header cannot be changed after creating the widget.

'multiSelection' indicates whether or not the user can select multiple items at the same time (e.g., with shift-click or ctrl-click). This can only be set in the constructor.

Definition at line 50 of file YTable.cc.

Here is the call graph for this function:



3.132.2.2 YTable::∼YTable ( ) [virtual]

Destructor.

Definition at line 64 of file YTable.cc.

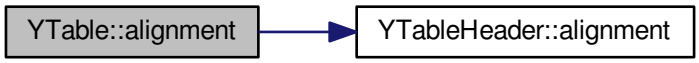
3.132.3 Member Function Documentation

3.132.3.1 YAlignmentType YTable::alignment (int *column*) const

Return the alignment for the specified column.

Definition at line 106 of file YTable.cc.

Here is the call graph for this function:



3.132.3.2 virtual void YTable::cellChanged (const YTableCell * *cell*) [pure virtual]

Notification that a cell (its text and/or its icon) was changed from the outside. Applications are required to call this whenever a table cell is changed after adding the corresponding table item (the row) to the table widget.

Derived classes are required to implement this and update the display accordingly.

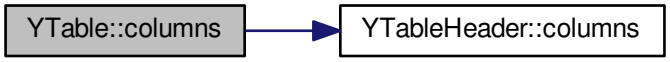
Note that the position of this cell can be retrieved with cell->column() and cell->itemIndex().

3.132.3.3 int YTable::columns ( ) const

Return the number of columns of this table.

Definition at line 85 of file YTable.cc.

Here is the call graph for this function:



3.132.3.4 YPropertyValue YTable::getProperty (const std::string & *propertyName*) [virtual]

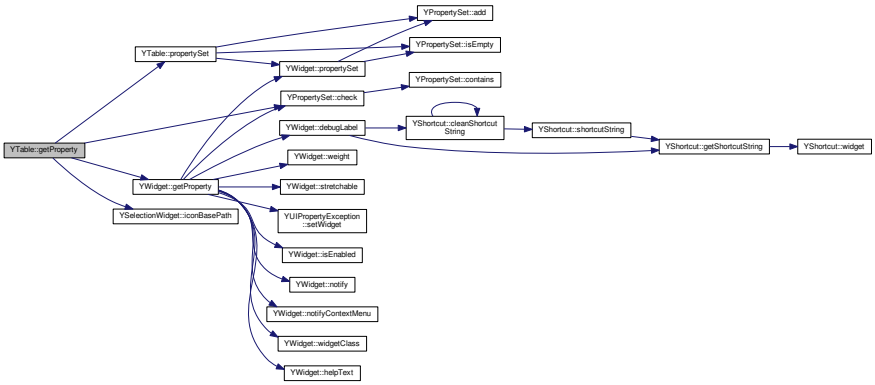
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 207 of file YTable.cc.

Here is the call graph for this function:

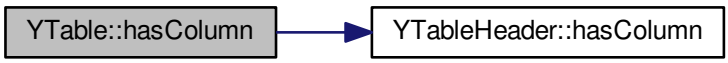


3.132.3.5 bool YTable::hasColumn (int *column*) const

Return 'true' if this table has a column no. 'column' (counting from 0 on).

Definition at line 92 of file YTable.cc.

Here is the call graph for this function:

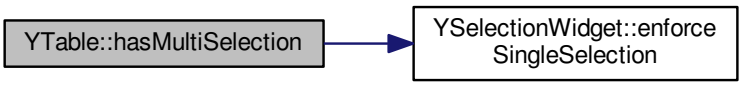


3.132.3.6 bool YTable::hasMultiSelection ( ) const

Return 'true' if the user can select multiple items at the same time (e.g., with shift-click or ctrl-click).

Definition at line 144 of file YTable.cc.

Here is the call graph for this function:

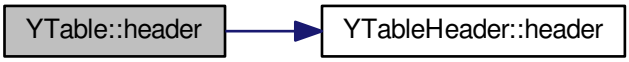


3.132.3.7 std::string YTable::header (int *column*) const

Return the header text for the specified column.

Definition at line 99 of file YTable.cc.

Here is the call graph for this function:



3.132.3.8 bool YTable::immediateMode ( ) const

Deliver even more events than with notify() set.

With "notify" alone, a table widget sends an ActivatedEvent when the user double-clicks an item or presses the "space" key on it. It does not send an event when the user just sends another item.

With "immediate", it also sends a SelectionChangedEvent when the user selects another item. "immediate" implicitly includes "notify".

Definition at line 113 of file YTable.cc.

3.132.3.9 bool YTable::keepSorting ( ) const

Return 'true' if the sort order is to be kept in item insertion order, i.e. if sorting the table by clicking on a column header should be disabled.

Definition at line 130 of file YTable.cc.

3.132.3.10 const YPropertySet & YTable::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 151 of file YTable.cc.

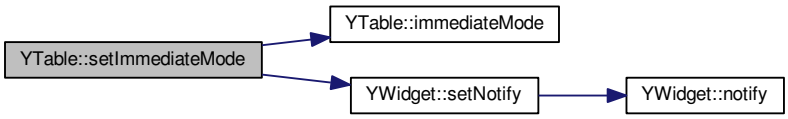
Here is the call graph for this function:

3.132.3.11 void YTable::setImmediateMode (bool *immediateMode* = `true`)

Set immediateMode() on or off.

Definition at line 120 of file YTable.cc.

Here is the call graph for this function:



3.132.3.12 void YTable::setKeepSorting (bool *keepSorting*) `[virtual]`

Switch between sorting by item insertion order (keepSorting: true) or allowing the user to sort by an arbitrary column (by clicking on the column header).

Derived classes can overwrite this function, but they should call this base class function in the new implementation.

Definition at line 137 of file YTable.cc.

Here is the call graph for this function:

3.132.3.13 bool YTable::setProperty (const std::string & *propertyName,* const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

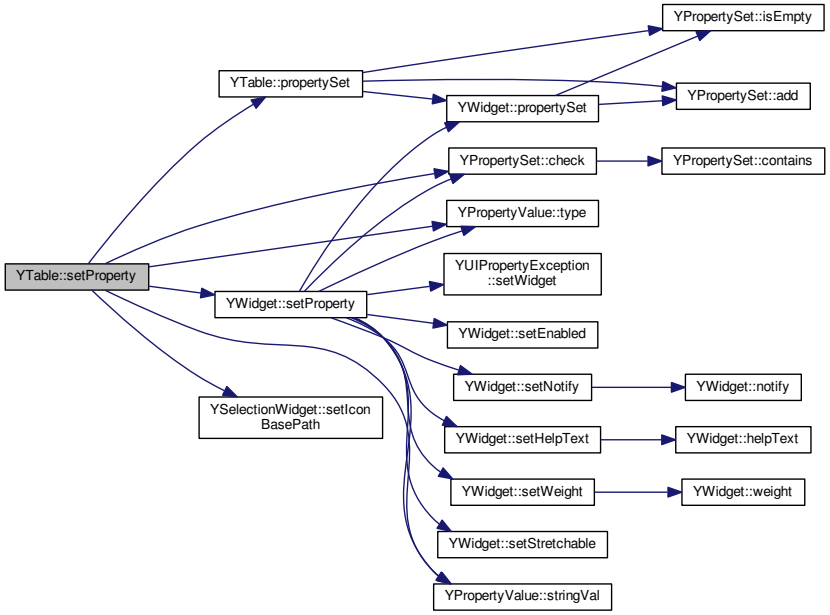
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 186 of file YTable.cc.

Here is the call graph for this function:



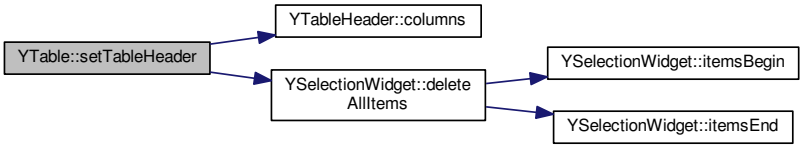
3.132.3.14 void YTable::setTableHeader (YTableHeader * *newHeader*) [protected]

Exchange the previous table header with a new one. This will delete the old YTableHeader object.

If the new header has a different number of columns than the old one, all items will implicitly be deleted.

Definition at line 72 of file YTable.cc.

Here is the call graph for this function:



3.132.3.15 const char* YTable::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 194 of file YTable.h.

3.132.3.16 virtual const char* YTable::widgetClass ( ) const [inline], [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YSelectionWidget.

Definition at line 83 of file YTable.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTable.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTable.cc

3.133 YTableCell Class Reference

```
#include <YTableItem.h>
```

Public Member Functions

- YTableCell (const std::string &label, const std::string &iconName="")
- YTableCell (YTableItem *parent, int column, const std::string &label, const std::string &iconName="")
- virtual ~YTableCell ()
- std::string label () const
- void setLabel (const std::string &newLabel)
- std::string iconName () const
- bool hasIconName () const
- void setIconName (const std::string &newIconName)
- YTableItem * parent () const
- int column () const
- int itemIndex () const
- void reparent (YTableItem *parent, int column)

3.133.1 Detailed Description

One cell (one column in one row) of a YTableItem. Each cell has a label (a user visible text) and optionally an icon (∗).

Note that cells don't have individual IDs; they have just an index. The first cell in an item is cell(0). In an ideal world, each YTableItem would have exactly as many cells as there are columns in the YTable, but these classes make no such assumptions. A YTableItem might have any number of cells, including none.

The YTable widget is free to ignore any excess cells if there are more than the YTable widget has columns. If there are less cells than the table has columns, the nonexistent cells will be treated as empty.

(∗) Not all UIs can handle icons. UIs that can't handle them will simply ignore any icons specified for YTableCells. Thus, applications should either check the UI capabilities if it can handle icons or use icons only as an additional visual cue that still has a text counterpart (so the user can still make sense of the table content when no icons are visible).

Definition at line 212 of file YTableItem.h.

3.133.2 Constructor & Destructor Documentation

3.133.2.1 YTableCell::YTableCell (const std::string & *label,* const std::string & *iconName* = " ") [inline]

Constructor with label and optional icon name for cells that don't have a parent item yet (that will be added to a parent later with setParent()).

Definition at line 220 of file YTableItem.h.

3.133.2.2 YTableCell::YTableCell (YTableItem ∗ *parent,* int *column,* const std::string & *label,* const std::string & *iconName* = " ") [inline]

Constructor with parent, column no., label and optional icon name for cells that are created with a parent.

Definition at line 231 of file YTableItem.h.

3.133.2.3 virtual YTableCell::∼YTableCell ( ) [inline],[virtual]

Destructor. Not strictly needed inside this class, but useful for derived classes. Since this is the only virtual method of this class, the cost of this is a vtable for this class and a pointer to the vtable in each instance.

Definition at line 247 of file YTableItem.h.

3.133.3 Member Function Documentation

3.133.3.1 int YTableCell::column ( ) const [inline]

Return this cell's column no. (counting from 0on) or -1 if it doesn't have a parent yet.

Definition at line 292 of file YTableItem.h.

3.133.3.2 bool YTableCell::hasIconName ( ) const [inline]

Return 'true' if this cell has an icon name.

Definition at line 272 of file YTableItem.h.

3.133.3.3 std::string YTableCell::iconName ( ) const [inline]

Return this cell's icon name.

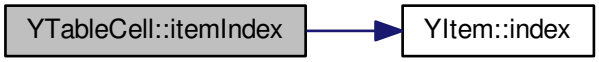
Definition at line 267 of file YTableItem.h.

3.133.3.4 int YTableCell::itemIndex ( ) const [inline]

Convenience function: Return this cell's parent item's index within its table widget or -1 if there is no parent item or no parent table.

Definition at line 298 of file YTableItem.h.

Here is the call graph for this function:



3.133.3.5 std::string YTableCell::label ( ) const [inline]

Return this cells's label. This is what the user sees in a dialog, so this will usually be a translated text.

Definition at line 253 of file YTableItem.h.

3.133.3.6 YTableItem* YTableCell::parent ( ) const [inline]

Return this cell's parent item or 0 if it doesn't have one yet.

Definition at line 286 of file YTableItem.h.

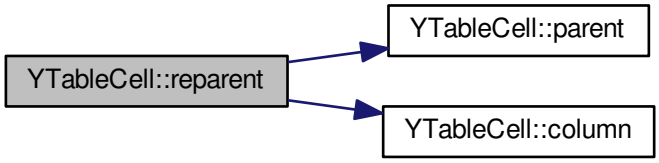
3.133.3.7 void YTableCell::reparent (YTableItem * *parent*, int *column*)

Set this cell's parent item and column no. if it doesn't have a parent yet.

This method will throw an exception if the cell already has a parent.

Definition at line 171 of file YTableItem.cc.

Here is the call graph for this function:



3.133.3.8 void YTableCell::setIconName (const std::string & *newIconName*) [inline]

Set this cell's icon name.

If this is called after the corresponding table item (table row) is added to the table widget, call YTable::cellChanged() to notify the table widget about the fact. Only then will the display be updated.

Definition at line 281 of file YTableItem.h.

3.133.3.9 void YTableCell::setLabel (const std::string & *newLabel*) [inline]

Set this cell's label.

If this is called after the corresponding table item (table row) is added to the table widget, call YTable::cellChanged() to notify the table widget about the fact. Only then will the display be updated.

Definition at line 262 of file YTableItem.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTableItem.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTableItem.cc

3.134 YTableHeader Class Reference

```
#include <YTableHeader.h>
```

Public Member Functions

- YTableHeader ()
- virtual ~YTableHeader ()
- void addColumn (const std::string &header, YAlignmentType alignment=YAlignBegin)
- int columns () const
- bool hasColumn (int column) const
- std::string header (int column) const
- YAlignmentType alignment (int column) const

3.134.1 Detailed Description

Helper class for YTable for table column properties:

- number of columns
- header for each column
- alignment for each column

Definition at line 43 of file YTableHeader.h.

3.134.2 Constructor & Destructor Documentation

3.134.2.1 YTableHeader::YTableHeader ( )

Constructor.

Definition at line 48 of file YTableHeader.cc.

3.134.2.2 YTableHeader::~YTableHeader ( ) [virtual]

Destructor.

Definition at line 55 of file YTableHeader.cc.

3.134.3 Member Function Documentation

3.134.3.1 void YTableHeader::addColumn (const std::string & *header*, YAlignmentType *alignment* = YAlignBegin)

Add a column with the specified colum header text and alignment.

Definition at line 62 of file YTableHeader.cc.

3.134.3.2 YAlignmentType YTableHeader::alignment (int *column*) const

Return the alignment for the specified column.

Definition at line 94 of file YTableHeader.cc.

3.134.3.3 int YTableHeader::columns ( ) const

Return the number of columns.

Definition at line 70 of file YTableHeader.cc.

3.134.3.4 bool YTableHeader::hasColumn (int *column*) const

Return 'true' if this table header has a column no. 'column' (counting from 0 on).

Definition at line 77 of file YTableHeader.cc.

3.134.3.5 std::string YTableHeader::header (int *column*) const

Return the header text for the specified column.

Definition at line 84 of file YTableHeader.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTableHeader.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTableHeader.cc

3.135 YTableHeaderPrivate Struct Reference

Public Attributes

- std::vector< std::string > **headers**
- std::vector< YAlignmentType > **alignments**

3.135.1 Detailed Description

Definition at line 36 of file YTableHeader.cc.

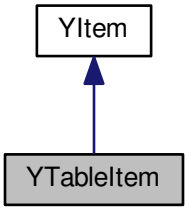
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTableHeader.cc

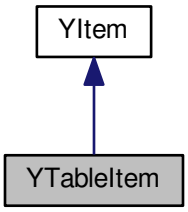
3.136 YTableItem Class Reference

```
#include <YTableItem.h>
```

Inheritance diagram for YTableItem:



Collaboration diagram for YTableItem:



Public Member Functions

- YTableItem ()
- YTableItem (const std::string &label_0, const std::string &label_1=std::string(), const std::string &label_2=std::string(), const std::string &label_3=std::string(), const std::string &label_4=std::string(), const std::string &label_5=std::string(), const std::string &label_6=std::string(), const std::string &label_7=std::string(), const std::string &label_8=std::string(), const std::string &label_9=std::string())
- virtual ~YTableItem ()
- void addCell (YTableCell *cell_disown)
- void addCell (const std::string &label, const std::string &iconName=std::string())
- void deleteCells ()
- YTableCellIterator cellsBegin ()
- YTableCellConstIterator **cellsBegin** () const
- YTableCellIterator cellsEnd ()
- YTableCellConstIterator **cellsEnd** () const
- const YTableCell * cell (int index) const
- YTableCell * **cell** (int index)
- int cellCount () const
- bool hasCell (int index) const
- std::string label (int index) const
- std::string iconName (int index) const
- bool hasIconName (int index) const
- std::string label () const

3.136.1 Detailed Description

Item class for YTable items. Each YTableItem corresponds to one row in a YTable.

A YTableItem might have any number of cells (columns within this row), including none. The YTable widget is free to ignore any excess cells if there are more than the YTable widget has columns. The YTable widget is to treat nonexistent cells like empty ones.

Note that while YTable items and their cells can be manipulated through pointers, their visual representation on screen might be updated only upon calling certain methods of the YTable widget. See the YTable reference for details.

Definition at line 52 of file YTableItem.h.

3.136.2 Constructor & Destructor Documentation

3.136.2.1 YTableItem::YTableItem ( )

Default constructor. Use addCell() to give it any content.

Definition at line 29 of file YTableItem.cc.

3.136.2.2 YTableItem::YTableItem (const std::string & *label_0*, const std::string & *label_1* = `std::string()`, const std::string & *label_2* = `std::string()`, const std::string & *label_3* = `std::string()`, const std::string & *label_4* = `std::string()`, const std::string & *label_5* = `std::string()`, const std::string & *label_6* = `std::string()`, const std::string & *label_7* = `std::string()`, const std::string & *label_8* = `std::string()`, const std::string & *label_9* = `std::string()`)

Convenience constructor for table items without any icons.

This will create up to 10 (0..9) cells. Empty cells for empty labels at the end of the labels are not created, but empty cells in between are.

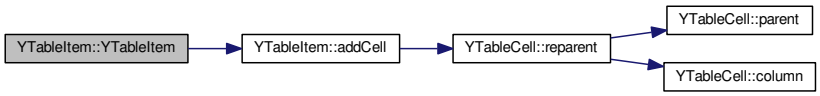
```
new YTableItem( "one", "two", "", "", "five" );
```

will create an item with 5 cells:

```
cell[0] ==> "one"
cell[1] ==> "two"
cell[2] ==> ""
cell[3] ==> ""
cell[4] ==> "five"
```

Definition at line 36 of file YTableItem.cc.

Here is the call graph for this function:



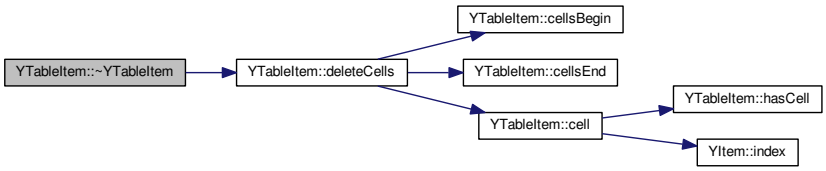
3.136.2.3 YTableItem::~YTableItem ( ) [virtual]

Destructor.

This will delete all cells.

Definition at line 82 of file YTableItem.cc.

Here is the call graph for this function:



3.136.3 Member Function Documentation

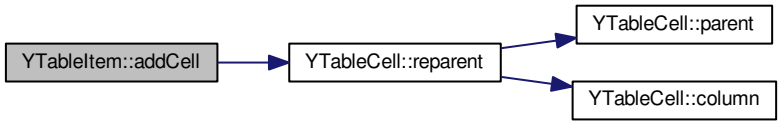
3.136.3.1 void YTableItem::addCell (YTableCell * *cell_disown*)

Add a cell. This item will assume ownership over the cell and delete it when appropriate (when the table is destroyed or when table items are replaced), at which time the pointer will become invalid.

Cells can still be changed after they (and the item they belong to) are added, but in that case, YTable::cellChanged() needs to be called to update the table display accordingly.

Definition at line 105 of file YTableItem.cc.

Here is the call graph for this function:

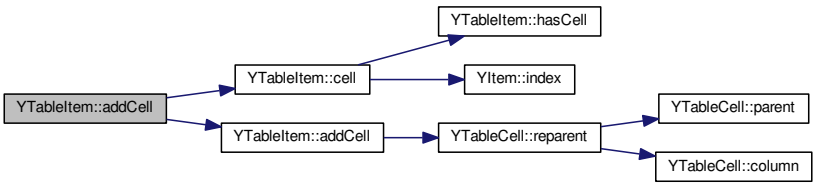


3.136.3.2 void YTableItem::addCell (const std::string & *label,* const std::string & *iconName* = `std::string()`)

Create a new cell and add it (even if both 'label' and 'iconName' are empty).

Definition at line 115 of file YTableItem.cc.

Here is the call graph for this function:

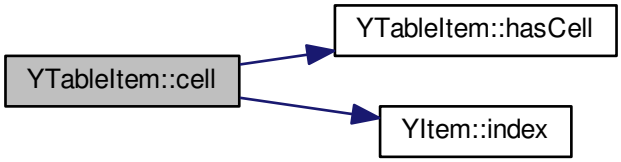


3.136.3.3 const YTableCell * YTableItem::cell (int *index*) const

Return the cell at the specified index (counting from 0 on) or 0 if there is none.

Definition at line 132 of file YTableItem.cc.

Here is the call graph for this function:



3.136.3.4 int YTableItem::cellCount ( ) const [inline]

Return the number of cells this item has.

Definition at line 139 of file YTableItem.h.

3.136.3.5 YTableCellIterator YTableItem::cellsBegin ( ) [inline]

Return an iterator that points to the first cell of this item.

Definition at line 120 of file YTableItem.h.

3.136.3.6 YTableCellIterator YTableItem::cellsEnd ( ) [inline]

Return an iterator that points after the last cell of this item.

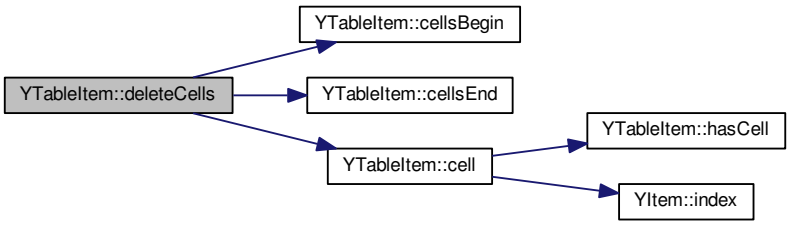
Definition at line 126 of file YTableItem.h.

3.136.3.7 void YTableItem::deleteCells ( )

Delete all cells.

Definition at line 89 of file YTableItem.cc.

Here is the call graph for this function:



3.136.3.8 bool YTableItem::hasCell (int *index*) const

Return 'true' if this item has a cell with the specified index (counting from 0 on), 'false' otherwise.

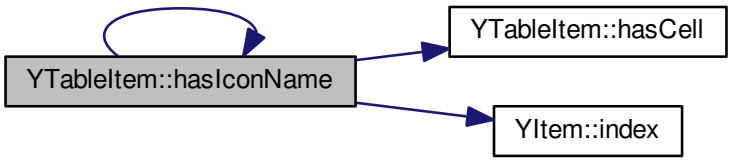
Definition at line 125 of file YTableItem.cc.

3.136.3.9 bool YTableItem::hasIconName (int *index*) const

Return 'true' if there is a cell with the specified index that has an icon name.

Definition at line 162 of file YTableItem.cc.

Here is the call graph for this function:

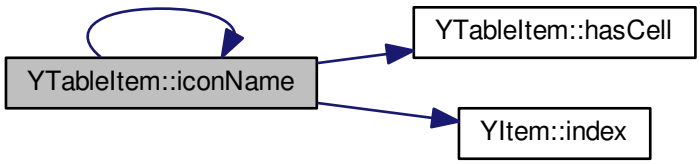


3.136.3.10 std::string YTableItem::iconName (int *index*) const

Return the icon name of cell no. 'index' (counting from 0 on) or an empty string if there is no cell with that index.

Definition at line 155 of file YTableItem.cc.

Here is the call graph for this function:

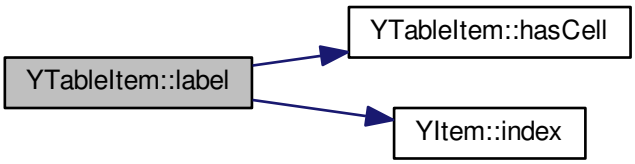


3.136.3.11 std::string YTableItem::label (int *index*) const

Return the label of cell no. 'index' (counting from 0 on) or an empty string if there is no cell with that index.

Definition at line 148 of file YTableItem.cc.

Here is the call graph for this function:

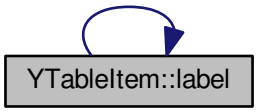


3.136.3.12 std::string YTableItem::label ( ) const [inline]

Just for debugging.

Definition at line 168 of file YTableItem.h.

