BlendSplitter

3.0.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

BlendSplitter

A blender-like Qt Widget management library, version 3.

You can download the whole documentation in pdf on (https://genabitu.github.io/BlendSplitter/latex/BlendSplitter.pdf).

This library offers 2 kinds of functionality - one implemented by the BlendSplitter class, the other by the SwitchingWidget class. Although these are intended to be used together, each one of them can be used separately.

BlendSplitter

This widget implements the functionality of Blender (Open-source 3D modelling software) widget management. This widget displays a splitter similar to QSplitter. However, each widget in BlendSplitter has a pair of Expanders (one in top right and one in bottom left corner). By dragging from these Expanders inwards a new widget is created in the direction of the drag. If the direction is different to that of the BlendSplitter, a new BlendSplitter with parallel direction is created in place of the widget with the widget and the new widget in it. By dragging from these expanders outwards, a neighbouring widget (or a collection of widgets) can be closed. While the mouse is held, the widgets to be closed are marked with black overlay. When the mouse is released, they are closed. BlendSplitter can be used like any other QWidget, although setting one as the central widget is recommended. A BlendSplitter can contain objects of any class inheriting from QWidget. Note that you have to manually set the initial state of the BlendSplitter. You need to add at least 1 widget, otherwise nothing will be displayed.

BlendSplitter provides 3 static variables that allow some customization of the library design. These are expanderSize, switchingBarHeight and expanderImage. These are all initialized with default values. The default Expander image is provided by the library.

The default Expander image:

Example

```
{C++}
#include <QApplication>
#include <QMainWindow>

#include <BlendSplitter>

int main(int argc, char** argv)
{
    new QApplication{argc, argv};
     QMainWindow* window{new QMainWindow{}};
     BlendSplitter* splitter{new BlendSplitter{[]()->QWidget* {return new QLabel{"My Widget"};}}};

    window->setCentralWidget(splitter);
    window->resize(400, 200);
    window->setWindowTitle("BlendSplitter example");

    splitter->addWidget();

    window->show();
    return qApp->exec();
}
```

On Gnome 3.22, this example looks like:

Example with composite splitters

```
{C++}
#include <QApplication>
#include <QMainWindow>

#include <BlendSplitter>

int main(int argc, char** argv)
{
    new QApplication{argc, argv};
     QMainWindow* window{new QMainWindow{}};
     BlendSplitter* splitter{new BlendSplitter{[]()->QWidget* {return new QLabel{"My Widget"};}}};

     window->setCentralWidget(splitter);
     window->resize(400, 200);
     window->setWindowTitle("BlendSplitter example 2");

     splitter->addWidget();
     BlendSplitter* splitter2{new BlendSplitter{[]()->QWidget* {return new QLabel{"My Widget"};},
     Qt::Vertical}};
     splitter->addSplitter(splitter2);
     splitter2->addWidget();
     splitter2->addWidget();

     window->show();
     return qApp->exec();
}
```

On Gnome 3.22, this example looks like:

SwitchingWidget

This class displays a Widget with a SwitchingBar on the bottom. The widget displayed is one from WidgetRegistry and it can be selected using a combo box in the SwitchingBar. The SwitchingBar is like a QMenuBar, but can also contain plain widgets. A SwitchingWidget can contain objects of any class inheriting from QWidget.

Note that constructing an object of this class when WidgetRegistry is empty will cause a default RegistryItem to be added to it. The height of the SwitchingBar can be modified by changing BlendSplitter::switchingBarHeight.

