# Qt 之模式、非模式、半模式对话框

# 简述

关于“模式”和“非模式”对话框，相信大家都比较熟悉，但其中有一个可能很多人都比较陌生，介于两者之间的状态，我们称之为“半模式“。

# 

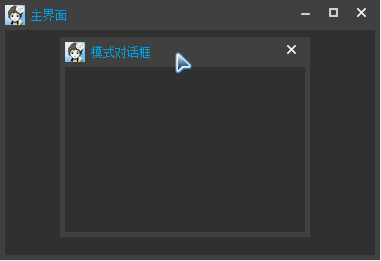
# 模式对话框

## 描述

阻塞同一应用程序中其它可视窗口输入的对话框。模式对话框有自己的事件循环，用户必须完成这个对话框中的交互操作，并且关闭了它之后才能访问应用程序中的其它任何窗口。模式对话框仅阻止访问与对话相关联的窗口，允许用户继续使用其它窗口中的应用程序。

显示模态对话框最常见的方法是调用其exec()函数，当用户关闭对话框，exec()将提供一个有用的返回值，并且这时流程控制继续从调用exec()的地方进行。通常情况下，要获得对话框关闭并返回相应的值，我们连接默认按钮，例如：”确定”按钮连接到accept()槽，”取消”按钮连接到reject()槽。另外我们也可以连接done()槽，传递给它Accepted或Rejected。

## 效果



## 源码

MainWindow \*pMainWindow = new MainWindow();

pMainWindow->setWindowTitle(QStringLiteral("主界面"));

pMainWindow->show();

CustomWindow \*pDialog = new CustomWindow(pMainWindow);

pDialog->setWindowTitle(QStringLiteral("模式对话框"));

*// 关键代码*

pDialog->exec();

*// 关闭模态对话框以后才会执行下面的代码*

pMainWindow->setWindowTitle(QStringLiteral("主界面-模式对话框"));

qDebug() << QStringLiteral("关闭模态对话框以后，可以继续向下执行");

1. 主界面被阻塞，不能进行点击、拖动等任何操作。
2. exec()之后的代码不会执行，直到关闭模态对话框。

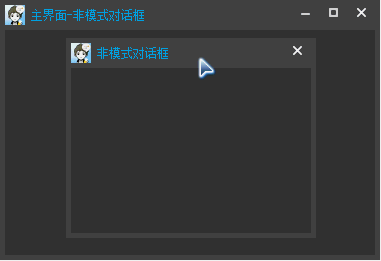
# 非模式对话框

## 描述

和同一个程序中其它窗口操作无关的对话框。在文字处理中的查找和替换对话框通常是非模式的，允许用户同时与应用程序的主窗口和对话框进行交互。调用show()来显示非模式对话框，并立即将控制返回给调用者。

如果隐藏对话框后调用show()函数，对话框将显示在其原始位置，这是因为窗口管理器决定的窗户位置没有明确由程序员指定，为了保持被用户移动的对话框位置，在closeEvent()中进行处理，然后在显示之前，将对话框移动到该位置。

## 效果



## 源码

MainWindow \*pMainWindow = new MainWindow();

pMainWindow->setWindowTitle(QStringLiteral("主界面"));

pMainWindow->show();

CustomWindow \*pDialog = new CustomWindow(pMainWindow);

pDialog->setWindowTitle(QStringLiteral("非模式对话框"));

*// 关键代码*

pDialog->show();

*// 下面的代码会立即运行*

pMainWindow->setWindowTitle(QStringLiteral("主界面-非模式对话框"));

qDebug() << QStringLiteral("立即运行");

1. 主界面不会被阻塞，可以进行点击、拖动等任何操作。
2. show()之后的代码会立即执行。

# 

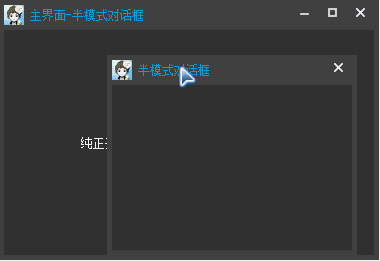
# 半模式对话框

## 描述

调用setModal(true)或者setWindowModality()，然后show()。有别于exec()，show() 立即返回给控制调用者。

对于进度对话框来说，调用setModal(true)是非常有用的，用户必须拥有与其交互的能力，例如：取消长时间运行的操作。如果使用show()和setModal(true)共同执行一个长时间操作，则必须定期在执行过程中调用QApplication ::processEvents()，以使用户能够与对话框交互（可以参考QProgressDialog）。

