原创 Qt核心剖析：信息隐藏(2)

2010-06-04 13:51:43

标签：源码 Qt 剖析 休闲 核心

下面在上一篇的基础上，我们进入Qt的源代码，看看Qt4.x是如何实现 Private Classes 的。

正如前面我们说的，或许你会看到很多类似 Q_D 或者 Q_Q 这类的宏。那么，我们来试着看一下这样的代码：

```
01. void MyClass::setFoo( int i )
02. {
03.     Q_D(MyClass);
04.     d-&gt;m_foo = i;
05. }
06.
07. int MyClass::foo() const
08. {
09.     Q_D(const MyClass);
10.     return d-&gt;m_foo;
11. }
```

按照传统 C++ 的类，如果我们要实现这样的 getter 和 setter，我们应该使用一个私有变量 _i，然后操作这个变量。按照上一篇说的 Private Class 的做法，我们就要建一个 MyClassPrivateData 这样的类，然后使用指针对所有的数据操作进行委托。

热门文章

Qt学习之路(17)：Qt标准..

Qt学习之路(2)：Hello, w..

Qt学习之路(1)：前言

Qt学习之路(50)：QString

Qt学习之路(7)：创建一个..

Qt学习之路(8)：创建一个..

Qt学习之路(15)：Qt标准..

深入Java单例模式

搜索BLOG文章

搜 索

最近访客

10552..
72974..
15192..
jim003
qinzy212
61940..
sucha..
肇庆..
一心永成
a9341..
loserfly
gaoku..

最新评论

飘雪夏娜样：书上说的都说了，书上没有讲清楚的..

再来看一个比较 Qt 的例子：

```
01. class MyObject: public QObject
02. {
03.     Q_OBJECT
04. 
05. public:
06.     MyObject();
07.     virtual ~ MyObject();
08.     void setMemberX( int x );
09.     int memberX() const;
10.     void setMemberY( double y);
11.     double memberY() const;
12. 
13. signals:
14.     void priorityChanged( MyObject::Priority priority );
15. 
16. private:
17.     int    m_memberX;
18.     double m_memberY;
19. };
```

在来看一下 Qt 的实现：

```
01. class MyObjectPrivate;
02. class MyObject: public QObject
03. {
04.     Q_OBJECT
05. 
06. public:
07.     MyObject();
08.     virtual ~ MyObject();
09.     void setMemberX( int x );
10.     int memberX() const;
11.     void setMemberY( double y);
12.     double memberY() const;
13. 
14. signals:
15.     void priorityChanged( MyObject::Priority priority );
16. 
17. protected:
18.     MyObjectPrivate * const d_ptr;
19. 
20. private:
21.     Q_DECLARE_PRIVATE(MyObject);
22. };
```

这个例子很简单，一个使用传统方法实现，另一个采用了 Qt4.x 的方法。Qt4.x 的方法被称为 D-Pointer，因为它会使用一个名为 d 的指针，正如上面写的那个 d_ptr。使用传统方法，我们需要在 private 里面写上所有的私有变量，通常这会让整个文件变得很长，更为重要的是，用户并不需要这些信息。而使用 D-Pointer 的方法，我们的接口变得很漂亮：再也没有那一串长长的私有变量了。你不再需要将你的私有变量一起发布出去，它们就在你的 d 指针里面。如果你要修改数据类型这些信息，你也不需要去修改头文件，只需改变私有数据类即可。

需要注意的一点是，与单纯的 C++ 类不同，如果你的私有类需要定义 signals 和 slots，就应该把这个定义放在头文件中，而不是像上一篇所说的放在 cpp 文件中。这是因为 qmake 只会检测 .h 文件中的 Q_OBJECT 宏（这一点大家务必注意）。当然，你不应该把这样的 private class 放在你的类的同一个头文件中，因为这样做的话就没有意义了。常见做法是，定义一个 private 的头文件，例如使用 myclass_p.h 的命名方式（这也是 Qt 的命名方式）。并且记住，不要把 private 头文件放到你发布的 include 下面！因为这不是你发布的一部分，它们是私有的。然后，在你的 myclass 头文件中，使用

```
01. class MyClassPrivate;
```

这种前向声明而不是直接

```
01. #include "myclass_p.h"
```

这种方式。这也是为了避免将私有的头文件发布出去，并且前向声明可以缩短编译时间。

在这个类的 private 部分，我们使用了一个 MyClassPrivate 的 const 指针 d_ptr。如果你需要让这个类的子类也能够使用这个指针，就应该把这个 d_ptr 放在 protected 部分，正如上面的代码那样。并且，我们还加上了 const 关键字，来确保它只能被初始化一次。

下面，我们遇到了一个神奇的宏：Q_DECLARE_PRIVATE。这是干什么用的？那么，我们先来看一下这个宏的展开：

```
01. #define Q_DECLARE_PRIVATE(Class) \
02.     inline Class##Private* d_func() { return reinterpret_cast(qGetPtrHelper(d_ptr)); } \
03.     inline const Class##Private* d_func() const { return reinterpret_cast(qGetPtrHelper(d_ptr)
); } \
04.     friend class Class##Private;
```