Example

```
{C++}
#include <QApplication>
#include <QMainWindow>

#include <BlendSplitter>

int main(int argc, char** argv)
{
    new QApplication{argc, argv};
     QMainWindow* window{new QMainWindow{}};

    WidgetRegistry::getRegistry()->addItem();
    WidgetRegistry::getRegistry()->addItem("Type1", []()->QWidget* {return new QLabel{"Type 1 Label"};}, [](SwitchingBar* bar, QWidget*)->void {
        QMenu* menu{new QMenu{"My first menu"}};
        bar->addMenu(menu);
        QMenu* menu2{new QMenu{"My second menu"}};
        menu2->addAction(new QAction{"New", 0});
        menu2->addAction(new QAction{"Close", 0});
        bar->addMenu(menu2);
        QLabel* lab{new QLabel{"My third not-so-menu"}};
        bar->addWidget(lab);
    });
    WidgetRegistry::getRegistry()->addItem(new RegistryItem{"Type2", []()->QWidget* {return new QLabel{"Type 2 Label"};}});
    WidgetRegistry::getRegistry()->setDefault(1);

    SwitchingWidget* widget{new SwitchingWidget{}};

    window->setCentralWidget(widget);
    window->resize(600, 400);
    window->setWindowTitle("SwitchingWidget example");

    window->show();
    return qApp->exec();
}
```

On Gnome 3.22, this example looks like:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

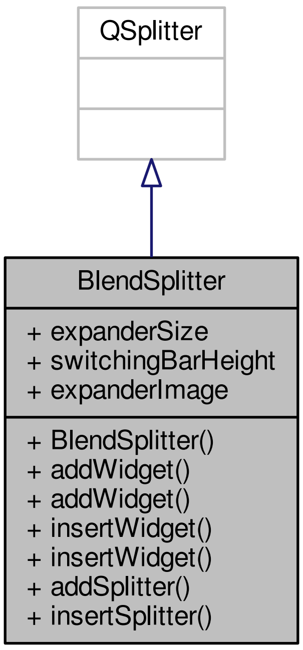
Class Documentation

4.1 BlendSplitter Class Reference

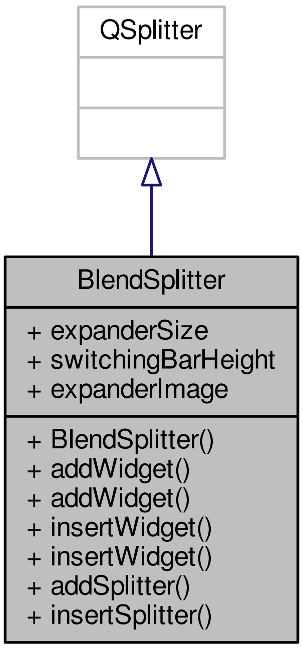
A user-defined Splitter.

```
#include <BlendSplitter.hpp>
```

Inheritance diagram for BlendSplitter:



Collaboration diagram for BlendSplitter:



Public Member Functions

- BlendSplitter (QWidget ∗(∗defaultWidget)()=[ ]() ->QWidget ∗{return new SwitchingWidget{};}, Qt::Orientation orientation=Qt::Horizontal)

  BlendSplitter class constructor.
- void addWidget ()

  Add a widget to the BlendSplitter.
- void addWidget (QWidget ∗widget)

  Add a widget to the BlendSplitter.
- void insertWidget (int index)

  Insert a widget into the BlendSplitter.
- void insertWidget (int index, QWidget ∗widget)

  Insert a widget into the BlendSplitter.
- void addSplitter (BlendSplitter ∗splitter)

  Add another BlendSplitter to this BlendSplitter.
- void insertSplitter (int index, BlendSplitter ∗splitter)

  Insert another BlendSplitter into this BlendSplitter.

Static Public Attributes

- static int expanderSize
- static int switchingBarHeight
- static QString expanderImage

4.1.1 Detailed Description

A user-defined Splitter.

This widget implements the functionality of Blender (Open-source 3D modelling software) widget management. This widget displays a splitter similar to QSplitter. However, each widget in BlendSplitter has a pair of Expanders (one in top right and one in bottom left corner). By dragging from these Expanders inwards a new widget is created in the direction of the drag. If the direction is different to that of the BlendSplitter, a new BlendSplitter with parallel direction is created in place of the widget with the widget and the new widget in it. By dragging from these expanders outwards, a neighbouring widget (or a collection of widgets) can be closed. While the mouse is held, the widgets to be closed are marked with black overlay. When the mouse is released, they are closed. BlendSplitter can be used like any other QWidget, although setting one as the central widget is recommended. A BlendSplitter can contain any QWidget, but to achieve best results, use it together with SwitchingWidget.

