# Qt 之 QAbstractButton

# 简述

QAbstractButton类是按钮部件的抽象基类，提供了按钮所共有的功能。

QAbstractButton类实现了一个抽象按钮，并且让它的子类来指定如何处理用户的动作，并指定如何绘制按钮。

QAbstractButton提供了点击和勾选按钮。QRadioButton和QCheckBox类只提供了勾选按钮，QPushButton和QToolButton提供了点击按钮，如果需要的话，它们还可以提供切换行为。

任何按钮，都可以显示一个包含文本和图标的标签。setText()用来设置文本，setIcon()可以置图标。如果按钮被禁用，其标签更改为“disabled”样式。

# 常用状态

Qt提供的所有按钮（QPushButton、QToolButton、QCheckBox和QRadioButton）都能够既显示文本也可以显示图标。

默认按钮（在很多对话框中被使用的）由QPushButton::setDefault()和QPushButton::setAutoDefault()提供。

QAbstractButton提供了按钮所要用到的绝大多数状态：

* isDown()   
  按钮是否被按下。
* isChecked()   
  按钮是否被选中。只有切换按钮才能被切换为选中或不选中（请看下面）。
* isEnabled()   
  按钮是否可以被用户按下。
* setAutoRepeat()   
  如果用户按下按钮，按钮是否可以自动回复。
* setToggleButton()   
  按钮是否是一个切换按钮。

**isDown()和isChecked()的不同点：**

当用户点击一个切换按钮并且把它选中，按钮首先被按下并且然后被释放变为选中状态。当用户再次点击（非选中），按钮首先被置为按下状态，然后变为非选中状态（isOn()和isChecked()都为false）。

# 信号

QAbstractButton提供了四个信号：

* pressed()   
  当鼠标光标在按钮内，鼠标左键被按下时，发送此信号。
* released()   
  当鼠标左键被释放时，发送此信号。
* clicked()   
  当按钮被按下然后又被释放，或按下快捷键，或当animateClick()被调用时，发送此信号。
* toggled()   
  当切换按钮的状态变化时，发送此信号。

要继承QAbstractButton，你至少不得不重新实现 paintEvent()来绘制按钮的外形、文本、图标。通常也建议重新实现sizeHint()，并且有时也建议重新实现hitButton()（为了决定一个按钮按下是否在按钮内）。

对于具有两个以上状态（如三态按钮）按钮，也将不得不重新实现checkStateSet()和nextCheckState()。

如果一个文本按钮的文本中有“&”，QAbstractButton会创建一个自动快捷键。下面的代码创建一个标签为“Rock & Roll”（c下面有下划线）的点击按钮。这个按钮有一个自动快捷键，Alt+c。更多快捷键文档请参考：QShortcut 。如果要显示一个“&”，请使用’&&’。

QAbstractButton \*pButton = new QPushButton("Ro&ck && Roll", this);

在这个实例中，当用户按下Alt+c时，按钮将会调用animateClick()。

你也可以使用setShortcut()函数来设置一个定制的快捷键。这个对于绝大多数没有文本的按钮是很有用的，因为它们没有自动快捷键。

QAbstractButton \*pButton = new QPushButton(this);

pButton->setIcon(QIcon(":/Images/logo"));

pButton->setShortcut(QKeySequence(Qt::AltModifier + Qt::Key\_F7));

# 使用

通常按钮是要有具体的行为的，也就是说，当我点击一个按钮时，肯定是要有事件发生的。比如：输出一段文本，弹出一个提示框等。

那么，如何响应按钮的操作呢，Qt中通过信号和槽的机制（后面讲解）来联系起来的，我们可以先简单的了解下connect…

QAbstractButton \*pButton = new QPushButton(this);

*// 设置图标、图标大小、文本*

pButton->setIcon(QIcon(":/Images/logo"));

pButton->setIconSize(QSize(18, 18));

pButton->setText("Click Me");

*// 连接信号与槽*

connect(pButton, &QAbstractButton::clicked, this, &MainWindow::onClicked);

void MainWindow::onClicked()

{

qDebug() << "Enter...";

}

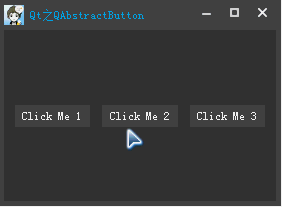
这时，当我们点击按钮时，就会输出信息”Enter…”。

# 选中/非选中

很多时候，我们会看到一些分组按钮，他们可以进行选中高亮，可以多个选中，也可以单个选中。

## 多选

### 效果



### 源码

主要使用QAbstractButton的setCheckable(true)来设置。

...

pButton->setCheckable(true);

...

为了有比较明显效果，我们使用了样式来进行控制。

QPushButton{

background: rgb(68, 69, 73);

border: 1px solid rgb(50, 50, 50);

color: white;

width: 75px;

height: 22px;

}

QPushButton:hover{

background: rgb(85, 85, 85);

}

QPushButton:pressed{

background: rgb(80, 80, 80);

}

QPushButton:checked{

background: green;

}

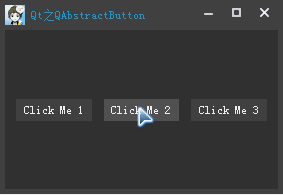
## 互斥

也有很多情况下我们只能选中一个按钮，比如360的导航栏，Button实现毫无压力。。。

主要设置以下两点：

* QPushButton：使用setCheckable(true)设置可选中
* QButtonGroup：设置互斥（多选一），并将QPushButton添加至按钮组。

### 效果



### 源码

QButtonGroup \*pButtonGroup = new QButtonGroup(this);

*// 设置互斥*

pButtonGroup->setExclusive(true);

for (int i = 0; i < 3; ++i)

{

QAbstractButton \*pButton = new QPushButton(this);

*// 设置可选中*

pButton->setCheckable(true);

*// 设置文本*

pButton->setText(QString("Click Me %1").arg(i + 1));

pButtonGroup->addButton(pButton);

}

connect(pButtonGroup, SIGNAL(buttonClicked(QAbstractButton\*)), this, SLOT(onButtonClicked(QAbstractButton\*)));

void MainWindow::onButtonClicked(QAbstractButton \*button)

{

qDebug() << "Clicked Button : " << button->text();

}

关于QAbstractButton常用的功能基本就讲完了，上面介绍的多选、互斥属于高级功能，如果可以很好地运用，在项目中可以实现很棒的效果。