Here is the call graph for this function:

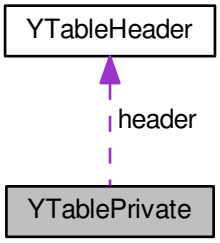


The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTableItem.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTableItem.cc

3.137 YTablePrivate Struct Reference

Collaboration diagram for YTablePrivate:



Public Member Functions

- **YTablePrivate** (YTableHeader *header)

Public Attributes

- YTableHeader * **header**
- bool **keepSorting**
- bool **immediateMode**

3.137.1 Detailed Description

Definition at line 33 of file YTable.cc.

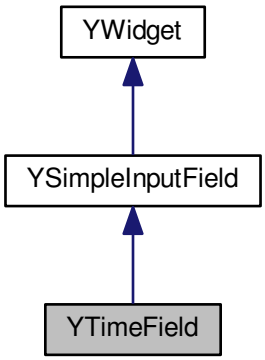
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTable.cc

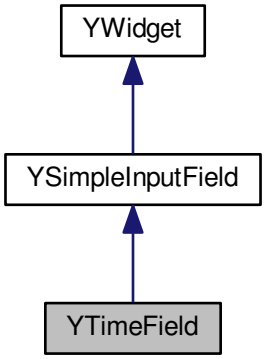
3.138 YTimeField Class Reference

```
#include <YTimeField.h>
```

Inheritance diagram for YTimeField:



Collaboration diagram for YTimeField:



Public Member Functions

- virtual ~YTimeField ()
- virtual const char * widgetClass () const

Protected Member Functions

- YTimeField (YWidget *parent, const std::string &label)

3.138.1 Detailed Description

Input field for entering a time in "hh:mm:ss" format.

Derived classes are required to implement: value() setValue() See YSimpleInputField.h for details.

Definition at line 41 of file YTimeField.h.

3.138.2 Constructor & Destructor Documentation

3.138.2.1 YTimeField::YTimeField (YWidget * *parent*, const std::string & *label*) [protected]

Constructor.

Definition at line 43 of file YTimeField.cc.

3.138.2.2 YTimeField::~YTimeField ( ) [virtual]

Destructor.

Definition at line 51 of file YTimeField.cc.

3.138.3 Member Function Documentation

3.138.3.1 virtual const char* YTimeField::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 59 of file YTimeField.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTimeField.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTimeField.cc

3.139 YTimeFieldPrivate Struct Reference

Public Attributes

- bool **dummy**

3.139.1 Detailed Description

Definition at line 32 of file YTimeField.cc.

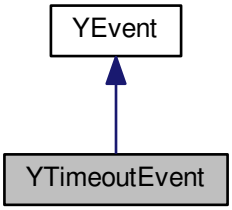
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTimeField.cc

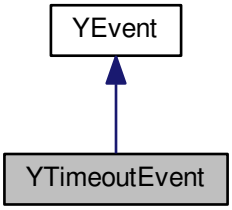
3.140 YTimeoutEvent Class Reference

```
#include <YEvent.h>
```

Inheritance diagram for YTimeoutEvent:



Collaboration diagram for YTimeoutEvent:



Protected Member Functions

- virtual ~YTimeoutEvent ()

Additional Inherited Members

3.140.1 Detailed Description

Event to be returned upon timeout (i.e. no event available in the specified timeout)

Definition at line 346 of file YEvent.h.

3.140.2 Constructor & Destructor Documentation

3.140.2.1 virtual YTimeoutEvent::~YTimeoutEvent ( ) [inline], [protected], [virtual]

Protected destructor - events can only be deleted via YDialog::deleteEvent(). The associated dialog will take care of this event and delete it when appropriate.

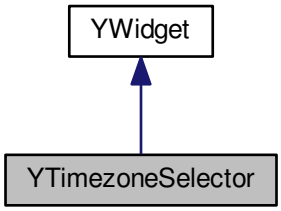
Definition at line 358 of file YEvent.h.

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

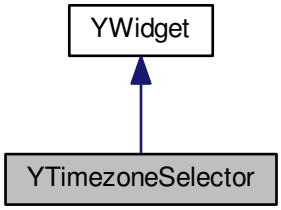
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEvent.h

3.141 YTimezoneSelector Class Reference

Inheritance diagram for YTimezoneSelector:



Collaboration diagram for YTimezoneSelector:



Public Member Functions

- virtual ~YTimezoneSelector ()
- virtual const char * widgetClass () const
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- virtual std::string currentZone () const =0
- virtual void setCurrentZone (const std::string &zone, bool zoom)=0

Protected Member Functions

- YTimezoneSelector (YWidget *parent, const std::string &pixmap, const std::map< std::string, std::string > &timezones)

3.141.1 Detailed Description

Definition at line 35 of file YTimezoneSelector.h.

3.141.2 Constructor & Destructor Documentation

3.141.2.1 YTimezoneSelector::YTimezoneSelector (YWidget * *parent,* const std::string & *pixmap,* const std::map< std::string, std::string > & *timezones*) [protected]

Constructor. This widget isn't doing much on it's own, but the UI may have some fancy use.

- pixmap should be a png or jpg of a world map with centered 0°0° and the timezones are a map between zone.tab entry and user visible string.

The widget is only displaying timezones/cities in that map

Definition at line 41 of file YTimezoneSelector.cc.

3.141.2.2 YTimezoneSelector::~YTimezoneSelector ( ) [virtual]

Destructor.

Definition at line 49 of file YTimezoneSelector.cc.

3.141.3 Member Function Documentation

3.141.3.1 virtual std::string YTimezoneSelector::currentZone ( ) const [pure virtual]

subclasses have to implement this to return value

3.141.3.2 YPropertyValue YTimezoneSelector::getProperty (const std::string & *propertyName*) [virtual]

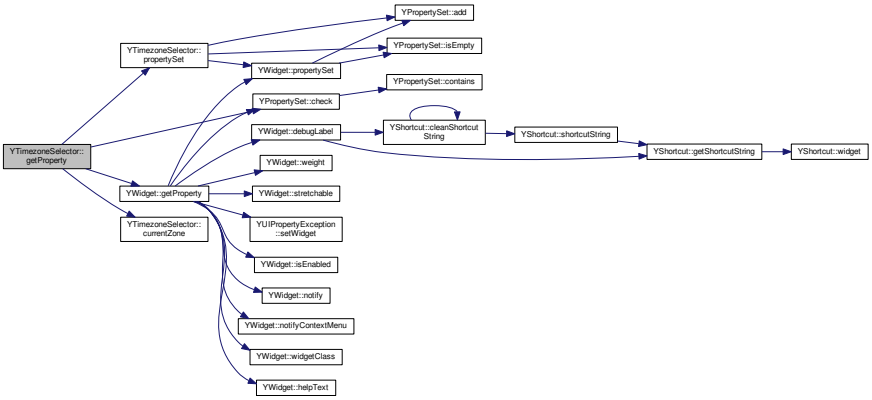
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 91 of file YTimezoneSelector.cc.

Here is the call graph for this function:



3.141.3.3 const YPropertySet & YTimezoneSelector::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 56 of file YTimezoneSelector.cc.

Here is the call graph for this function:

3.141.3.4 virtual void YTimezoneSelector::setCurrentZone (const std::string & *zone*, bool *zoom*) [pure virtual]

subclasses have to implement this to set value

3.141.3.5 bool YTimezoneSelector::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

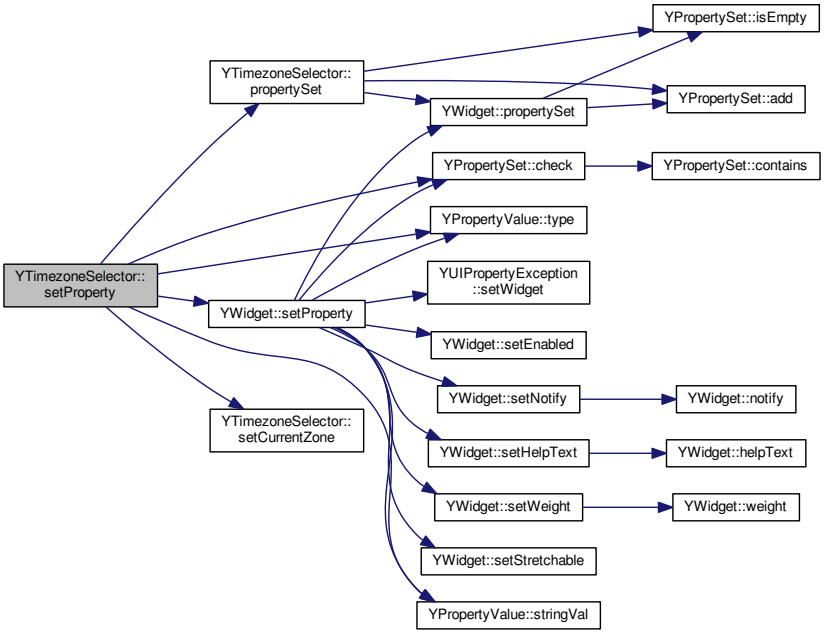
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 72 of file YTimezoneSelector.cc.

Here is the call graph for this function:



3.141.3.6 virtual const char* YTimezoneSelector::widgetClass ( ) const [inline], [virtual]

Return a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 60 of file YTimezoneSelector.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTimezoneSelector.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTimezoneSelector.cc

3.142 YTimezoneSelectorPrivate Class Reference

3.142.1 Detailed Description

Definition at line 33 of file YTimezoneSelector.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTimezoneSelector.cc

3.143 YTransText Class Reference

```
#include <YTransText.h>
```

Public Member Functions

- YTransText (const std::string &orig, const std::string &translation)
- YTransText (const std::string &orig)
- YTransText (const YTransText &src)
- YTransText & operator= (const YTransText &src)
- const std::string & orig () const
- const std::string & translation () const
- const std::string & trans () const
- void setOrig (const std::string &newOrig)
- void setTranslation (const std::string &newTrans)
- bool operator< (const YTransText &other) const
- bool operator> (const YTransText &other) const
- bool operator== (const YTransText &other) const

3.143.1 Detailed Description

Helper class for translated strings: Stores a message in the original (untranslated) version along with the translation into the current locale.

Definition at line 36 of file YTransText.h.

3.143.2 Constructor & Destructor Documentation

3.143.2.1 YTransText::YTransText (const std::string & *orig*, const std::string & *translation*) [inline]

Constructor with both original and translated message.

Definition at line 43 of file YTransText.h.

3.143.2.2 YTransText::YTransText (const std::string & *orig*) [inline]

Constructor that automatically translates the original message.

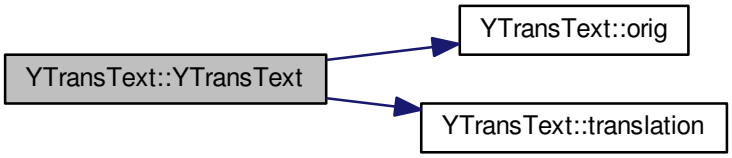
Definition at line 50 of file YTransText.h.

3.143.2.3 YTransText::YTransText (const YTransText & *src*) [inline]

Copy constructor.

Definition at line 58 of file YTransText.h.

Here is the call graph for this function:



3.143.3 Member Function Documentation

3.143.3.1 bool YTransText::operator< (const YTransText & *other*) const [inline]

operator< : Compares translations.

Definition at line 105 of file YTransText.h.

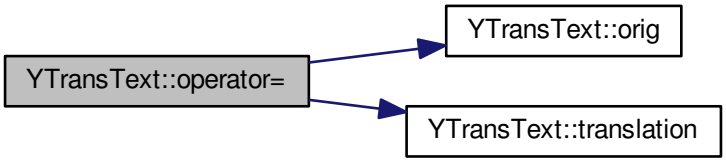
Here is the call graph for this function:

3.143.3.2 YTransText& YTransText::operator= (const YTransText & *src*) [inline]

Assignment operator.

Definition at line 67 of file YTransText.h.

Here is the call graph for this function:



3.143.3.3 bool YTransText::operator== (const YTransText & *other*) const [inline]

operator== : Compares translations.

Definition at line 117 of file YTransText.h.

Here is the call graph for this function:

3.143.3.4 bool YTransText::operator> (const YTransText & *other*) const [inline]

operator> : Compares translations.

Definition at line 111 of file YTransText.h.

Here is the call graph for this function:

3.143.3.5 const std::string& YTransText::orig ( ) const [inline]

Return the original message.

Definition at line 78 of file YTransText.h.

3.143.3.6 void YTransText::setOrig (const std::string & *newOrig*) [inline]

Set the original message. Does not touch the translation, so make sure you change both if you want to keep them synchronized!

Definition at line 95 of file YTransText.h.

3.143.3.7 void YTransText::setTranslation (const std::string & *newTrans*) [inline]

Set the translation.

Definition at line 100 of file YTransText.h.

3.143.3.8 const std::string& YTransText::trans ( ) const [inline]

Return the translation. (alias, just as a shortcut)

Definition at line 89 of file YTransText.h.

3.143.3.9 const std::string& YTransText::translation ( ) const [inline]

Return the translation.

Definition at line 83 of file YTransText.h.

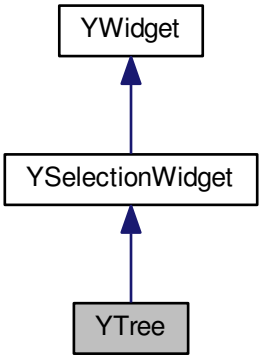
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTransText.h

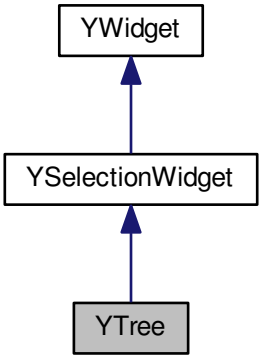
3.144 YTree Class Reference

```
#include <YTree.h>
```

Inheritance diagram for YTree:



Collaboration diagram for YTree:



Public Member Functions

- virtual ~YTree ()
- virtual const char * widgetClass () const
- virtual void rebuildTree ()=0
- virtual void addItems (const YItemCollection &itemCollection)
- bool immediateMode () const
- void setImmediateMode (bool on=true)
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)

- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()
- const char * userInputProperty ()
- bool hasMultiSelection () const
- virtual YTreeItem * currentItem ()=0

Protected Member Functions

- YTree (YWidget *parent, const std::string &label, bool multiSelection, bool recursiveSelection)

3.144.1 Detailed Description

Tree: List box that displays a (scrollable) list of hierarchical items from which the user can select exactly one. Each item has a label text and an optional icon (*).

This is very similar to SelectionBox, but each item can have subitems that can be open (expanded) or closed (collapsed).

The tree widget also has a caption label that is displayed above the tree. The hotkey displayed in that caption label will move the keyboard focus into the tree item list.

(*) Not all UIs (in particular not text-based UIs) support displaying icons, so an icon should never be an exclusive means to display any kind of information.

'multiSelection' indicates whether or not the user can select multiple items at the same time. This can only be set in the constructor.

Definition at line 56 of file YTree.h.

3.144.2 Constructor & Destructor Documentation

3.144.2.1 YTree::YTree (YWidget * *parent*, const std::string & *label*, bool *multiSelection*, bool *recursiveSelection*) [protected]

Constructor.

Definition at line 44 of file YTree.cc.

Here is the call graph for this function:

3.144.2.2 YTree::~YTree ( ) [virtual]

Destructor.

Definition at line 57 of file YTree.cc.

3.144.3 Member Function Documentation

3.144.3.1 void YTree::addItems (const YItemCollection & *itemCollection*) [virtual]

Add multiple items. For some UIs, this can be more efficient than calling addItem() multiple times. This function also automatically calls rebuildTree() at the end.

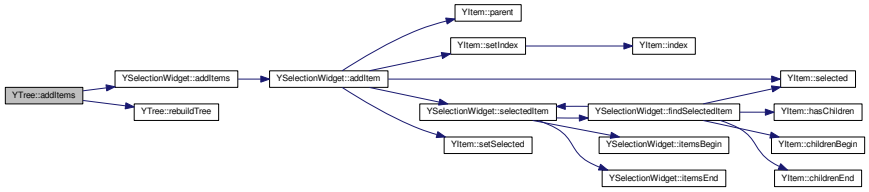
Derived classes can overwrite this function, but they should call this base class function at the end of the new implementation.

Reimplemented from YSelectionWidget.

Reimplemented from YSelectionWidget.

Definition at line 81 of file YTree.cc.

Here is the call graph for this function:



3.144.3.2 virtual YTreeItem* YTree::currentItem ( ) [pure virtual]

Return the the item that currently has the keyboard focus or 0 if no item currently has the keyboard focus.

Notice that for a MultiSelectionBox the current item is not necessarily selected, i.e., its check box may or may not be checked.

Derived classes are required to implement this function.

3.144.3.3 YPropertyValue YTree::getProperty (const std::string & *propertyName*) [virtual]

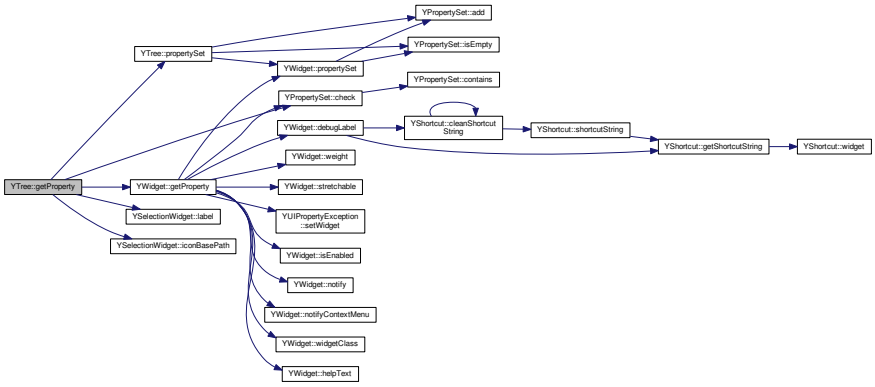
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 146 of file YTree.cc.

Here is the call graph for this function:

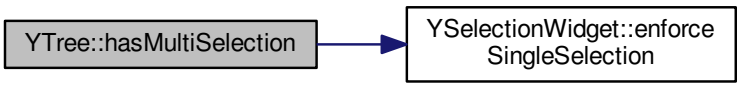


3.144.3.4 bool YTree::hasMultiSelection ( ) const

Return 'true' if the user can select multiple items at the same time

Definition at line 165 of file YTree.cc.

Here is the call graph for this function:



3.144.3.5 bool YTree::immediateMode ( ) const

Deliver even more events than with notify() set.

For YTree, this is relevant mostly for the NCurses UI:

In graphical UIs like the Qt UI, the user can use the mouse to select an item in a tree. With notify() set, this will send an event right away (i.e., it will make UserInput and related return, while normally it would only return when the user clicks a PushButton).

In the NCurses UI, there is no mouse, so the user has to use the cursor keys to move to the item he wants to select. In immediateMode(), every cursor key press will make the tree send an event. Without immediateMode(), the NCTree will wait until the user hits the [Return] key until an event is sent. Depending on what the application does upon each selection box event, immediateMode() might make the application less responsive.

Definition at line 64 of file YTree.cc.

3.144.3.6 const YPropertySet & YTree::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 89 of file YTree.cc.

Here is the call graph for this function:

3.144.3.7 virtual void YTree::rebuildTree ( ) [pure virtual]

Rebuild the displayed tree from the internally stored YTreeItems.

The application should call this (once) after all items have been added with addItem(). YTree::addItems() calls this automatically.

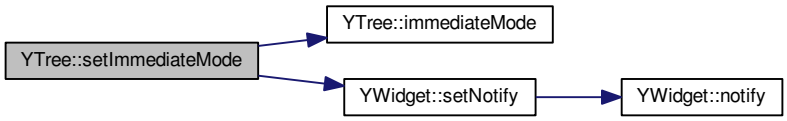
Derived classes are required to implement this.

3.144.3.8 void YTree::setImmediateMode (bool *on* = true)

Set immediateMode() on or off.

Definition at line 71 of file YTree.cc.

Here is the call graph for this function:



3.144.3.9 bool YTree::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) [virtual]

Set a property. Reimplemented from YWidget.

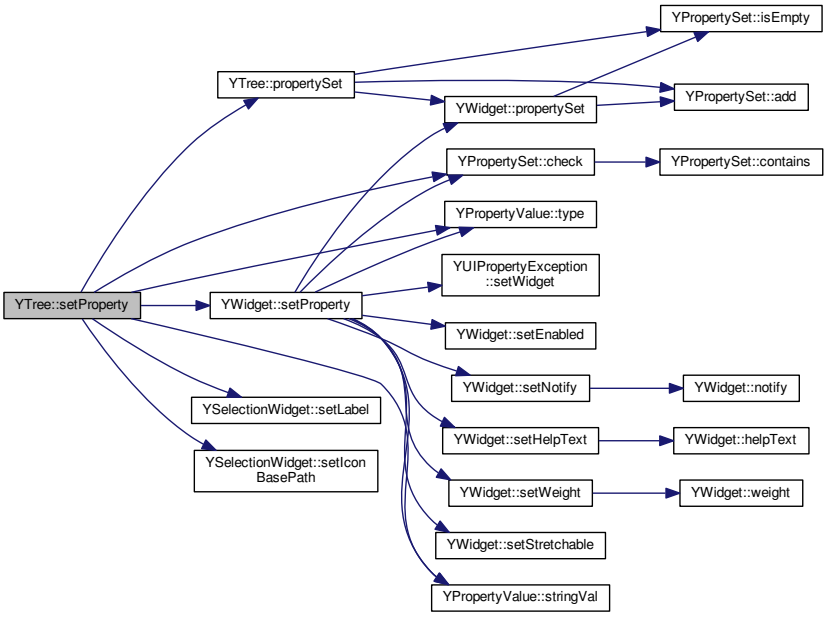
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from YWidget.

Definition at line 123 of file YTree.cc.

Here is the call graph for this function:



3.144.3.10 const char* YTree::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input. Inherited from YWidget.

Reimplemented from YWidget.

Definition at line 156 of file YTree.h.

3.144.3.11 virtual const char* YTree::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YSelectionWidget.

Definition at line 74 of file YTree.h.

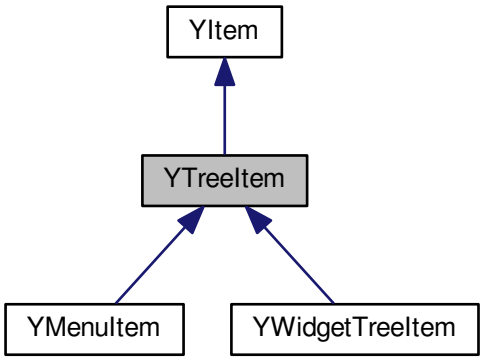
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTree.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTree.cc

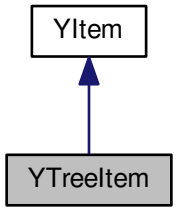
3.145 YTreeItem Class Reference

```
#include <YTreeItem.h>
```

Inheritance diagram for YTreeItem:



Collaboration diagram for YTreeItem:



Public Member Functions

- YTreeItem (const std::string &label, bool isOpen=false)
- **YTreeItem** (const std::string &label, const std::string &iconName, bool isOpen=false)
- YTreeItem (YTreeItem *parent, const std::string &label, bool isOpen=false)
- **YTreeItem** (YTreeItem *parent, const std::string &label, const std::string &iconName, bool isOpen=false)
- virtual ~YTreeItem ()
- virtual bool hasChildren () const
- virtual YItemIterator childrenBegin ()
- virtual YItemConstIterator **childrenBegin** () const

- virtual YItemIterator childrenEnd ()
- virtual YItemConstIterator **childrenEnd** () const
- virtual void addChild (YItem *item_disown)
- virtual void deleteChildren ()
- bool isOpen () const
- void setOpen (bool open)
- virtual YTreeItem * parent () const

3.145.1 Detailed Description

Item class for tree items.

This class implements children management.

Definition at line 37 of file YTreeItem.h.

3.145.2 Constructor & Destructor Documentation

3.145.2.1 YTreeItem::YTreeItem (const std::string & *label*, bool *isOpen* = `false`)

Constructors for toplevel items.

Definition at line 28 of file YTreeItem.cc.

3.145.2.2 YTreeItem::YTreeItem (YTreeItem * *parent*, const std::string & *label*, bool *isOpen* = `false`)

Constructors for items that have a parent item.

They will automatically register this item with the parent item. The parent assumes ownership of this item and will delete it in its (the parent's) destructor.

Definition at line 47 of file YTreeItem.cc.

Here is the call graph for this function:

3.145.2.3 YTreeItem::~YTreeItem ( ) `[virtual]`

Destructor.

This will delete all children.

Definition at line 72 of file YTreeItem.cc.

Here is the call graph for this function:

3.145.3 Member Function Documentation

3.145.3.1 void YTreeItem::addChild (YItem * *item_disown*) [virtual]

Add a child item to this item.

Note that the constructors that accept a parent pointer will automatically add themselves to their parent, so applications will normally not have to call this function.

Definition at line 78 of file YTreeItem.cc.

3.145.3.2 virtual YItemIterator YTreeItem::childrenBegin ( ) [inline],[virtual]

Return an iterator that points to the first child item of this item.

Reimplemented from YItem.

Reimplemented from YItem.

Definition at line 85 of file YTreeItem.h.