BlendSplitter provides 3 static variables that allow some customization of the library design. These are expander↩ Size, switchingBarHeight and expanderImage. These are all initialized with default values. The default Expander image is provided by the library.

The default Expander image:

Definition at line 31 of file BlendSplitter.hpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BlendSplitter::BlendSplitter (QWidget ∗(∗)() *defaultWidget* = `[]() ->QWidget *{return new` **SwitchingWidget**`{};}`, Qt::Orientation *orientation* = `Qt::Horizontal`)

BlendSplitter class constructor.

Parameters

defaultWidget	A pointer to function constructing the default widget. This function is called when a new widget is added to BlendSplitter.
orientation	Orientation of the main BlendSplitter

4.1.3 Member Function Documentation

4.1.3.1 void BlendSplitter::addSplitter (**BlendSplitter** ∗ *splitter*)

Add another BlendSplitter to this BlendSplitter.

Adds a BlendSplitter (usually with parallel orientation) to the BlendSplitter

Parameters

splitter	A pointer to the BlendSplitter to be added

4.1.3.2 void BlendSplitter::addWidget ( )

Add a widget to the BlendSplitter.

Adds the default widget to the very bottom/right of the BlendSplitter.

4.1.3.3 void BlendSplitter::addWidget (QWidget ∗ *widget*)

Add a widget to the BlendSplitter.

Adds the specified widget to the very bottom/right of the BlendSplitter

Parameters

widget	A pointer to the widget to be added

4.1.3.4 void BlendSplitter::insertSplitter (int *index*, BlendSplitter ∗ *splitter*)

Insert another BlendSplitter into this BlendSplitter.

Inserts a BlendSplitter (usually with parallel orientation) into the BlendSplitter at the given position counting from top/left (counting starts at 0).

Parameters

index	The desired position
splitter	A pointer to the BlendSplitter to be inserted

4.1.3.5 void BlendSplitter::insertWidget (int *index*)

Insert a widget into the BlendSplitter.

Inserts the default widget into the BlendSplitter at the given position counting from top/left (counting starts at 0). This function should NOT be called with a BlendSplitter as a parameter.

Parameters

index	The desired position

4.1.3.6 void BlendSplitter::insertWidget (int *index*, QWidget ∗ *widget*)

Insert a widget into the BlendSplitter.

Inserts the specified widget into the BlendSplitter at the given position counting from top/left (counting starts at 0). This function should NOT be called with a BlendSplitter as a parameter.

Parameters

index	The desired position
widget	A pointer to the widget to be inserted

4.1.4 Member Data Documentation

4.1.4.1 QString BlendSplitter::expanderImage `[static]`

The image to be used for the top left expander. The bottom right one will rotate this by pi (180 degrees). Default value: ":/BlendSplitter/Expander"

Definition at line 38 of file BlendSplitter.hpp.

4.1.4.2 int BlendSplitter::expanderSize `[static]`

Size of the expanders in the corners. Default value: 12

Definition at line 36 of file BlendSplitter.hpp.

4.1.4.3 int BlendSplitter::switchingBarHeight `[static]`

Height of the SwitchingBar. Default value: 36

Definition at line 37 of file BlendSplitter.hpp.

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

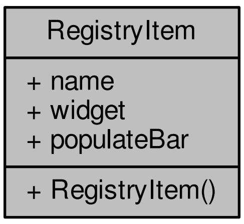
- include/BlendSplitter.hpp

4.2 RegistryItem Class Reference

An item intended to be put into WidgetRegistry.

