QDialog Class

QDialog 类是对话框窗口的基类。更多的...

Header:	**#include**
CMake:	find_package(Qt6 REQUIRED COMPONENTS Widgets) target_link_libraries(mytarget PRIVATE Qt6::Widgets)
qmake:	QT += widgets
Inherits:	QWidget
Inherited By:	QColorDialog, QErrorMessage, QFileDialog, QFontDialog, QInputDialog, QMessageBox, QProgressDialog, and QWizard

- 所有成员的列表，包括继承的成员

公共类型

enum	**DialogCode { Accepted, Rejected }**

特性

- **modal** : bool
- **sizeGripEnabled** : bool

公共方法

	QDialog(QWidget **parent* = nullptr, Qt::WindowFlags *f* = Qt::WindowFlags())
virtual	**~QDialog**()
bool	**isSizeGripEnabled**() const
int	**result**() const
void	**setModal**(bool *modal*)

	QDialog(QWidget **parent* = nullptr, Qt::WindowFlags *f* = Qt::WindowFlags())
void	**setResult**(int *i*)
void	**setSizeGripEnabled**(bool)

重载的公共方法

virtual QSize	**minimumSizeHint() const override**
virtual void	**setVisible**(bool *visible*) override
virtual QSize	**sizeHint**() const override

公共槽

virtual void	**accept()**
virtual void	**done**(int *r*)
virtual int	**exec**()
virtual void	**open**()
virtual void	**reject**()

信号

void	**accepted()**
void	**finished**(int *result*)
void	**rejected**()

重载的protected function

virtual void	**closeEvent(QCloseEvent **e*) override**
virtual void	**contextMenuEvent**(QContextMenuEvent **e*) override
virtual bool	**eventFilter**(QObject **o*, QEvent** e*) override
virtual void	**keyPressEvent**(QKeyEvent **e*) override
virtual void	**resizeEvent**(QResizeEvent *) override

virtual void **showEvent**(QShowEvent *event) override
virtual void closeEvent(QCloseEvent *e) override

详细说明

对话窗口是顶级窗口，主要用于短期任务和与用户的简短通信。QDialog 可以是模态的也可以是非模态的。QDialogs 可以提供return value，他们可以有default buttons。QDialogs 还可以有一个QSizeGrip在右下角，使用setSizeGripEnabled()。

请注意，QDialog（以及具有 type 的任何其他小部件`Qt::Dialog`）使用父小部件与 Qt 中的其他类略有不同。对话框始终是顶级小部件，但如果它有父级，则其默认位置位于父级顶级小部件顶部的中心（如果它本身不是顶级）。它还将共享父级的任务栏条目。

使用的过载QWidget::setParent() 函数更改 QDialog 小部件的所有权。此函数允许您显式设置重新设置父级的小部件的窗口标志；使用重载函数将清除指定小部件窗口系统属性的窗口标志（特别是它将重置Qt::Dialog旗帜）。

注意：对话框的父关系并不*意味着*对话框将始终堆叠在父窗口之上。为了确保对话框始终位于顶部，请将对话框设为模态。这也适用于对话框本身的子窗口。为了确保对话框的子窗口保持在对话框的顶部，也使子窗口成为模态的。

模态对话框

模式对话框是阻止对同一应用程序中其他可见窗口**进行**输入的对话框。用于向用户请求文件名或用于设置应用程序首选项的对话框通常是模态的。对话框可以是application modal（默认）或window modal。

打开应用程序模式对话框时，用户必须完成与该对话框的交互并关闭它，然后才能访问应用程序中的任何其他窗口。窗口模式对话框仅阻止对与该对话框关联的窗口的访问，从而允许用户继续使用应用程序中的其他窗口。

显示模式对话框最常见的方法是调用它exec（） 功能。当用户关闭对话框时，exec() 将提供有用的return value。要关闭对话框并返回适当的值，您必须连接一个默认按钮，例如**OK**按钮到accept() 槽和一个**Cancel**按钮到reject（） 投币口。或者，您可以致电done() 插槽带有`Accepted`或`Rejected`。

另一种方法是致电setModal（正确）或setWindowModality（），然后show()。不像exec(),show() 立即将控制权返回给调用者。呼唤setModal(true) 对于进度对话框特别有用，其中用户必须能够与对话框交互，例如取消长时间运行的操作。如果你使用show（） 和setModal(true) 一起执行长操作，必须调用QCoreApplication::processEvents() 在处理过程中定期进行，以使用户能够与对话框交互。（看QProgressDialog.）

