QListWidget Class Reference

[QtGui module]

The QListWidget class provides an item-based list widget. More...

Inherits QListView.

Methods

- __init__ (*self*, QWidget *parent* = None)
- addItem (*self*, QListWidgetItem *aitem*)
- addItem (*self*, QString *label*)
- addItems (*self*, QStringList *labels*)
- clear (*self*)
- closePersistentEditor (*self*, QListWidgetItem *item*)
- int count (*self*)
- QListWidgetItem currentItem (*self*)
- int currentRow (*self*)
- dropEvent (*self*, QDropEvent *event*)
- bool dropMimeData (*self*, int *index*, QMimeData *data*, Qt.DropAction *action*)
- editItem (*self*, QListWidgetItem *item*)
- bool event (*self*, QEvent *e*)
- list-of-QListWidgetItem findItems (*self*, QString *text*, Qt.MatchFlags *flags*)
- QModelIndex indexFromItem (*self*, QListWidgetItem *item*)
- insertItem (*self*, int *row*, QListWidgetItem *item*)
- insertItem (*self*, int *row*, QString *label*)
- insertItems (*self*, int *row*, QStringList *labels*)
- bool isItemHidden (*self*, QListWidgetItem *item*)
- bool isItemSelected (*self*, QListWidgetItem *item*)
- bool isSortingEnabled (*self*)
- QListWidgetItem item (*self*, int *row*)
- QListWidgetItem itemAt (*self*, QPoint *p*)
- QListWidgetItem itemAt (*self*, int *ax*, int *ay*)
- QListWidgetItem itemFromIndex (*self*, QModelIndex *index*)
- list-of-QListWidgetItem items (*self*, QMimeData *data*)
- QWidget itemWidget (*self*, QListWidgetItem *item*)
- QMimeData mimeData (*self*, list-of-QListWidgetItem *items*)
- QStringList mimeTypes (*self*)
- openPersistentEditor (*self*, QListWidgetItem *item*)
- removeItemWidget (*self*, QListWidgetItem *aItem*)
- int row (*self*, QListWidgetItem *item*)
- scrollToItem (*self*, QListWidgetItem *item*, QAbstractItemView.ScrollHint *hint* = QAbstractItemView.EnsureVisible)
- list-of-QListWidgetItem selectedItems (*self*)
- setCurrentItem (*self*, QListWidgetItem *item*)
- setCurrentItem (*self*, QListWidgetItem *item*, QItemSelectionModel.SelectionFlags *command*)
- setCurrentRow (*self*, int *row*)
- setCurrentRow (*self*, int *row*, QItemSelectionModel.SelectionFlags *command*)
- setItemHidden (*self*, QListWidgetItem *item*, bool *hide*)
- setItemSelected (*self*, QListWidgetItem *item*, bool *select*)
- setItemWidget (*self*, QListWidgetItem *item*, QWidget *widget*)
- setSortingEnabled (*self*, bool *enable*)
- sortItems (*self*, Qt.SortOrder *order* = Qt.AscendingOrder)
- Qt.DropActions supportedDropActions (*self*)

- QListWidgetItem takeItem (*self*, int *row*)
- QRect visualItemRect (*self*, QListWidgetItem *item*)

Special Methods

- __len__ (*self*)

Qt Signals

- void currentItemChanged (QListWidgetItem *,QListWidgetItem *)
- void currentRowChanged (int)
- void currentTextChanged (const QString&)
- void itemActivated (QListWidgetItem *)
- void itemChanged (QListWidgetItem *)
- void itemClicked (QListWidgetItem *)
- void itemDoubleClicked (QListWidgetItem *)
- void itemEntered (QListWidgetItem *)
- void itemPressed (QListWidgetItem *)
- void itemSelectionChanged ()

Detailed Description

The QListWidget class provides an item-based list widget.

QListWidget is a convenience class that provides a list view similar to the one supplied by QListView, but with a classic item-based interface for adding and removing items. QListWidget uses an internal model to manage each QListWidgetItem in the list.

For a more flexible list view widget, use the QListView class with a standard model.

List widgets are constructed in the same way as other widgets:

```
QListWidget *listWidget = new QListWidget(this);
```

The selectionMode() of a list widget determines how many of the items in the list can be selected at the same time, and whether complex selections of items can be created. This can be set with the setSelectionMode() function.