`#include <RegistryItem.hpp>`

Collaboration diagram for RegistryItem:



Public Member Functions

- RegistryItem (QString name="Default", QWidget ∗(∗widget)()=[ ]() ->QWidget ∗{return new QLabel{"Default Widget"};}, void(∗populateBar)(SwitchingBar ∗, QWidget ∗)=[ ](SwitchingBar ∗, QWidget ∗) ->void{})

    A constructor setting all the internal values.

Public Attributes

- QString name
- QWidget ∗(∗ widget)()

    A function constructing the widget.
- void(∗ populateBar)(SwitchingBar ∗, QWidget ∗)

    A function populating the SwitchingBar.

4.2.1 Detailed Description

An item intended to be put into WidgetRegistry.

Each RegistryItem corresponds to one widget that can be displayed in a BlendSplitter. It describes how this widget should be constructed, what is its name and what items should be in the SwitchingBar when this widget is selected.

Definition at line 17 of file RegistryItem.hpp.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 RegistryItem::RegistryItem (QString *name* = `"Default"`, QWidget ∗(∗)() *widget* = `[]() ->QWidget *{return new QLabel{"Default Widget"};}`, void(∗)(SwitchingBar ∗, QWidget ∗) *populateBar* = `[](`**SwitchingBar** `*, QWidget *) ->void{}`)

A constructor setting all the internal values.

This constructor takes 3 parameters corresponding to the 3 members of the RegistryItem class. See their desription for more details.

Parameters

name	The name of the widget, used in the SwitchingBar combo box
widget	A pointer to a function constructing the widget
populateBar	A pointer to a function populating the SwitchingBar

4.2.3 Member Data Documentation

4.2.3.1 QString RegistryItem::name

The name of the widget, used in the SwitchingBar combo box.

Definition at line 20 of file RegistryItem.hpp.

4.2.3.2 void(∗ RegistryItem::populateBar) (SwitchingBar ∗, QWidget ∗)

A function populating the SwitchingBar.

A pointer to a function populating the SwitchingBar. This function is called each time this widget is selected in any SwitchingWidget. Usually this function makes use of the interface provided by SwitchingBar to populate it.

Parameters

A	pointer to the SwitchingBar to be populated
A	pointer to the newly-created widget in the SwitchingWidget

Definition at line 33 of file RegistryItem.hpp.

4.2.3.3 QWidget∗(∗ RegistryItem::widget) ()

A function constructing the widget.

A pointer to a function returning QWidget∗. This function is called to construct the widget each time it is selected in any SwitchingWidget. Usually in this function the widget is dynamically created using `new` operator and the pointer is returned.

Returns

A pointer to the newly-created QWidget

Definition at line 26 of file RegistryItem.hpp.

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

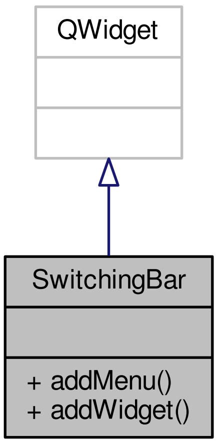
- include/BS/RegistryItem.hpp

4.3 SwitchingBar Class Reference

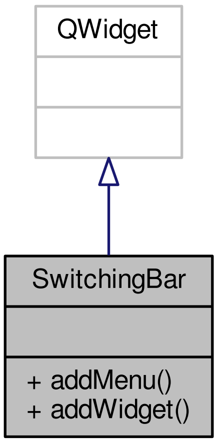
A menu bar which is always found on the bottom of SwitchingWidget.

```
#include <SwitchingBar.hpp>
```

Inheritance diagram for SwitchingBar:



Collaboration diagram for SwitchingBar:



Public Member Functions

- void addMenu (QMenu ∗menu)

  Add a QMenu.
- void addWidget (QWidget ∗widget)

  Add a QWidget.

4.3.1 Detailed Description

A menu bar which is always found on the bottom of SwitchingWidget.

This menu bar is similar to the built-in QMenuBar, but can also contain plain QWidgets. The first item on the left is always a combo box for selecting which widget should be displayed in the SwitchingWidget.