无模式对话框

无**模式**对话框是独立于同一应用程序中的其他窗口运行的对话框。文字处理程序中的查找和替换对话框通常是无模式的，以允许用户与应用程序的主窗口和对话框进行交互。

非模式对话框显示使用show()，它立即将控制权返回给调用者。

如果您调用show() 函数隐藏对话框后，对话框将显示在原来的位置。这是因为窗口管理器决定了程序员未明确放置的窗口的位置。要保留用户移动的对话框的位置，请将其位置保存在您的closeEvent() 处理程序，然后将对话框移动到该位置，然后再次显示它。

默认按钮

对话框的*默认*按钮是当用户按 Enter（回车）键时按下的按钮。该按钮用于表示用户接受对话框的设置并想要关闭对话框。使用QPushButton::setDefault(),QPushButton::isDefault （） 和QPushButton::autoDefault() 设置和控制对话框的默认按钮。

退出键

如果用户在对话框中按 Esc 键，QDialog::reject() 将被调用。这将导致窗口关闭：close event不可能是ignored。

可扩展性

可扩展性是指以两种方式显示对话框的能力：显示最常用选项的部分对话框和显示所有选项的完整对话框。通常，可扩展对话框最初会显示为部分对话框，但带有**More**切换按钮。如果用户按下**More**按下按钮，对话框展开。

返回值（模态对话框）

模态对话框通常用于需要返回值的情况，例如指示用户是否按下**OK**或者**Cancel**。可以通过调用关闭对话框accept （） 或者reject() 插槽，以及exec() 将返回`Accepted`或`Rejected`视情况而定。这exec() 调用返回对话框的结果。结果也可从result() 如果对话框尚未被销毁。

为了修改对话框的关闭行为，您可以重新实现以下功能accept(),reject （） 或者done()。这closeEvent() 函数仅应重新实现以保留对话框的位置或覆盖标准关闭或拒绝行为。

代码示例

模态对话框：

```
void EditorWindow::countWords()
{
    WordCountDialog dialog(this);
    dialog.setWordCount(document().wordCount());
    dialog.exec();
}
```

无模式对话框：

```
void EditorWindow::find()
{
    if (!findDialog) {
        findDialog = new FindDialog(this);
        connect(findDialog, &FindDialog::findNext,
                this, &EditorWindow::findNext);
    }

    findDialog->show();
    findDialog->raise();
    findDialog->activateWindow();
}
```

带有扩展名的对话框：

```
    findButton = new QPushButton(tr("&Find"));
    moreButton = new QPushButton(tr("&More..."));
    moreButton->setCheckable(true);

    extension = new ExtendedControls;
    mainLayout->addWidget(extension);
    extension->hide();

    connect(moreButton, &QAbstractButton::toggled, extension, &QWidget::setVisible);
```

可以参阅QDialogButtonBox,QTabWidget,QWidget,QProgressDialog， 和Standard Dialogs Example。

成员类型文档

enum QDialog::DialogCode

模式对话框返回的值。

持续的	价值
QDialog::Accepted	1
QDialog::Rejected	0

属性文档

modal : bool

该属性是否持有show() 应该弹出模态或非模态对话框

默认情况下，该属性是false并且show() 以无模式方式弹出对话框。将此属性设置为 true 相当于设置QWidget::windowModality到Qt::ApplicationModal。

exec() 忽略该属性的值并始终以模态方式弹出对话框。

访问功能：

bool	**isModal() const**
void	**setModal**(bool *modal*)

可以参阅QWidget::windowModality,show（），和exec()。

sizeGripEnabled : bool

该属性保存是否启用尺寸夹点

AQSizeGrip启用此属性后，将放置在对话框的右下角。默认情况下，尺寸夹点处于禁用状态。

访问功能：

bool	**isSizeGripEnabled() const**
void	**setSizeGripEnabled**(bool)

成员函数文档

*[explicit]QDialog::QDialog(QWidget *parent = nullptr, Qt::WindowFlags f = Qt::WindowFlags())*

与父级构建对话*parent*。

对话框始终是顶级小部件，但如果它有父级，则其默认位置位于父级顶部的中心。它还将共享父级的任务栏条目。

小部件标志*f*被传递到QWidget构造函数。例如，如果您不想在对话框的标题栏中显示“这是什么”按钮，请传递Qt::WindowTitleHint|Qt::WindowSystemMenuHint在*f*。

可以参阅QWidget::setWindowFlags()。

[virtual]QDialog::~QDialog()

摧毁了QDialog，删除其所有子项。

[virtual slot]void QDialog::accept()

