QScrollBar Class

QScrollBar 小部件提供垂直或水平滚动条。更多的...

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

- 所有成员的列表，包括继承的成员

公共职能

	QScrollBar(QWidget **parent* = nullptr)
	QScrollBar(Qt::Orientation *orientation*, QWidget **parent* = nullptr)
virtual	~QScrollBar()

重载公共职能

virtual bool	**event(QEvent **event*) override**
virtual QSize	sizeHint() const override

protected功能

virtual void	**initStyleOption(QStyleOptionSlider **option*) const**

重载protected功能

virtual void	**contextMenuEvent**(QContextMenuEvent ***event***) **override**
virtual void	**hideEvent**(QHideEvent *) override
virtual void	**mouseMoveEvent**(QMouseEvent **e*) override
virtual void	**mousePressEvent**(QMouseEvent **e*) override
virtual void	**mouseReleaseEvent**(QMouseEvent **e*) override
virtual void	**paintEvent**(QPaintEvent *) override
virtual void	**sliderChange**(QAbstractSlider::SliderChange *change*) override
virtual void	**wheelEvent**(QWheelEvent **event*) override

详细说明

滚动条是一种控件，使用户能够访问大于用于显示文档的小部件的文档部分。它提供了用户在文档中的当前位置以及可见文档的数量的视觉指示。滚动条通常配备有其他控件，可以实现更准确的导航。Qt 以适合每个平台的方式显示滚动条。

如果您需要在另一个小部件上提供滚动视图，那么使用QScrollArea类，因为它提供了视口小部件和滚动条。如果您需要使用以下命令为专用小部件实现类似的功能，QScrollBar 非常有用QAbstractScrollArea; 例如，如果您决定子类化QAbstractItemView。对于使用滑块控件获取给定范围内的值的大多数其他情况，QSlider课程可能更适合您的需求。

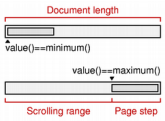
滚动条通常包括四个独立的控件：滑块、滚动箭头和页面控件。A。滑块提供了一种快速转到文档任何部分的方法，但不支持在大型文档中进行精确导航。b. 滚动箭头是按钮，可用于准确导航到文档中的特定位置。对于连接到文本编辑器的垂直滚动条，它们通常将当前位置向上或向下移动一“行”，并稍微调整滑块的位置。在编辑器和列表框中，“行”可能表示一行文本；在图像查看器中，它可能意味着 20 像素。C。页面控件是滑块拖动的区域（滚动条的背景）。单击此处，滚动条将向单击的方向移动一“页”。该值通常与滑块的长度相同。

每个滚动条都有一个值，指示滑块距离滚动条起点有多远；这是通过以下方式获得的value() 并设置为setValue()。该值始终位于为滚动条定义的值范围内，从minimum （） 到maximum （） 包括的。可接受值的范围可以设置为setMinimum （） 和setMaximum()。在最小值时，滑块的上边缘（对于垂直滚动条）或左边缘（对于水平滚动条）将位于滚动条的顶部（或左端）。在最大值时，滑块的下（或右）边缘将位于滚动条的下（或右）端。

滑块的长度通常与页面步长的值相关，并且通常表示滚动视图中显示的文档区域的比例。页步是当用户按下按钮时值改变的量**Page Up**和**Page Down**键，并设置为setPageStep()。使用光标键对线步定义的值进行较小的更改，并且该数量通过setSingleStep()。

请注意，使用的值范围与滚动条小部件的实际大小无关。当您选择范围和页面步长的值时，无需考虑这一点。

为滚动条指定的值范围通常与为滚动条指定的值范围不同。QSlider因为需要考虑滑块的长度。如果我们有一个 100 行的文档，而我们只能在小部件中显示 20 行，我们可能希望构造一个页面步长为 20、最小值为 0、最大值为 80 的滚动条。这将给我们一个有五个“页面”的滚动条。



在许多常见情况下，文档长度、滚动条中使用的值范围和页面步长之间的关系很简单。滚动条的值范围是通过从表示文档长度的某个值中减去选定的页面步长来确定的。在这种情况下，以下等式很有用：*文档长度*=maximum()-minimum() +pageStep()。

