# Qt 之 QCheckBox

# 简述

QCheckBox继承自QAbstractButton，它提供了一个带文本标签的复选框。

QCheckBox（复选框）和QRadioButton（单选框）都是选项按钮。这是因为它们都可以在开（选中）或者关（未选中）之间切换。区别是对用户选择的限制：单选框定义了“多选一”的选择，而复选框提供的是“多选多”的选择。

尽管在技术上可以通过复选框来实现单选框的行为，反之亦然，但还是强烈建议使用众所周知的约定。

# 详细描述

QButtonGroup可以用来在视觉上把许多复选框组织在一起。

只要复选框被选中或者清除，都会发射一个stateChanged()信号。如果想在复选框状态改变的时候触发一个行为，请连接这个信号，可以使用 isChecked()来查询复选框是否被选中。

除了常用的选中和未选中两个状态，QCheckBox还可选地提供了第三种状态（半选）来表明“没有变化”。当需要给用户一个选中或者未选中复选框的选择时，这是很有用的。如果需要第三种状态，可以通过setTristate()来使它生效，并使用checkState()来查询当前的切换状态。

和QPushButton一样，复选框可以显示文本或者图标。文本可以通过构造函数或者setText()来设置，图标可以通过setIcon()来设置。

QCheckBox \*checkbox = new QCheckBox("C&ase sensitive", this);

这个例子中，快捷键为Alt+a，详情请参阅QShortcut文档。要显示一个实际的符号，使用“&&”。

重要的继承函数：text()、setText()、pixmap()、setPixmap()、accel()、setAccel()、isToggleButton()、setDown()、isDown()、isOn()、checkState()、 autoRepeat()、isExclusiveToggle()、group()、setAutoRepeat()、toggle()、pressed()、released()、clicked()、toggled()、checkState()、stateChanged()。

# 共有函数

* Qt::CheckState checkState() const   
  返回复选框的选中状态。如果不需要三态的支持，可以使用QAbstractButton::isChecked()，它返回一个布尔值。
* bool isTristate() const   
  复选框是否为一个三态复选框。

默认的是false，也就是说复选框只有两个状态。

* void setCheckState(Qt::CheckState state)   
  设置复选框的选中状态。如果不需要三态的支持，可以使用QAbstractButton:setChecked()，它接受一个布尔值。
* void setTristate(bool y = true)   
  设置复选框为一个三态复选框。

# 信号

* void stateChanged(int state)   
  当复选框状态发生改变，这个信号就会被发射。即：用户选中或者取消选中。

# 示例

## 三态复选框

下面，我们实现一个三态复选框，并监听状态变化。

### 效果



### 源码

构造一个复选框QCheckBox，然后使用setTristate()开启三态模式。

QCheckBox \*pCheckBox = new QCheckBox(this);

m\_pLabel = new QLabel(this);

m\_pLabel->setText("Click CheckBox...");

pCheckBox->setText(QString::fromLocal8Bit("三态复选框"));

*// 开启三态模式*

pCheckBox->setTristate();

*// 连接信号槽*

connect(pCheckBox, SIGNAL(stateChanged(int)), this, SLOT(onStateChanged(int)));

槽函数，判断当前复选框状态，其中包括：选中（Qt::Checked）、半选（Qt::PartiallyChecked）、未选中（Qt::Unchecked）。

void MainWindow::onStateChanged(int state)

{

if (state == Qt::Checked) *// "选中"*

{

m\_pLabel->setText("Checked");

}

else if(state == Qt::PartiallyChecked) *// "半选"*

{

m\_pLabel->setText("PartiallyChecked");

}

else *// 未选中 - Qt::Unchecked*

{

m\_pLabel->setText("Unchecked");

}

}

### QSS

复选框样式。

QCheckBox{

spacing: 5px;

color: white;

}

QCheckBox::indicator {

width: 17px;

height: 17px;

}

QCheckBox::indicator:enabled:unchecked {

image: url(:/Images/checkBox);

}

QCheckBox::indicator:enabled:unchecked:hover {

image: url(:/Images/checkBoxHover);

}

QCheckBox::indicator:enabled:unchecked:pressed {

image: url(:/Images/checkBoxPressed);

}

QCheckBox::indicator:enabled:checked {

image: url(:/Images/checkBoxChecked);

}

QCheckBox::indicator:enabled:checked:hover {

image: url(:/Images/checkBoxCheckedHover);