隐藏模式对话框并将结果代码设置为Accepted。

可以参阅reject（） 和done()。

[signal]void QDialog::accepted()

当用户或通过调用接受对话框时，会发出此信号accept（） 或者done() 与QDialog::Accepted争论。

请注意，隐藏对话框时*不会*发出此信号hide（） 或者setVisible（错误的）。这包括在对话框可见时将其删除。

可以参阅finished（） 和rejected()。

*[override virtual protected]void QDialog::closeEvent(QCloseEvent *e)*

重新实现：QWidget::closeEvent（QCloseEvent *事件）。

*[override virtual protected]void QDialog::contextMenuEvent(QContextMenuEvent *e)*

重新实现：QWidget::contextMenuEvent（QContextMenuEvent *事件）。

[virtual slot]void QDialog::done(int r)

关闭对话框并将其结果代码设置为*r*。这finished() 信号将发出*r*; 如果*r*是QDialog::Accepted或者QDialog::Rejected，这accepted（） 或者rejected() 信号也将分别发出。

如果此对话框显示为exec()、done() 也会导致本地事件循环完成，并且exec（） 回来*r*。

与QWidget::close()、done() 删除对话框，如果Qt::WA_DeleteOnClose标志已设置。如果该对话框是应用程序的主窗口部件，则应用程序将终止。如果该对话框是最后关闭的窗口，则QGuiApplication::lastWindowClosed() 信号被发射。

可以参阅accept(),reject(),QApplication::activeWindow（） ， 和QCoreApplication::quit()。

[override virtual protected]bool QDialog::eventFilter(QObject o, QEvent e*)*

重新实现：QObject::eventFilter（QObject *观看，QEvent *事件）。

[virtual slot]int QDialog::exec()

将对话框显示为modal dialog，阻塞直到用户关闭它。该函数返回一个DialogCode结果。

如果对话框是application modal，用户在关闭对话框之前无法与同一应用程序中的任何其他窗口交互。如果对话框是window modal，当对话框打开时，仅阻止与父窗口的交互。默认情况下，该对话框是应用程序模式的。

注意：避免使用该功能；相反，使用open()。与 exec() 不同，open() 是异步的，并且不会旋转额外的事件循环。这可以防止发生一系列危险的错误（例如，在通过 exec() 打开对话框时删除对话框的父级）。使用时open() 您可以连接到finished() 信号QDialog当对话框关闭时收到通知。

可以参阅open(),show(),result（），和setWindowModality()。

[signal]void QDialog::finished(int result)

当对话框的*result*代码已由用户或通过调用设置done(),accept（），或者reject()。

请注意，隐藏对话框时不会发出此信号hide（）或者setVisible（错误的）。这包括在对话框可见时将其删除。

可以参阅accepted（）和rejected()。

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

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

[override virtual]QSize QDialog::minimumSizeHint() const

重新实现属性的访问函数：QWidget::minimumSizeHint。

[virtual slot]void QDialog::open()

将对话框显示为window modal dialog，立即返回。

可以参阅exec(),show(),result（），和setWindowModality()。

[virtual slot]void QDialog::reject()

隐藏模式对话框并将结果代码设置为Rejected。

可以参阅accept（） 和done()。

[signal]void QDialog::rejected()

当对话框被用户或调用拒绝时，会发出此信号reject（） 或者done() 与QDialog::Rejected争论。

请注意，隐藏对话框时不会发出此信号hide（） 或者setVisible（错误的）。这包括在对话框可见时将其删除。

可以参阅finished（） 和accepted()。

[override virtual protected]void QDialog::resizeEvent(QResizeEvent *)

重新实现：QWidget::resizeEvent（QResizeEvent *事件）。

int QDialog::result() const

通常返回模式对话框的结果代码，Accepted或Rejected.

注意：当调用QMessageBox实例，返回值是QMessageBox::StandardButton枚举。

如果对话框是用以下函数构造的，则不要调用此函数Qt::WA_DeleteOnClose属性。

可以参阅setResult()。

void QDialog::setResult(int i)

将模式对话框的结果代码设置为*i*。

注意：我们建议您使用由定义的值之一QDialog::DialogCode。

可以参阅result()。

[override virtual]void QDialog::setVisible(bool visible)

重新实现属性的访问函数：QWidget::visible。

*[override virtual protected]void QDialog::showEvent(QShowEvent *event)*

重新实现：QWidget::showEvent（QShowEvent *事件）。

[override virtual]QSize QDialog::sizeHint() const

重新实现属性的访问函数：QWidget::sizeHint。