3.145.3.3 virtual YItemIterator YTreeItem::childrenEnd ( ) [inline],[virtual]

Return an iterator that points after the last child item of this item.

Reimplemented from YItem.

Reimplemented from YItem.

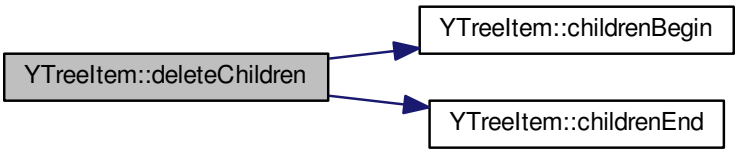
Definition at line 93 of file YTreeItem.h.

3.145.3.4 void YTreeItem::deleteChildren ( ) [virtual]

Delete all child items.

Definition at line 84 of file YTreeItem.cc.

Here is the call graph for this function:



3.145.3.5 virtual bool YTreeItem::hasChildren ( ) const [inline], [virtual]

Return 'true' if this item has any child items.

Reimplemented from YItem.

Reimplemented from YItem.

Definition at line 78 of file YTreeItem.h.

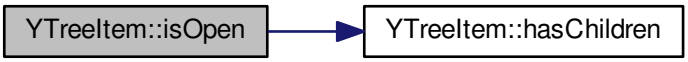
3.145.3.6 bool YTreeItem::isOpen ( ) const

Return 'true' if this tree item should be displayed open (with its children visible) by default.

Notice that this will always return 'false' for tree items without children.

Definition at line 99 of file YTreeItem.cc.

Here is the call graph for this function:



3.145.3.7 virtual YTreeItem* YTreeItem::parent ( ) const [inline], [virtual]

Returns this item's parent item or 0 if it is a toplevel item.

Reimplemented from YItem.

Reimplemented from YItem.

Reimplemented in YMenuItem.

Definition at line 129 of file YTreeItem.h.

3.145.3.8 void YTreeItem::setOpen (bool *open*)

Change the 'isOpen' flag.

Definition at line 105 of file YTreeItem.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTreeItem.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTreeItem.cc

3.146 YTreePrivate Struct Reference

Public Attributes

- bool **immediateMode**

3.146.1 Detailed Description

Definition at line 34 of file YTree.cc.

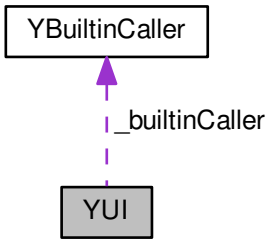
The documentation for this struct was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YTree.cc

3.147 YUI Class Reference

```
#include <YUI.h>
```

Collaboration diagram for YUI:



Public Member Functions

- virtual ~YUI ()
- void shutdownThreads ()

- virtual void blockEvents (bool block=true)
- void unblockEvents ()
- virtual bool eventsBlocked () const
- virtual void deleteNotify (YWidget *widget)
- void topmostConstructorHasFinished ()
- bool runningWithThreads () const
- void uiThreadMainLoop ()
- YBuiltinCaller * builtinCaller () const
- void setBuiltinCaller (YBuiltinCaller *caller)
- virtual YEvent * runPkgSelection (YWidget *packageSelector)=0

Static Public Member Functions

- static YUI * ui ()
- static YWidgetFactory * widgetFactory ()
- static YOptionalWidgetFactory * optionalWidgetFactory ()
- static YApplication * app ()
- static YApplication * application ()
- static YApplication * **yApp** ()
- static void ensureUICreated ()

Protected Member Functions

- YUI (bool withThreads)
- virtual YWidgetFactory * createWidgetFactory ()=0
- virtual YOptionalWidgetFactory * createOptionalWidgetFactory ()=0
- virtual YApplication * createApplication ()=0
- virtual void idleLoop (int fd_ycp)=0
- void terminateUIThread ()
- void createUIThread ()
- virtual void uiThreadDestructor ()
- void signalUIThread ()
- bool waitForUIThread ()
- void signalYCPThread ()
- bool waitForYCPThread ()
- void setButtonOrderFromEnvironment ()

Protected Attributes

- bool _withThreads
- pthread_t _uiThread
- YBuiltinCaller * _builtinCaller
- int pipe_to_ui [2]
- int pipe_from_ui [2]
- bool _terminate_ui_thread
- bool _eventsBlocked

Friends

- class **YUIFunction**
- class **YUITerminator**
- void ∗ **start_ui_thread** (void ∗ui_int)

3.147.1 Detailed Description

Abstract base class of a libYUI user interface.

Definition at line 48 of file YUI.h.

3.147.2 Constructor & Destructor Documentation

3.147.2.1 YUI::YUI (bool *withThreads*) [protected]

Constructor.

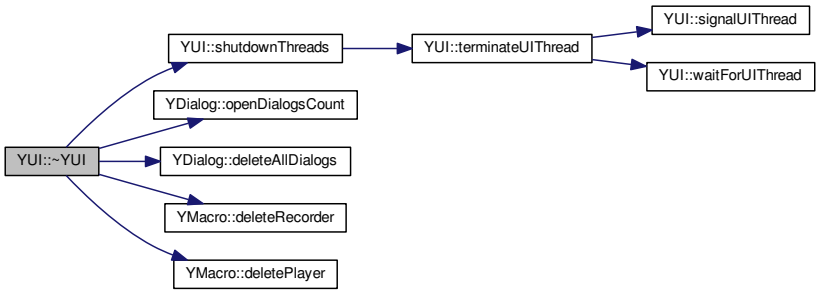
Definition at line 69 of file YUI.cc.

3.147.2.2 YUI::∼YUI ( ) [virtual]

Destructor.

Definition at line 82 of file YUI.cc.

Here is the call graph for this function:



3.147.3 Member Function Documentation

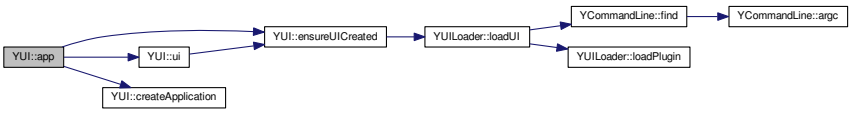
3.147.3.1 YApplication ∗ YUI::app ( ) [static]

Return the global YApplication object.

This will create the YApplication upon the first call and return a pointer to the one and only (singleton) YApplication upon each subsequent call. This may throw exceptions if the YApplication cannot be created.

Definition at line 156 of file YUI.cc.

Here is the call graph for this function:



3.147.3.2 static YApplication* YUI::application ( ) [inline], [static]

Aliases for YUI::app()

Definition at line 112 of file YUI.h.

Here is the call graph for this function:

3.147.3.3 virtual void YUI::blockEvents (bool *block* = true) [inline], [virtual]

Block (or unblock) events. If events are blocked, any event sent should be ignored until events are unblocked again.

This default implementation keeps track of a simple internal flag that can be queried with eventsBlocked(), so if you reimplement blockEvents(), be sure to reimplement eventsBlocked() as well.

Definition at line 161 of file YUI.h.

3.147.3.4 YBuiltinCaller* YUI::builtinCaller ( ) const [inline]

Return the transparent inter-thread communication. This will return 0 until set from the outside.

Definition at line 212 of file YUI.h.

3.147.3.5 virtual YApplication* YUI::createApplication ( ) [protected], [pure virtual]

Create the YApplication object that provides global methods.

Derived classes are required to implement this.

3.147.3.6 virtual YOptionalWidgetFactory* YUI::createOptionalWidgetFactory ( ) [protected], [pure virtual]

Create the widget factory that provides all the createXY() methods for optional ("special") widgets and the corresponding hasXYWidget() methods.

Derived classes are required to implement this.

3.147.3.7 void YUI::createUIThread ( ) [protected]

Creates and launches the ui thread.

Definition at line 235 of file YUI.cc.

3.147.3.8 virtual YWidgetFactory* YUI::createWidgetFactory ( ) [protected], [pure virtual]

Create the widget factory that provides all the createXY() methods for standard (mandatory, i.e. non-optional) widgets.

Derived classes are required to implement this.

3.147.3.9 virtual void YUI::deleteNotify (YWidget * *widget*) [inline], [virtual]

Notification that a widget is being deleted. This is called from the YWidget destructor.

Derived classes can implement this for any clean-up actions such as deleting any events that might be pending for that widget.

Definition at line 185 of file YUI.h.

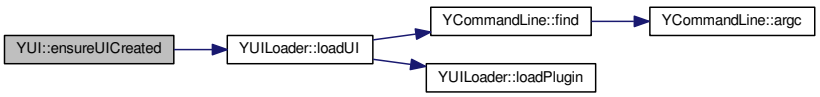
3.147.3.10 void YUI::ensureUICreated ( ) [static]

Make sure there is a UI (with a UI plug-in) created.

If there is none yet, this will use all-default parameters to load a UI plug-in and create a UI (without threads).

Definition at line 170 of file YUI.cc.

Here is the call graph for this function:



3.147.3.11 virtual bool YUI::eventsBlocked ( ) const [inline], [virtual]

Returns 'true' if events are currently blocked.

Reimplement this if you reimplement blockEvents().

Definition at line 176 of file YUI.h.

3.147.3.12 virtual void YUI::idleLoop (int *fd_ycp*) [protected], [pure virtual]

This virtual method is called when threads are activated in case the execution control is currently on the side of the module. This means that no UserInput() or PollInput() is pending. The module just does some work. The UI <->

module protocol is in the "UI waits for the next command" state. The UI can override this method when it wants to react to user input or other external events such as repaint requests from the X server.

'fd_ycp' file descriptor that should be used to determine when to leave the idle loop. As soon as it is readable, the loop must be left. In order to avoid polling you can combine it with other ui-specific fds and do a common select() call.

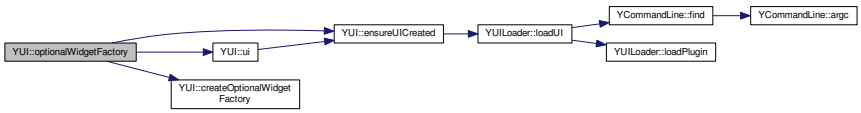
3.147.3.13 YOptionalWidgetFactory * YUI::optionalWidgetFactory ( ) [static]

Return the widget factory that provides all the createXY() methods for optional ("special") widgets and the corresponding hasXYWidget() methods.

This will create the factory upon the first call and return a pointer to the one and only (singleton) factory upon each subsequent call. This may throw exceptions if the factory cannot be created.

Definition at line 141 of file YUI.cc.

Here is the call graph for this function:



3.147.3.14 bool YUI::runningWithThreads ( ) const [inline]

Running with threads?

Definition at line 196 of file YUI.h.

3.147.3.15 virtual YEvent* YUI::runPkgSelection (YWidget * *packageSelector*) [pure virtual]

UI-specific runPkgSelection method.

Derived classes are required to implement this.

The packageSelector's dialog will take care of the event and delete it when appropriate. The returned pointer is valid until the next call to YDialog::userInput(), YDialog::pollInput(), or YUI::runPkgSelection() or until the dialog with the packageSelector is destroyed.

3.147.3.16 void YUI::setBuiltinCaller (YBuiltinCaller * *caller*) [inline]

Set the transparent inter-thread communication. Built-ins are only really called if there is a valid YBuiltinCaller set.

Definition at line 218 of file YUI.h.

3.147.3.17 void YUI::setButtonOrderFromEnvironment ( ) [protected]

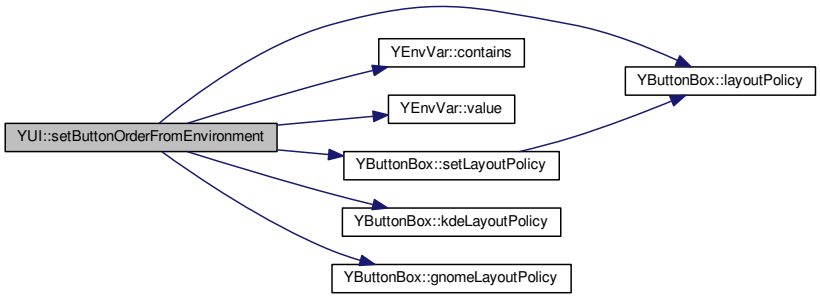
Set the button order (in YButtonBox widgets) from environment variables:

```
$Y2_BUTTON_ORDER="KDE"
$Y2_BUTTON_ORDER="Gnome"
```

(all case insensitive)

Definition at line 386 of file YUI.cc.

Here is the call graph for this function:



3.147.3.18 void YUI::shutdownThreads ( )

Shut down multithreading. This needs to be called before the destructor if the UI was created with threads. If the UI was created without threads, this does nothing.

Definition at line 259 of file YUI.cc.

Here is the call graph for this function:

3.147.3.19 void YUI::signalUIThread ( ) [protected]

Signals the ui thread by sending one byte through the pipe to it.

Definition at line 273 of file YUI.cc.

3.147.3.20 void YUI::signalYCPThread ( ) [protected]

Signals the ycp thread by sending one byte through the pipe to it.

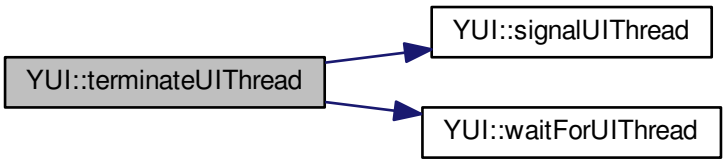
Definition at line 313 of file YUI.cc.

3.147.3.21 void YUI::terminateUIThread ( ) [protected]

Tells the ui thread that it should terminate and waits until it does so.

Definition at line 246 of file YUI.cc.

Here is the call graph for this function:

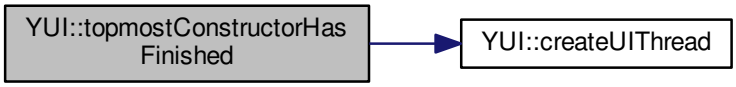


3.147.3.22 void YUI::topmostConstructorHasFinished ( )

Must be called after the constructor of the Qt/NCurses ui is ready. Starts the ui thread.

Definition at line 182 of file YUI.cc.

Here is the call graph for this function:

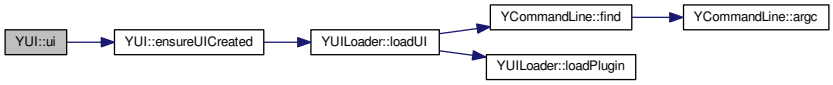


3.147.3.23 YUI * YUI::ui ( ) [static]

Access the global UI.

Definition at line 118 of file YUI.cc.

Here is the call graph for this function:



3.147.3.24 void YUI::uiThreadDestructor ( ) [protected],[virtual]

Destructor for the UI thread. This will be called as the last thing the UI thread does.

Derived classes can overwrite this. In most cases it makes sense to call this base class method in the new implementation.

Definition at line 111 of file YUI.cc.

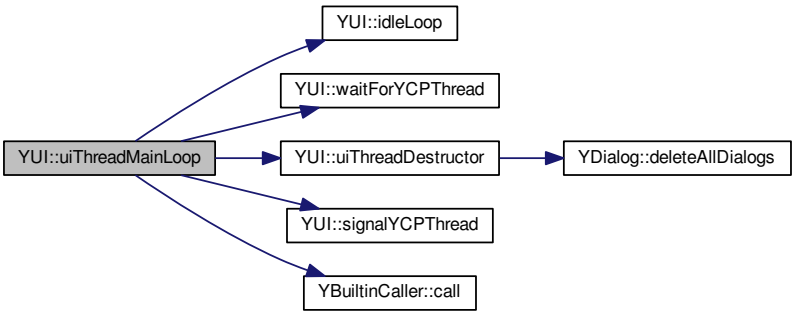
Here is the call graph for this function:

3.147.3.25 void YUI::uiThreadMainLoop ( )

This method implements the UI thread in case it is existing. The loop consists of calling idleLoop, getting the next command from the YCPUIComponent, evaluating it, which possibly invovles calling userInput() or pollInput() and writes the answer back to the other thread where the request came from.

Definition at line 353 of file YUI.cc.

Here is the call graph for this function:



3.147.3.26 void YUI::unblockEvents ( ) [inline]

Unblock events previously blocked. This is just an alias for blockEvents(false) for better readability.

Note: This method is intentionally not virtual.

Definition at line 169 of file YUI.h.

Here is the call graph for this function:

3.147.3.27 bool YUI::waitForUIThread ( ) [protected]

Waits for the ui thread to send one byte through the pipe to the ycp thread and reads this byte from the pipe.

Definition at line 285 of file YUI.cc.

3.147.3.28 bool YUI::waitForYCPThread ( ) [protected]

Waits for the ycp thread to send one byte through the pipe to the ycp thread and reads this byte from the pipe.

Definition at line 325 of file YUI.cc.

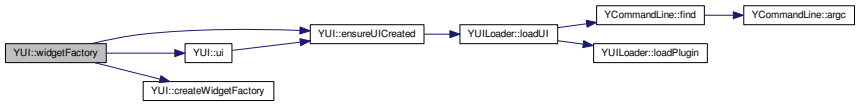
3.147.3.29 YWidgetFactory * YUI::widgetFactory ( ) [static]

Return the widget factory that provides all the createXY() methods for standard (mandatory, i.e. non-optional) widgets.

This will create the factory upon the first call and return a pointer to the one and only (singleton) factory upon each subsequent call. This may throw exceptions if the factory cannot be created.

Definition at line 126 of file YUI.cc.

Here is the call graph for this function:



3.147.4 Member Data Documentation

3.147.4.1 YBuiltinCaller* YUI::_builtinCaller [protected]

Inter-thread communication between the YCP thread and the UI thread: The YCP thread supplies data here and signals the UI thread, the UI thread picks up the data, executes the function, puts the result here and signals the YCP thread that waits until the result is available.

Definition at line 330 of file YUI.h.

3.147.4.2 bool YUI::_eventsBlocked [protected]

Flag that keeps track of blocked events. Never query this directly, use eventsBlocked() instead.

Definition at line 358 of file YUI.h.

3.147.4.3 bool YUI::_terminate_ui_thread [protected]

This is a flag that signals the ui thread that it should terminate. This is done by setting the flag to true. The ui thread replies by setting the flag back to false directly after terminating itself.

Definition at line 352 of file YUI.h.

3.147.4.4 pthread_t YUI::_uiThread [protected]

Handle to the ui thread.

Definition at line 321 of file YUI.h.

3.147.4.5 bool YUI::_withThreads [protected]

true if a seperate UI thread is created

Definition at line 316 of file YUI.h.

3.147.4.6 int YUI::pipe_from_ui[2] [protected]

Used to synchronize data transfer with the ui thread. It stores a pair of file descriptors of a pipe. For each YCP value we get from the ui thread, we read one aribrary byte from here.

Definition at line 344 of file YUI.h.

3.147.4.7 int YUI::pipe_to_ui[2] [protected]

Used to synchronize data transfer with the ui thread. It stores a pair of file descriptors of a pipe. For each YCP value we send to the ui thread, we write one aribrary byte here.

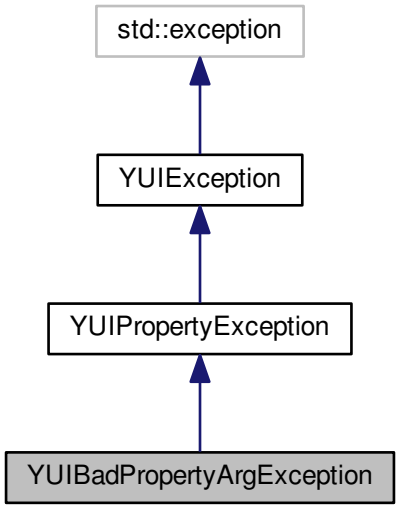
Definition at line 337 of file YUI.h.

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

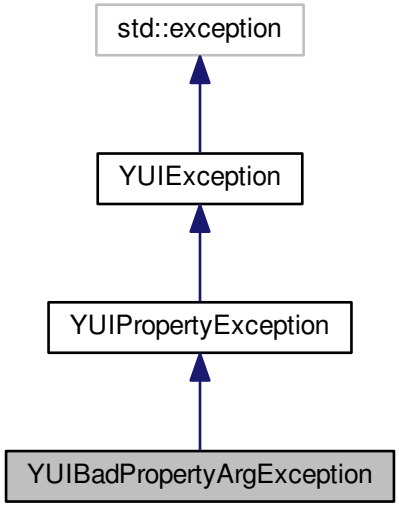
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUI.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUI.cc

3.148 YUIBadPropertyArgException Class Reference

Inheritance diagram for YUIBadPropertyArgException:



Collaboration diagram for YUIBadPropertyArgException:



Public Member Functions

- **YUIBadPropertyArgException** (const YProperty &property, YWidget *widget, const std::string &message="")

Protected Member Functions

- virtual std::ostream & dumpOn (std::ostream &str) const

Additional Inherited Members

3.148.1 Detailed Description

Definition at line 619 of file YUIException.h.

3.148.2 Member Function Documentation

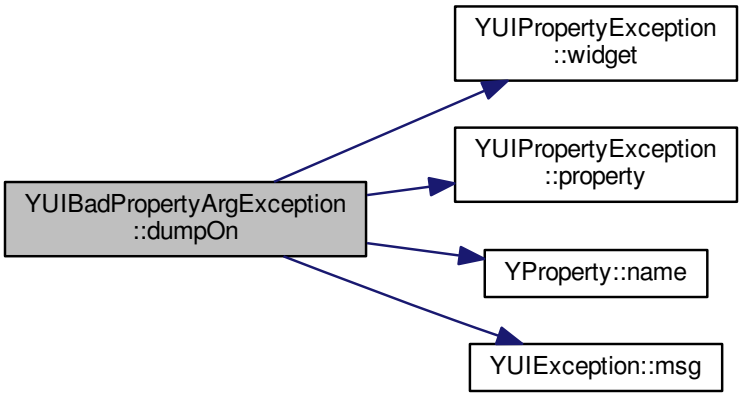
3.148.2.1 std::ostream & YUIBadPropertyArgException::dumpOn (std::ostream & *str*) const [protected], [virtual]

Write proper error message with all relevant data. Reimplemented from YUIException.

Implements YUIPropertyException.

Definition at line 196 of file YUIException.cc.

Here is the call graph for this function:



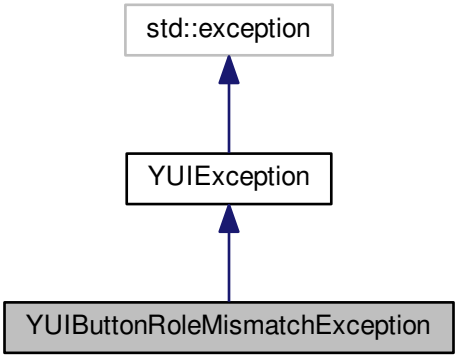
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.cc

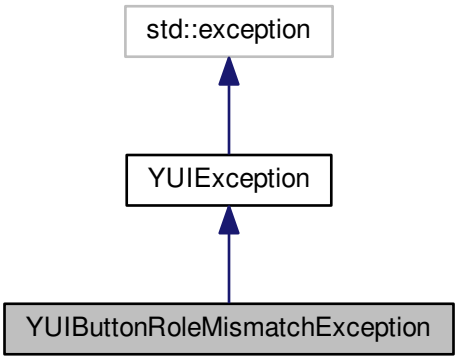
3.149 YUIButtonRoleMismatchException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIButtonRoleMismatchException:



Collaboration diagram for YUIButtonRoleMismatchException:



Public Member Functions

- **YUIButtonRoleMismatchException** (const std::string &msg)

Additional Inherited Members

3.149.1 Detailed Description

Exception class for "wrong button roles in YButtonBox"

Definition at line 889 of file YUIException.h.

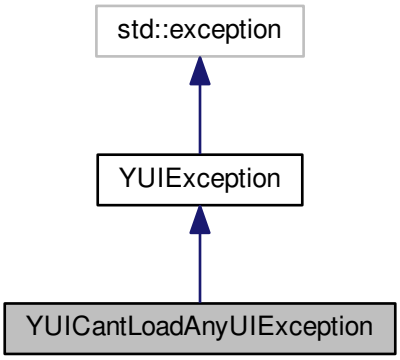
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

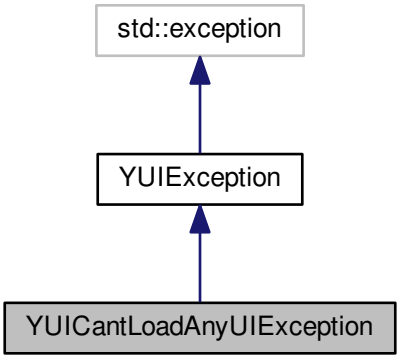
3.150 YUICantLoadAnyUIException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUICantLoadAnyUIException:



Collaboration diagram for YUICantLoadAnyUIException:



Additional Inherited Members

3.150.1 Detailed Description

Exception class for UI plugin load failure

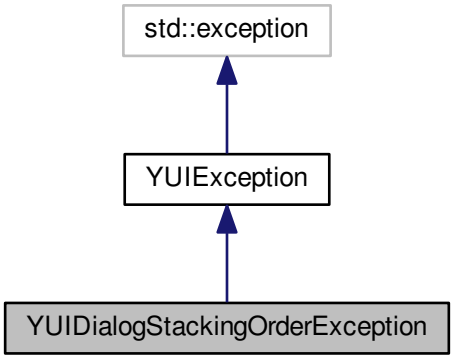
Definition at line 874 of file YUIException.h.