Definition at line 20 of file SwitchingBar.hpp.

4.3.2 Member Function Documentation

4.3.2.1 void SwitchingBar::addMenu (QMenu ∗ *menu*)

Add a QMenu.

This function adds a QMenu to the very right of the SwitchingBar. The menu is wrapped in an invisible QMenuBar.

Parameters

menu	A pointer to the QMenu to be added

4.3.2.2 void SwitchingBar::addWidget (QWidget ∗ *widget*)

Add a QWidget.

This function adds a QWidget to the very right of the SwitchingBar. The widget is placed in a QHBoxLayout.

Parameters

widget	A pointer to the QWidget to be added

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

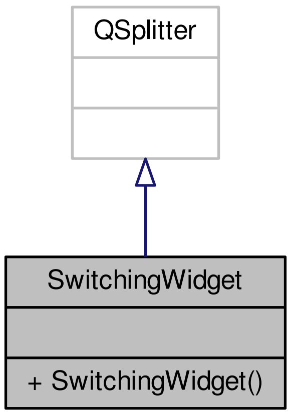
- include/BS/SwitchingBar.hpp

4.4 SwitchingWidget Class Reference

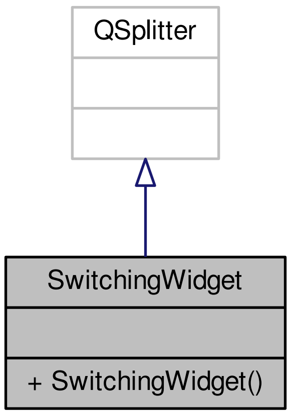
A widget whose actual content can be selected from a combo box.

```
#include <SwitchingWidget.hpp>
```

Inheritance diagram for SwitchingWidget:



Collaboration diagram for SwitchingWidget:



Public Member Functions

- SwitchingWidget (QWidget ∗parent=nullptr)

    A default constructor similar to that of QWidget.

4.4.1 Detailed Description

A widget whose actual content can be selected from a combo box.

This widget displays a Widget with a SwitchingBar on the bottom. The widget displayed is one from WidgetRegistry and it can be selected using a combo box in the SwitchingBar.

Note that constructing an object of this class when WidgetRegistry is empty will cause a default RegistryItem to be added to it. The height of the SwitchingBar can be modified by changing BlendSplitter::switchingBarHeight.

Definition at line 19 of file SwitchingWidget.hpp.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 SwitchingWidget::SwitchingWidget (QWidget ∗ *parent* = `nullptr`) `[explicit]`

A default constructor similar to that of QWidget.

Creates a SwitchingWidget containg the default widget specified in WidgetRegistry

Parameters

parent	A parent widget

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

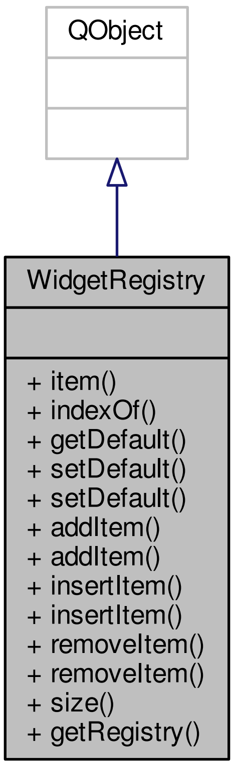
- include/BS/SwitchingWidget.hpp

4.5 WidgetRegistry Class Reference

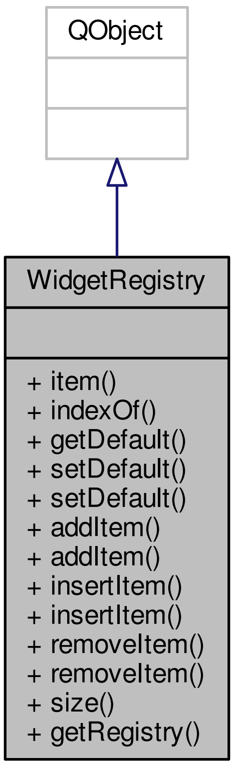
A registry of all widgets that can be displayed in a SwitchingWidget.

