QAbstractButton Class

QAbstractButton 类是按钮小部件的抽象基类，提供按钮通用的功能。更多的...

Header:	**#include < QAbstractButton>**
CMake:	find_package(Qt6 REQUIRED COMPONENTS Widgets) target_link_libraries(mytarget PRIVATE Qt6::Widgets)
qmake:	QT += widgets
Inherits:	QWidget
Inherited By:	QCheckBox, QPushButton, QRadioButton, and QToolButton

- 所有成员的列表，包括继承的成员

特性

autoExclusive : boolautoRepeat : boolautoRepeatDelay : intautoRepeatInterval : intcheckable : boolchecked : bool

down : boolicon : QIconiconSize : QSizeshortcut : QKeySequencetext : QString

公共方法

	QAbstractButton(QWidget **parent* = nullptr)
virtual	**~QAbstractButton**()
bool	**autoExclusive**() const
bool	**autoRepeat**() const
int	**autoRepeatDelay**() const
int	**autoRepeatInterval**() const
QButtonGroup *	**group**() const
QIcon	**icon**() const
QSize	**iconSize**() const
bool	**isCheckable**() const
bool	**isChecked**() const
bool	**isDown**() const

	QAbstractButton(QWidget **parent* = nullptr)
void	**setAutoExclusive**(bool)
void	**setAutoRepeat**(bool)
void	**setAutoRepeatDelay**(int)
void	**setAutoRepeatInterval**(int)
void	**setCheckable**(bool)
void	**setDown**(bool)
void	**setIcon**(const QIcon &*icon*)
void	**setShortcut**(const QKeySequence &*key*)
void	**setText**(const QString &*text*)
QKeySequence	**shortcut**() const
QString	**text**() const

公共槽

void	**animateClick()**
void	**click**()
void	**setChecked**(bool)
void	**setIconSize**(const QSize &*size*)
void	**toggle**()

信号

void	**clicked(bool *checked* = false)**
void	**pressed**()
void	**released**()
void	**toggled**(bool *checked*)

protected function

virtual void	**checkStateSet()**
virtual bool	**hitButton**(const QPoint &*pos*) const
virtual void	**nextCheckState**()

重载的protected function

virtual void	**changeEvent(QEvent *e) override**
virtual bool	**event**(QEvent *e) override
virtual void	**focusInEvent**(QFocusEvent *e) override
virtual void	**focusOutEvent**(QFocusEvent *e) override
virtual void	**keyPressEvent**(QKeyEvent *e) override
virtual void	**keyReleaseEvent**(QKeyEvent *e) override
virtual void	**mouseMoveEvent**(QMouseEvent *e) override
virtual void	**mousePressEvent**(QMouseEvent *e) override
virtual void	**mouseReleaseEvent**(QMouseEvent *e) override
virtual void	**paintEvent**(QPaintEvent *e) override = 0
virtual void	**timerEvent**(QTimerEvent *e) override

详细说明

该类实现了一个*抽象*按钮。此类的子类处理用户操作，并指定按钮的绘制方式。

QAbstractButton 提供对普通按钮和可检查（切换）按钮的支持。可检查的按钮在QRadioButton和QCheckBox类。按钮是在QPushButton和QToolButton课程；如果需要的话，它们还提供切换行为。

任何按钮都可以显示包含文本和图标的标签。setText() 设置文本；setIcon() 设置图标。如果按钮被禁用，其标签将更改以使按钮呈现“禁用”外观。

如果该按钮是一个文本按钮，其字符串包含与号（'&'），QAbstractButton 会自动创建一个快捷键。例如：

```
QPushButton *button = new QPushButton(tr("Ro&ck && Roll"), this);
```

这**Alt+C**快捷键被分配给按钮，即当用户按下时**Alt+C**该按钮将调用animateClick()。请参阅QShortcut文档了解详细信息。要显示实际的 & 符号，请使用“&&”。

您还可以使用以下命令设置自定义快捷键setShortcut （） 功能。这对于没有任何文本的按钮非常有用，因此不能有任何自动快捷方式。

```
button->setIcon(QIcon(":/images/print.png"));
button->setShortcut(tr("Alt+F7"));
```

Qt 提供的所有按钮（QPushButton,QToolButton,QCheckBox， 和QRadioButton) 可以同时显示text和icons。

可以通过以下方式将按钮设置为对话框中的默认按钮QPushButton::setDefault（） 和QPushButton::setAutoDefault()。

QAbstractButton 提供了大多数用于按钮的状态：

- isDown()表示按钮是否*按下*。
- isChecked() 表示该按钮是否被*选中*。只能选中和取消选中可选中的按钮（见下文）。
- isEnabled

  () 表示该按钮是否可以被用户按下。

  注意：与其他小部件不同，从 QAbstractButton 派生的按钮在禁用时接受鼠标和上下文菜单事件。
- setAutoRepeat() 设置如果用户按住按钮是否会自动重复。autoRepeatDelay和autoRepeatInterval定义如何完成自动重复。
- setCheckable() 设置按钮是否为切换按钮。

