## Qt入门学习——Qt Creator 中 ui 文件和 Qt 代码关系

标签: Qt5 入门教程 界面设计器 ui uic

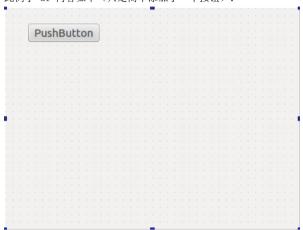
2015-08-28 10:20 263人阅读 评论(0) 收藏 编辑 删除

**Ⅲ** 分类: 【图形界面编程】(53)▼

▶ 版权声明:本博客文章,大多是本人整理编写,或在网络中收集,转载请注明出处!!

通过《Qt Creator的使用》的学习,我们可以借助 Designer (界面设计器) 快速设计界面。

此例子 ui 内容如下(只是简单添加了一个按钮):



## 工程的代码目录结构如下:



最终在工程所在目录会生成一个 ui 文件:



此 ui 文件实际上是xml 文件:

```
[WidgetWithUi]ls
main.cpp
              mywidget.h
                           WidgetWithUi.pro
mywidget.cpp (mywidget.ui) WidgetWithUi.pro.user
[WidgetWithUi]cat mywidget.ui
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
<class>MyWidget</class>
 <widget class="QWidget" name="MyWidget">
  property name="geometry">
   <rect>
                        ui文件内容为xml
    < x > 0 < / x >
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
```

```
当我们编译 Qt 程序代码,Qt Creator 用 uic 工具把 ui 文件的内容转换成 C++ 代码,在工程目录同一级目录的 build- 目录下自动生成 ui_类名.h 文件,如本例子中的 ui_mywidget.h,是由 mywidget.ui 利用uic 工具转换而成,只要通过Designer 修改了图形界面里的内容,ui_mywidget.h 也会跟着自动同步更新内容,ui_mywidget.h 是自动生成,用户写代码时可以不用关心其实现过程:
[qt_code]ls
build-WidgetWithUi-Desktop_Qt_5_4_2_GCC_32bit-Debug WidgetWithUi
[qt_code]cd build-WidgetWithUi-Desktop_Qt_5_4_2_GCC_32bit-Debug/[build-WidgetWithUi-Desktop_Qt_5_4_2_GCC_32bit-Debug]ls
main.o moc_mywidget.o WidgetWithUi
Makefile moc_mywidget.o WidgetWithUi
[build-WidgetWithUi-Desktop_Qt_5_4_2_GCC_32bit-Debug]]
```

接下来我们一起分析一下此 ui\_mywidget.h 和 Qt 程序代码如何关联起来(即 ui 文件和 Qt 代码关系)。

```
mywidget.h 中自动多了一个命名空间的声明,类中多了 ui 指针对象成员:
 ▼ 📠 WidgetWithUi
                        #ifndef MYWIDGET H
    WidgetWithUi.pro
                        #define MYWIDGET H
  ▼ 具文件
    mywidget.h
                        #include <QWidget>
  ▼ 🖳 源文件
                                         此命名空间的作用:提前声明
     main.cpp
                        namespace Ui {
                                         告诉编译器,此命名空间在别
     mywidget.cpp
                        class MyWidget;
                                         处定义。
  ▼ 网 界面文件
     mywidget.ui
                    10 ▼ class MyWidget : public QWidget
                            Q OBJECT
                        public:
                            explicit MyWidget(QWidget *parent = 0);
                            ~MyWidget();
                                         ui对象指针,所对应类为6行
                                         Ui命名空间中MyWidget
                        private:
                    19
                            Ui::MyWidget *ui;
                        #endif // MYWIDGET H
```