## 效果



## 源码

MainWindow \*pMainWindow = new MainWindow();

pMainWindow->setWindowTitle(QStringLiteral("主界面"));

pMainWindow->show();

CustomWindow \*pDialog = new CustomWindow(pMainWindow);

pDialog->setWindowTitle(QStringLiteral("半模式对话框"));

*// 关键代码*

pDialog->setModal(true);

pDialog->show();

*// 下面的代码会立即运行*

pMainWindow->setWindowTitle(QStringLiteral("主界面-半模式对话框"));

qDebug() << QStringLiteral("立即运行");

1. 主界面被阻塞，不能进行点击、拖动等任何操作。
2. show()之后的代码会立即执行。

# Qt - QDialog,QWidget实现模态及非模态

对于 QDialog 的模态及非模态是直接可以实现的，很多课本中都会提到，此处总结下。

**模态QDialog**

方式一：

QDialog dlg(this); dlg.exec();

方式二：

QDialog \*pDlg=new QDialog(this); pDlg->setModal(true); pDlg->show();

**非模态QDialog**

QDialog \*pDlg=new QDialog(this); pDlg->show();

QDialog实现模态非模态很简单，但是对于QWidget有点迷茫，QWidget中没有exec()，也没有setModal()方式，但是想想看，QWidget作为QDialog的基类，而且QWidget作为“窗口”使用也是在平常不过了，所以会意识到QWidget中是否存在一个相对exec()或setModal()更基本的操作来实现模态和非模态呢？就这样，我找到了setWindowModality()，此函数就是用来设置QWidget运行时的程序阻塞方式的，参数解释如下：

Qt::**NonModal**不阻塞

Qt::**WindowModal**阻塞父窗口，所有祖先窗口及其子窗口

Qt::**ApplicationModal**阻塞整个应用程序

看来，**setModal**()也就是使用**setWindowModality**()设置Qt::**ApplicationModal**参数来实现的模态。

如此，要实现QWidget的模态和非模态，只要调用**setWindowModality**()设置阻塞类型就好了：

// QWidget \*pWid = new QWidget(this); ----- 注意这样设置不能实现非模态，改成如下

QWidget \*pWid = new QWidget (NULL);

pWid->setWindowModality(Qt::ApplicationModal);

//pWid->setAttribute(Qt::WA\_ShowModal, true);

pWid->show();

但是运行发现并未实现模态效果。这里需要注意，当希望使用**setWindowModality**()将QWidget设置为模态时应该保证QWidget父部件为0，这里修改QWidget \*pWid = new QWidget(this);为QWidget \*pWid = new QWidget(NULL);在运行就好了。

此外，通过**setWindowModality**()设置模态窗口并不是唯一方式，直接设置部件（或窗口）属性也可以：

pWid->setAttribute(Qt::WA\_ShowModal, true)

---------------

还有很多地方需要注意，当创建QDialog后使用setWindowFlags(Qt::FramelessWindowHint);去掉标题栏时此对话框不再阻塞父窗口，如果需要实现阻塞效果可再次指定Qt::Dialog，即使用：

**setWindowFlags**(Qt::**Dialog**| Qt::**FramelessWindowHint**); //这样就会阻塞父窗口了！

但是，这样会影响对话框的半透明（或透明）显示。使用Qt::Dialog之前半透明显示正常：

**总而言之**

是否是模态和QDialog 和QWidget都可以模态和非模态.exec(), show() 等函数无直接关系，只和窗口属性有关,使用以下两种方式都行：

setAttribute(Qt::WA\_ShowModal, true);//属性设置

setWindowModality(Qt::ApplicationModal);//设置阻塞类型

-----------------

QDialog中的成员函数setModal(true)及exec()之所以是模态是因为他先设置了窗口属性:setAttribute()再show()的(具体看源码)！