和...之间的不同isDown（） 和isChecked()如下。当用户单击切换按钮进行检查时，首先*按下*按钮，然后释放按钮进入*检查*状态。当用户再次单击它（取消选中它）时，按钮首先移动到按下*状态*，然后移动到*未选中*状态（isChecked（） 和isDown() 均为假）。

QAbstractButton 提供四种信号：

1. pressed当鼠标光标位于按钮内并按下鼠标左键时，会发出 () 。
2. released释放鼠标左键时会发出 ()。
3. clicked当第一次按下按钮然后释放时，当键入快捷键时，或者当click（） 或者animateClick（） 叫做。
4. toggled当切换按钮的状态发生变化时，会发出 ()。

要子类化 QAbstractButton，您必须至少重新实现paintEvent() 绘制按钮的轮廓及其文本或像素图。一般建议重新实现sizeHint() 也是如此，有时hitButton()（确定按钮按下是否在按钮内）。对于具有两种以上状态的按钮（例如三态按钮），您还必须重新实现checkStateSet（） 和nextCheckState()。

也可以看看QButtonGroup。

属性文档

autoExclusive : bool

该属性保存是否启用自动排他性

如果启用自动独占性，则属于同一父窗口小部件的可检查按钮的行为就像它们是同一独占按钮组的一部分一样。专属按钮组中，任何时候只能选中一个按钮；选中另一个按钮会自动取消选中之前选中的按钮。

该属性对属于按钮组的按钮没有影响。

默认情况下，autoExclusive 处于关闭状态，单选按钮除外。

访问功能：

bool	**autoExclusive() const**

bool	autoExclusive() const
void	setAutoExclusive(bool)

也可以看看QRadioButton。

autoRepeat : bool

该属性保存是否启用autoRepeat

如果启用自动重复，则pressed(),released（），和clicked当按下按钮时，会定期发出 () 信号。默认情况下，自动重复处于关闭状态。初始延迟和重复间隔以毫秒为单位定义为autoRepeatDelay和autoRepeatInterval。

注意：如果通过快捷键按下按钮，则自动重复将由系统而不是此类启用并计时。这pressed(),released（），和clicked() 信号将像正常情况下一样发出。

访问功能：

bool	autoRepeat() const
void	setAutoRepeat(bool)

autoRepeatDelay : int

该属性保存自动重复的初始延迟

如果autoRepeat启用后，autoRepeatDelay 定义自动重复开始之前的初始延迟（以毫秒为单位）。

访问功能：

int	autoRepeatDelay() const
void	setAutoRepeatDelay(int)

也可以看看autoRepeat和autoRepeatInterval。

autoRepeatInterval : int

该属性保存自动重复的间隔

如果autoRepeat启用后，autoRepeatInterval 定义自动重复间隔的长度（以毫秒为单位）。

访问功能：

int	autoRepeatInterval() const
void	setAutoRepeatInterval(int)

也可以看看autoRepeat和autoRepeatDelay。

checkable : bool

该属性保存按钮是否可检查

默认情况下，该按钮不可选中。

访问功能：

bool	**isCheckable() const**
void	**setCheckable**(bool)

也可以看看checked。

checked : bool

该属性保存按钮是否被选中

只能检查可检查的按钮。默认情况下，该按钮处于未选中状态。

访问功能：

bool	**isChecked() const**
void	**setChecked**(bool)

通知器信号：

void	**toggled**(**bool *checked***)

也可以看看checkable。

down : bool

该属性保存按钮是否被按下

如果此属性为`true`，则按钮被按下。信号pressed () 和clicked如果将此属性设置为 true，则不会发出 ()。默认为 false。

访问功能：

bool	**isDown() const**

bool	isDown() const
void	**setDown**(bool)

icon : QIcon

该属性保存按钮上显示的图标

图标的默认大小由 GUI 样式定义，但可以通过设置来调整iconSize财产。

访问功能：

QIcon	icon() const
void	**setIcon**(const QIcon &*icon*)

iconSize : QSize

该属性保存用于该按钮的图标大小。

默认大小由 GUI 样式定义。这是图标的最大尺寸。较小的图标不会放大。

访问功能：

QSize	iconSize() const
void	**setIconSize**(const QSize &*size*)

shortcut : QKeySequence

该属性保存与按钮关联的助记符

访问功能：

QKeySequence	shortcut() const
void	**setShortcut**(const QKeySequence &*key*)

text : QString

该属性保存按钮上显示的文本

如果按钮没有文本，text() 函数将返回空字符串。

如果文本包含与字符 ('&')，则会自动为其创建快捷方式。“&”后面的字符将用作快捷键。如果文本没有定义快捷方式，则任何先前的快捷方式都将被覆盖或清除。请参阅QShortcut文档了解详细信息。要显示实际的 & 符号，请使用“&&”。

没有默认文本。

访问功能：

QString	text() const
void	**setText**(const QString &*text*)

成员函数文档

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

构造一个抽象按钮*parent*。

[virtual]QAbstractButton::~QAbstractButton()

销毁按钮。

[slot]void QAbstractButton::animateClick()

执行动画单击：按钮立即按下，100 毫秒后释放。

在释放按钮之前再次调用此函数将重置释放计时器。

与点击相关的所有信号都会根据需要发出。

如果按钮是，则此函数不执行任何操作disabled.