QScrollBar仅提供整数范围。请注意，尽管 QScrollBar 可以处理非常大的数字，但当前屏幕上的滚动条无法有效地表示大约 100,000 像素以上的范围。除此之外，用户使用键盘或鼠标控制滑块变得困难，并且滚动箭头的用途也受到限制。

ScrollBar 继承了一组全面的信号QAbstractSlider：

- valueChanged当滚动条的值发生更改时，会发出 () 。Tracking() 确定在用户交互期间是否发出该信号。
- rangeChanged当滚动条的值范围发生更改时，会发出 () 。
- sliderPressed当用户开始拖动滑块时，会发出 ()。
- sliderMoved当用户拖动滑块时会发出 ()。
- sliderReleased当用户释放滑块时，会发出 ()。
- actionTriggered当滚动条通过用户交互或通过triggerAction （） 功能。

滚动条可以通过键盘控制，但它有一个默认值focusPolicy （） 的Qt::NoFocus。使用setFocusPolicy() 启用键盘与滚动条的交互：

- 向左/向右将水平滚动条移动一步。
- 向上/向下将垂直滚动条移动一步。
- PageUp 向上移动一页。
- PageDown 向下移动一页。
- Home 移动到开头（最小）。
- End 移动到末尾（最大）。

滑块本身可以通过使用来控制triggerAction() 函数模拟用户与滚动条控件的交互。如果您有许多不同的小部件使用相同的值范围，这非常有用。

大多数 GUI 样式使用pageStep() 值来计算滑块的大小。

也可以看看QScrollArea,QSlider,QDial,QSpinBox， 和Sliders Example。

成员函数文档

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

构造一个垂直滚动条。

这*parent*参数被发送到QWidget构造函数。

这minimum默认为 0，maximum到 99，带有singleStep1 和 a 的大小pageStep大小为 10，初始值value为 0。

*[explicit]QScrollBar::QScrollBar(Qt::Orientation orientation, QWidget *parent = nullptr)*

使用给定的构造一个滚动条*orientation*。

这*parent*参数被传递给QWidget构造函数。

这minimum默认为 0，maximum到 99，带有singleStep1 和 a 的大小pageStep大小为 10，初始值value为 0。

[virtual]QScrollBar::~QScrollBar()

销毁滚动条。

*[override virtual protected]void QScrollBar::contextMenuEvent(QContextMenuEvent *event)*

重新实现：QWidget::contextMenuEvent（QContextMenuEvent *事件）。

*[override virtual]bool QScrollBar::event(QEvent *event)*

重新实现：QAbstractSlider::event（QEvent *e）。

*[override virtual protected]void QScrollBar::hideEvent(QHideEvent *)*

重新实现：QWidget::hideEvent（QHideEvent *事件）。

*[virtual protected]void QScrollBar::initStyleOption(QStyleOptionSlider *option) const*

初始化*option*与此值QScrollBar。当子类需要时，此方法非常有用QStyleOptionSlider，但不想自己填写所有信息。

也可以看看QStyleOption::initFrom()。

*[override virtual protected]void QScrollBar::mouseMoveEvent(QMouseEvent *e)*

重新实现：QWidget::mouseMoveEvent（QMouseEvent *事件）。

*[override virtual protected]void QScrollBar::mousePressEvent(QMouseEvent *e)*

重新实现：QWidget::mousePressEvent（QMouseEvent *事件）。

*[override virtual protected]void QScrollBar::mouseReleaseEvent(QMouseEvent *e)*

重新实现：QWidget::mouseReleaseEvent（QMouseEvent *事件）。

*[override virtual protected]void QScrollBar::paintEvent(QPaintEvent *)*

重新实现：QWidget::paintEvent（QPaintEvent *事件）。

[override virtual]QSize QScrollBar::sizeHint() const

重新实现属性的访问函数：QWidget::sizeHint。

[override virtual protected]void QScrollBar::sliderChange(QAbstractSlider::SliderChange change)

重新实现：QAbstractSlider::sliderChange（QAbstractSlider::SliderChange 更改）。

*[override virtual protected]void QScrollBar::wheelEvent(QWheelEvent *event)*

重新实现：QAbstractSlider::wheelEvent(QWheelEvent *e)。