There are two ways to add items to the list: they can be constructed with the list widget as their parent widget, or they can be constructed with no parent widget and added to the list later. If a list widget already exists when the items are constructed, the first method is easier to use:

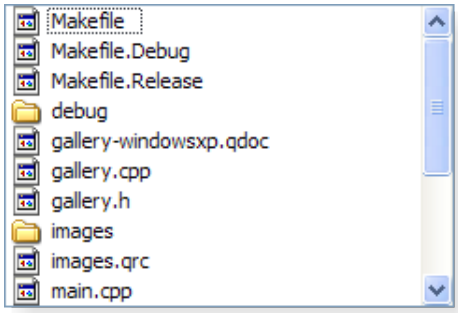
```
new QListWidgetItem(tr("Oak"), listWidget);
new QListWidgetItem(tr("Fir"), listWidget);
new QListWidgetItem(tr("Pine"), listWidget);
```

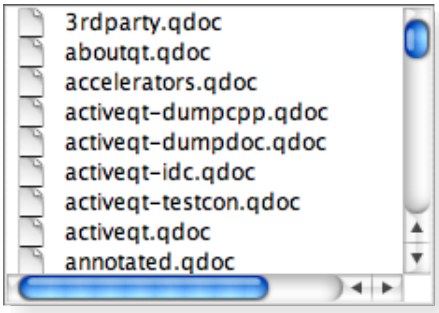
If you need to insert a new item into the list at a particular position, then it should be constructed without a parent widget. The insertItem() function should then be used to place it within the list. The list widget will take ownership of the item.

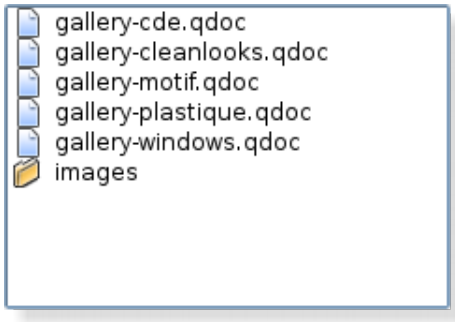
```
QListWidgetItem *newItem = new QListWidgetItem;
newItem->setText(itemText);
listWidget->insertItem(row, newItem);
```

For multiple items, insertItems() can be used instead. The number of items in the list is found with the count() function. To remove items from the list, use takeItem().

The current item in the list can be found with currentItem(), and changed with setCurrentItem(). The user can also change the current item by navigating with the keyboard or clicking on a different item. When the current item changes, the currentItemChanged() signal is emitted with the new current item and the item that was previously current.







A Windows XP style list widget. A Macintosh style list widget. A Plastique style list widget.

Method Documentation

QListWidget.__init__ (*self*, QWidget *parent* = None)

The *parent* argument, if not None, causes *self* to be owned by Qt instead of PyQt.

Constructs an empty QListWidget with the given *parent*.

QListWidget.addItem (*self*, QListWidgetItem *aitem*)

The *aitem* argument has it's ownership transferred to Qt.

Inserts an item with the text *label* at the end of the list widget.

QListWidget.addItem (*self*, QString *label*)

Inserts the *item* at the end of the list widget.

Warning: A QListWidgetItem can only be added to a QListWidget once. Adding the same QListWidgetItem multiple times to a QListWidget will result in undefined behavior.

See also insertItem().

QListWidget.addItems (*self*, QStringList *labels*)

Inserts items with the text *labels* at the end of the list widget.

See also insertItems().

QListWidget.clear (*self*)

This method is also a Qt slot with the C++ signature `void clear()`.

Removes all items and selections in the view.

Warning: All items will be permanently deleted.

QListWidget.closePersistentEditor (*self*, QListWidgetItem *item*)

Closes the persistent editor for the given *item*.

See also openPersistentEditor().

int QListWidget.count (*self*)

QListWidgetItem QListWidget.currentItem (*self*)

Returns the current item.