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

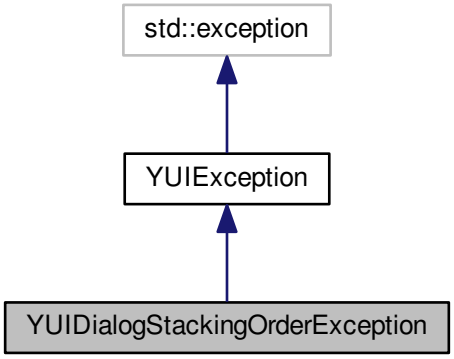
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

3.151 YUIDialogStackingOrderException Class Reference

Inheritance diagram for YUIDialogStackingOrderException:



Collaboration diagram for YUIDialogStackingOrderException:



Additional Inherited Members

3.151.1 Detailed Description

Definition at line 463 of file YUIException.h.

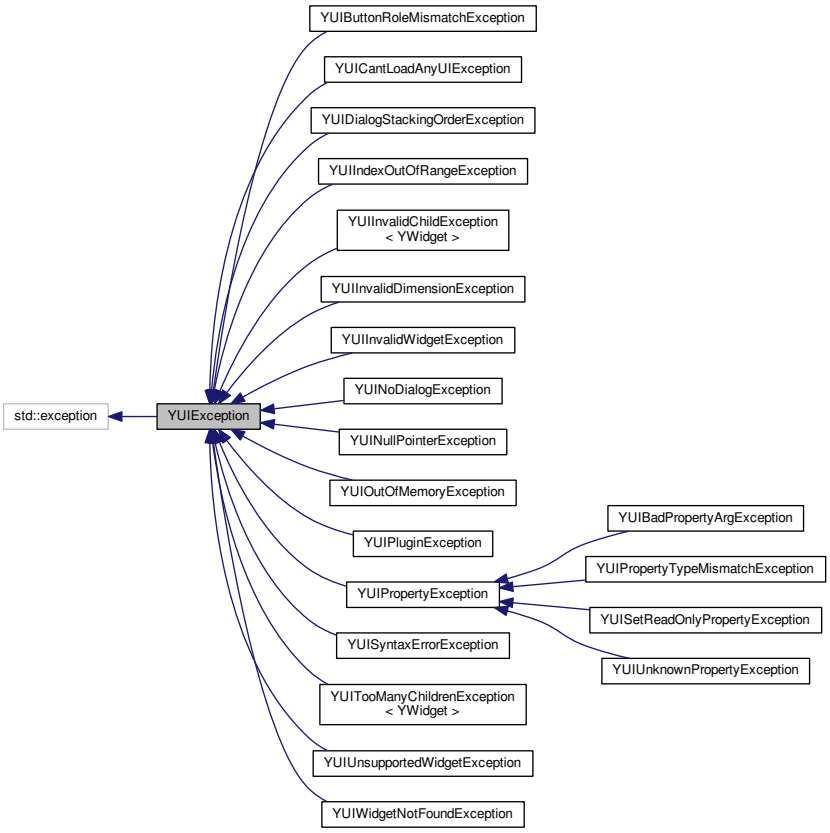
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

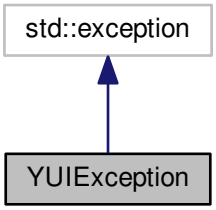
3.152 YUIException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIException:



Collaboration diagram for YUIException:



Public Member Functions

- YUIException ()
- YUIException (const std::string &msg_r)
- virtual ~YUIException () throw ()
- const YCodeLocation & where () const
- void relocate (const YCodeLocation &newLocation) const
- const std::string & msg () const
- void setMsg (const std::string &msg)
- std::string asString () const
- virtual const char * what () const throw ()

Static Public Member Functions

- static std::string strErrno (int errno_r)
- static std::string strErrno (int errno_r, const std::string &msg)
- static void log (const YUIException &exception, const YCodeLocation &location, const char *const prefix)

Protected Member Functions

- virtual std::ostream & dumpOn (std::ostream &str) const

Friends

- std::ostream & operator<< (std::ostream &str, const YUIException &obj)

3.152.1 Detailed Description

Base class for UI Exceptions.

Exception offers to store a message string passed to the constructor. Derived classes may provide additional information. Overload dumpOn to provide a proper error text.

Definition at line 281 of file YUIException.h.

3.152.2 Constructor & Destructor Documentation

3.152.2.1 YUIException::YUIException ( )

Default constructor. Use YUI_THROW to throw exceptions.

Definition at line 62 of file YUIException.cc.

3.152.2.2 YUIException::YUIException (const std::string & *msg_r*)

Constructor taking a message. Use YUI_THROW to throw exceptions.

Definition at line 67 of file YUIException.cc.

3.152.2.3 YUIException::~YUIException ( ) throw () [virtual]

Destructor.

Definition at line 74 of file YUIException.cc.

3.152.3 Member Function Documentation

3.152.3.1 std::string YUIException::asString ( ) const

Error message provided by dumpOn as string.

Definition at line 81 of file YUIException.cc.

Here is the call graph for this function:

3.152.3.2 std::ostream & YUIException::dumpOn (std::ostream & *str*) const [protected], [virtual]

Overload this to print a proper error message.

Reimplemented in YUIIndexOutOfRangeException, YUIInvalidChildException< YWidget >, YUITooManyChildrenException< YWidget >, YUIBadPropertyArgException, YUISetReadOnlyPropertyException, YUIPropertyTypeMismatchException, YUIUnknownPropertyException, and YUIPropertyException.

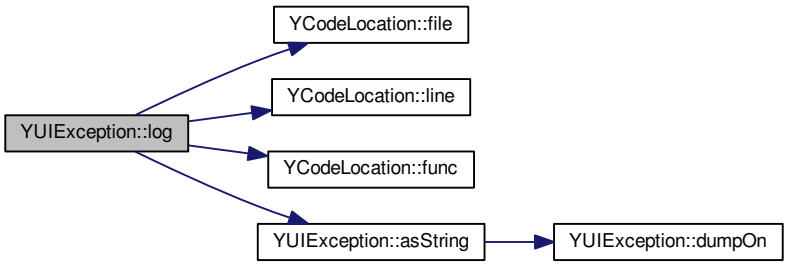
Definition at line 90 of file YUIException.cc.

3.152.3.3 void YUIException::log (const YUIException & *exception*, const YCodeLocation & *location*, const char *const *prefix*) [static]

Drop a log line on throw, catch or rethrow. Used by YUI_THROW macros.

Definition at line 127 of file YUIException.cc.

Here is the call graph for this function:



3.152.3.4 const std::string& YUIException::msg ( ) const [inline]

Return the message string provided to the constructor. Note: This is not neccessarily the complete error message. The whole error message is provided by asString or dumpOn.

Definition at line 318 of file YUIException.h.

3.152.3.5 void YUIException::relocate (const YCodeLocation & *newLocation*) const [inline]

Exchange location on rethrow.

Definition at line 310 of file YUIException.h.

3.152.3.6 void YUIException::setMsg (const std::string & *msg*) [inline]

Set a new message string.

Definition at line 324 of file YUIException.h.

Here is the call graph for this function:

3.152.3.7 std::string YUIException::strErrno (int *errno_r*) [static]

Make a string from errno_r.

Definition at line 111 of file YUIException.cc.

3.152.3.8 std::string YUIException::strErrno (int *errno_r,* const std::string & *msg*) [static]

Make a string from errno_r and msg_r.

Definition at line 118 of file YUIException.cc.

Here is the call graph for this function:

3.152.3.9 virtual const char* YUIException::what ( ) const throw () [inline], [virtual]

Return message string.

Reimplemented from std::exception.

Definition at line 354 of file YUIException.h.

3.152.3.10 const YCodeLocation& YUIException::where ( ) const [inline]

Return YCodeLocation.

Definition at line 304 of file YUIException.h.

3.152.4 Friends And Related Function Documentation

3.152.4.1 std::ostream& operator<< (std::ostream & *str,* const YUIException & *obj*) [friend]

YUIException stream output

Definition at line 104 of file YUIException.cc.

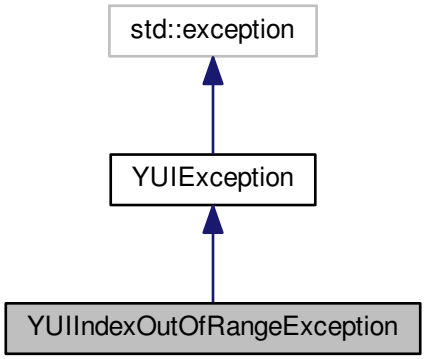
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.cc

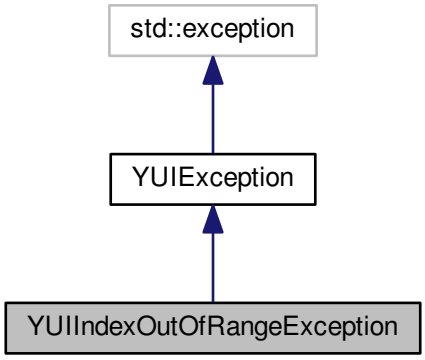
3.153 YUIIndexOutOfRangeException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIIndexOutOfRangeException:



Collaboration diagram for YUIIndexOutOfRangeException:



Public Member Functions

- YUIIndexOutOfRangeException (int invalidIndex, int validMin, int validMax, const std::string &msg="")
- int invalidIndex () const
- int validMin () const
- int validMax () const

Protected Member Functions

- virtual std::ostream & dumpOn (std::ostream &str) const

Additional Inherited Members

3.153.1 Detailed Description

Exception class for "index out of range"

Definition at line 791 of file YUIException.h.

3.153.2 Constructor & Destructor Documentation

3.153.2.1 YUIIndexOutOfRangeException::YUIIndexOutOfRangeException (int *invalidIndex*, int *validMin*, int *validMax*, const std::string & *msg* = " ") [inline]

Constructor.

'invalidIndex' is the offending index value. It should be between 'validMin' and 'validMax':

```
validMin <= index <= validMax
```

Definition at line 802 of file YUIException.h.

3.153.3 Member Function Documentation

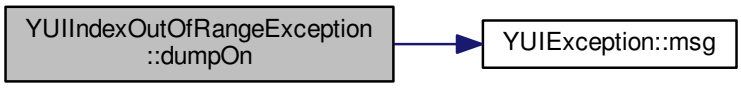
3.153.3.1 virtual std::ostream& YUIIndexOutOfRangeException::dumpOn (std::ostream & *str*) const [inline], [protected], [virtual]

Write proper error message with all relevant data. Reimplemented from YUIException.

Reimplemented from YUIException.

Definition at line 836 of file YUIException.h.

Here is the call graph for this function:



3.153.3.2 int YUIIndexOutOfRangeException::invalidIndex ( ) const [inline]

Return the offending index value.

Definition at line 818 of file YUIException.h.

3.153.3.3 int YUIIndexOutOfRangeException::validMax ( ) const [inline]

Return the valid maximum index.

Definition at line 828 of file YUIException.h.

3.153.3.4 int YUIIndexOutOfRangeException::validMin ( ) const [inline]

Return the valid minimum index.

Definition at line 823 of file YUIException.h.

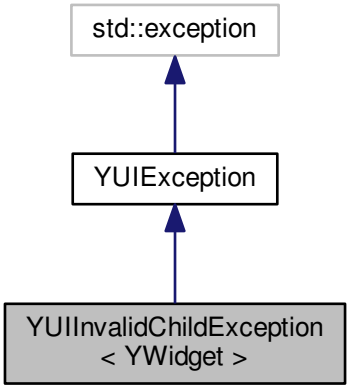
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

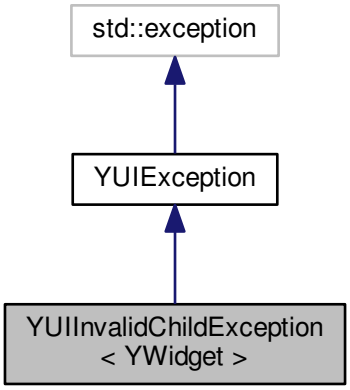
3.154 YUIInvalidChildException< YWidget > Class Template Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIInvalidChildException< YWidget >:



Collaboration diagram for YUIInvalidChildException< YWidget >:



Public Member Functions

- **YUIInvalidChildException** (YWidget *container, YWidget *child=0)
- YWidget * container () const
- YWidget * child () const

Protected Member Functions

- virtual std::ostream & dumpOn (std::ostream &str) const

Additional Inherited Members

3.154.1 Detailed Description

template<class YWidget>class YUIInvalidChildException< YWidget >

Exception class for "invalid child". One of:

- Attempt to remove a child from a children manager that is not in that manager's children list.
- Child widget of wrong type added to a container widget, e.g., anything other than a YPushButton added to a YButtonBox.

Definition at line 696 of file YUIException.h.

3.154.2 Member Function Documentation

3.154.2.1 template<class YWidget > YWidget* YUIInvalidChildException< YWidget >::child ( ) const [inline]

Returns the child widget.

Definition at line 718 of file YUIException.h.

3.154.2.2 template<class YWidget > YWidget* YUIInvalidChildException< YWidget >::container ( ) const [inline]

Returns the container widget whose child should be removed etc.

Definition at line 713 of file YUIException.h.

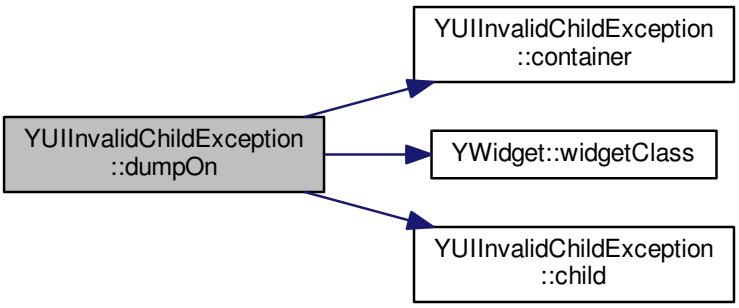
3.154.2.3 template<class YWidget > virtual std::ostream& YUIInvalidChildException< YWidget >::dumpOn (std::ostream & *str*) const [inline], [protected], [virtual]

Write proper error message with all relevant data. Reimplemented from YUIException.

Reimplemented from YUIException.

Definition at line 726 of file YUIException.h.

Here is the call graph for this function:



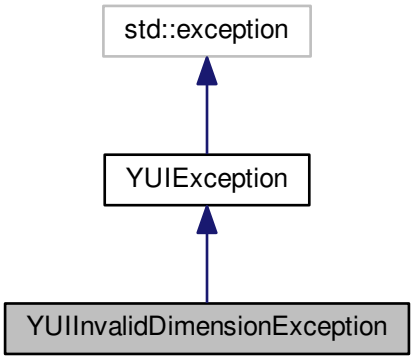
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

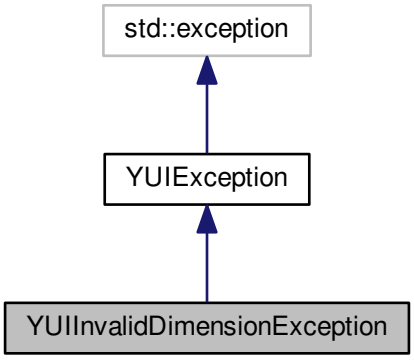
3.155 YUIInvalidDimensionException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIInvalidDimensionException:



Collaboration diagram for YUIInvalidDimensionException:



Additional Inherited Members

3.155.1 Detailed Description

Exception class for "value other than YD_HORIZ or YD_VERT used for dimension".

Definition at line 776 of file YUIException.h.

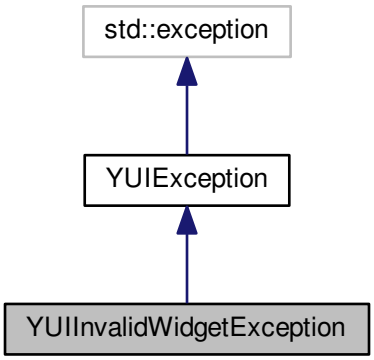
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

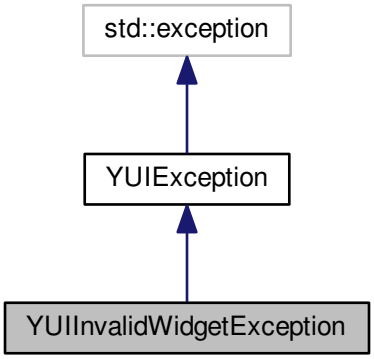
3.156 YUIInvalidWidgetException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIInvalidWidgetException:



Collaboration diagram for YUIInvalidWidgetException:



Additional Inherited Members

3.156.1 Detailed Description

Exception class for invalid widgets. This is typically caused by widget pointers that continue living after the corresponding widget has already been deleted.

Definition at line 424 of file YUIException.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

3.157 YUILoader Class Reference

```
#include <YUILoader.h>
```

Static Public Member Functions

- static void loadUI (bool withThreads=false)
- static void loadPlugin (const std::string &name, bool withThreads=false)
- static bool **pluginExists** (const std::string &pluginBaseName)

3.157.1 Detailed Description

Class to load one of the concrete UI plug-ins: Qt, NCurses, Gtk.

Definition at line 44 of file YUILoader.h.

3.157.2 Member Function Documentation

3.157.2.1 void YUILoader::loadPlugin (const std::string & *name*, bool *withThreads* = `false`) [static]

Load a UI plug-in. 'name' is one of the YUIPlugin_ -defines above.

This might throw exceptions.

Definition at line 99 of file YUILoader.cc.

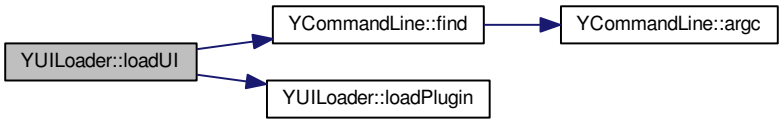
3.157.2.2 void YUILoader::loadUI (bool *withThreads* = `false`) [static]

Load any of the available UI plug-ins in this order:

- Qt if $DISPLAY is set
- NCurses if stdout is a tty

Definition at line 39 of file YUILoader.cc.

Here is the call graph for this function:



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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUILoader.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUILoader.cc

3.158 YUILog Class Reference

```
#include <YUILog.h>
```

Public Member Functions

- std::ostream & log (YUILogLevel_t logLevel, const char ∗logComponent, const char ∗sourceFileName, int lineNo, const char ∗functionName)

Static Public Member Functions

- static std::ostream & debug (const char ∗logComponent, const char ∗sourceFileName, int lineNo, const char ∗functionName)
- static std::ostream & **milestone** (const char ∗logComponent, const char ∗sourceFileName, int lineNo, const char ∗functionName)
- static std::ostream & **warning** (const char ∗logComponent, const char ∗sourceFileName, int lineNo, const char ∗functionName)
- static std::ostream & **error** (const char ∗logComponent, const char ∗sourceFileName, int lineNo, const char ∗functionName)
- static YUILog ∗ instance ()
- static void enableDebugLogging (bool debugLogging=true)
- static bool debugLoggingEnabled ()
- static bool setLogFileName (const std::string &logFileName)
- static std::string logFileName ()
- static void setLoggerFunction (YUILoggerFunction loggerFunction)
- static YUILoggerFunction loggerFunction (bool returnStdLogger=false)
- static void setEnableDebugLoggingHooks (YUIEnableDebugLoggingFunction enableFunction, YUIDebugLoggingEnabledFunction isEnabledFunction)
- static YUIEnableDebugLoggingFunction enableDebugLoggingHook ()
- static YUIDebugLoggingEnabledFunction debugLoggingEnabledHook ()
- static std::string basename (const std::string &fileNameWithPath)

3.158.1 Detailed Description

UI logging.

Definition at line 97 of file YUILog.h.

3.158.2 Member Function Documentation

3.158.2.1 std::string YUILog::basename (const std::string & *fileNameWithPath*) [static]

Return the base name without path from a file name with path.

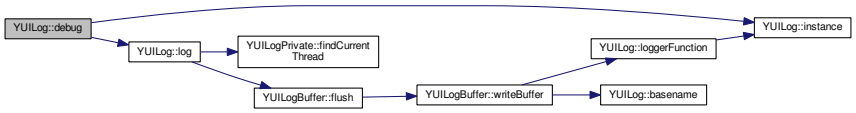
Definition at line 512 of file YUILog.cc.

3.158.2.2 std::ostream & YUILog::debug (const char * *logComponent*, const char * *sourceFileName*, int *lineNo*, const char * *functionName*) [static]

Logging functions for each log level. They all access the singleton object for this class. This means that the first call to any of those functions will create the singleton YUILog object.

Definition at line 483 of file YUILog.cc.

Here is the call graph for this function:



3.158.2.3 bool YUILog::debugLoggingEnabled ( ) [static]

Return 'true' if debug logging is enabled, 'false' if not.

Definition at line 397 of file YUILog.cc.

Here is the call graph for this function:

3.158.2.4 YUIDebugLoggingEnabledFunction YUILog::debugLoggingEnabledHook ( ) [static]

Return the hook function that checks if debug logging is enabled or 0 if no such hook function is set.

Definition at line 445 of file YUILog.cc.

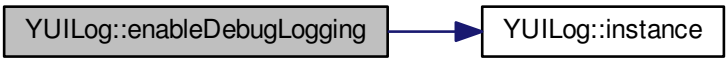
Here is the call graph for this function:

3.158.2.5 void YUILog::enableDebugLogging (bool *debugLogging* = `true`) [static]

Enable or disable debug logging.

Definition at line 387 of file YUILog.cc.

Here is the call graph for this function:



3.158.2.6 YUIEnableDebugLoggingFunction YUILog::enableDebugLoggingHook ( ) [static]

Return the hook function that enables or disables debug logging or 0 if no such hook function is set.

Definition at line 438 of file YUILog.cc.

Here is the call graph for this function:

3.158.2.7 YUILog * YUILog::instance ( ) [static]

Return the singleton object for this class. This will create the singleton if it doesn't exist yet.

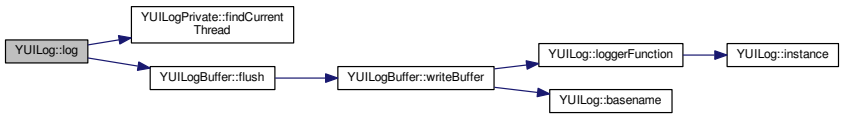
Definition at line 329 of file YUILog.cc.

3.158.2.8 std::ostream & YUILog::log (YUILogLevel_t *logLevel*, const char * *logComponent*, const char * *sourceFileName*, int *lineNo*, const char * *functionName*)

Generic log function. debug(), milestone() etc. ultimately all call this function.

Definition at line 452 of file YUILog.cc.

Here is the call graph for this function:

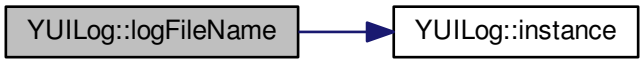


3.158.2.9 std::string YUILog::logFileName ( ) [static]

Return the current log file name or an empty string if stderr is used. Notice that this information is only relevant as long as the standard logger function is used.

Definition at line 380 of file YUILog.cc.

Here is the call graph for this function:



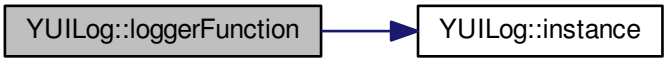
3.158.2.10 YUILoggerFunction YUILog::loggerFunction (bool *returnStdLogger* = false) [static]

Return the UI logger function.

If stderr is used for logging (i.e. no logger function set), 0 is returned (unless 'returnStdLogger' is 'true', in which case the internally used stderr-logger is returned).

Definition at line 417 of file YUILog.cc.

Here is the call graph for this function:



3.158.2.11 void YUILog::setEnableDebugLoggingHooks (YUIEnableDebugLoggingFunction *enableFunction*, YUIDebugLoggingEnabledFunction *isEnabledFunction*) [static]

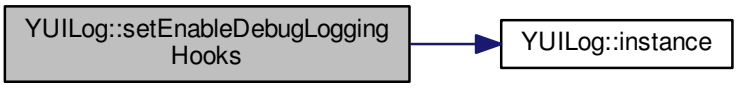
Set the hook functions to enable/disable debug logging and to query if debug logging is enabled:

```
void enableDebugLogging( bool enable );
bool debugLoggingEnabled();
```

If those functions are set, they will be used instead of the internal "debugLogging" flag.

Definition at line 429 of file YUILog.cc.

Here is the call graph for this function:



3.158.2.12 bool YUILog::setLogFileName (const std::string & *logFileName*) [static]

Set the log file name to be used with the standard logger function. Output will be appended to this file.

Until this file name is set, the standard logger function logs to stderr. Set the log file name to an empty string to log to stderr again.

This returns 'true' upon success (opening the file was successful), 'false' upon error.

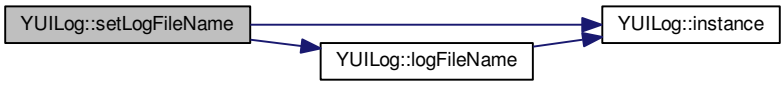
Notice:

(1) This file name is only relevant as long as the standard logger function is used. Custom logger functions may or may not use this file name.

