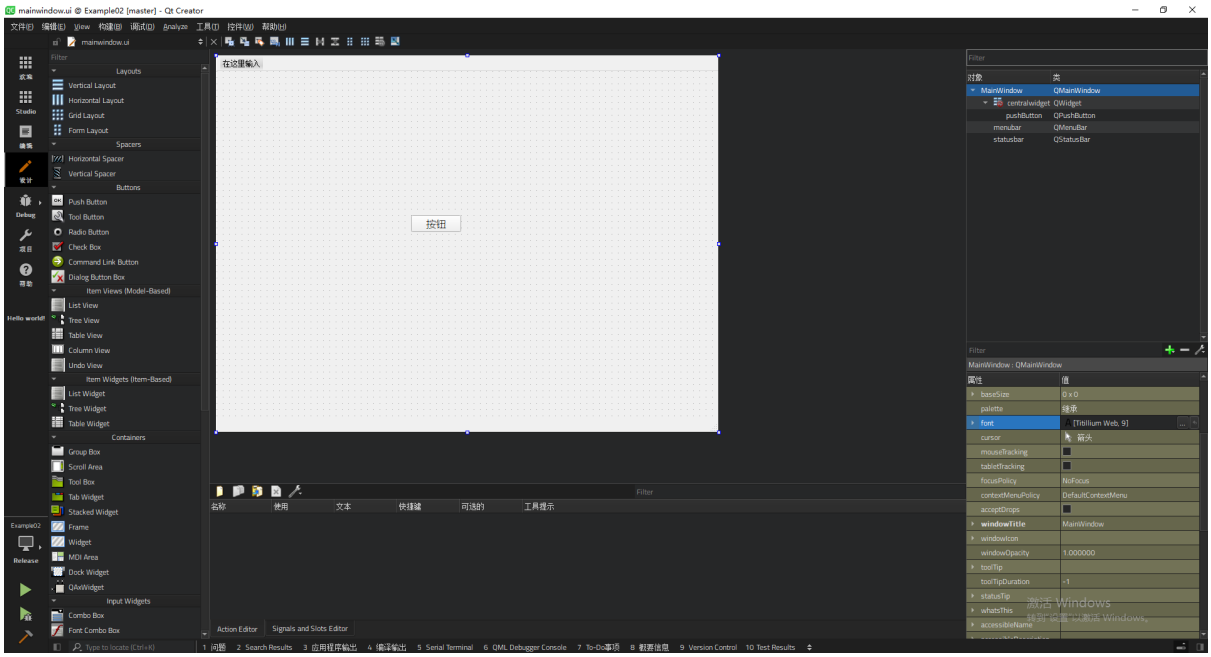
第2篇 (基础) 编写Qt多窗口程序

一、添加主窗口
二、添加主窗口
三、添加登录对话框
四、使用自定义的对话框类

地址：https://github.com/Lornatang/QtStartQuicklyTutorial/tree/main/Example02

一、添加主窗口

1. 我们打开Qt Creator，新建Qt Gui应用，项目名称设置为`nWindows`，在类信息界面保持基类为`QMainWindow`，类名为`MainWindow`，这样将会生成一个主窗口界面。
2. 完成项目创建后，打开`mainwindow.ui`文件进入设计模式，向界面上拖入一个`Push Button`，然后对其双击并修改显示文本为“按钮”，如下图所示。



二、添加主窗口

我们点击Qt Creator左侧的“编辑”按钮进入编辑模式，然后双击`mainwindow.cpp`文件对其进行编辑。在构造函数`MainWindow()`中添加代码：

```
MainWindow::MainWindow(QWidget *parent) :
        QMainWindow(parent),
        ui(new Ui::MainWindow)
{
        ui->setupUi(this);
        ui->pushButton->setText("新窗口"); //将界面上按钮的显示文本更改为“新窗口”
}
```

