# Qt 之 QSlider

# 

# 简述

QSlider部件提供了一个垂直或水平滑动条。

滑块是一个用于控制有界值的典型部件。它允许用户沿水平或垂直方向移动滑块，并将滑块所在的位置转换成一个合法范围内的值。

# 详细描述

QSlider很少有自己的函数，大部分功能在QAbstractSlider中。最有用的函数是setValue()，用来设置滑块的当前值；triggerAction()来模拟点击的效果（对快捷键有用），setSingleStep()、setPageStep()用来设置步长，setMinimum()和setMaximum()用于定义滚动条的范围。

QSlider提供了一些方法来控制刻度标记。可以使用setTickPosition()来表示刻度标记的位置，使用setTickInterval()来指定刻度的间隔；当前设置的刻度位置和间隔可以分别使用tickPosition()和tickInterval()函数来查询。

QSlider继承了一组全面的信号：

| **信号** | **描述** |
| --- | --- |
| valueChanged() | 当滑块的值发生了改变，发射此信号。tracking()确定在用户交互时，是否发出此信号。 |
| sliderPressed() | 当用户按下滑块，发射此信号。 |
| sliderMoved() | 当用户拖动滑块，发射此信号。 |
| sliderReleased() | 当用户释放滑块，发射此信号。 |

QSlider只提供整数范围。

**注意：**尽管QSlider可以处理非常大的数字，但是对于用户来说，难以准确使用很大范围的滑块。

滑块接受Tab键的焦点，并同时提供了一个鼠标滚轮和键盘接口。键盘接口如下：

* Left/Right 移动水平滑块一个步长。
* Up/Down 移动垂直滑块一个步长。
* PageUp 上移一页。
* PageDown 下移一页。
* Home 移动至起始位置（最小值）。
* End 移动至结束位置（最大值）

# 刻度位置

枚举 QSlider::TickPosition

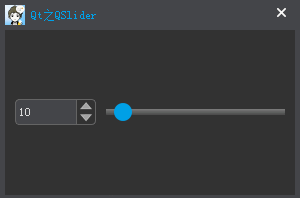
这个枚举指定刻度线相对于滑块和用户操作的位置。

| **常量** | **值** | **描述** |
| --- | --- | --- |
| QSlider::NoTicks | 0 | 不绘制任何刻度线 |
| QSlider::TicksBothSides | 3 | 在滑块的两侧绘制刻度线 |
| QSlider::TicksAbove | 1 | 在（水平）滑块上方绘制刻度线 |
| QSlider::TicksBelow | 2 | 在（水平）滑块下方绘制刻度线 |
| QSlider::TicksLeft | TicksAbove | 在（垂直）滑块左侧绘制刻度线 |
| QSlider::TicksRight | TicksBelow | 在（垂直）滑块右侧绘制刻度线 |

# 基本使用

下面我们来看一个示例，结合QSlider与上节分享的QSpinBox（微调框），来让他们彼此之间相互更新。

## 效果



## 源码

int nMin = 0;

int nMax = 200;

int nSingleStep = 10;

*// 微调框*

QSpinBox \*pSpinBox = new QSpinBox(this);

pSpinBox->setMinimum(nMin); *// 最小值*

pSpinBox->setMaximum(nMax); *// 最大值*

pSpinBox->setSingleStep(nSingleStep); *// 步长*

*// 滑动条*

QSlider \*pSlider = new QSlider(this);

pSlider->setOrientation(Qt::Horizontal); *// 水平方向*

pSlider->setMinimum(nMin); *// 最小值*

pSlider->setMaximum(nMax); *// 最大值*

pSlider->setSingleStep(nSingleStep); *// 步长*

*// 连接信号槽（相互改变）*

connect(pSpinBox, SIGNAL(valueChanged(int)), pSlider, SLOT(setValue(int)));

connect(pSlider, SIGNAL(valueChanged(int)), pSpinBox, SLOT(setValue(int)));

pSpinBox->setValue(10);

通过setMinimum()与setMaximum()函数，我们将取固定到一个合适的范围（0 - 200），连接信号槽后，当QSpinBox的值发生改变时，QSlider的值也会发生相应变化；反之亦然。最后，我们使用setValue()将QSpinBox的值设置为10，由于信号槽已经连接，所以这时QSlider的值也会发生改变。

## QSS

样式一并奉上：

QSlider::groove:horizontal {

height: 6px;

background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 rgb(124, 124, 124), stop: 1.0 rgb(72, 71, 71));

}

QSlider::handle:horizontal {

width: 1px;

background: rgb(0, 160, 230);

margin: -6px 0px -6px 0px;

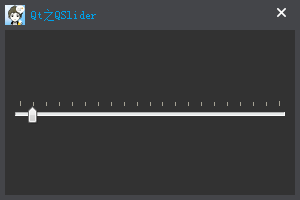
border-radius: 9px;

}

# 刻度位置及间隔

为了演示效果，这里我们不再采用QSS样式控制。

## 效果



## 源码

QSlider \*pSlider = new QSlider(this);

pSlider->setOrientation(Qt::Horizontal); *// 水平方向*

pSlider->setMinimum(nMin); *// 最小值*

pSlider->setMaximum(nMax); *// 最大值*

pSlider->setSingleStep(nSingleStep); *// 步长*

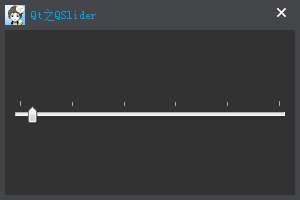
*// pSlider->setTickInterval(40); // 设置刻度间隔*

pSlider->setTickPosition(QSlider::TicksAbove); *//刻度在上方*

由于我们的取值范围是：0 - 200，步长为10。所以，在绘制刻度的时候，一共有21个刻度点（从0开始，每隔步长10绘制一个点，到200处结束）。

当我们使用setTickInterval(40)设置刻度间隔后，在绘制刻度的时候，就会有6个刻度点（从0开始，每隔步长40绘制一个点，到200处结束）。

如下所示：



也就是说，刻度点的个数 = 最大值 / 刻度间隔 + 1。