(2) No attempt is made to do anything fancy with the log file like log file rotation when a certain file size is reached. Applications that need this should use a custom logger function. See also setLoggerFunction().

Definition at line 344 of file YUILog.cc.

Here is the call graph for this function:



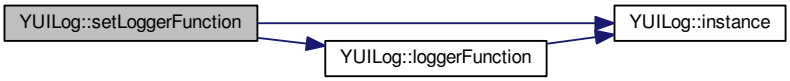
3.158.2.13 void YUILog::setLoggerFunction (YUILoggerFunction *loggerFunction*) [static]

Set the UI logger function. This is the function that will ultimately receive all UI log output (except debug logging if debug logging is disabled).

By default, all logging is output to stderr. This behaviour can be restored if 0 is passed as a function pointer here.

Definition at line 407 of file YUILog.cc.

Here is the call graph for this function:

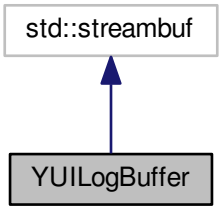


The documentation for this class was generated from the following files:

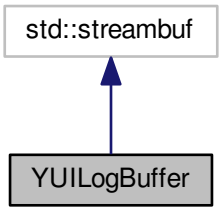
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUILog.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUILog.cc

3.159 YUILogBuffer Class Reference

Inheritance diagram for YUILogBuffer:



Collaboration diagram for YUILogBuffer:



Public Member Functions

- YUILogBuffer ()
- virtual ~YUILogBuffer ()
- virtual std::streamsize xsputn (const char *sequence, std::streamsize maxLength)
- virtual int overflow (int ch=EOF)
- std::streamsize writeBuffer (const char *sequence, std::streamsize seqLen)
- void flush ()

Friends

- class **YUILog**

3.159.1 Detailed Description

Stream buffer class that will use the YUILog's logger function.

See also http://blogs.awesomeplay.com/elanthis/archives/2007/12/10/

Definition at line 54 of file YUILog.cc.

3.159.2 Constructor & Destructor Documentation

3.159.2.1 YUILogBuffer::YUILogBuffer ( ) [inline]

Constructor.

Definition at line 63 of file YUILog.cc.

3.159.2.2 virtual YUILogBuffer::~YUILogBuffer ( ) [inline], [virtual]

Destructor.

Definition at line 73 of file YUILog.cc.

Here is the call graph for this function:

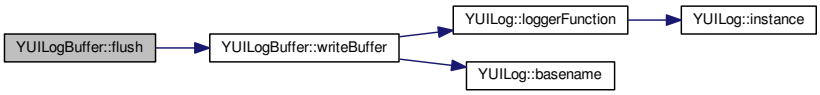
3.159.3 Member Function Documentation

3.159.3.1 void YUILogBuffer::flush ( )

Flush the output buffer: Write any data unwritten so far.

Definition at line 178 of file YUILog.cc.

Here is the call graph for this function:



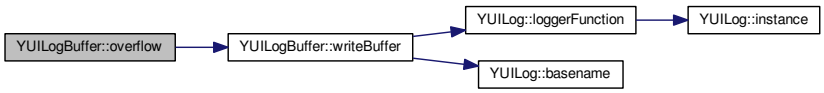
3.159.3.2 int YUILogBuffer::overflow (int *ch* = EOF) [virtual]

Write one character in case of buffer overflow.

Reimplemented from std::streambuf.

Definition at line 166 of file YUILog.cc.

Here is the call graph for this function:



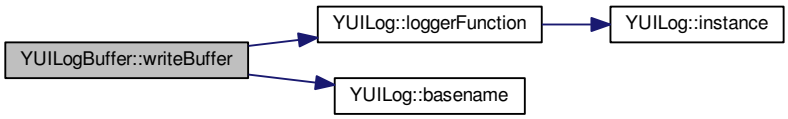
3.159.3.3 std::streamsize YUILogBuffer::writeBuffer (const char * *sequence*, std::streamsize *seqLen*)

Write (no more than maxLength characters of) a sequence of characters and return the number of characters written.

This is the actual worker function that uses the YUILog::loggerFunction to actually write characters.

Definition at line 121 of file YUILog.cc.

Here is the call graph for this function:



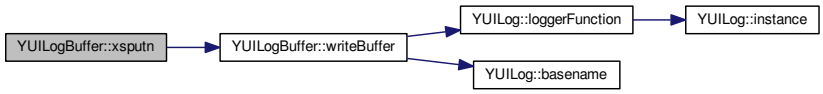
3.159.3.4 std::streamsize YUILogBuffer::xsputn (const char * *sequence*, std::streamsize *maxLength*) [virtual]

Write (no more than maxLength characters of) a sequence of characters and return the number of characters written.

Reimplemented from std::streambuf. This is called for all output operations on the associated ostream.

Definition at line 159 of file YUILog.cc.

Here is the call graph for this function:



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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUILog.cc

3.160 YUILogPrivate Struct Reference

Public Member Functions

- YUILogPrivate ()
- ∼YUILogPrivate ()
- YPerThreadLogInfo ∗ findCurrentThread ()

Public Attributes

- std::string **logFileName**
- std::ofstream **stdLogStream**
- YUILoggerFunction **loggerFunction**
- YUIEnableDebugLoggingFunction **enableDebugLoggingHook**
- YUIDebugLoggingEnabledFunction **debugLoggingEnabledHook**
- bool **enableDebugLogging**
- std::vector< YPerThreadLogInfo ∗ > **threadLogInfo**

3.160.1 Detailed Description

Definition at line 250 of file YUILog.cc.

3.160.2 Constructor & Destructor Documentation

3.160.2.1 YUILogPrivate::YUILogPrivate ( ) [inline]

Constructor

Definition at line 255 of file YUILog.cc.

3.160.2.2 YUILogPrivate::∼YUILogPrivate ( ) [inline]

Destructor

Definition at line 265 of file YUILog.cc.

3.160.3 Member Function Documentation

3.160.3.1 YPerThreadLogInfo∗ YUILogPrivate::findCurrentThread ( ) [inline]

Find the per-thread logging information for the current thread. Create a new one if it doesn't exist yet.

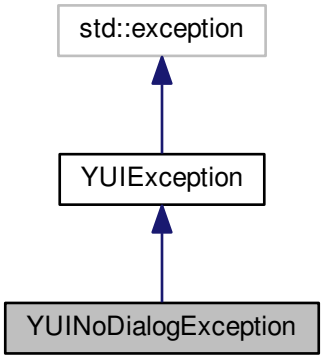
Definition at line 275 of file YUILog.cc.

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

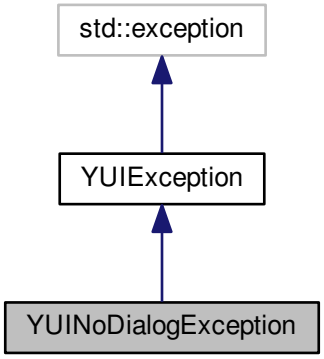
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUILog.cc

3.161 YUINoDialogException Class Reference

Inheritance diagram for YUINoDialogException:



Collaboration diagram for YUINoDialogException:



Additional Inherited Members

3.161.1 Detailed Description

Definition at line 451 of file YUIException.h.

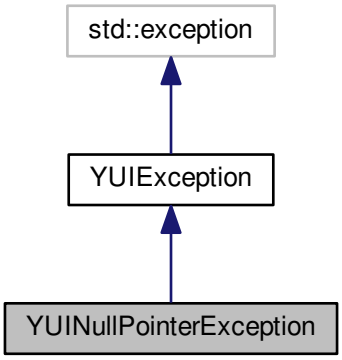
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

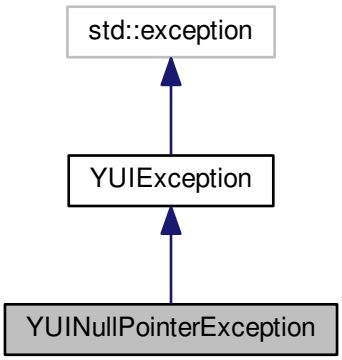
3.162 YUINullPointerException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUINullPointerException:



Collaboration diagram for YUINullPointerException:



Additional Inherited Members

3.162.1 Detailed Description

Exception class for generic null pointer exceptions. When available, a more specialized exception class should be used.

Definition at line 391 of file YUIException.h.

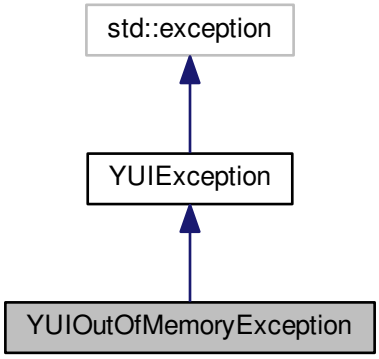
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

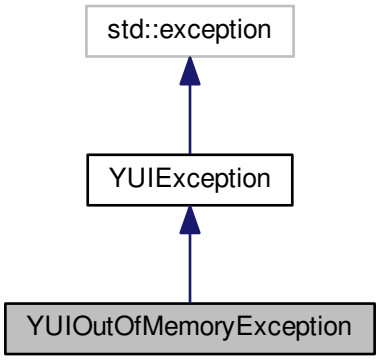
3.163 YUIOutOfMemoryException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIOutOfMemoryException:



Collaboration diagram for YUIOutOfMemoryException:



Additional Inherited Members

3.163.1 Detailed Description

Exception class for "out of memory". Typically used if operator new returned 0.

Definition at line 407 of file YUIException.h.

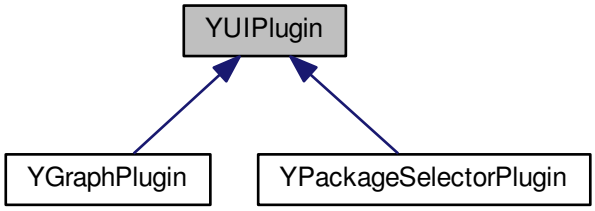
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

3.164 YUIPlugin Class Reference

```
#include <YUIPlugin.h>
```

Inheritance diagram for YUIPlugin:



Public Member Functions

- YUIPlugin (const char *pluginLibBaseName)
- virtual ~YUIPlugin ()
- void unload ()
- void * locateSymbol (const char *symbol)
- bool error () const
- bool success () const
- std::string errorMsg () const

Protected Member Functions

- void * pluginLibHandle ()
- std::string pluginLibBaseName () const
- std::string pluginLibFullPath () const

3.164.1 Detailed Description

Wrapper class for dlopen() and related.

Definition at line 35 of file YUIPlugin.h.

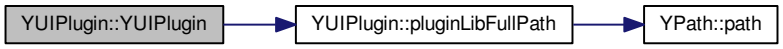
3.164.2 Constructor & Destructor Documentation

3.164.2.1 YUIPlugin::YUIPlugin (const char * *pluginLibBaseName*)

Constructor: Load the specified plugin library from the standard UI plugin directory (/usr/lib/yui/).

Definition at line 37 of file YUIPlugin.cc.

Here is the call graph for this function:



3.164.2.2 YUIPlugin::~YUIPlugin ( ) [virtual]

Destructor.

Please note that this will NOT attempt to unload the plugin library since this is usually counterproductive. If unloading the plugin is desired, call unload() manually.

Definition at line 57 of file YUIPlugin.cc.

3.164.3 Member Function Documentation

3.164.3.1 bool YUIPlugin::error ( ) const

Returns 'true' if there was an error loading the plugin.

Definition at line 104 of file YUIPlugin.cc.

3.164.3.2 std::string YUIPlugin::errorMsg ( ) const

Returns a human readable (but in most cases untranslated) error message if there was an error.

Definition at line 116 of file YUIPlugin.cc.

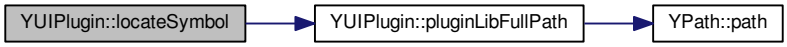
3.164.3.3 void * YUIPlugin::locateSymbol (const char * *symbol*)

Try to locate the specified symbol (function or global variable) in the plugin library.

Returns the in-memory address of that symbol or 0 if it could not be found or if loading the plugin library had failed in the constructor.

Definition at line 86 of file YUIPlugin.cc.

Here is the call graph for this function:



3.164.3.4 std::string YUIPlugin::pluginLibBaseName ( ) const [inline],[protected]

Returns the base name of the plugin library.

Definition at line 96 of file YUIPlugin.h.

3.164.3.5 std::string YUIPlugin::pluginLibFullPath ( ) const [protected]

Returns the full path of the plugin library.

Definition at line 73 of file YUIPlugin.cc.

Here is the call graph for this function:

3.164.3.6 void* YUIPlugin::pluginLibHandle ( ) [inline], [protected]

Returns the dlopen() handle of the plugin library.

Definition at line 91 of file YUIPlugin.h.

3.164.3.7 bool YUIPlugin::success ( ) const

Returns 'true' if there was no error loading the plugin.

Definition at line 110 of file YUIPlugin.cc.

3.164.3.8 void YUIPlugin::unload ( )

Unload this plugin. This calls dlclose() which will unload the plugin library if it is no longer used, i.e. if the reference count dlopen() uses reaches 0.

Definition at line 65 of file YUIPlugin.cc.

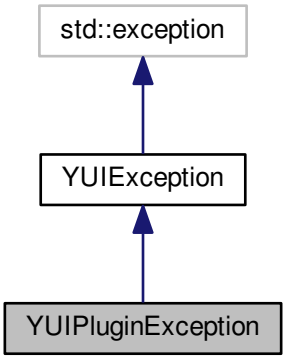
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIPlugin.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIPlugin.cc

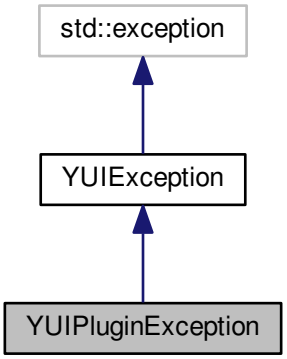
3.165 YUIPluginException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIPluginException:



Collaboration diagram for YUIPluginException:



Public Member Functions

- **YUIPluginException** (const std::string &pluginName)

Additional Inherited Members

3.165.1 Detailed Description

Exception class for plugin load failure

Definition at line 859 of file YUIException.h.

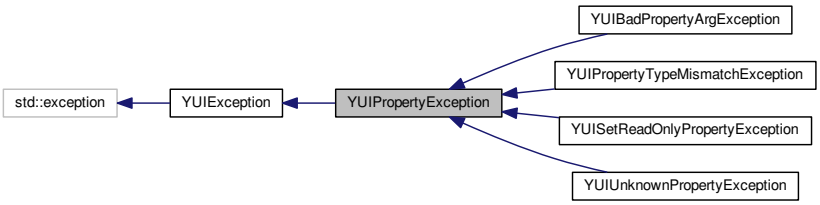
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

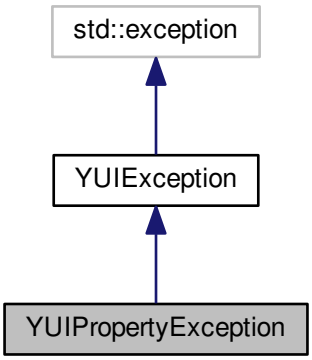
3.166 YUIPropertyException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIPropertyException:



Collaboration diagram for YUIPropertyException:



Public Member Functions

- YProperty property () const
- YWidget * widget () const
- void setWidget (YWidget *w)

Protected Member Functions

- **YUIPropertyException** (const YProperty &prop, YWidget *widget=0)
- virtual std::ostream & dumpOn (std::ostream &str) const =0

Additional Inherited Members

3.166.1 Detailed Description

Abstract base class for widget property exceptions.

Definition at line 490 of file YUIException.h.

3.166.2 Member Function Documentation

3.166.2.1 virtual std::ostream& YUIPropertyException::dumpOn (std::ostream & *str*) const [protected], [pure virtual]

Write proper error message with all relevant data. Reimplemented from YUIException.

Reimplemented from YUIException.

Implemented in YUIBadPropertyArgException, YUISetReadOnlyPropertyException, YUIPropertyTypeMismatchException, and YUIUnknownPropertyException.

3.166.2.2 YProperty YUIPropertyException::property ( ) const [inline]

Returns the property that caused this exception.

Definition at line 507 of file YUIException.h.

3.166.2.3 void YUIPropertyException::setWidget (YWidget * *w*) [inline]

Set the corresponding widget.

Definition at line 517 of file YUIException.h.

3.166.2.4 YWidget* YUIPropertyException::widget ( ) const [inline]

Returns the corresponding widget or 0 if there was none.

Definition at line 512 of file YUIException.h.

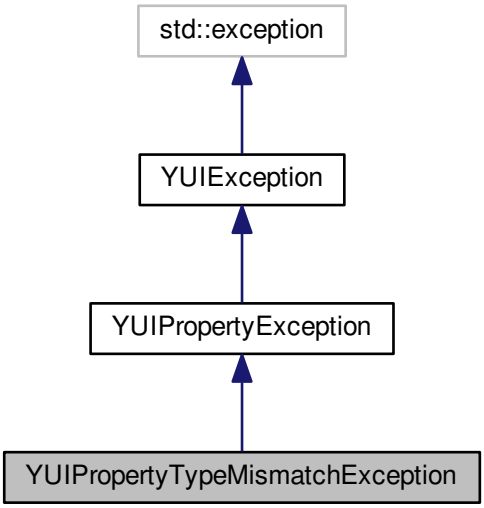
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

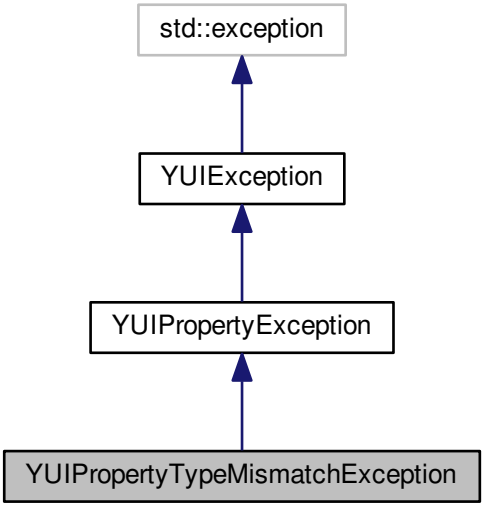
3.167 YUIPropertyTypeMismatchException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIPropertyTypeMismatchException:



Collaboration diagram for YUIPropertyTypeMismatchException:



Public Member Functions

- **YUIPropertyTypeMismatchException** (const YProperty &property, YPropertyType type, YWidget *widget=0)
- YPropertyType type () const

Protected Member Functions

- virtual std::ostream & dumpOn (std::ostream &str) const

Additional Inherited Members

3.167.1 Detailed Description

Exception class for "property type mismatch": The application tried to set a property with a wrong type.

Definition at line 562 of file YUIException.h.

3.167.2 Member Function Documentation

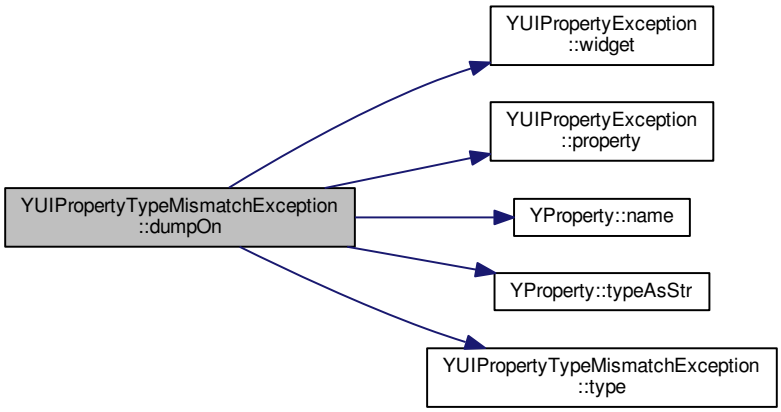
3.167.2.1 std::ostream & YUIPropertyTypeMismatchException::dumpOn (std::ostream & *str*) const [protected], [virtual]

Write proper error message with all relevant data. Reimplemented from YUIException.

Implements YUIPropertyException.

Definition at line 161 of file YUIException.cc.

Here is the call graph for this function:



3.167.2.2 YPropertyType YUIPropertyTypeMismatchException::type ( ) const [inline]

Return the property type the application tried to set.

Definition at line 579 of file YUIException.h.

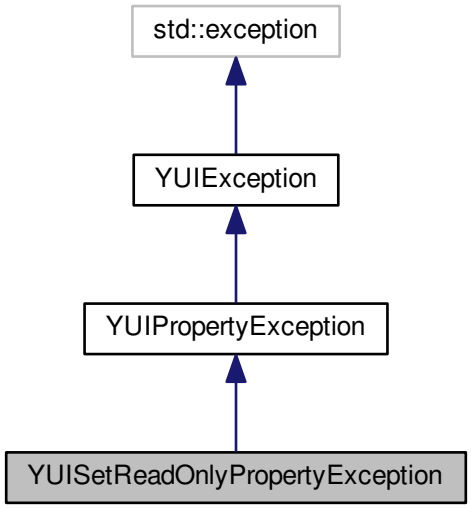
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.cc

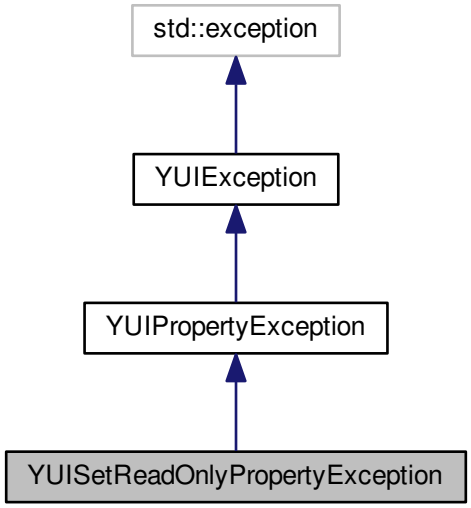
3.168 YUISetReadOnlyPropertyException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUISetReadOnlyPropertyException:



Collaboration diagram for YUISetReadOnlyPropertyException:



Public Member Functions

- **YUISetReadOnlyPropertyException** (const YProperty &property, YWidget *widget=0)

Protected Member Functions

- virtual std::ostream & dumpOn (std::ostream &str) const

Additional Inherited Members

3.168.1 Detailed Description

Exception class for attempt to set a read-only property.

Definition at line 597 of file YUIException.h.

3.168.2 Member Function Documentation

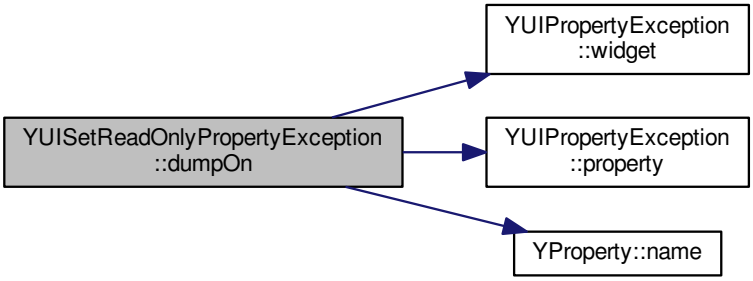
3.168.2.1 std::ostream & YUISetReadOnlyPropertyException::dumpOn (std::ostream & *str*) const [protected], [virtual]

Write proper error message with all relevant data. Reimplemented from YUIException.

Implements YUIPropertyException.

Definition at line 180 of file YUIException.cc.

Here is the call graph for this function:

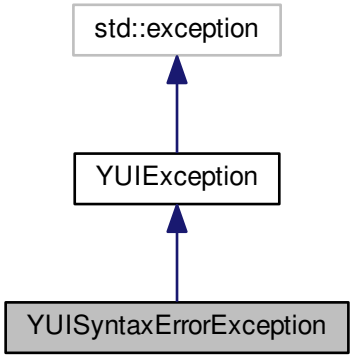


The documentation for this class was generated from the following files:

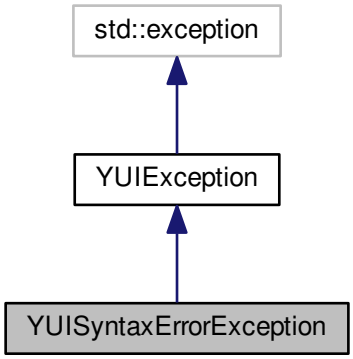
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.cc

3.169 YUISyntaxErrorException Class Reference

Inheritance diagram for YUISyntaxErrorException:



Collaboration diagram for YUISyntaxErrorException:



Public Member Functions

- **YUISyntaxErrorException** (const std::string &msg)

Additional Inherited Members

3.169.1 Detailed Description

Definition at line 475 of file YUIException.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

3.170 YUITerminator Class Reference

Public Member Functions

- ~YUITerminator ()

3.170.1 Detailed Description

Helper class to make sure the UI is properly shut down.

Definition at line 506 of file YUI.cc.

3.170.2 Constructor & Destructor Documentation

3.170.2.1 YUITerminator::~YUITerminator ( )

Destructor.

If there still is a UI, it will be deleted. If there is none, this will do nothing.

Definition at line 521 of file YUI.cc.