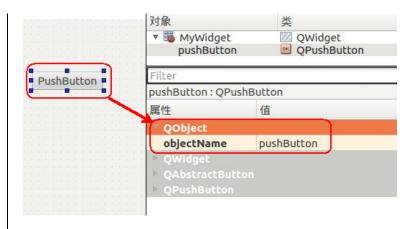
mywidget.h 对应的 mywidget.cpp 实现过程如下:

```
由mywidget.ui转换而成的头文件
MidgetWithUi
                           #include
                                     "mywidget.h"
                                                       Ui::MyWidget, setupUi()在此文件
定义。
  widgetWithUi.pro
                       2
                           #include (ui_mywidget.h")
▼ 🔐 头文件
    mywidget.h
                           MyWidget::MyWidget(QWidget *parent):
▼ 🔐 源文件
                                QWidget(parent),
                               (ui(new Ui::MyWidget)) 通过参数列表方式给ui对象指针动态分配空间
    c. main.cpp
   mywidget.cpp
    界面义件
                                ui->setupUi(this);
    mywidget.ui
                           }
                                                       ui调用setupUi()成员函数,此函数
在ui_mywidget.h定义,参数为this,
this指向main()函数的w的地址,即
                         MyWidget::~MyWidget()
                           {
                                delete ui:
                                                       相当于把&w传递过去。
                           }
```

```
接下来,我们一起看看 ui_mywidget.h 的实现过程:
      #ifndef UI MYWIDGET H
      #define UI MYWIDGET H
      #include <QtCore/QVariant>
      #include <Qttore/Qvariant>
#include <QtWidgets/QAction>
#include <QtWidgets/QApplication> 成员都是公有的,Ui::MyWidget继承过来后
#include <QtWidgets/OButtonGroup> 操体保留公方复数
      #include <QtWidgets/QButtonGroup>继续保留公有属性。
      #include <QtWidgets/QHeaderView>
                                            这样的话, Ui∷M
这样麻烦绕一圈,
                                                     Ui::MyWidget等价于Ui_MyWidget,
绕一圈,主要是为了防止用户自定义
      #include < QtWidgets/QPushButton>
                                            的类名和这个自动生成的类名重名, 防止名字
      #include <QtWidgets/QWidget>
      OT BEGIN NAMESPACE
                                             Ui {
                                  namespace
                                      class MyWidget: public Ui MyWidget {};
    // namespace Ui
                                              在界面设计器里设置一个按钮,自动在这
      publ
                                             声明QPushButton类型的对象指针。
          QPushButton *pushButton;
                                                    参数为this,即为main()函数的w地址.
           void setupUi((QWidget *MyWidget))
                                                   即为&w。
               if (MyWidget->objectName().isEmpty())
                    MyWidget->setObjectName(QStringLiteral("MyWidget"));
               MyWidget->resize(400, 300);
               (pushButton = new QPushButton(MyWidget);)
               pushButton->setObjectName(QSfringLiteral("pushButton"));
pushButton->setGeometry(QRect (30, 20, 99, 27));
                                            动态分配空间,通过给构造函数传参指定父对象,
               retranslateUi(MyWidget);即按钮放在main()函数中主窗口w对象的上面。
        DMetaObject::connectSlotsByName(MyWidget);
} // setupUi
38
        void(retranslateUi(QWidget *MyWidget)) 此函数在36行被调用
           MyWidget->setWindowTitle(QApplication::translate("MyWidget", "MyWidget", 0));
           pushButton->setText(QApplication::translate("MyWidget",
                                                              "PushButton", 0));
            retranslateUi
                      给按钮设置内容
    namespace Ui {
        class MyWidget: public Ui_MyWidget {};
                                               ■ Ui::MyWidget等价于22行中Ui_MyWidget
     // namespace Ui
    OT END NAMESPACE
    #endif // UI MYWIDGET H
```

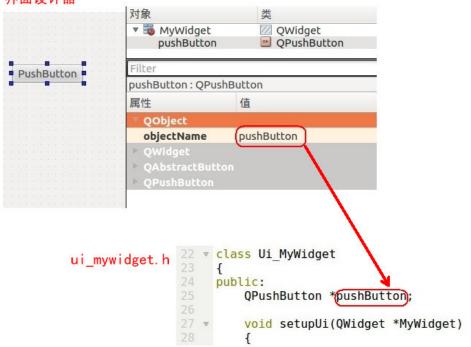
通过这样的一步步分析,我们发现 Designer(界面设计器)设计 ui 界面最终转换为 C++ 代码(ui\_类 名.h),和我们写的 Qt 代码是大同小异的,也就是说代码才是王道,Designer(界面设计器)只是辅助我们快速设计界面,没有它,我们同样可以写 Qt 程序。假如我们对某些部件操作不熟悉,不知道该如何用其相应函数,这时候我们通过 Designer(界面设计器)拖拽此部件,修改其所需属性,接着编译,看其自动转换的 C++ 代码如何实现,这样既可学习其相应函数的用法,所以说,Designer(界面设计器)本身就是学习Qt 的好老师。

那我们如何通过代码修改 ui 文件部件的属性呢? 在设计模式下,每个部件在"QObject"中都有一个"objectName"(对象名字)的属性:

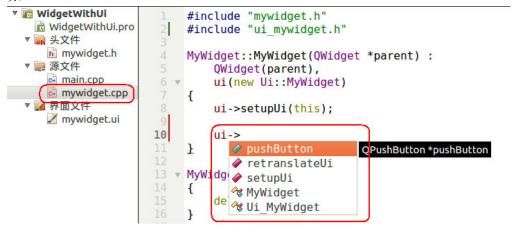


"objectName" (对象名字)的属性的值 (如此例子中的值 "pushButton"),此值则为 "ui\_类名.h"中自动创建的对象指针名:

## 界面设计器



所以,我们在.cpp 文件某函数中,可以通过" ui-> "引用此成员,在 Qt Creator 中,".(点)"键比较智能,如果操作的是普通对象,则为".(点)",如果是对象指向,自动变为"->",而且提示可以引用的成员:



这里, 通过代码修改 pushButton 的内容:



编译运行程序后,按钮的内容确实被修改:

