QListWidget Class

QListWidget 类提供了一个基于项目的列表小部件。更多的...

Header:	**#include < QListWidget>**
CMake:	find_package(Qt6 REQUIRED COMPONENTS Widgets) target_link_libraries(mytarget PRIVATE Qt6::Widgets)
qmake:	QT += widgets
Inherits:	QListView

- 所有成员的列表，包括继承的成员

特性

- **count** : const int
- **currentRow** : int
- **sortingEnabled** : bool

公共职能

	QListWidget(QWidget **parent* = nullptr)
virtual	**~QListWidget**()
void	**addItem**(const QString &*label*)
void	**addItem**(QListWidgetItem ***item*)
void	**addItems**(const QStringList &*labels*)
void	**closePersistentEditor**(QListWidgetItem ***item*)
int	**count**() const
QListWidgetItem *	**currentItem**() const
int	**currentRow**() const
void	**editItem**(QListWidgetItem ***item*)
QList<QListWidgetItem *>	**findItems**(const QString &*text*, Qt::MatchFlags *flags*) const

QModelIndex	**indexFromItem**(const QListWidgetItem *item*) const
	QListWidget(QWidget *parent* = nullptr)
void	**insertItem**(int *row*, QListWidgetItem *item*)
void	**insertItem**(int *row*, const QString &*label*)
void	**insertItems**(int *row*, const QStringList &*labels*)
bool	**isPersistentEditorOpen**(QListWidgetItem *item*) const
bool	**isSortingEnabled**() const
QListWidgetItem *	**item**(int *row*) const
QListWidgetItem *	**itemAt**(const QPoint &*p*) const
QListWidgetItem *	**itemAt**(int *x*, int *y*) const
QListWidgetItem *	**itemFromIndex**(const QModelIndex &*index*) const
QWidget *	**itemWidget**(QListWidgetItem *item*) const
QList<QListWidgetItem *>	**items**(const QMimeData *data*) const
void	**openPersistentEditor**(QListWidgetItem *item*)
void	**removeItemWidget**(QListWidgetItem *item*)
int	**row**(const QListWidgetItem *item*) const
QList<QListWidgetItem *>	**selectedItems**() const
void	**setCurrentItem**(QListWidgetItem *item*)
void	**setCurrentItem**(QListWidgetItem *item*, QItemSelectionModel::SelectionFlags *command*)
void	**setCurrentRow**(int *row*)
void	**setCurrentRow**(int *row*, QItemSelectionModel::SelectionFlags *command*)
void	**setItemWidget**(QListWidgetItem **item*, **QWidget** widget*)
void	**setSortingEnabled**(bool *enable*)
void	**sortItems**(Qt::SortOrder *order* = Qt::AscendingOrder)
QListWidgetItem *	**takeItem**(int *row*)
QRect	**visualItemRect**(const QListWidgetItem *item*) const

重新实施公共职能

virtual void	**setSelectionModel(QItemSelectionModel **selectionModel*) override**

公共老虎机

void	**clear()**
void	**scrollToItem**(const QListWidgetItem **item*, QAbstractItemView::ScrollHint *hint* = EnsureVisible)

信号

void	**currentItemChanged(QListWidgetItem current*, QListWidgetItem previous*)**
void	**currentRowChanged**(int *currentRow*)
void	**currentTextChanged**(const QString &*currentText*)
void	**itemActivated**(QListWidgetItem **item*)
void	**itemChanged**(QListWidgetItem **item*)
void	**itemClicked**(QListWidgetItem **item*)
void	**itemDoubleClicked**(QListWidgetItem **item*)
void	**itemEntered**(QListWidgetItem **item*)
void	**itemPressed**(QListWidgetItem **item*)
void	**itemSelectionChanged**()

受保护的功能

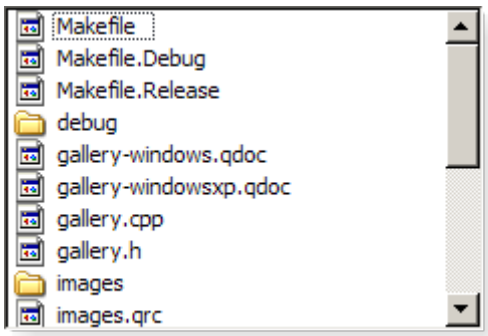
virtual bool	**dropMimeData(int *index*, const QMimeData **data*, Qt::DropAction *action*)**
virtual QMimeData *	**mimeData**(const QList<QListWidgetItem > &items*) const
virtual QStringList	**mimeTypes**() const
virtual Qt::DropActions	**supportedDropActions**() const

重新实现受保护的功能

virtual void	**dropEvent(QDropEvent **event*) override**

virtual void	**dropEvent**(QDropEvent **event*) override
virtual bool	**event**(QEvent **e*) override

详细说明



QListWidget 是一个方便的类，它提供了一个类似于由QListView，但具有用于添加和删除项目的经典的基于项目的界面。QListWidget使用内部模型来管理每个QListWidgetItem在列表中。

对于更灵活的列表视图小部件，请使用QListView具有标准模型的类。

列表小部件的构造方式与其他小部件相同：

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

这selectionMode列表小部件的 () 决定可以同时选择列表中的多少个项目，以及是否可以创建复杂的项目选择。这可以通过设置setSelectionMode （） 功能。

有两种方法可以将项目添加到列表中：可以使用列表小部件作为其父小部件来构造它们，也可以不使用父小部件来构造它们，然后将其添加到列表中。如果在构造项目时列表小部件已经存在，则第一种方法更容易使用：

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

如果您需要将新项目插入到列表中的特定位置，那么应该在没有父窗口小部件的情况下构造它。这insertItem然后应该使用 () 函数将其放入列表中。列表小部件将取得该项目的所有权。

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

对于多个项目，insertItems可以用()代替。列表中的项目数可以通过以下命令找到count （） 功能。要从列表中删除项目，请使用takeItem()。

列表中的当前项目可以通过以下方式找到currentItem()，并更改为setCurrentItem()。用户还可以通过使用键盘导航或单击不同的项目来更改当前项目。当当前项目发生变化时，currentItemChanged() 信号与新的当前项目和先前的当前项目一起发出。

也可以看看QListWidgetItem,QListView,QTreeView,Model/View Programming， 和Tab Dialog Example。

财产文件

[read-only]count : const int

此属性保存列表中的项目数，包括任何隐藏的项目。

访问功能：

int	**count() const**

currentRow : int

该属性保存当前项目的行。

根据当前的选择模式，还可以选择该行。

访问功能：

int	**currentRow() const**
void	**setCurrentRow**(int *row*)
void	**setCurrentRow**(int *row*, QItemSelectionModel::SelectionFlags *command*)

通知器信号：

void	**currentRowChanged(int *currentRow*)**

sortingEnabled : bool

该属性保存是否启用排序

如果此属性为`true`，则为列表启用排序；如果该属性为 false，则不启用排序。

默认值为 false。

访问功能：

bool	**isSortingEnabled() const**
void	**setSortingEnabled**(bool *enable*)

成员函数文档

*[explicit]QListWidget::QListWidget(QWidget *parent = nullptr)*

使用给定的值构造一个空的 QListWidget*parent*。

[virtual]QListWidget::~QListWidget()

销毁列表小部件及其所有项目。

void QListWidget::addItem(const QString &label)

插入带有文本的项目*label*在列表小部件的末尾。

*void QListWidget::addItem(QListWidgetItem *item)*

插入*item*在列表小部件的末尾。

警告： AQListWidgetItem只能添加到QListWidget一次。添加相同的QListWidgetItem多次到QListWidget将导致未定义的行为。

也可以看看insertItem()。

void QListWidget::addItems(const QStringList &labels)

插入带有文本的项目*labels*在列表小部件的末尾。

也可以看看insertItems()。

[slot]void QListWidget::clear()

删除视图中的所有项目和选择。

警告：所有项目将被永久删除。

*void QListWidget::closePersistentEditor(QListWidgetItem *item)*

关闭给定的持久编辑器*item*。

也可以看看openPersistentEditor（） 和isPersistentEditorOpen()。

*QListWidgetItem *QListWidget::currentItem() const*

返回当前项目。

也可以看看setCurrentItem()。

*[signal]void QListWidget::currentItemChanged(QListWidgetItem **current***, ***QListWidgetItem** previous*)*

每当当前项发生更改时，都会发出此信号。

*previous*是先前获得焦点的项目；*current*是新的当前项目。

[signal]void QListWidget::currentRowChanged(int currentRow)

每当当前项发生更改时，都会发出此信号。

*currentRow*是当前项目的行。如果没有当前项目，则*currentRow*是-1。

注意：属性的通知程序信号currentRow。

[signal]void QListWidget::currentTextChanged(const QString &currentText)

每当当前项发生更改时，都会发出此信号。

*currentText*是当前项目中的文本数据。如果没有当前项目，则*currentText*是无效的。

*[override virtual protected]void QListWidget::dropEvent(QDropEvent *event)*

重新实现：QListView::dropEvent（QDropEvent *事件）。

*[virtual protected]bool QListWidget::dropMimeData(int index, const QMimeData *data, Qt::DropAction action)*

手柄*data*由以给定结尾的外部拖放操作提供*action*在给定的*index*。返回`true`如果*data*和*action*可以由模型处理；否则返回`false`.

也可以看看supportedDropActions()。

*void QListWidget::editItem(QListWidgetItem *item)*

开始编辑*item*如果它是可编辑的。

*[override virtual protected]bool QListWidget::event(QEvent *e)*

重新实现：QListView::event（QEvent *e）。

QList<QListWidgetItem > QListWidget::findItems(const QString &text, Qt::MatchFlags flags) const*

查找文本与字符串匹配的项目*text*使用给定的*flags*。

*QModelIndex QListWidget::indexFromItem(const QListWidgetItem *item) const*

返回QModelIndex与给定相关的*item*。

注意：在 5.10 之前的 Qt 版本中，此函数采用非`const` *item*。

*void QListWidget::insertItem(int row, QListWidgetItem *item)*

插入*item*在由给出的列表中的位置*row*。

也可以看看addItem()。

void QListWidget::insertItem(int row, const QString &label)

插入带有文本的项目*label*在列表小部件中指定的位置*row*。

也可以看看addItem()。

void QListWidget::insertItems(int row, const QStringList &labels)

从列表中插入项目*labels*进入列表，从给定的位置开始*row*。

也可以看看insertItem（） 和addItem()。

*bool QListWidget::isPersistentEditorOpen(QListWidgetItem *item) const*

返回持久编辑器是否为项目打开*item*。

也可以看看openPersistentEditor（） 和closePersistentEditor()。

*QListWidgetItem *QListWidget::item(int row) const*

返回占据给定值的项目*row*如果已设置，则在列表中；否则返回`nullptr`.

也可以看看row()。

*[signal]void QListWidget::itemActivated(QListWidgetItem *item)*

当*item*被激活。这*item*当用户单击或双击它时激活，具体取决于系统配置。当用户按下激活键时它也会被激活（在 Windows 和 X11 上，这是**Return**键，在 Mac OS X 上是**Command+O**）。

QListWidgetItem QListWidget::itemAt(const QPoint &p) const*

返回指向坐标处的项目的指针*p*。坐标是相对于列表小部件的viewport()。

*QListWidgetItem *QListWidget::itemAt(int x, int y) const*

这是一个过载功能。

返回指向坐标处的项目的指针 (*x*,*y*) 。坐标是相对于列表小部件的viewport()。

*[signal]void QListWidget::itemChanged(QListWidgetItem *item)*

每当数据*item*已经改变。

*[signal]void QListWidget::itemClicked(QListWidgetItem *item)*

该信号以指定的方式发出*item*当鼠标按钮单击小部件中的项目时。

也可以看看itemPressed（） 和itemDoubleClicked()。

*[signal]void QListWidget::itemDoubleClicked(QListWidgetItem *item)*

该信号以指定的方式发出*item*当鼠标按钮双击小部件中的项目时。

也可以看看itemClicked（） 和itemPressed()。

*[signal]void QListWidget::itemEntered(QListWidgetItem *item)*

当鼠标光标进入某个项目时，会发出此信号。这*item*是输入的项目。仅当打开 mouseTracking 或在移动到某个项目时按下鼠标按钮时，才会发出此信号。

也可以看看QWidget::setMouseTracking()。

QListWidgetItem QListWidget::itemFromIndex(const QModelIndex &index) const*

返回一个指向QListWidgetItem与给定相关的*index*。

*[signal]void QListWidget::itemPressed(QListWidgetItem *item)*

该信号以指定的方式发出*item*当在小部件中的某个项目上按下鼠标按钮时。

也可以看看itemClicked（） 和itemDoubleClicked()。

[signal]void QListWidget::itemSelectionChanged()

每当选择发生变化时就会发出此信号。

也可以看看selectedItems(),QListWidgetItem::isSelected（），和currentItemChanged()。

*QWidget QListWidget::itemWidget(QListWidgetItem **item) const*

返回给定中显示的小部件*item*。

也可以看看setItemWidget（） 和removeItemWidget()。

*QList<QListWidgetItem > QListWidget::items(const QMimeData **data) const*

返回指向包含在其中的项目的指针列表*data*目的。如果该对象不是由QListWidget在同一过程中，列表为空。

*[virtual protected]QMimeData *QListWidget::mimeData(const QList<QListWidgetItem > &items*) const*

返回一个对象，其中包含指定的序列化描述*items*。用于描述项目的格式是从mimeTypes（） 功能。

如果项目列表为空，`nullptr`则返回而不是序列化的空列表。

[virtual protected]QStringList QListWidget::mimeTypes() const

返回可用于描述列表小部件项目列表的 MIME 类型列表。

也可以看看mimeData()。

*void QListWidget::openPersistentEditor(QListWidgetItem *item)*

打开给定的编辑器*item*。编辑器在编辑后保持打开状态。

也可以看看closePersistentEditor（） 和isPersistentEditorOpen()。

*void QListWidget::removeItemWidget(QListWidgetItem *item)*

删除给定的小部件集*item*。

要从列表中完全删除项目（行），请删除该项目或使用takeItem()。

也可以看看itemWidget（） 和setItemWidget()。

*int QListWidget::row(const QListWidgetItem *item) const*

返回包含给定值的行*item*。

也可以看看item()。

*[slot]void QListWidget::scrollToItem(const QListWidgetItem *item, QAbstractItemView::ScrollHint hint = EnsureVisible)*

如有必要，滚动视图以确保*item*是可见的。

*hint*指定在哪里*item*应在手术后定位。

*QList<QListWidgetItem *> QListWidget::selectedItems() const*

返回列表小部件中所有选定项目的列表。

*void QListWidget::setCurrentItem(QListWidgetItem *item)*

将当前项目设置为*item*。

除非选择模式是NoSelection，该项目也被选中。

也可以看看currentItem()。

*void QListWidget::setCurrentItem(QListWidgetItem *item, QItemSelectionModel::SelectionFlags command)*

将当前项目设置为*item*，使用给定的*command*。

void QListWidget::setCurrentRow(int row, QItemSelectionModel::SelectionFlags command)

将当前行设置为给定的*row*，使用给定的*command*,

注意：属性的 Setter 函数currentRow。

*void QListWidget::setItemWidget(QListWidgetItem **item*, QWidget *widget*)*

设置*widget*要显示在给定的*item*。

此函数只能用于在列表小部件项目的位置显示静态内容。如果您想显示自定义动态内容或实现自定义编辑器小部件，请使用QListView和子类QStyledItemDelegate反而。

也可以看看itemWidget(),removeItemWidget（），和Delegate Classes。

*[override virtual]void QListWidget::setSelectionModel(QItemSelectionModel *selectionModel)*

重新实现：QAbstractItemView::setSelectionModel(QItemSelectionModel *selectionModel)。

void QListWidget::sortItems(Qt::SortOrder order = Qt::AscendingOrder)

根据指定对列表小部件中的所有项目进行排序*order*。

[virtual protected]Qt::DropActions QListWidget::supportedDropActions() const

返回此视图支持的放置操作。

也可以看看Qt::DropActions。

*QListWidgetItem *QListWidget::takeItem(int row)*

删除并返回给定的项目*row*在列表小部件中；否则返回`nullptr`.

从列表小部件中删除的项目将不会由 Qt 管理，并且需要手动删除。

也可以看看insertItem（） 和addItem()。

*QRect QListWidget::visualItemRect(const QListWidgetItem *item) const*

返回视口中项目占据的矩形*item*。