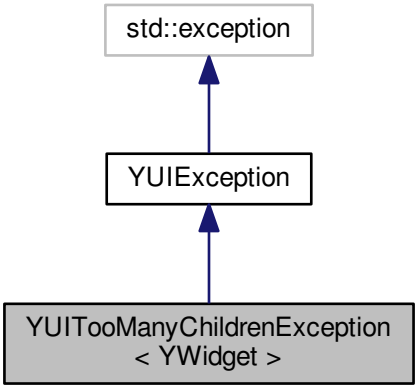
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUI.cc

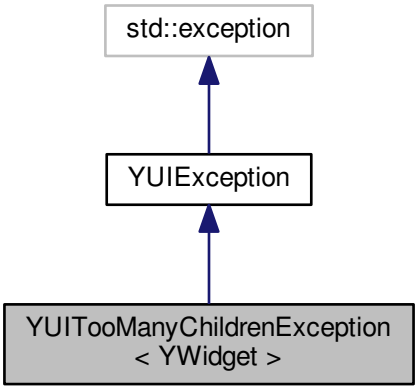
3.171 YUITooManyChildrenException< YWidget > Class Template Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUITooManyChildrenException< YWidget >:



Collaboration diagram for YUITooManyChildrenException< YWidget >:



Public Member Functions

- **YUITooManyChildrenException** (YWidget *container)
- YWidget * container () const

Protected Member Functions

- virtual std::ostream & dumpOn (std::ostream &str) const

Additional Inherited Members

3.171.1 Detailed Description

template<class YWidget>class YUITooManyChildrenException< YWidget >

Exception class for "too many children": Attempt to add a child to a widget class that can't handle children (YPushButton etc.) or just one child (YFrame, YDialog).

Definition at line 647 of file YUIException.h.

3.171.2 Member Function Documentation

3.171.2.1 template<class YWidget > YWidget* YUITooManyChildrenException< YWidget >::container ( ) const [inline]

Returns the container widget that can't handle that many children.

Definition at line 662 of file YUIException.h.

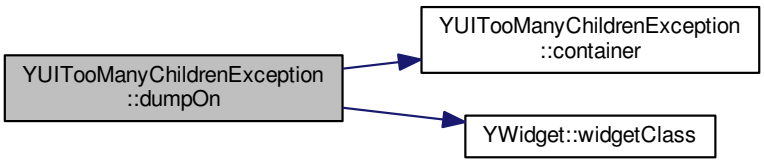
3.171.2.2 template<class YWidget > virtual std::ostream& YUITooManyChildrenException< YWidget >::dumpOn (std::ostream & *str*) const [inline], [protected], [virtual]

Write proper error message with all relevant data. Reimplemented from YUIException.

Reimplemented from YUIException.

Definition at line 670 of file YUIException.h.

Here is the call graph for this function:



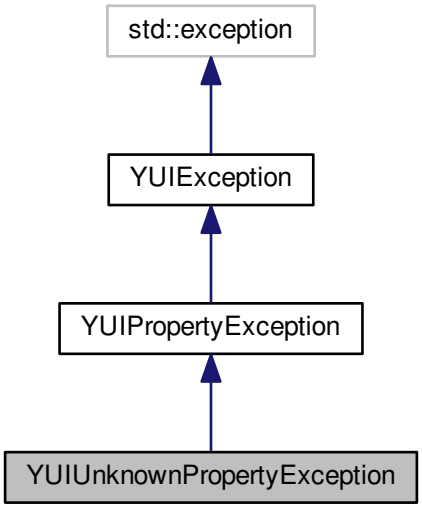
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

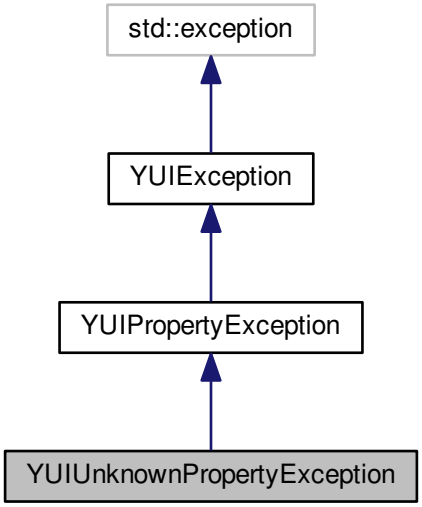
3.172 YUIUnknownPropertyException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIUnknownPropertyException:



Collaboration diagram for YUIUnknownPropertyException:



Public Member Functions

- **YUIUnknownPropertyException** (const std::string &propertyName, YWidget *widget=0)

Protected Member Functions

- virtual std::ostream & dumpOn (std::ostream &str) const

Additional Inherited Members

3.172.1 Detailed Description

Exception class for "unknown property name": The application tried to set (or query) a property that doesn't exist.

Definition at line 537 of file YUIException.h.

3.172.2 Member Function Documentation

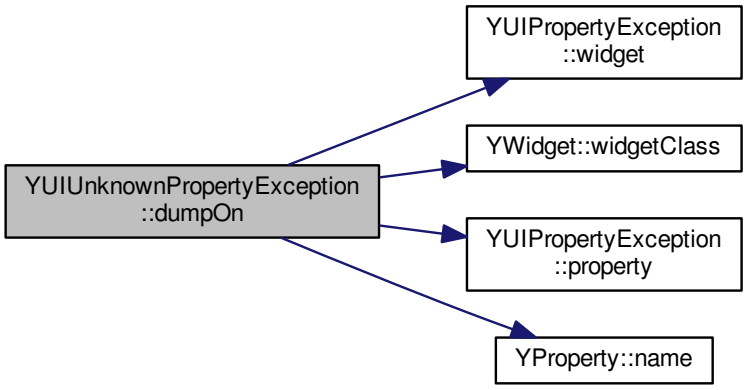
3.172.2.1 std::ostream & YUIUnknownPropertyException::dumpOn (std::ostream & *str*) const [protected], [virtual]

Write proper error message with all relevant data. Reimplemented from YUIException.

Implements YUIPropertyException.

Definition at line 140 of file YUIException.cc.

Here is the call graph for this function:



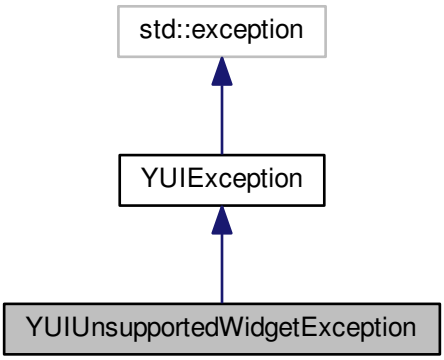
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.cc

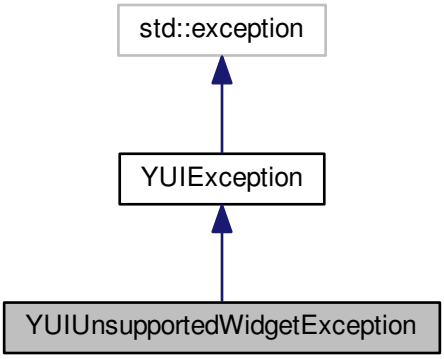
3.173 YUIUnsupportedWidgetException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIUnsupportedWidgetException:



Collaboration diagram for YUIUnsupportedWidgetException:



Public Member Functions

- **YUIUnsupportedWidgetException** (const std::string &widgetType)

Additional Inherited Members

3.173.1 Detailed Description

Exception class for "optional widget not supported".

Note that applications are supposed to check with YUI::optionalWidgetFactory()->hasXYWidget() before trying to create such a widget. This exception is thrown if that check wasn't done, the application tried to create that kind of widget anyway, but the UI doesn't support that widget.

Definition at line 759 of file YUIException.h.

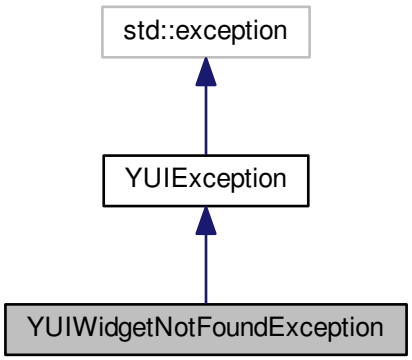
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

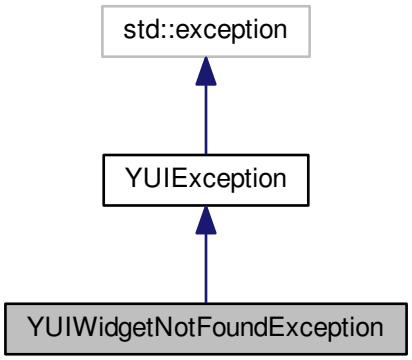
3.174 YUIWidgetNotFoundException Class Reference

```
#include <YUIException.h>
```

Inheritance diagram for YUIWidgetNotFoundException:



Collaboration diagram for YUIWidgetNotFoundException:



Public Member Functions

- **YUIWidgetNotFoundException** (const std::string &idString)

Additional Inherited Members

3.174.1 Detailed Description

Exception class for "No widget found with that ID".

Definition at line 439 of file YUIException.h.

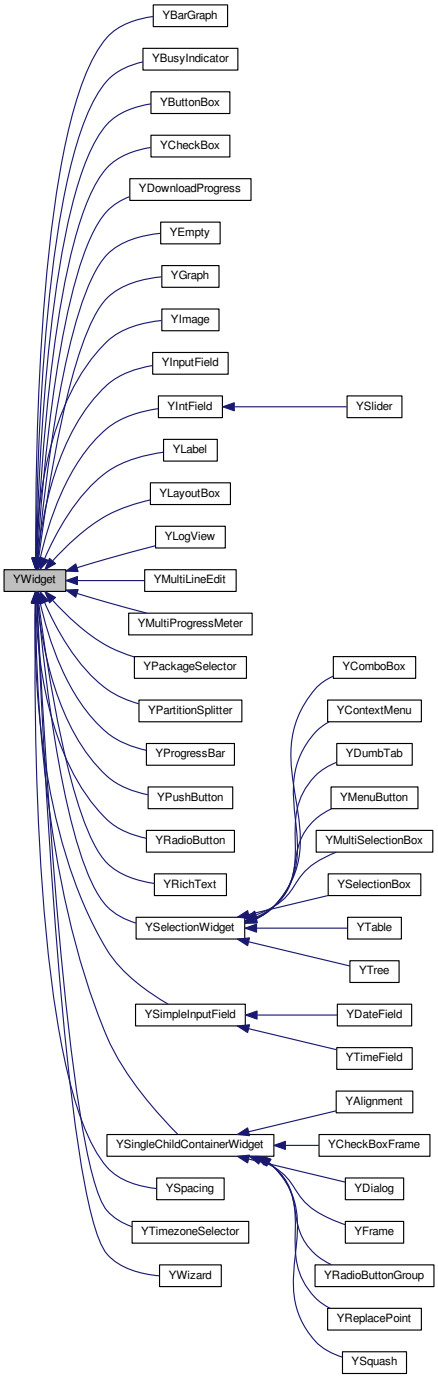
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YUIException.h

3.175 YWidget Class Reference

```
#include <YWidget.h>
```

Inheritance diagram for YWidget:



Classes

- class OptimizeChanges

Public Member Functions

- virtual ~YWidget ()
- virtual const char * widgetClass () const
- virtual std::string debugLabel () const
- std::string helpText () const
- void setHelpText (const std::string &helpText)
- virtual const YPropertySet & propertySet ()
- virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)
- virtual YPropertyValue getProperty (const std::string &propertyName)
- bool hasChildren () const
- YWidget * firstChild () const
- YWidget * lastChild () const
- YWidgetListConstIterator childrenBegin () const
- YWidgetListConstIterator childrenEnd () const
- int childrenCount () const
- bool contains (YWidget *child) const
- virtual void addChild (YWidget *child)
- virtual void removeChild (YWidget *child)
- void deleteChildren ()
- YWidget * parent () const
- bool hasParent () const
- void setParent (YWidget *newParent)
- YDialog * findDialog ()
- YWidget * findWidget (YWidgetID *id, bool doThrow=true) const
- virtual int preferredWidth ()=0
- virtual int preferredHeight ()=0
- virtual int preferredSize (YUIDimension dim)
- virtual void setSize (int newWidth, int newHeight)=0
- bool isValid () const
- bool beingDestroyed () const
- void * widgetRep () const
- void setWidgetRep (void *toolkitWidgetRep)
- bool hasId () const
- YWidgetID * id () const
- void setId (YWidgetID *newId_disown)
- virtual void setEnabled (bool enabled=true)
- void setDisabled ()
- virtual bool isEnabled () const
- virtual bool stretchable (YUIDimension dim) const
- void setStretchable (YUIDimension dim, bool newStretch)
- void setDefaultStretchable (YUIDimension dim, bool newStretch)
- virtual int weight (YUIDimension dim)
- bool hasWeight (YUIDimension dim)
- void setWeight (YUIDimension dim, int weight)
- void setNotify (bool notify=true)
- bool notify () const
- void setNotifyContextMenu (bool notifyContextMenu=true)
- bool notifyContextMenu () const
- bool sendKeyEvents () const
- void setSendKeyEvents (bool doSend)

- bool autoShortcut () const
- void setAutoShortcut (bool _newAutoShortcut)
- int functionKey () const
- bool hasFunctionKey () const
- virtual void setFunctionKey (int fkey_no)
- virtual bool setKeyboardFocus ()
- virtual std::string shortcutString () const
- virtual void setShortcutString (const std::string &str)
- virtual const char * userInputProperty ()
- void dumpWidgetTree (int indentationLevel=0)
- void dumpDialogWidgetTree ()
- void setChildrenEnabled (bool enabled)
- virtual void saveUserInput (YMacroRecorder *macroRecorder)
- void * operator new (size_t size)
- virtual void startMultipleChanges ()
- virtual void **doneMultipleChanges** ()

Protected Member Functions

- YWidget (YWidget *parent)
- YWidgetChildrenManager * childrenManager () const
- void setChildrenManager (YWidgetChildrenManager *manager)
- void setBeingDestroyed ()
- void dumpWidget (YWidget *w, int indentationLevel)

3.175.1 Detailed Description

Abstract base class of all UI widgets

Definition at line 54 of file YWidget.h.

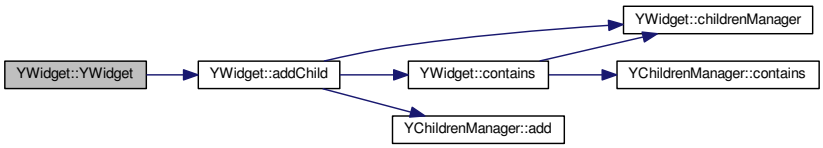
3.175.2 Constructor & Destructor Documentation

3.175.2.1 YWidget::YWidget (YWidget * *parent*) [protected]

Constructor.

Definition at line 104 of file YWidget.cc.

Here is the call graph for this function:

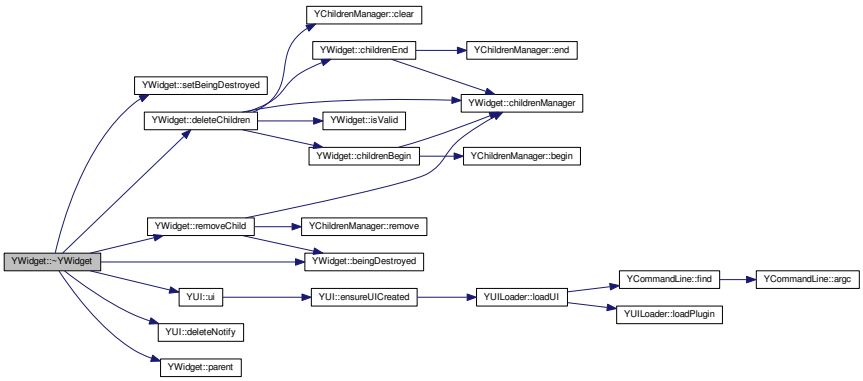


3.175.2.2 YWidget::~YWidget ( ) [virtual]

Destructor.

Definition at line 135 of file YWidget.cc.

Here is the call graph for this function:



3.175.3 Member Function Documentation

3.175.3.1 void YWidget::addChild (YWidget * *child*) [virtual]

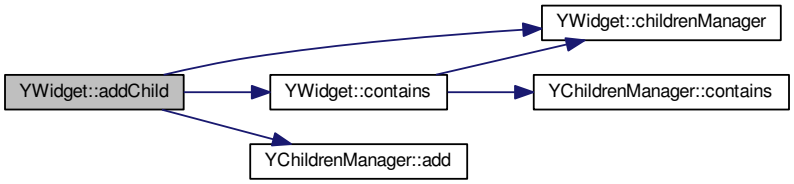
Add a new child.

This may throw exceptions if more children are added than this widget can handle.

Reimplemented in YAlignment.

Definition at line 174 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.2 bool YWidget::autoShortcut ( ) const

Returns 'true' if a keyboard shortcut should automatically be assigned to this widget - without complaints in the log file.

Definition at line 310 of file YWidget.cc.

3.175.3.3 bool YWidget::beingDestroyed ( ) const

Check if this widget is in the process of being destroyed.

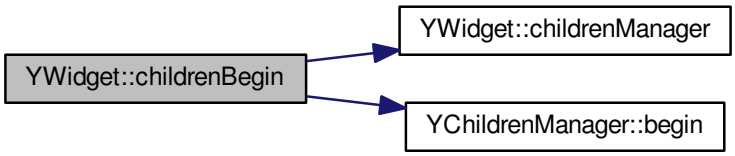
Definition at line 256 of file YWidget.cc.

3.175.3.4 YWidgetListConstIterator YWidget::childrenBegin ( ) const [inline]

Return an iterator that points to the first child or to childrenEnd() if there are no children.

Definition at line 212 of file YWidget.h.

Here is the call graph for this function:

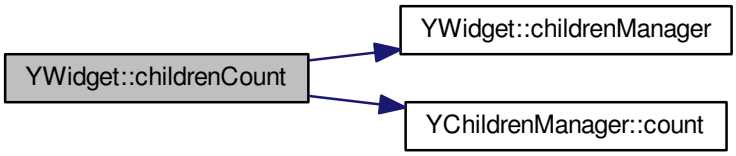


3.175.3.5 int YWidget::childrenCount ( ) const [inline]

Returns the current number of children.

Definition at line 224 of file YWidget.h.

Here is the call graph for this function:

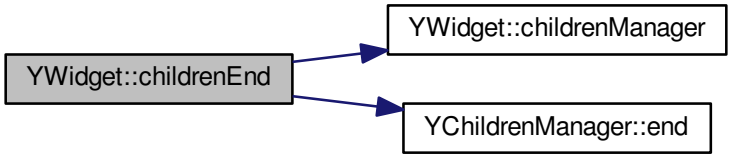


3.175.3.6 YWidgetListConstIterator YWidget::childrenEnd ( ) const [inline]

Return an interator that points after the last child.

Definition at line 218 of file YWidget.h.

Here is the call graph for this function:



3.175.3.7 YWidgetChildrenManager * YWidget::childrenManager ( ) const [protected]

Returns this widget's children manager.

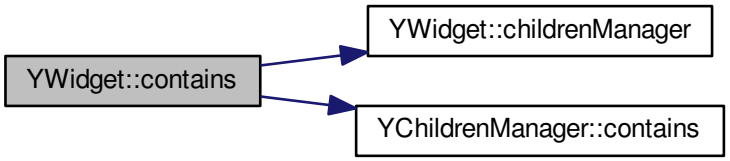
Definition at line 157 of file YWidget.cc.

3.175.3.8 bool YWidget::contains (YWidget * *child*) const [inline]

Checks if 'child' is a (direct!) child of this widget.

Definition at line 229 of file YWidget.h.

Here is the call graph for this function:



3.175.3.9 std::string YWidget::debugLabel ( ) const [virtual]

Returns a descriptive label of this widget instance.

This default implementation returns this widget's "shortcut property" (possibly trunctated to avoid over-long texts) - the property that contains the keyboard shortcut used to activate this widget or to move the keyboard focus to it. In most cases this is this widget's label.

Note: This is usually translated to the user's target language. This makes this useful for debugging only.

Reimplemented in YLabel, and YDumbTab.

Definition at line 221 of file YWidget.cc.

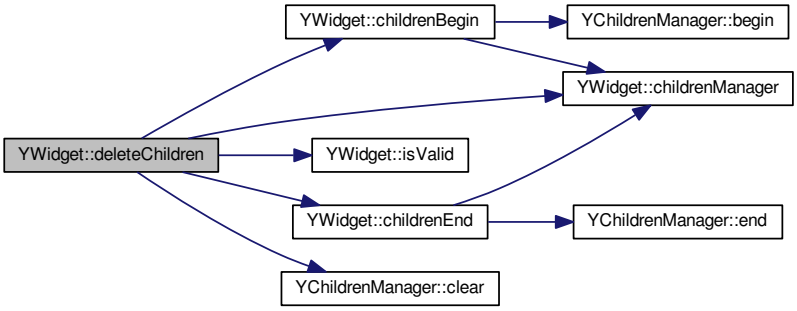
Here is the call graph for this function:

3.175.3.10 void YWidget::deleteChildren ( )

Delete all children and remove them from the children manager's list.

Definition at line 200 of file YWidget.cc.

Here is the call graph for this function:

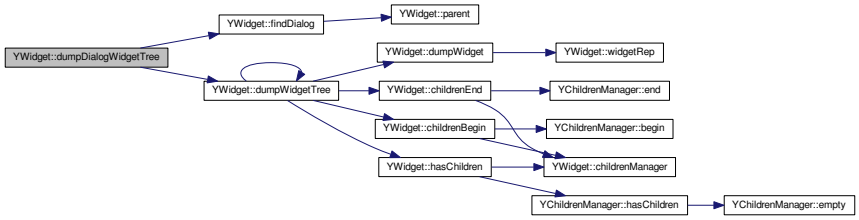


3.175.3.11 void YWidget::dumpDialogWidgetTree ( )

Debugging function: Dump the widget tree from this widget's dialog parent. If there is no such dialog parent, dump the widget tree from here on.

Definition at line 658 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.12 void YWidget::dumpWidget (YWidget * *w*, int *indentationLevel*) [protected]

Helper function for dumpWidgetTree(): Dump one widget to the log file.

Definition at line 687 of file YWidget.cc.

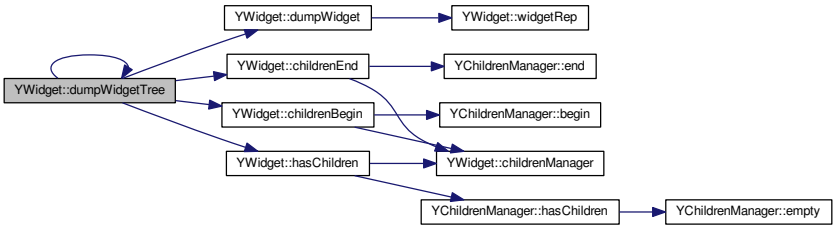
Here is the call graph for this function:

3.175.3.13 void YWidget::dumpWidgetTree (int *indentationLevel* = 0)

Debugging function: Dump the widget tree from here on to the log file.

Definition at line 669 of file YWidget.cc.

Here is the call graph for this function:

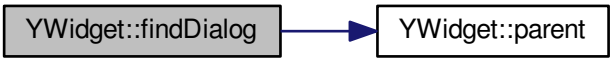


3.175.3.14 YDialog * YWidget::findDialog ( )

Traverse up the widget hierarchy and find the dialog this widget belongs to. Returns 0 if there is none.

Definition at line 374 of file YWidget.cc.

Here is the call graph for this function:

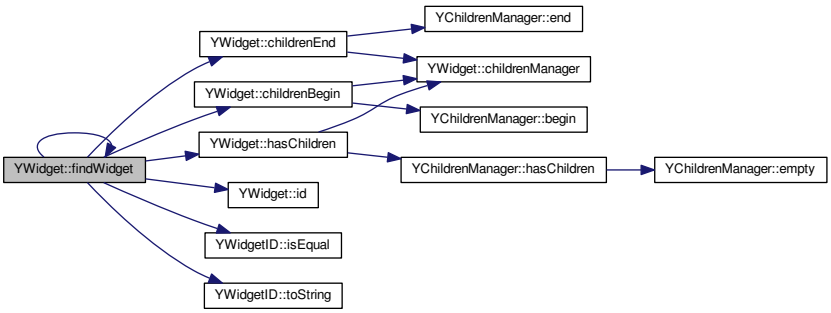


3.175.3.15 YWidget * YWidget::findWidget (YWidgetID * *id*, bool *doThrow* = `true`) const

Recursively find a widget by its ID. If there is no widget with that ID, this function throws a YUIWidgetNotFoundException if 'doThrow' is 'true'. It returns 0 if 'doThrow' is 'false'.

Definition at line 602 of file YWidget.cc.

Here is the call graph for this function:

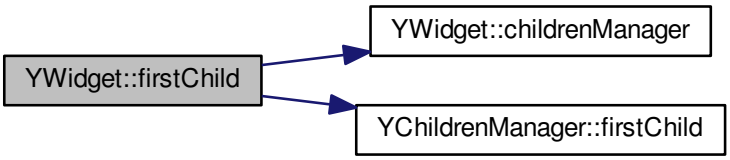


3.175.3.16 YWidget* YWidget::firstChild ( ) const [inline]

Returns the first child or 0 if there is none. Useful mostly for children managers that handle only one child.