See also setCurrentItem().

int QListWidget.currentRow (*self*)

QListWidget.dropEvent (*self*, QDropEvent *event*)

Reimplemented from QWidget.dropEvent().

bool QListWidget.dropMimeData (*self*, int *index*, QMimeData *data*, Qt.DropAction *action*)

Handles *data* supplied by an external drag and drop operation that ended with the given *action* in the given *index*. Returns true if *data* and *action* can be handled by the model; otherwise returns false.

See also supportedDropActions().

QListWidget.editItem (*self*, QListWidgetItem *item*)

Starts editing the *item* if it is editable.

bool QListWidget.event (*self*, QEvent *e*)

Reimplemented from QObject.event().

list-of-QListWidgetItem QListWidget.findItems (*self*, QString *text*, Qt.MatchFlags *flags*)

Finds items with the text that matches the string *text* using the given *flags*.

QModelIndex QListWidget.indexFromItem (*self*, QListWidgetItem *item*)

Returns the QModelIndex assocated with the given *item*.

QListWidget.insertItem (*self*, int *row*, QListWidgetItem *item*)

The *item* argument has it's ownership transferred to Qt.

Inserts the *item* at the position in the list given by *row*.

See also addItem().

QListWidget.insertItem (*self*, int *row*, QString *label*)

Inserts an item with the text *label* in the list widget at the position given by *row*.

See also addItem().

QListWidget.insertItems (*self*, int *row*, QStringList *labels*)

Inserts items from the list of *labels* into the list, starting at the given *row*.

See also insertItem() and addItem().

bool QListWidget.isItemHidden (*self*, QListWidgetItem *item*)

bool QListWidget.isItemSelected (*self*, QListWidgetItem *item*)

bool QListWidget.isSortingEnabled (*self*)

QListWidgetItem QListWidget.item (*self*, int *row*)

Returns the item that occupies the given *row* in the list if one has been set; otherwise returns 0.

See also row().

QListWidgetItem QListWidget.itemAt (*self*, QPoint *p*)

Returns a pointer to the item at the coordinates *p*. The coordinates are relative to the list widget's viewport().

QListWidgetItem QListWidget.itemAt (*self*, int *ax*, int *ay*)

This is an overloaded function.

Returns a pointer to the item at the coordinates (*x*, *y*). The coordinates are relative to the list widget's viewport().

QListWidgetItem QListWidget.itemFromIndex (*self*, QModelIndex *index*)

Returns a pointer to the QListWidgetItem assocated with the given *index*.

list-of-QListWidgetItem QListWidget.items (*self*, QMimeData *data*)

Returns a list of pointers to the items contained in the *data* object. If the object was not created by a QListWidget in the same process, the list is empty.

QWidget QListWidget.itemWidget (*self*, QListWidgetItem *item*)

Returns the widget displayed in the given *item*.

This function was introduced in Qt 4.1.

See also setItemWidget().

QMimeData QListWidget.mimeData (*self*, list-of-QListWidgetItem *items*)

The *QMimeData* result

Returns an object that contains a serialized description of the specified *items*. The format used to describe the items is obtained from the mimeTypes() function.

If the list of items is empty, 0 is returned instead of a serialized empty list.

QStringList QListWidget.mimeTypes (*self*)

Returns a list of MIME types that can be used to describe a list of listwidget items.

See also mimeData().

QListWidget.openPersistentEditor (*self*, QListWidgetItem *item*)

Opens an editor for the given *item*. The editor remains open after editing.

See also closePersistentEditor().

QListWidget.removeItemWidget (*self*, QListWidgetItem *aItem*)

Removes the widget set on the given *item*.

This function was introduced in Qt 4.3.

int QListWidget.row (*self*, QListWidgetItem *item*)

Returns the row containing the given *item*.

See also item().

QListWidget.scrollToItem (*self*, QListWidgetItem *item*,
QAbstractItemView.ScrollHint *hint* = QAbstractItemView.EnsureVisible)

This method is also a Qt slot with the C++ signature void scrollToItem(const QListWidgetItem *,QAbstractItemView::ScrollHint = QAbstractItemView.EnsureVisible).

Scrolls the view if necessary to ensure that the *item* is visible.

hint specifies where the *item* should be located after the operation.

list-of-QListWidgetItem QListWidget.selectedItems (*self*)

Returns a list of all selected items in the list widget.