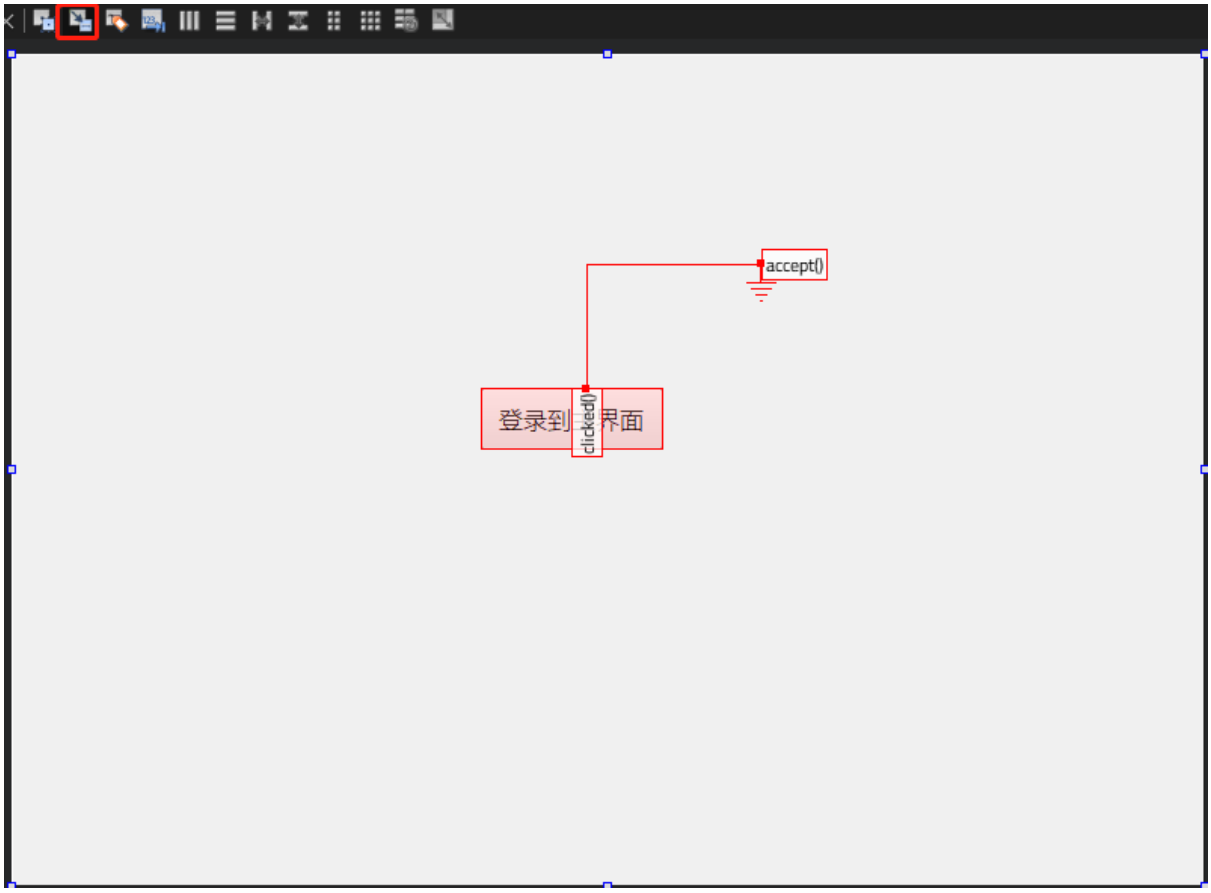
这里的`ui`对象就是界面文件对应的类的对象，在`mainwindow.h`文件中对其进行了定义，我们可以通过它来访问设计模式添加到界面上的部件。前面添加的按钮部件`Push Button`，在其属性面板上可以看到它的`objectName`属性的默认值为`pushButton`，这里就是通过这个属性来获取部件对象的。