Definition at line 199 of file YWidget.h.

Here is the call graph for this function:



3.175.3.17 int YWidget::functionKey ( ) const

Return a function key number that is assigned to this widget. (1 for F1, 2 for F2, etc.; 0 for none)

Definition at line 322 of file YWidget.cc.

3.175.3.18 YPropertyValue YWidget::getProperty (const std::string & *propertyName*) [virtual]

Get a property. Derived classes need to implement this.

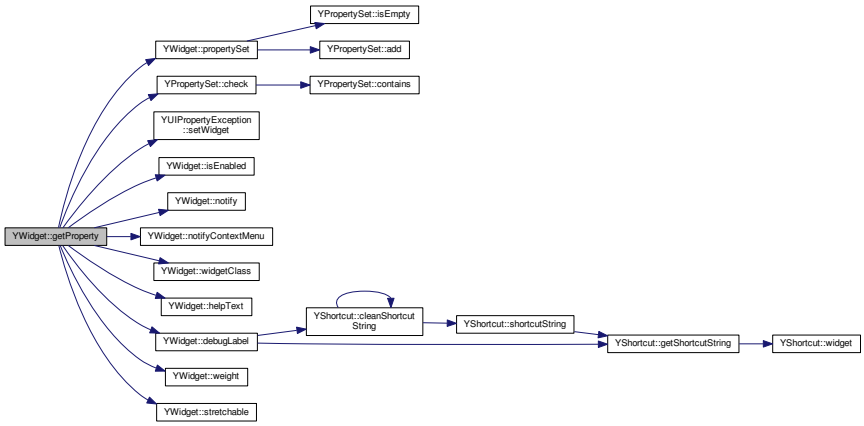
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented in YWizard, YComboBox, YTable, YInputField, YPushButton, YCheckBoxFrame, YPartitionSplitter, YCheckBox, YLogView, YIntField, YRadioButton, YMultiProgressMeter, YLabel, YRichText, YBarGraph, YMultiLineEdit, YDownloadProgress, YTree, YContextMenu, YMenuButton, YSelectionBox, YBusyIndicator, YRadioButtonGroup, YProgressBar, YDumbTab, YSimpleInputField, YGraph, YFrame, YMultiSelectionBox, and YTimezoneSelector.

Definition at line 453 of file YWidget.cc.

Here is the call graph for this function:

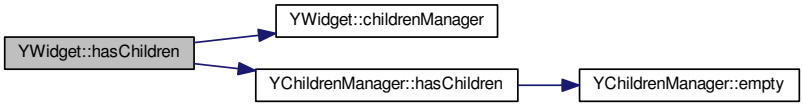


3.175.3.19 bool YWidget::hasChildren ( ) const [inline]

Returns 'true' if this widget has any children.

Definition at line 192 of file YWidget.h.

Here is the call graph for this function:



3.175.3.20 bool YWidget::hasFunctionKey ( ) const

Check if a function key is assigned to this widget.

Definition at line 328 of file YWidget.cc.

3.175.3.21 bool YWidget::hasId ( ) const

Returns 'true' if this widget has an ID.

Definition at line 368 of file YWidget.cc.

3.175.3.22 bool YWidget::hasParent ( ) const

Return 'true' if this widget has a parent, 'false' if not.

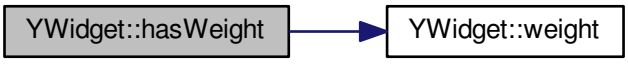
Definition at line 276 of file YWidget.cc.

3.175.3.23 bool YWidget::hasWeight (YUIDimension *dim*)

Return whether or not the widget has a weight in the specified dimension.

Definition at line 585 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.24 std::string YWidget::helpText ( ) const

Return the help text for this widget.

Definition at line 340 of file YWidget.cc.

3.175.3.25 YWidgetID * YWidget::id ( ) const

Returns this widget's ID.

Definition at line 353 of file YWidget.cc.

3.175.3.26 bool YWidget::isEnabled ( ) const [virtual]

Returns 'true' if this widget is enabled.

Definition at line 502 of file YWidget.cc.

3.175.3.27 bool YWidget::isValid ( ) const

Checks whether or not this object is valid. This is to enable dangling pointer error checking (i.e. this object is already deallocated, but a pointer to it is still in use).

See also the YUI_CHECK_WIDGET() macro in YUIException.h

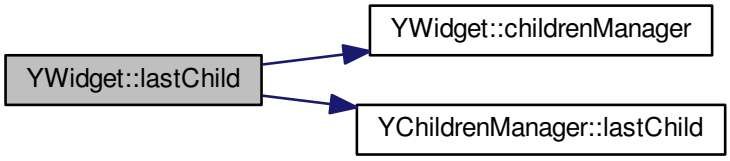
Definition at line 242 of file YWidget.cc.

3.175.3.28 YWidget* YWidget::lastChild ( ) const [inline]

Returns the last child or 0 if there is none.

Definition at line 205 of file YWidget.h.

Here is the call graph for this function:



3.175.3.29 bool YWidget::notify ( ) const

Returns whether the widget will notify, i.e. will case UserInput to return.

Definition at line 529 of file YWidget.cc.

3.175.3.30 bool YWidget::notifyContextMenu ( ) const

Returns whether the widget will send an event when the user clicks selects the context menu e.g. via right click.

Definition at line 535 of file YWidget.cc.

3.175.3.31 void * YWidget::operator new (size_t *size*)

Overloaded operator new to ensure widgets are always created on the heap, never on the stack.

Simpler implementations of this have a tendency to be fooled by poorly implemented derived classes.

Definition at line 128 of file YWidget.cc.

3.175.3.32 YWidget * YWidget::parent ( ) const

Return this widget's parent or 0 if it doesn't have a parent.

Definition at line 269 of file YWidget.cc.

3.175.3.33 virtual int YWidget::preferredHeight ( ) [pure virtual]

Preferred height of the widget.

Derived classes are required to implement this.

Implemented in YButtonBox, YAlignment, YSpacing, YLayoutBox, YEmpty, and YSingleChildContainerWidget.

3.175.3.34 int YWidget::preferredSize (YUIDimension *dim*) [virtual]

Preferred size of the widget in the specified dimension. This default implementation calls preferredWidth() or preferredHeight() which makes sense for most cases.

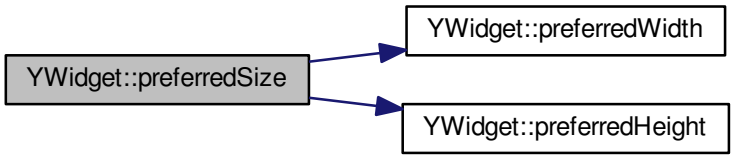
Derived classes can reimplement this, but this is discouraged.

Note: Even in that case, preferredWidth() and preferredHeight() need to be implemented, but they might then call preferredSize().

Reimplemented in YLayoutBox.

Definition at line 541 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.35 virtual int YWidget::preferredWidth ( ) [pure virtual]

Preferred width of the widget.

Derived classes are required to implement this.

Implemented in YButtonBox, YAlignment, YSpacing, YLayoutBox, YEmpty, and YSingleChildContainerWidget.

3.175.3.36 const YPropertySet & YWidget::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Derived classes should reimplement this.

Remember to add the base class's property set to your own in reimplemented versions, e.g.:

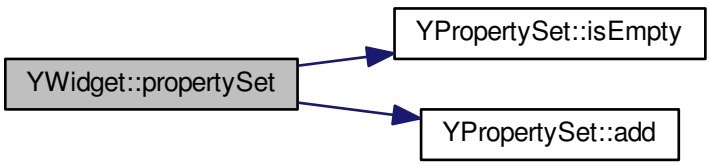
```
const YPropertySet &
MyWidgetClass::propertySet()
{
    static YPropertySet propSet;

    if ( propSet.isEmpty() )
    {
        // Add properties for the derived class
       propSet.add( YProperty( YUIProperty_Value, YStringProperty ) );
       propSet.add( YProperty( YUIProperty_Label, YStringProperty ) );

        // Add base class properties
       propSet.add( YWidget::propertySet() );
    }

    return propSet;
}
```

Otherwise the base class's properties will not be available in the derived class. It is also important that the base class's properties are added after those of the derived class so the derived class's properties have priority over those of the base class.

Reimplemented in YWizard, YComboBox, YTable, YInputField, YPushButton, YCheckBoxFrame, YPartitionSplitter, YCheckBox, YLogView, YIntField, YRadioButton, YMultiProgressMeter, YLabel, YRichText, YBarGraph, YMultiLineEdit, YDownloadProgress, YTree, YContextMenu, YMenuButton, YSelectionBox, YBusyIndicator, YRadioButtonGroup, YProgressBar, YDumbTab, YSimpleInputField, YGraph, YFrame, YMultiSelectionBox, and YTimezoneSelector.

Definition at line 393 of file YWidget.cc.

Here is the call graph for this function:

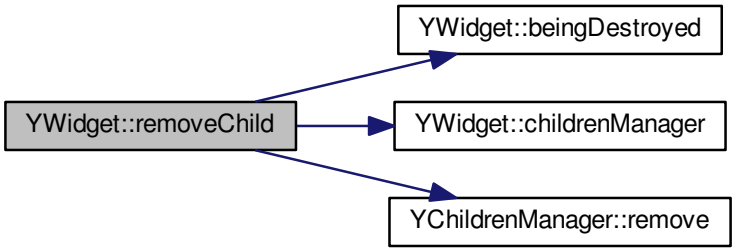


3.175.3.37 void YWidget::removeChild (YWidget * *child*) [virtual]

Remove a child. This only removes the child from the children manager's list; it does not delete it.

Definition at line 189 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.38 void YWidget::saveUserInput (YMacroRecorder * *macroRecorder*) [virtual]

Recursively save the user input of all child widgets to a macro recorder:

All child widgets that could contain data entered by the user are requested to send their contents to the macro recorder, e.g. input fields, check boxes etc.

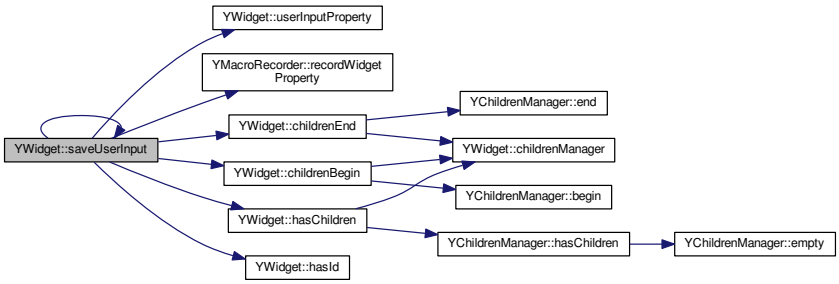
This default implementation records this widget's user input property (the property returned by userInputProperty) and then recursively calls saveUserInput() for all child widgets. This is suitable for most cases, for container widgets as well as for leaf widgets that have no or exactly one property that needs to be recorded.

Widgets that need another number of properties recorded should reimplement this method (and NOT call this default method in the new implementation).

Reimplemented in YInputField, YRadioButton, and YMultiSelectionBox.

Definition at line 714 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.39 bool YWidget::sendKeyEvents ( ) const

Returns 'true' if this widget should send key events, i.e. if it has `opt` (keyEvent) set.

Definition at line 298 of file YWidget.cc.

3.175.3.40 void YWidget::setAutoShortcut (bool *_newAutoShortcut*)

Sets the 'autoShortcut' flag.

Definition at line 316 of file YWidget.cc.

3.175.3.41 void YWidget::setBeingDestroyed ( ) [protected]

Set the "being destroyed" flag, i.e. indicate that this widget is in the process of being destroyed. The base class method already sets this, but sometimes it might be useful to call this in a derived class's destructor so certain optimizations work better.

This status intentionally cannot be reverted to "not being destroyed".

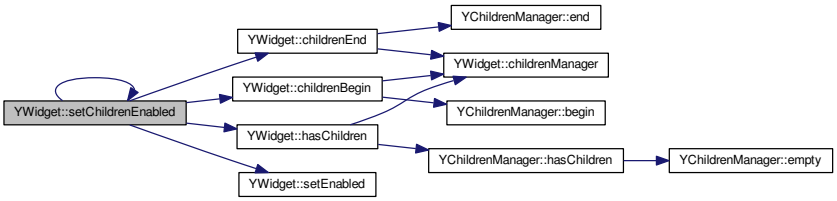
Definition at line 262 of file YWidget.cc.

3.175.3.42 void YWidget::setChildrenEnabled (bool *enabled*)

Enable or disable all widgets in this widget tree.

Definition at line 638 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.43 void YWidget::setChildrenManager (**YWidgetChildrenManager** * *manager*) [protected]

Sets a new children manager for this widget. The widget assumes ownership of this children manager and will delete it when appropriate.

The default children manager (a YWidgetChildrenRejector) rejects all children. This is useful for leaf widgets such as PushButton, ComboBox etc.

Derived classes that can handle children might want to set the children manager to a YWidgetChildrenManager (the base class that does not reject children) or to a YSingleWidgetChildManager (the class that handles exactly one child widget).

Definition at line 164 of file YWidget.cc.

3.175.3.44 void YWidget::setDefaultStretchable (YUIDimension *dim*, bool *newStretch*)

Set the stretchable state to "newStretch". `hstretch` or`vstretch` options may override this.

Definition at line 561 of file YWidget.cc.

3.175.3.45 void YWidget::setDisabled ( ) [inline]

Disable this widget (overloaded for better readability).

Definition at line 399 of file YWidget.h.

Here is the call graph for this function:

3.175.3.46 void YWidget::setEnabled (bool *enabled* = `true`) [virtual]

Enable or disable this widget, i.e. make it accept or reject user input.

Derived classes should call the base class method to update the internal "enabled" flag.

Definition at line 495 of file YWidget.cc.

3.175.3.47 void YWidget::setFunctionKey (int *fkey_no*) [virtual]

Assign a function key to this widget (1 for F1, 2 for F2, etc.; 0 for none)

Derived classes may want to overwrite this function, but they should call this base class function in the new function.

Reimplemented in YPushButton.

Definition at line 334 of file YWidget.cc.

3.175.3.48 void YWidget::setHelpText (const std::string & *helpText*)

Set a help text for this widget.

Currently, the UI does not do anything with this text but store it. Displaying the text at a convenient time is currently the application's responsibility. This may change in future versions.

Definition at line 346 of file YWidget.cc.

Here is the call graph for this function:

3.175.3.49 void YWidget::setId (YWidgetID * *newId_disown*)

Set this widget's ID.

The widget assumes ownership of this ID and will delete it when needed. (In the widget's destructor or when a new ID is set)

Widget IDs are purely for application use. C++ applications don't need to use them; they are much better off using widget pointers. For other languages, though, that can't use C++ pointers (e.g., YCP) it makes sense to have widget IDs to identify widgets.

Definition at line 359 of file YWidget.cc.

3.175.3.50 bool YWidget::setKeyboardFocus ( ) [virtual]

Set the keyboard focus to this widget. The default implementation just emits a warning message. Overwrite this function for all widgets that can accept the keyboard focus.

This function returns true if the widget did accept the keyboard focus, and false if not.

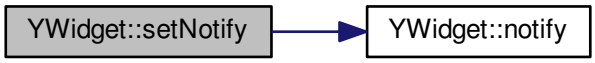
Definition at line 594 of file YWidget.cc.

3.175.3.51 void YWidget::setNotify (bool *notify* = true)

Sets the Notify property

Definition at line 517 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.52 void YWidget::setNotifyContextMenu (bool *notifyContextMenu* = `true`)

Sets the notifyContextMenu property

Definition at line 523 of file YWidget.cc.

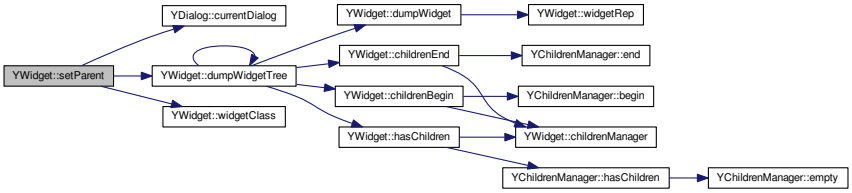
Here is the call graph for this function:

3.175.3.53 void YWidget::setParent (YWidget * *newParent*)

Set this widget's parent.

Definition at line 283 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.54 bool YWidget::setProperty (const std::string & *propertyName*, const YPropertyValue & *val*) `[virtual]`

Set a property. Derived classes need to implement this.

This method may throw exceptions, for example

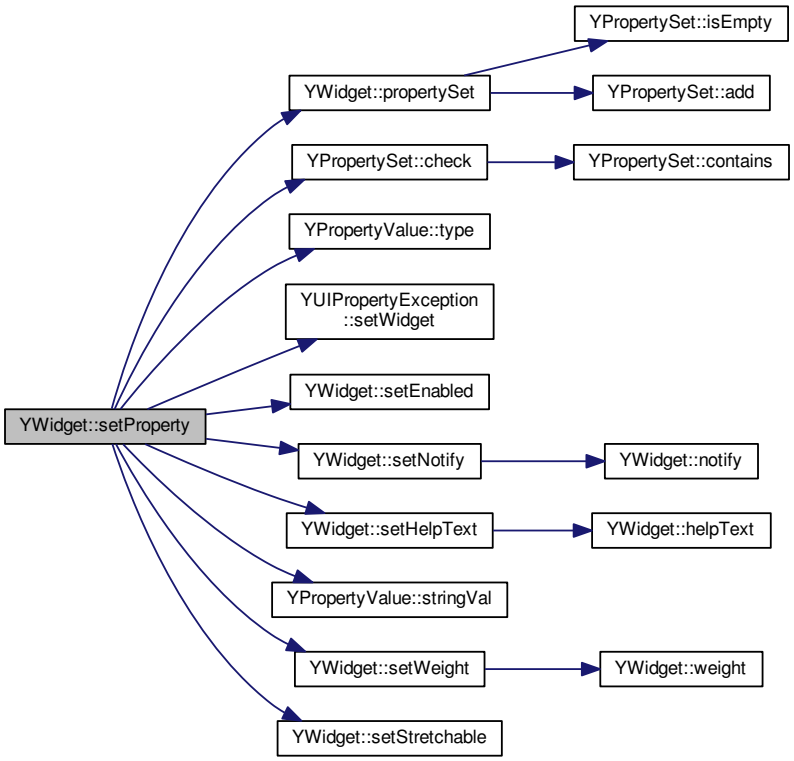
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented in YComboBox, YTable, YInputField, YPushButton, YCheckBoxFrame, YPartitionSplitter, YCheckBox, YLogView, YIntField, YRadioButton, YMultiProgressMeter, YRichText, YLabel, YBarGraph, YMultiLineEdit, YDownloadProgress, YTree, YContextMenu, YMenuButton, YSelectionBox, YBusyIndicator, YRadioButtonGroup, YProgressBar, YDumbTab, YSimpleInputField, YGraph, YFrame, YMultiSelectionBox, and YTimezoneSelector.

Definition at line 428 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.55 void YWidget::setSendKeyEvents (bool *doSend*)

Specify whether or not this widget should send key events.

Definition at line 304 of file YWidget.cc.

3.175.3.56 void YWidget::setShortcutString (const std::string & *str*) [virtual]

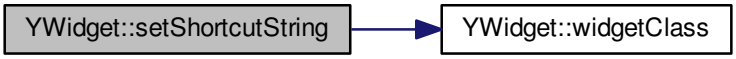
Set the string of this widget that holds the keyboard shortcut, if any. Most widgets will call setLabel().

Overwrite this for widgets that can have keyboard shortcuts.

Reimplemented in YSelectionWidget, YInputField, YPushButton, YCheckBox, YLogView, YIntField, YRadioButton, YMultiLineEdit, YCheckBoxFrame, YDumbTab, and YSimpleInputField.

Definition at line 508 of file YWidget.cc.

Here is the call graph for this function:



3.175.3.57 virtual void YWidget::setSize (int *newWidth*, int *newHeight*) [pure virtual]

Set the new size of the widget.

Layout manager widgets (like YLayoutBox) call this during geometry management after all widgets are queried about their preferred widths and heights. Depending on layout constraints, widgets might be resized beyond or below their preferred size.

The sizes passed here are not meant to affect any future preferredWidth() or preferredHeight() calls; they are just the outcome of all kinds of compromises (too little screen space or too much) for the current geometry management calculation.

Derived classes are required to implement this function.

Implemented in YButtonBox, YAlignment, YLayoutBox, and YSingleChildContainerWidget.

3.175.3.58 void YWidget::setStretchable (YUIDimension *dim*, bool *newStretch*)

Set the stretchable state to "newStretch" regardless of any `hstretch or` vstretch options.

Definition at line 555 of file YWidget.cc.

3.175.3.59 void YWidget::setWeight (YUIDimension *dim*, int *weight*)

Set a weight in the specified dimension.

Definition at line 579 of file YWidget.cc.

Here is the call graph for this function:

3.175.3.60 void YWidget::setWidgetRep (void * *toolkitWidgetRep*)

Set the pointer to the underlying toolkit's (Qt, ...) widget representing this abstract UI widget.

This pointer might be useful for derived UIs to store a counterpart of the toolkit widget in each YWidget. The abstract UI does not need that, though; this is purely for the convenience of derived UIs. All the abstract UI ever does with that pointer is store it.

Definition at line 488 of file YWidget.cc.

3.175.3.61 virtual std::string YWidget::shortcutString ( ) const [inline], [virtual]

Get the string of this widget that holds the keyboard shortcut, if any. Most widgets will return label().

Overwrite this for widgets that can have keyboard shortcuts.

Reimplemented in YSelectionWidget, YInputField, YPushButton, YCheckBox, YLogView, YIntField, YRadioButton, YMultiLineEdit, YCheckBoxFrame, YDumbTab, and YSimpleInputField.

Definition at line 533 of file YWidget.h.

3.175.3.62 virtual void YWidget::startMultipleChanges ( ) [inline], [virtual]

In some UIs updating the screen content is an expensive operation. Use startMultipleChanges() to tell the ui that you're going to perform multiple chages to the widget. The UI may delay any screen updates until doneMultipleChanges() is called.

Definition at line 613 of file YWidget.h.

3.175.3.63 bool YWidget::stretchable (YUIDimension *dim*) const [virtual]

This is a boolean value that determines whether the widget is resizable beyond its preferred size in the specified dimension. A selection box is stretchable in both dimensions, a push button is not stretchable by default, a frame is stretchable if its contents are stretchable. Most widgets accept a `hstretch` or`vstretch` option to become stretchable even when by default they are not.

Reimplemented in YButtonBox, YAlignment, YDumbTab, YLayoutBox, YSquash, and YSingleChildContainerWidget.

Definition at line 567 of file YWidget.cc.

3.175.3.64 virtual const char* YWidget::userInputProperty ( ) [inline], [virtual]

The name of the widget property that will return user input, if there is any. Widgets that do have user input (such as InputField, ComboBox, SelBox) should overwrite this methods. Widgets that are purely passive (such as Label, RichText) should not.

Reimplemented in YComboBox, YInputField, YCheckBox, YTable, YIntField, YRadioButton, YPartitionSplitter, YMultiLineEdit, YTree, YSelectionBox, YCheckBoxFrame, YSimpleInputField, and YMultiSelectionBox.

Definition at line 549 of file YWidget.h.

3.175.3.65 int YWidget::weight (YUIDimension *dim*) [virtual]

The weight is used in situations where all widgets can get their preferred size and yet space is available. The remaining space will be devided between all stretchable widgets according to their weights. A widget with greater weight will get more space. The default weight for all widgets is 0.

Derived classes can overwrite this function, but they should call this base class function in the new function.

Definition at line 573 of file YWidget.cc.

3.175.3.66 virtual const char* YWidget::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented in YButtonBox, YWizard, YPartitionSplitter, YMultiProgressMeter, YRadioButton, YTable, YGraph, YSelectionBox, YComboBox, YTree, YSlider, YContextMenu, YInputField, YLabel, YMenuButton, YSelectionWidget, YDialog, YSpacing, YIntField, YLogView, YDownloadProgress, YSquash, YAlignment, YCheckBox, YDateField, YRichText, YTimezoneSelector, YLayoutBox, YTimeField, YDumbTab, YImage, YBarGraph, YPackageSelector, YRadioButtonGroup, YReplacePoint, YBusyIndicator, YCheckBoxFrame, YEmpty, YFrame, YProgressBar, YPushButton, YMultiLineEdit, and YMultiSelectionBox.

Definition at line 72 of file YWidget.h.

3.175.3.67 void * YWidget::widgetRep ( ) const

Return a pointer to the underlying toolkit's (Qt, ...) widget representing this abstract UI widget.

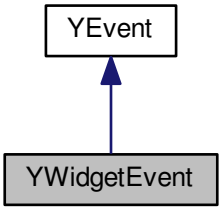
Definition at line 481 of file YWidget.cc.