也可以看看click()。

*[override virtual protected]void QAbstractButton::changeEvent(QEvent *e)*

重新实现：QWidget::changeEvent（QEvent *事件）。

[virtual protected]void QAbstractButton::checkStateSet()

该虚拟处理程序在以下情况下被调用setChecked使用 ()，除非从内部调用nextCheckState()。它允许子类重置其中间按钮状态。

也可以看看nextCheckState()。

[slot]void QAbstractButton::click()

执行单击操作。

所有与点击相关的常用信号都会根据需要发出。如果该按钮是可选的，则该按钮的状态将被切换。

如果按钮是，则此函数不执行任何操作disabled.

也可以看看animateClick()。

[signal]void QAbstractButton::clicked(bool checked = false)

当按钮被激活（即，当鼠标光标位于按钮内时按下然后释放）、键入快捷键时或当click（） 或者animateClick（） 叫做。值得注意的是，如果您调用，*则不会*发出此信号setDown(),setChecked（） 或者toggle()。

如果该按钮是可选的，*checked*如果选中按钮，则为 true；如果未选中按钮，则为 false。

也可以看看pressed(),released（），和toggled()。

*[override virtual protected]bool QAbstractButton::event(QEvent *e)*

重新实现：QWidget::event（QEvent *事件）。

*[override virtual protected]void QAbstractButton::focusInEvent(QFocusEvent *e)*

重新实现：QWidget::focusInEvent（QFocusEvent *事件）。

*[override virtual protected]void QAbstractButton::focusOutEvent(QFocusEvent *e)*

重新实现：QWidget::focusOutEvent（QFocusEvent *事件）。

*QButtonGroup *QAbstractButton::group() const*

返回该按钮所属的组。

如果该按钮不属于任何QButtonGroup，该函数返回`nullptr`。

也可以看看QButtonGroup。

[virtual protected]bool QAbstractButton::hitButton(const QPoint &pos) const

返回`true`如果*pos*位于可点击按钮矩形内；否则返回`false`.

默认情况下，可点击区域是整个小部件。子类可以重新实现此功能，以提供对不同形状和大小的可点击区域的支持。

*[override virtual protected]void QAbstractButton::keyPressEvent(QKeyEvent *e)*

重新实现：QWidget::keyPressEvent（QKeyEvent *事件）。

*[override virtual protected]void QAbstractButton::keyReleaseEvent(QKeyEvent *e)*

重新实现：QWidget::keyReleaseEvent（QKeyEvent *事件）。

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

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

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

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

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

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

[virtual protected]void QAbstractButton::nextCheckState()

单击按钮时会调用此虚拟处理程序。默认实现调用setChecked（！isChecked()）如果按钮isCheckable()。它允许子类实现中间按钮状态。

也可以看看checkStateSet()。

*[override pure virtual protected]void QAbstractButton::paintEvent(QPaintEvent *e)*

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

[signal]void QAbstractButton::pressed()

当按下按钮时会发出此信号。

也可以看看released（） 和clicked()。

[signal]void QAbstractButton::released()

释放按钮时会发出此信号。

也可以看看pressed(),clicked（），和toggled()。

[override virtual protected]void QAbstractButton::timerEvent(QTimerEvent *e)

重新实现：QObject::timerEvent（QTimerEvent *事件）。

[slot]void QAbstractButton::toggle()

切换可检查按钮的状态。

也可以看看checked。

[signal]void QAbstractButton::toggled(bool checked)

每当可检查按钮改变其状态时，就会发出此信号。*checked*如果选中按钮，则为 true；如果未选中按钮，则为 false。

这可能是用户操作的结果，click() 槽激活，或者因为setChecked（） 叫做。

在发出该信号之前，独占按钮组中按钮的状态会被更新。这意味着插槽可以对状态已更改的组中的按钮发出的“关闭”信号或“打开”信号起作用。

例如，一个对新选中的按钮发出的信号做出反应但忽略来自未选中的按钮的信号的插槽可以使用以下模式实现：

```
void MyWidget::reactToToggle(bool checked)
{
   if (checked) {
      // Examine the new button states.
      ...
   }
}
```

可以使用以下命令创建按钮组QButtonGroup类，并更新使用监视的按钮状态QButtonGroup::buttonClicked（） 信号。

注意：属性的通知程序信号checked。

也可以看看checked和clicked()。