`#include <WidgetRegistry.hpp>`

Inheritance diagram for WidgetRegistry:



Collaboration diagram for WidgetRegistry:



Signals

- void registryChanged ()

  Signal emited when WidgetRegistry changes its contents.

Public Member Functions

- RegistryItem ∗ item (int i) const

  Get the item at position i.
- int indexOf (RegistryItem ∗item) const

  Get the position of an item in WidgetRegistry.
- RegistryItem ∗ getDefault ()

  Get the default RegistryItem.
- void setDefault (RegistryItem ∗item)

  Set the default RegistryItem.
- void setDefault (int index=0)

  Set the default RegistryItem.
- void addItem (RegistryItem ∗item)

  Add an item to WidgetRegistry.

- void addItem (QString name="Default", QWidget ∗(∗widget)()=[ ]() ->QWidget ∗{return new QLabel{"Default widget"};}, void(∗populateBar)(SwitchingBar ∗, QWidget ∗)=[ ](SwitchingBar ∗, QWidget ∗) ->void{})

    Add an item to WidgetRegistry.
- void insertItem (int index, RegistryItem ∗item)

    Insert an item into WidgetRegistry.
- void insertItem (int index, QString name="Default", QWidget ∗(∗widget)()=[ ]() ->QWidget ∗{return new Q←Label{"Default widget"};}, void(∗populateBar)(SwitchingBar ∗, QWidget ∗)=[ ](SwitchingBar ∗, QWidget ∗) ->void{})

    Insert an item into widgetRegistry.
- void removeItem (RegistryItem ∗item)

    Remove a RegistryItem from WidgetRegistry.
- void removeItem (int index)

    Remove a RegistryItem from WidgetRegistry.
- int size () const

    Get the size of WidgetRegistry.

Static Public Member Functions

- static WidgetRegistry ∗ getRegistry ()

    Registry getter.

4.5.1 Detailed Description

A registry of all widgets that can be displayed in a SwitchingWidget.

This singleton-class acts as a registry of widgets that can be displayed in a SwitchingWidget by selecting from a combo box in the SwitchingBar. Each item is represented as one RegistryItem. The Registry also contains a pointer to the default RegistryItem, which is shown when a new SwitchingWidget is created.

Definition at line 18 of file WidgetRegistry.hpp.

4.5.2 Member Function Documentation

4.5.2.1 void WidgetRegistry::addItem (**RegistryItem** ∗ *item*)

Add an item to WidgetRegistry.

Adds a given RegistryItem at the end of WidgetRegistry.

Parameters

item	A pointer to the RegistryItem to be added

4.5.2.2 void WidgetRegistry::addItem (QString *name* = `"Default"`, QWidget ∗(∗)() *widget* = `[]() ->QWidget *{return new QLabel{"Default widget"};}`, void(∗)(SwitchingBar ∗, QWidget ∗) *populateBar* = `[](`**SwitchingBar** `*, QWidget *) ->void{}`)

Add an item to WidgetRegistry.

Adds a RegistryItem constructed with the given parameters at the end of WidgetRegistry. This is equal to calling addItem(new RegistryItem{name, widget, populateBar}).

Parameters

name	The name of the widget, used in the SwitchingBar combo box
widget	A pointer to a function constructing the widget
populateBar	A pointer to a function populating the SwitchingBar

4.5.2.3 RegistryItem∗ WidgetRegistry::getDefault ( )

Get the default RegistryItem.

This function gives you the default RegistryItem. Note that if no item was set as default, the currently first item is set as default by this function. If the registry is empty, a RegistryItem is added to the registry (using the default constructor) and set as default.

Returns

A pointer to the default RegistryItem

4.5.2.4 static WidgetRegistry∗ WidgetRegistry::getRegistry ( ) [static]

Registry getter.