}

QCheckBox::indicator:enabled:checked:pressed {

image: url(:/Images/checkBoxCheckedPressed);

}

QCheckBox::indicator:enabled:indeterminate {

image: url(:/Images/checkBoxIndeterminate);

}

QCheckBox::indicator:enabled:indeterminate:hover {

image: url(:/Images/checkBoxIndeterminateHover);

}

QCheckBox::indicator:enabled:indeterminate:pressed {

image: url(:/Images/checkBoxIndeterminatePressed);

}

连接stateChanged()信号和槽函数，当用户点击复选框时，状态发生改变就会调用槽函数。

那么，如果我们不是手动在界面上点击，如何执行槽函数呢？

这时，有些人就会说这很简单啊，看connect就清楚了，直接发射stateChanged()信号或直接调用槽函数：

emit pCheckBox->stateChanged(Qt::PartiallyChecked);

onStateChanged(Qt::PartiallyChecked);

这当然可以。这里我想说的是除了这种方式外，还有木有其它方式呢？我们追朔到QCheckBox的基类QAbstractButton，它里面有一系列的信号和函数，我们可以尝试一下。

经过试验，我们归结了两类，分别是可触发和不可触发槽函数的：

* 不可触发槽函数：

emit pCheckBox->clicked(true);

emit pCheckBox->toggled(true);

emit pCheckBox->pressed();

emit pCheckBox->released();

* 可触发槽函数：

pCheckBox->animateClick();

pCheckBox->click();

pCheckBox->toggle();

pCheckBox->setChecked(true);

pCheckBox->setCheckState(Qt::PartiallyChecked); *// QCheckBox自身接口*

也就是说，通过调用这些信号或函数，我们可以很容易判定其是否可以触发stateChanged()信号，从而调用槽函数onStateChanged()。

## 开关效果

我们来实现一个iphone中常见的开关效果 - 多选。

### 效果



### 源码

构建复选框QCheckBox，然后将它们添加至按钮组QButtonGroup中。

QHBoxLayout \*pLayout = new QHBoxLayout();

m\_pButtonGroup = new QButtonGroup(this);

*// 设置不互斥*

m\_pButtonGroup->setExclusive(false);

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

{

QCheckBox \*pCheckBox = new QCheckBox(this);

*// 设置文本*

pCheckBox->setText(QString::fromLocal8Bit("切换%1").arg(i + 1));

pLayout->addWidget(pCheckBox);

m\_pButtonGroup->addButton(pCheckBox);

}

pLayout->setSpacing(10);

pLayout->setContentsMargins(10, 10, 10, 10);

setLayout(pLayout);

*// 连接信号槽*

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

槽函数，用来判断当前点击的按钮，以及获取按钮组各个按钮的选中状态。

void MainWindow::onButtonClicked(QAbstractButton \*button)

{

*// 当前点击的按钮*

qDebug() << QString("Clicked Button : %1").arg(button->text());

*// 遍历按钮，获取选中状态*

QList<QAbstractButton\*> list = m\_pButtonGroup->buttons();

foreach (QAbstractButton \*pCheckBox, list)

{

QString strStatus = pCheckBox->isChecked() ? "Checked" : "Unchecked";

qDebug() << QString("Button : %1 is %2").arg(pCheckBox->text()).arg(strStatus);

}

}

### QSS

复选框样式

QCheckBox{

spacing: 2px;

color: white;

}

QCheckBox::indicator {

width: 45px;

height: 30px;

}

QCheckBox::indicator:unchecked {

image: url(:/Images/switchOff);

}

QCheckBox::indicator:unchecked:hover {

image: url(:/Images/switchOffHover);

}

QCheckBox::indicator:unchecked:pressed {

image: url(:/Images/switchOffPressed);

}

QCheckBox::indicator:checked {

image: url(:/Images/switchOn);

}

QCheckBox::indicator:checked:hover {

image: url(:/Images/switchOnHover);

}

QCheckBox::indicator:checked:pressed {

image: url(:/Images/switchOnPressed);

}

上面，我们通过调用QButtonGroup的setExclusive(false)来设置按钮组中的复选框不互斥（可以多选）。

如上，我们使用了QButtonGroup，仅仅为了演示复选框实现单选功能（只需设置setExclusive(true)打开互斥即可）。即使这样，如之前所言 - 强烈建议使用众所周知的约定。所以，如果要实现单选功能，建议选择QRadioButton。