# Qt 之 QSpinBox 和 QDoubleSpinBox

# 简述

QSpinBox和QDoubleSpinBox均派生自QAbstractSpinBox。

QSpinBox旨在处理整数和离散值（例如：月份名称），QDoubleSpinBox则用于处理浮点值。他们之间的区别就是处理数据的类型不同，其他功能都基本相同。

QDoubleSpinBox的默认的精度是2位小数，但可以通过setDecimals()来改变。

下面主要以QSpinBox为例，来讲解常用的功能。最后部分，会单独分享QDoubleSpinBox的精度设置。

# 详细描述

QSpinBox类提供了一个微调框部件。

QSpinBox允许用户选择一个值，通过单击向上/向下按钮或按下键盘的上/下箭头来增加/减少当前显示的值，用户也可以输入值。微调框支持整数值，但可以被扩展为不同的字符串，使用validate()、textFromValue()和valueFromText()。

当QSpinBox的值发生改变时，会发射两个valueChanged()信号，其中一个提供int类型，另一个则是QString类型，该QString提供了prefix()和suffix()。当前值可以用value()来读取，setValue()来设置。

单击向上/向下按钮或按下键盘的上/下箭头时，将以singleStep()为步长增加/减少当前值。如果想改变这种行为，可以重载虚函数stepBy()。最小值、最大值和步长可以使用其中的一个构造函数来设置，以后可以用setMinimum()、setMaximum()和setSingleStep()来修改。

大多数微调框是定向的，但也可以设置为循环的。例如：如果取值范围是0 - 99，当前值是99，如果wrapping()被设置为true，点击“向上”按钮值会变为0。如果你想要一个循环微调框，可以使用setWrapping()函数。

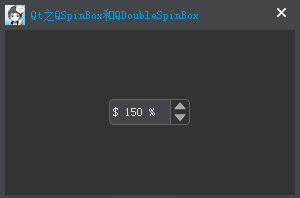
显示的值可以和任意的字符串进行附加，适用setPrefix()和setSuffix()分别可以设置前缀和后缀，例如：货币或计量单位。微调框中的文本可以用text()（包括任何前缀和后缀）或者通过cleanText()（没有前缀()、没有后缀()、无前导或结尾空白）来获取。

除了数值的范围，通常需要使用setSpecialValueText()给用户一个特殊（默认）的选择。

# 基本使用

构建一个QSpinBox，范围：20 - 200，步长：10，开启循环。

## 效果



## 源码

设置步长为10以后，当值发生改变时，就会在当前值的基础上±10（相当于一个等差数列，公差为10）。当我们开启循环后，当前值达到最大或者最小时，就会循环（类似于听歌的列表循环）。

#include <QSpinBox>

class MainWindow : public CustomWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0)

: CustomWindow(parent)

{

*// ...*

QSpinBox \*pSpinBox = new QSpinBox(this);

pSpinBox->setRange(20, 200); *// 范围*

pSpinBox->setSingleStep(10); *// 步长*

pSpinBox->setValue(150); *// 当前值*

pSpinBox->setPrefix("$ "); *// 前缀*

pSpinBox->setSuffix(" %"); *// 后缀*

pSpinBox->setWrapping(true); *// 开启循环*

connect(pSpinBox, static\_cast<void(QSpinBox::\*)(int)>(&QSpinBox::valueChanged),

[=](int value)

{

qDebug() << "Value : " << value;

qDebug() << "Text : " << pSpinBox->text();

});

connect(pSpinBox, static\_cast<void(QSpinBox::\*)(const QString &)>(&QSpinBox::valueChanged),

[=](const QString &text)

{

qDebug() << "Text Value : " << text;

qDebug() << "Clean Text : " << pSpinBox->cleanText();

});

}

};

这里使用了Qt5的信号与槽的语法（后面详细讲解），由于valueChanged()是一个重载信号，所以需要进行参数类别区分。

输出如下：

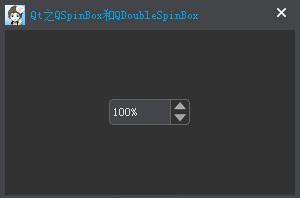
Text Value : "$ 160 %"   
Clean Text : "160"   
Value : 160   
Text : "$ 160 %"

# 特殊文本值

如果设置了specialValueText，只要当前值等于微调框的最小值时，将显示该文本，而不是一个数值。典型的用途是表明此选择具有特殊（默认）的意思。

例如，如果你的微调框允许用户可以选择一个比例系数（或缩放级别），用于显示图像，并且应用程序能够自动选择一个，将使图像完全符合显示窗口，可以像这样设置微调框：

## 效果



## 源码

QSpinBox \*zoomSpinBox = new QSpinBox(this);

zoomSpinBox->setRange(0, 1000); *// 范围*

zoomSpinBox->setSingleStep(10); *// 步长*

zoomSpinBox->setSuffix("%"); *// 前缀*

zoomSpinBox->setSpecialValueText(tr("Automatic")); *// 特殊文本值*

zoomSpinBox->setValue(100); *// 当前值*

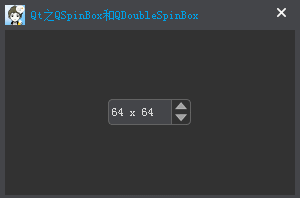
一旦当前值变为最小值时，显示的就是Automatic。

# 自定义

如果使用prefix()、suffix()和specialValueText()没有提供足够的控制，可以子类化QSpinBox，重写valueFromText()和textFromValue()。

例如，自定义一个微调框，允许用户输入图标大小（例如：”32 x 32”）：

## 效果



## 源码

#include <QSpinBox>

class IconSizeSpinBox : public QSpinBox

{

Q\_OBJECT

public:

explicit IconSizeSpinBox(QWidget \*parent = 0){}

protected:

// 将输入的文本解读为适当的值

virtual int valueFromText(const QString &text) const Q\_DECL\_OVERRIDE

{

QRegExp regExp(tr("(\\d+)(\\s\*[xx]\\s\*\\d+)?"));

if (regExp.exactMatch(text)) {

return regExp.cap(1).toInt();

} else {

return 0;

}

}

*// 根据输入的值返回文本*

virtual QString textFromValue(int value) const Q\_DECL\_OVERRIDE

{

return tr("%1 x %1").arg(value);

}

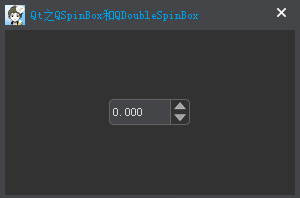
};

# QDoubleSpinBox

上面的所有功能对于QDoubleSpinBox同样适用。

下面，我们看下精度的用法，其实比较简单，一般主要使用setDecimals()设置精度，然后利用setSingleStep()来设置步长即可。

## 效果



## 源码

QDoubleSpinBox \*pSpinBox = new QDoubleSpinBox(this);

pSpinBox->setRange(0, 20); *// 范围*

pSpinBox->setDecimals(3); *// 精度*

pSpinBox->setSingleStep(0.005); *// 步长*

connect(pSpinBox, static\_cast<void(QDoubleSpinBox::\*)(const QString &)>(&QDoubleSpinBox::valueChanged),[=](const QString &text)

{

qDebug() << text;

});