This is a singleton class, i. e. you can't construct any object yourself. To get the one-and-only instance of this class, you need to call WidgetRegistry::getRegistry(). The function will create the object if neccessary (= when called for the first time) and return a pointer to it.

Returns

A pointer to the one-and-only instance of WidgetRegistry

4.5.2.5 int WidgetRegistry::indexOf (RegistryItem ∗ *item*) const

Get the position of an item in WidgetRegistry.

Get the index (counting starts at 0) of an item. Often used together with item(int i) const.

Parameters

item	A pointer to the item whose index is to be returned

Returns

Index of the item

4.5.2.6 void WidgetRegistry::insertItem (int *index*, **RegistryItem** ∗ *item*)

Insert an item into WidgetRegistry.

Inserts a given RegistryItem into WidgetRegistry at a given index.

Parameters

index	The desired index of the inserted RegistryItem (counting starts at 0)
item	A pointer to the RegistryItem to be added

4.5.2.7 void WidgetRegistry::insertItem (int *index*, QString *name* = `"Default"`, QWidget ∗(∗)() *widget* = `[]() ->QWidget *{return new QLabel{"Default widget"};}`, void(∗)(**SwitchingBar** ∗, QWidget ∗) *populateBar* = `[](`**SwitchingBar** `*, QWidget *) ->void{}`)

Insert an item into widgetRegistry.

Inserts a RegistryItem constructed with the given parameters into WidgetRegistry at a given index. This is equal to calling insertItem(index, new RegistryItem{name, widget, populateBar}).

Parameters

index	The desired index of the inserted RegistryItem (counting starts at 0)
name	The name of the widget, used in the SwitchingBar combo box
widget	A pointer to a function constructing the widget
populateBar	A pointer to a function populating the SwitchingBar

4.5.2.8 **RegistryItem**∗ WidgetRegistry::item (int *i*) const

Get the item at position i.

This function gives you the item at position i (counting starts at 0).

Parameters

i	Index of the item to be returned

Returns

A pointer to the RegistryItem at position i

4.5.2.9 void WidgetRegistry::registryChanged ( ) `[signal]`

Signal emited when WidgetRegistry changes its contents.

This signal is emited when a RegistryItem is added, inserted or removed from WidgetRegistry. It is NOT emited when the default item is changed unless this change requires adding a RegistryItem.

4.5.2.10 void WidgetRegistry::removeItem (RegistryItem ∗ *item*)

Remove a RegistryItem from WidgetRegistry.

Removes a given RegistryItem from WidgetRegistry. If this is also the default RegistryItem, the first RegistryItem is set as default if it exists. This is equal to calling removeItem(indexOf(item)).

Parameters

item	

4.5.2.11 void WidgetRegistry::removeItem (int *index*)

Remove a RegistryItem from WidgetRegistry.

Removes the RegistryItem at position index (counting starts at 0) from WidgetRegistry. If this is also the default RegistryItem, the first RegistryItem is set as default if it exists.

Parameters

index	Index of the RegistryItem to be removed

4.5.2.12 void WidgetRegistry::setDefault (RegistryItem ∗ *item*)

Set the default RegistryItem.

This function sets the default RegistryItem. Note that if the item is not in WidgetRegistry, it is added as the last entry. The default item is used when a new SwitchingWidget is created as the displayed widget.

Parameters

item	A pointer to the RegistryItem to be set as default

4.5.2.13 void WidgetRegistry::setDefault (int *index* = 0)

Set the default RegistryItem.

This function sets the default RegistryItem to be the RegistryItem at given index. This is equal to calling set↩ Default(item(index)).

Parameters

index	Index of the RegistryItem to be set as default (counting starts at 0)

4.5.2.14 int WidgetRegistry::size ( ) const

Get the size of WidgetRegistry.

This function returns the number of RegistryItems currently in WidgetRegistry. Note that these are indexed as 0, ..., (size() - 1).

Returns

Number of RegistryItems in WidgetRegistry

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

- include/BS/WidgetRegistry.hpp

Index