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

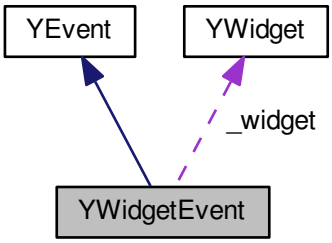
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWidget.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWidget.cc

3.176 YWidgetEvent Class Reference

Inheritance diagram for YWidgetEvent:



Collaboration diagram for YWidgetEvent:



Public Member Functions

- YWidgetEvent (YWidget *widget=0, EventReason reason=Activated, EventType eventType=WidgetEvent)
- virtual YWidget * widget () const
- EventReason reason () const

Protected Member Functions

- virtual ~YWidgetEvent ()

Protected Attributes

- YWidget * **_widget**
- EventReason **_reason**

Additional Inherited Members

3.176.1 Detailed Description

Definition at line 165 of file YEvent.h.

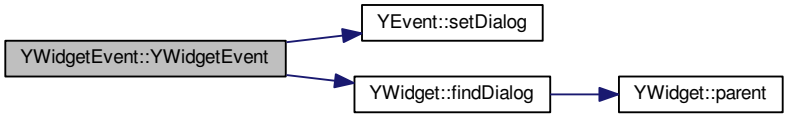
3.176.2 Constructor & Destructor Documentation

3.176.2.1 YWidgetEvent::YWidgetEvent (YWidget * *widget* = `0`, EventReason *reason* = `Activated`, EventType *eventType* = `WidgetEvent`)

Constructor.

Definition at line 110 of file YEvent.cc.

Here is the call graph for this function:



3.176.2.2 virtual YWidgetEvent::~YWidgetEvent ( ) [inline], [protected], [virtual]

Protected destructor - events can only be deleted via YDialog::deleteEvent(). The associated dialog will take care of this event and delete it when appropriate.

Definition at line 194 of file YEvent.h.

3.176.3 Member Function Documentation

3.176.3.1 EventReason YWidgetEvent::reason ( ) const [inline]

Returns the reason for this event. This very much like an event sub-type.

Definition at line 185 of file YEvent.h.

3.176.3.2 virtual YWidget* YWidgetEvent::widget ( ) const [inline], [virtual]

Returns the widget that caused this event. Reimplemented from YEvent.

Reimplemented from YEvent.

Definition at line 180 of file YEvent.h.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEvent.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YEvent.cc

3.177 YWidgetFactory Class Reference

```
#include <YWidgetFactory.h>
```

Public Member Functions

- YDialog * **createMainDialog** (YDialogColorMode colorMode=YDialogNormalColor)
- YDialog * **createPopupDialog** (YDialogColorMode colorMode=YDialogNormalColor)
- virtual YDialog * **createDialog** (YDialogType dialogType, YDialogColorMode colorMode=YDialogNormalColor)=0
- YLayoutBox * **createVBox** (YWidget *parent)

- YLayoutBox * **createHBox** (YWidget *parent)
- virtual YLayoutBox * **createLayoutBox** (YWidget *parent, YUIDimension dimension)=0
- virtual YButtonBox * **createButtonBox** (YWidget *parent)=0
- virtual YPushButton * **createPushButton** (YWidget *parent, const std::string &label)=0
- virtual YLabel * **createLabel** (YWidget *parent, const std::string &text, bool isHeading=false, bool isOutputField=false)=0
- YLabel * **createHeading** (YWidget *parent, const std::string &label)
- virtual YInputField * **createInputField** (YWidget *parent, const std::string &label, bool passwordMode=false)=0
- virtual YCheckBox * **createCheckBox** (YWidget *parent, const std::string &label, bool isChecked=false)=0
- virtual YRadioButton * **createRadioButton** (YWidget *parent, const std::string &label, bool isChecked=false)=0
- virtual YComboBox * **createComboBox** (YWidget *parent, const std::string &label, bool editable=false)=0
- virtual YSelectionBox * **createSelectionBox** (YWidget *parent, const std::string &label)=0
- virtual YTree * **createTree** (YWidget *parent, const std::string &label, bool multiselection=false, bool recursiveselection=false)=0
- virtual YTable * **createTable** (YWidget *parent, YTableHeader *header_disown, bool multiSelection=false)=0
- virtual YProgressBar * **createProgressBar** (YWidget *parent, const std::string &label, int maxValue=100)=0
- virtual YRichText * **createRichText** (YWidget *parent, const std::string &text=std::string(), bool plainTextMode=false)=0
- virtual YBusyIndicator * **createBusyIndicator** (YWidget *parent, const std::string &label, int timeout=1000)=0
- YPushButton * **createIconButton** (YWidget *parent, const std::string &iconName, const std::string &fallbackTextLabel)
- YLabel * **createOutputField** (YWidget *parent, const std::string &label)
- virtual YIntField * **createIntField** (YWidget *parent, const std::string &label, int minVal, int maxVal, int initialVal)=0
- YInputField * **createPasswordField** (YWidget *parent, const std::string &label)
- virtual YMenuButton * **createMenuButton** (YWidget *parent, const std::string &label)=0
- virtual YMultiLineEdit * **createMultiLineEdit** (YWidget *parent, const std::string &label)=0
- virtual YImage * **createImage** (YWidget *parent, const std::string &imageFileName, bool animated=false)=0
- virtual YLogView * **createLogView** (YWidget *parent, const std::string &label, int visibleLines, int storedLines=0)=0
- virtual YMultiSelectionBox * **createMultiSelectionBox** (YWidget *parent, const std::string &label)=0
- virtual YPackageSelector * **createPackageSelector** (YWidget *parent, long ModeFlags=0)=0
- virtual YWidget * **createPkgSpecial** (YWidget *parent, const std::string &subwidgetName)=0
- YSpacing * **createHStretch** (YWidget *parent)
- YSpacing * **createVStretch** (YWidget *parent)
- YSpacing * **createHSpacing** (YWidget *parent, YLayoutSize_t size=1.0)
- YSpacing * **createVSpacing** (YWidget *parent, YLayoutSize_t size=1.0)
- virtual YSpacing * **createSpacing** (YWidget *parent, YUIDimension dim, bool stretchable=false, YLayoutSize_t size=0.0)=0
- virtual YEmpty * **createEmpty** (YWidget *parent)=0
- YAlignment * **createLeft** (YWidget *parent)
- YAlignment * **createRight** (YWidget *parent)
- YAlignment * **createTop** (YWidget *parent)
- YAlignment * **createBottom** (YWidget *parent)
- YAlignment * **createHCenter** (YWidget *parent)
- YAlignment * **createVCenter** (YWidget *parent)
- YAlignment * **createHVCenter** (YWidget *parent)
- YAlignment * **createMarginBox** (YWidget *parent, YLayoutSize_t horMargin, YLayoutSize_t vertMargin)
- YAlignment * **createMarginBox** (YWidget *parent, YLayoutSize_t leftMargin, YLayoutSize_t rightMargin, YLayoutSize_t topMargin, YLayoutSize_t bottomMargin)
- YAlignment * **createMinWidth** (YWidget *parent, YLayoutSize_t minWidth)
- YAlignment * **createMinHeight** (YWidget *parent, YLayoutSize_t minHeight)

- YAlignment * **createMinSize** (YWidget *parent, YLayoutSize_t minWidth, YLayoutSize_t minHeight)
- virtual YAlignment * **createAlignment** (YWidget *parent, YAlignmentType horAlignment, YAlignmentType vertAlignment)=0
- YSquash * **createHSquash** (YWidget *parent)
- YSquash * **createVSquash** (YWidget *parent)
- YSquash * **createHVSquash** (YWidget *parent)
- virtual YSquash * **createSquash** (YWidget *parent, bool horSquash, bool vertSquash)=0
- virtual YFrame * **createFrame** (YWidget *parent, const std::string &label)=0
- virtual YCheckBoxFrame * **createCheckBoxFrame** (YWidget *parent, const std::string &label, bool checked)=0
- virtual YRadioButtonGroup * **createRadioButtonGroup** (YWidget *parent)=0
- virtual YReplacePoint * **createReplacePoint** (YWidget *parent)=0

Protected Member Functions

- YWidgetFactory ()
- virtual ~YWidgetFactory ()

Friends

- class **YUI**

3.177.1 Detailed Description

Abstract widget factory for mandatory widgets. Use YOptionalWidgetFactory for optional ("special") widgets.

Refer to the respective widget's documentation (in the header file) for documentation about the function parameters.

Definition at line 76 of file YWidgetFactory.h.

3.177.2 Constructor & Destructor Documentation

3.177.2.1 YWidgetFactory::YWidgetFactory ( ) [protected]

Constructor.

Use YUI::widgetFactory() to get the singleton for this class.

Definition at line 32 of file YWidgetFactory.cc.

3.177.2.2 YWidgetFactory::~YWidgetFactory ( ) [protected], [virtual]

Destructor.

Definition at line 37 of file YWidgetFactory.cc.

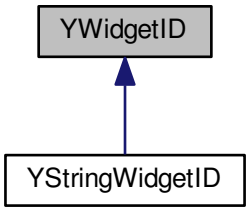
The documentation for this class was generated from the following files:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWidgetFactory.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWidgetFactory.cc

3.178 YWidgetID Class Reference

`#include <YWidgetID.h>`

Inheritance diagram for YWidgetID:



Public Member Functions

- virtual ~YWidgetID ()
- virtual bool isEqual (YWidgetID *otherID) const =0
- virtual std::string toString () const =0

Protected Member Functions

- YWidgetID ()

3.178.1 Detailed Description

Abstract base class for widget IDs.

Definition at line 36 of file YWidgetID.h.

3.178.2 Constructor & Destructor Documentation

3.178.2.1 YWidgetID::YWidgetID ( ) [inline],[protected]

Constructor. Protected since this is an abstract base class.

Definition at line 42 of file YWidgetID.h.

3.178.2.2 virtual YWidgetID::~YWidgetID ( ) [inline],[virtual]

Destructor.

Definition at line 48 of file YWidgetID.h.

3.178.3 Member Function Documentation

3.178.3.1 virtual bool YWidgetID::isEqual (YWidgetID * *otherID*) const [pure virtual]

Check if this ID is equal to another.

Implemented in YStringWidgetID.

3.178.3.2 virtual std::string YWidgetID::toString ( ) const [pure virtual]

Convert the ID value to string. Used for logging and debugging.

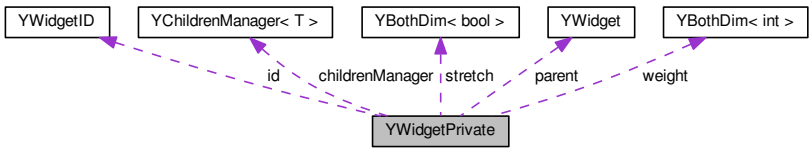
Implemented in YStringWidgetID.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWidgetID.h

3.179 YWidgetPrivate Struct Reference

Collaboration diagram for YWidgetPrivate:



Public Member Functions

- YWidgetPrivate (YWidgetChildrenManager *manager, YWidget *parentWidget=0)

Public Attributes

- YWidgetChildrenManager * **childrenManager**
- YWidget * **parent**
- bool **beingDestroyed**
- bool **enabled**
- bool **notify**
- bool **notifyContextMenu**
- bool **sendKeyEvents**
- bool **autoShortcut**
- void * **toolkitWidgetRep**
- YWidgetID * **id**
- YBothDim< bool > **stretch**
- YBothDim< int > **weight**

- int **functionKey**
- std::string **helpText**

3.179.1 Detailed Description

Definition at line 54 of file YWidget.cc.

3.179.2 Constructor & Destructor Documentation

3.179.2.1 YWidgetPrivate::YWidgetPrivate (YWidgetChildrenManager * *manager,* YWidget * *parentWidget* = 0) [inline]

Constructor

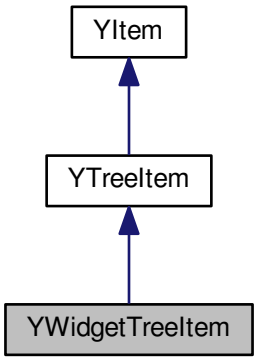
Definition at line 59 of file YWidget.cc.

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

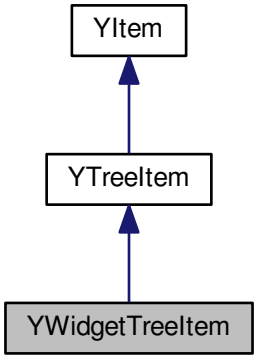
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWidget.cc

3.180 YWidgetTreeItem Class Reference

Inheritance diagram for YWidgetTreeItem:



Collaboration diagram for YWidgetTreeItem:



Public Member Functions

- **YWidgetTreeItem** (YWidget *widget, bool isOpen)
- **YWidgetTreeItem** (YWidgetTreeItem *parent, YWidget *widget, bool isOpen)
- YWidget * **widget** () const

Protected Member Functions

- void **setWidgetLabel** ()

3.180.1 Detailed Description

Custom tree item class to map tree items to widgets

Definition at line 59 of file YDialogSpy.cc.

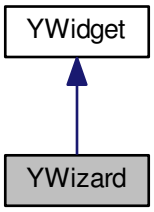
The documentation for this class was generated from the following file:

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YDialogSpy.cc

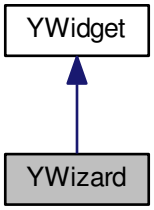
3.181 YWizard Class Reference

```
#include <YWizard.h>
```

Inheritance diagram for YWizard:



Collaboration diagram for YWizard:



Public Member Functions

- virtual ~YWizard ()
- virtual const char * widgetClass () const
- YWizardMode wizardMode () const
- virtual YPushButton * backButton () const =0
- virtual YPushButton * **abortButton** () const =0
- virtual YPushButton * **nextButton** () const =0
- virtual YReplacePoint * contentsReplacePoint () const =0
- void protectNextButton (bool protect)
- bool nextButtonIsProtected () const
- virtual void setButtonLabel (YPushButton *button, const std::string &newLabel)
- virtual void setHelpText (const std::string &helpText)=0
- virtual void setDialogIcon (const std::string &iconName)=0
- virtual void setDialogTitle (const std::string &titleText)=0
- virtual void setDialogHeading (const std::string &headingText)=0
- virtual void addStep (const std::string &text, const std::string &id)=0
- virtual void addStepHeading (const std::string &text)=0

- virtual void deleteSteps ()=0
- virtual void setCurrentStep (const std::string &id)=0
- virtual void updateSteps ()=0
- virtual void addTreeItem (const std::string &parentID, const std::string &text, const std::string &id)=0
- virtual void selectTreeItem (const std::string &id)=0
- virtual std::string currentTreeSelection ()=0
- virtual void deleteTreeItems ()=0
- virtual void addMenu (const std::string &text, const std::string &id)=0
- virtual void addSubMenu (const std::string &parentMenuID, const std::string &text, const std::string &id)=0
- virtual void addMenuEntry (const std::string &parentMenuID, const std::string &text, const std::string &id)=0
- virtual void addMenuSeparator (const std::string &parentMenuID)=0
- virtual void deleteMenus ()=0
- virtual void showReleaseNotesButton (const std::string &label, const std::string &id)=0
- virtual void hideReleaseNotesButton ()=0
- virtual void retranslateInternalButtons ()=0
- void ping ()
- virtual YPropertyValue getProperty (const std::string &propertyName)
- virtual const YPropertySet & propertySet ()

Protected Member Functions

- YWizard (YWidget *parent, const std::string &backButtonLabel, const std::string &abortButtonLabel, const std::string &nextButtonLabel, YWizardMode wizardMode=YWizardMode_Standard)

3.181.1 Detailed Description

A wizard is a more complex frame typically used for multi-step workflows:

+----------+----------+

Side bar	Content Area
	(YReplacePoint)
	[Back] [Abort] [Next]

+----------+----------+

The side bar can contain help text, a list of steps that are performed, or an embedded tree (much like the YTree widget).

The client application creates the wizard and replaces the widget in the content area for each step.

The wizard buttons can theoretically be used to do anything, but good UI design will stick to the model above: [Back], [Abort], [Next].

If only two buttons are desired, leave the [Back] button's label emtpy. The remaining two buttons will be rearranged accordingly in the button area.

In the last step of a multi-step workflow, the [Next] button's label is customarily replaced with a label that indicates that this is the last step. [Accept] is recommended for that button label: [Finish] (as sometimes used in other environments) by no means clearly indicates that this is the positive ending, the final "do it" button. Worse, translations of that are often downright miserable: To German, [Finish] gets translated as [Beenden] which is the same word as "Quit" (used in menus). This does not at all tell the user that that button really performs the requested action the multi-step wizard is all about.

Definition at line 92 of file YWizard.h.

3.181.2 Constructor & Destructor Documentation

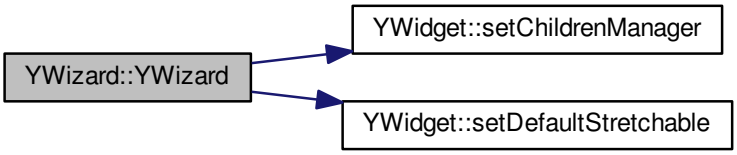
3.181.2.1 YWizard::YWizard (YWidget * *parent*, const std::string & *backButtonLabel*, const std::string & *abortButtonLabel*, const std::string & *nextButtonLabel*, YWizardMode *wizardMode* = `YWizardMode_Standard`) [protected]

Constructor.

If only two buttons are desired, leave 'backButtonLabel' empty.

Definition at line 47 of file YWizard.cc.

Here is the call graph for this function:



3.181.2.2 YWizard::~YWizard ( ) [virtual]

Destructor.

Definition at line 67 of file YWizard.cc.

3.181.3 Member Function Documentation

3.181.3.1 virtual void YWizard::addMenu (const std::string & *text*, const std::string & *id*) [pure virtual]

Add a menu to the menu bar. If the menu bar is not visible yet, it will be made visible. 'text' is the user-visible text for the menu bar (including keyboard shortcuts marked with '&'), 'id' is the menu ID for later addMenuEntry() etc. calls.

3.181.3.2 virtual void YWizard::addMenuEntry (const std::string & *parentMenuID,* const std::string & *text,* const std::string & *id*) [pure virtual]

Add a menu entry to the menu with ID 'parentMenuID'. 'id' is what will be returned by UI::UserInput() etc. when a user activates this menu entry.

3.181.3.3 virtual void YWizard::addMenuSeparator (const std::string & *parentMenuID*) [pure virtual]

Add a menu separator to a menu.

3.181.3.4 virtual void YWizard::addStep (const std::string & *text,* const std::string & *id*) [pure virtual]

Add a step for the steps panel on the side bar. This only adds the step to the internal list of steps. The display is only updated upon calling updateSteps().

3.181.3.5 virtual void YWizard::addStepHeading (const std::string & *text*) [pure virtual]

Add a step heading for the steps panel on the side bar. This only adds the heading to the internal list of steps. The display is only updated upon calling updateSteps().

3.181.3.6 virtual void YWizard::addSubMenu (const std::string & *parentMenuID,* const std::string & *text,* const std::string & *id*) [pure virtual]

Add a submenu to the menu with ID 'parentMenuID'.

3.181.3.7 virtual void YWizard::addTreeItem (const std::string & *parentID,* const std::string & *text,* const std::string & *id*) [pure virtual]

Add a tree item. If "parentID" is an empty string, it will be a root item. 'text' is the text that will be displayed in the tree, 'id' the ID with which this newly created item can be referenced - and that will be returned when the user clicks on a tree item.

3.181.3.8 virtual YPushButton* YWizard::backButton ( ) const [pure virtual]

Return the wizard buttons or 0 if there is no such button.

Derived classes are required to implement this.

3.181.3.9 virtual YReplacePoint* YWizard::contentsReplacePoint ( ) const [pure virtual]

Return the internal contents ReplacePoint.

Derived classes are required to implement this.

3.181.3.10 virtual std::string YWizard::currentTreeSelection ( ) [pure virtual]

Returns the current tree selection or an empty string if nothing is selected or there is no tree.

3.181.3.11 virtual void YWizard::deleteMenus ( ) [pure virtual]

Delete all menus and hide the menu bar.

3.181.3.12 virtual void YWizard::deleteSteps ( ) [pure virtual]

Delete all steps and step headings from the internal lists. The display is only updated upon calling updateSteps().

3.181.3.13 virtual void YWizard::deleteTreeItems ( ) [pure virtual]

Delete all tree items.

3.181.3.14 YPropertyValue YWizard::getProperty (const std::string & *propertyName*) [virtual]

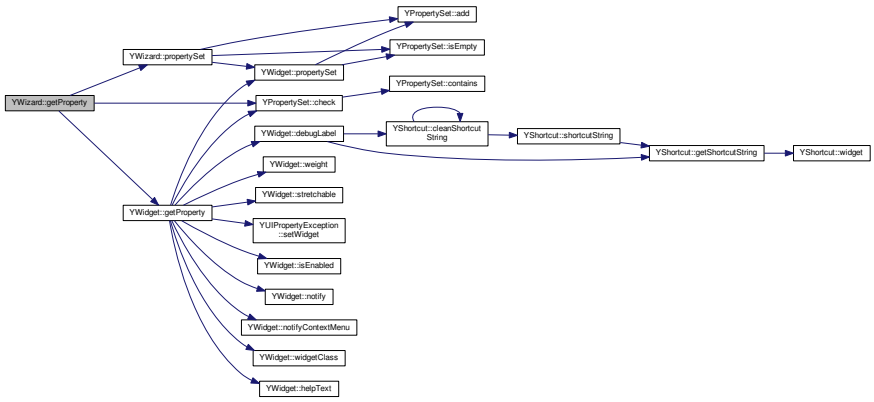
Get a property. Reimplemented from YWidget.

This method may throw YUIPropertyExceptions.

Reimplemented from YWidget.

Definition at line 131 of file YWizard.cc.

Here is the call graph for this function:



3.181.3.15 virtual void YWizard::hideReleaseNotesButton ( ) [pure virtual]

Hide an existing "Release Notes" button.

3.181.3.16 bool YWizard::nextButtonIsProtected ( ) const

Check if the wizard's "Next" button is currently protected against disabling.

Definition at line 80 of file YWizard.cc.

3.181.3.17 void YWizard::ping ( )

NOP command to check if a YWizard is running.

Definition at line 106 of file YWizard.cc.

3.181.3.18 const YPropertySet & YWizard::propertySet ( ) [virtual]

Return this class's property set. This also initializes the property upon the first call.

Reimplemented from YWidget.

Reimplemented from YWidget.

Definition at line 113 of file YWizard.cc.

Here is the call graph for this function:

3.181.3.19 void YWizard::protectNextButton (bool *protect*)

Protect the wizard's "Next" button against disabling.

Definition at line 87 of file YWizard.cc.

3.181.3.20 virtual void YWizard::retranslateInternalButtons ( ) [pure virtual]

Retranslate internal buttons that are not accessible from the outside:

- [Help]
- [Steps]
- [Tree]

3.181.3.21 virtual void YWizard::selectTreeItem (const std::string & *id*) [pure virtual]

Select the tree item with the specified ID, if such an item exists.

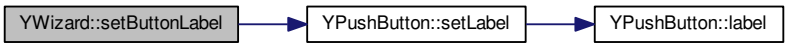
3.181.3.22 void YWizard::setButtonLabel (YPushButton * *button*, const std::string & *newLabel*) [virtual]

Set the label of one of the wizard buttons (backButton(), abortButton(), nextButton()) if that button is non-null.

The default implementation simply calls button->setLabel(newLabel).

Definition at line 94 of file YWizard.cc.

Here is the call graph for this function:



3.181.3.23 virtual void YWizard::setCurrentStep (const std::string & *id*) [pure virtual]

Set the current step. This also triggers updateSteps() if necessary.

3.181.3.24 virtual void YWizard::setDialogHeading (const std::string & *headingText*) [pure virtual]

Set the dialog heading.

3.181.3.25 virtual void YWizard::setDialogIcon (const std::string & *iconName*) [pure virtual]

Set the dialog icon. An empty icon name clears the current icon.

3.181.3.26 virtual void YWizard::setDialogTitle (const std::string & *titleText*) [pure virtual]

Set the dialog title shown in the window manager's title bar. An empty string clears the current title.

3.181.3.27 virtual void YWizard::setHelpText (const std::string & *helpText*) [pure virtual]

Set the help text.

3.181.3.28 virtual void YWizard::showReleaseNotesButton (const std::string & *label,* const std::string & *id*) [pure virtual]

Show a "Release Notes" button above the "Help" button in the steps panel with the specified label that will return the specified id to UI::UserInput() when clicked.

3.181.3.29 virtual void YWizard::updateSteps ( ) [pure virtual]

Update the steps display: Reflect the internal steps and heading lists in the layout.

3.181.3.30 virtual const char* YWizard::widgetClass ( ) const [inline], [virtual]

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from YWidget.

Definition at line 117 of file YWizard.h.

3.181.3.31 YWizardMode YWizard::wizardMode ( ) const

Return the wizard mode (what kind of wizard this is): YWizardMode_Standard, YWizardMode_Steps, YWizardMode_-Tree

Definition at line 74 of file YWizard.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWizard.h
- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWizard.cc

3.182 YWizardPrivate Struct Reference

Public Member Functions

- **YWizardPrivate** (YWizardMode wizardMode)

Public Attributes

- YWizardMode **wizardMode**
- bool **nextButtonIsProtected**

3.182.1 Detailed Description

Definition at line 33 of file YWizard.cc.

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

- /builddir/build/BUILD/libyui-libyui-master-3.0.10/src/YWizard.cc

Index