QListWidget.setCurrentItem (*self*, QListWidgetItem *item*)

Sets the current item to *item*.

Unless the selection mode is NoSelection, the item is also be selected.

See also currentItem().

QListWidget.setCurrentItem (*self*, QListWidgetItem *item*,
QItemSelectionModel.SelectionFlags *command*)

Set the current item to *item*, using the given *command*.

This function was introduced in Qt 4.4.

QListWidget.setCurrentRow (*self*, int *row*)

QListWidget.setCurrentRow (*self*, int *row*,
QItemSelectionModel.SelectionFlags *command*)

QListWidget.setItemHidden (*self*, QListWidgetItem *item*, bool *hide*)

QListWidget.setItemSelected (*self*, QListWidgetItem *item*, bool *select*)

QListWidget.setItemWidget (*self*, QListWidgetItem *item*, QWidget *widget*)

The *widget* argument has it's ownership transferred to Qt.

Sets the *widget* to be displayed in the give *item*.

This function should only be used to display static content in the place of a list widget item. If you want to display custom dynamic content or implement a custom editor widget, use QListView and subclass QItemDelegate instead.

This function was introduced in Qt 4.1.

See also itemWidget() and Delegate Classes.

QListWidget.setSortingEnabled (*self*, bool *enable*)

QListWidget.sortItems (*self*, Qt.SortOrder *order* = Qt.AscendingOrder)

Sorts all the items in the list widget according to the specified *order*.

Qt.DropActions QListWidget.supportedDropActions (*self*)

Returns the drop actions supported by this view.

See also Qt.DropActions.

QListWidgetItem QListWidget.takeItem (*self*, int *row*)

The *QListWidgetItem* result

Removes and returns the item from the given *row* in the list widget; otherwise returns 0.

Items removed from a list widget will not be managed by Qt, and will need to be deleted manually.

See also insertItem() and addItem().

QRect QListWidget.visualItemRect (*self*, QListWidgetItem *item*)

Returns the rectangle on the viewport occupied by the item at *item*.

QListWidget.__len__ (*self*)

Qt Signal Documentation

void currentItemChanged (QListWidgetItem *,QListWidgetItem *)

This is the default overload of this signal.

This signal is emitted whenever the current item changes.

previous is the item that previously had the focus; *current* is the new current item.

void currentRowChanged (int)

This is the default overload of this signal.

This signal is emitted whenever the current item changes.

currentRow is the row of the current item. If there is no current item, the *currentRow* is -1.

void currentTextChanged (const QString&)

This is the default overload of this signal.

This signal is emitted whenever the current item changes.

currentText is the text data in the current item. If there is no current item, the *currentText* is invalid.

void itemActivated (QListWidgetItem *)

This is the default overload of this signal.

This signal is emitted when the *item* is activated. The *item* is activated when the user clicks or double clicks on it, depending on the system configuration. It is also activated when the user presses the activation key (on Windows and X11 this is the **Return** key, on Mac OS X it is **Ctrl+O**).

void itemChanged (QListWidgetItem *)

This is the default overload of this signal.

This signal is emitted whenever the data of *item* has changed.

void itemClicked (QListWidgetItem *)

This is the default overload of this signal.

This signal is emitted with the specified *item* when a mouse button is clicked on an item in the widget.

See also itemPressed() and itemDoubleClicked().

void itemDoubleClicked (QListWidgetItem *)

This is the default overload of this signal.

This signal is emitted with the specified *item* when a mouse button is double clicked on an item in the widget.

See also itemClicked() and itemPressed().

void itemEntered (QListWidgetItem *)

This is the default overload of this signal.

This signal is emitted when the mouse cursor enters an item. The *item* is the item entered. This signal is only emitted when mouseTracking is turned on, or when a mouse button is pressed while moving into an item.

See also QWidget.setMouseTracking().

void itemPressed (QListWidgetItem *)

This is the default overload of this signal.

This signal is emitted with the specified *item* when a mouse button is pressed on an item in the widget.

See also itemClicked() and itemDoubleClicked().

void itemSelectionChanged ()

This is the default overload of this signal.

This signal is emitted whenever the selection changes.

See also selectedItems(), QListWidgetItem.isSelected(), and currentItemChanged().