我们使用了`QPushButton`类的`setText()`函数来设置按钮的显示文本，现在运行程序，效果如下图所示。

三、添加登录对话框

1. 往项目中添加新文件，这里可以在编辑模式的项目目录上点击鼠标右键，然后选择添加新文件菜单。当然也可以在文件菜单中进行添加。

2. 模板选择Qt设计师界面类，然后界面模板选择`Dialog without Button`。
3. 点击下一步进入类信息界面，这里将类名更改为`LoginDlg`（注意类名首字母一般大写）。
4. 当完成后会自动跳转到设计模式，对新添加的对话框进行设计。我们向界面上拖入一个`Push Button`，然后更改显示文本为“登录到主界面”。为了实现点击这个按钮后可以关闭该对话框并显示主窗口，我们需要设置信号和槽的关联。点击设计模式上方的 图标，或者按下F4，便进入了信号和槽编辑模式。按着鼠标左键，从按钮上拖向界面。
5. 当完成后会自动跳转到设计模式，对新添加的对话框进行设计。我们向界面上拖入一个`Push Button`，然后更改显示文本为“登录到主界面”。为了实现点击这个按钮后可以关闭该对话框并显示主窗口，我们需要设置信号和槽的关联。点击设计模式上方的 图标，或者按下F4，便进入了信号和槽编辑模式。按着鼠标左键，从按钮上拖向界面。
6. 当放开鼠标后，会弹出配置连接对话框，这里我们选择`pushButton`的`clicked()`信号和`LoginDlg`的`accept()`槽并按下确定按钮。如下图所示。如下图所示。



7. 这里简单介绍一下信号和槽，大家可以把它们都看做是函数，比如这里，当单击了按钮以后就会发射单击信号，即`clicked()`；然后对话框接收到信号就会执行相应的操作，即执行`accept()`槽。一般情况下，我们只需要修改槽函数即可，不过，这里的`accept()`已经实现了默认的功能，它会将对话框关闭并返回`Accepted`，所以我们无需再做更改。下面我们就是要使用返回的`Accepted`来判断是否按下了登录按钮。按下F3来返回控件编辑模式。

8. 上面讲述了一种显示对话框的情况，下面再来讲述一种情况。我们打开`mainwindow.ui`文件进入设计模式，然后在按钮部件上单击鼠标右键并选择转到槽菜单，如下图所示。

四、使用自定义的对话框类

1. 按下Ctrl+2返回代码编辑模式，在这里打开`main.cpp`文件，添加代码：

```
#include <QtGui/QApplication>
#include "mainwindow.h"
#include <QTextCodec> //添加头文件
#include "logindlg.h" //添加头文件
int main(int argc, char *argv[])
{
   QApplication a(argc, argv);
   // QTextCodec::setCodecForTr(QTextCodec::codecForLocale()); //设置编码
   QTextCodec::setCodecForTr(QTextCodec::codecForName("GB2312"));
   MainWindow w;
   LoginDlg dlg;                          // 建立自己新建的类的对象dlg
   if(dlg.exec() == QDialog::Accepted) // 利用Accepted返回值判断按钮是否被按下
    {
       w.show();                        // 如果被按下，显示主窗口
       return a.exec();                 // 程序一直执行，直到主窗口关闭
   }
   else return 0;            //如果没被按下，则不会进入主窗口，整个程序结束运行
}
```

在这里，我们先创建了`LoginDlg`类的对象`dlg`，然后让`dlg`运行，即执行`exec()`函数，并判断对话框的返回值，如果是按下了登录按钮，那么返回值应该是`Accepted`，这时就显示主窗口，并正常执行程序；如果没有按下登录按钮，那么就结束程序。

现在大家可以运行程序，测试一下效果。

2. 上面讲述了一种显示对话框的情况，下面再来讲述一种情况。我们打开`mainwindow.ui`文件进入设计模式，然后在按钮部件上单击鼠标右键并选择转到槽菜单。

3. 在弹出的转到槽对话框中选择`clicked()`信号并按下确定按钮。这时会跳转到编辑模式`mainwindow.cpp`文件的`on_pushButton_clicked()`函数处，这个就是自动生成的槽，它已经在`mainwindow.h`文件中进行了声明。我们只需要更改函数体即可。这里更改为：

```
void MainWindow::on_pushButton_clicked()
{
    QDialog *dlg = new QDialog(this);
    dlg->show();
}
```

我们创建了一个对话框对象，然后让其显示，这里的`this`参数表明这个对话框的父窗口是`MainWindow`。注意这里还需要添加`#include <QDialog>`头文件包含。有的童鞋可能会问，这里如果多次按下按钮，那么每次都会生成一个对话框，是否会造成内存泄露或者内存耗尽。这里简单说明一下，因为现在只是演示程序， Qt的对象树机制保证了不会造成内存泄露，而且不用写`delete`语句；而且因为是桌面程序，对于这样一个简单的对话框，其使用的内存可以被忽略。

当然，严谨的童鞋也可以在`mainwindow.h`文件中先定义一个对话框对象，并再在构造函数中进行创建，然后再到这里使用。

下面大家可以运行一下程序，查看效果。