如果你看不大懂，那么就用我们的 Q_DECLARE_PRIVATE(MyClass) 看看展开之后是什么吧：

```
01. inline MyClassPrivate* d_func() { return reinterpret_cast(qGetPtrHelper(d_ptr)); }
02. inline const MyClassPrivate* d_func() const { return reinterpret_cast(qGetPtrHelper(d_ptr)); }

03. friend class MyClassPrivate;
```

它实际上创建了两个 inline 的 d_func() 函数，返回值分别是我们的 d_ptr 指针和 const 指针。另外，它还把 MyClassPrivate 类声明为 MyClass 的 friend。这样的话，我们就可以在 MyClass 这个类里面使用 Q_D(MyClass) 以及 Q_D(const MyClass)。还记得我们最先看到的那段代码吗？现在我们来看看这个 Q_D 倒是是何方神圣！

```
01. // A template function for getting the instance to your private class instance.
02. template  static inline T *qGetPtrHelper(T *ptr) { return ptr; }
03.
04. // A macro for getting the d-pointer
05. #define Q_D(Class) Class##Private * const d = d_func()
```

下面还是自己展开一下这个宏，就成了

```
01. MyClassPrivate * const d = d_func()
```

简单来说，Qt 为我们把从 d_func() 获取 MyClassPrivate 指针的代码给封装起来了，这样我们就可以比较面向对

象的使用 getter 函数获取这个指针了。

现在我们已经比较清楚的知道 Qt 是如何使用 D-Pointer 实现我们前面所说的信息隐藏的了。但是，还有一个问题：如果我们把大部分代码集中到 MyClassPrivate 里面，很可能需要让 MyClassPrivate 的实现访问到 MyClass 的一些东西。现在我们让主类通过 D-Pointer 访问 MyClassPrivate 的数据，但是怎么反过来让 MyClassPrivate 访问主类的数据呢？Qt 也提供了相应的解决方案，那就是 Q_Q 宏，例如：

```
01.  class MyObjectPrivate
02.  {
03.  public:
04.      MyObjectPrivate(MyObject * parent):
05.              q_ptr( parent ),
06.              m_priority(MyObject::Low)
07.      {}
08.      void foo()
09.      {
10.         // Demonstrate how to make MyObject to emit a signal
11.         Q_Q(MyObject);
12.         emit q->priorityChanged( m_priority );
13.      }
14.
15.      //  Members
16.      MyObject * const q_ptr;
17.      Q_DECLARE_PUBLIC(MyObject);
18.      MyObject::Priority m_priority;
19.  };
```

在 private 类 MyObjectPrivate 中，通过构造函数将主类 MyObject 的指针传给 q_ptr。然后我们使用类似主类中使用的 Q_DECLARE_PRIVATE 的宏一样的另外的宏 Q_DECLARE_PUBLIC。这个宏所做的就是让你能够通过 Q_Q(Class) 宏使用主类指针。与 D-Pointer 不同，这时候你需要使用的是 Q_Pointer。这两个是完全相对的，这里也就不再赘述。

现在我们已经能够使用比较 Qt 的方式来使用 Private Classes 实现信息隐藏了。这不仅仅是 Qt 的实现，当然，你也可以不用 Q_D 和 Q_Q，而是使用自己的方式，这些都无关紧要。最主要的是，我们了解了一种 C++ 类的设计思路，这是 Qt 的源代码教给我们的。

本文出自 “豆子空间” 博客，请务必保留此出处http://devbean.blog.51cto.com/448512/326686

相关文章

- Qt核心剖析：信息隐藏(1)
- Qt学习之路(39)：QListWidget
- Qt学习之路(4)：初探信号槽
- Qt学习之路(1)：前言
- 利用C#实现生成PDF电子书源码
- Qt核心剖析：寻找 QObject 的源代码
- 源码包安装vsftp及相关配置
- Qt对话框--QMessageBox
- Qt Style Sheet的实现（Qt Designer Integra..
- Qt学习之路(15)：Qt标准对话框之QFileDialog

文章评论

[1楼] anjing6066 回复

2010-06-07 08:44:22

博主，我想问个问题，与这篇文章无关，Qt Creator自带的demo为什么有些带有.ui文件，有些不带呢，不带.ui的是不是就不能可视化的在上边修改界面了？谢谢

[2楼]楼主 FinderCheng 回复

2010-06-07 09:04:24

回复 anjing6066:[1楼]
ui 文件说到底也是编译成 C++ 代码，所以 ui 文件不是必须的，你用 C++ 代码也能写出界面来，因此没有 ui 文件的话也是可以有界面的。当然，没有 ui 文件就不能在 QtCreator 上面拖画了。

[3楼] ding404 回复

2010-06-09 23:04:02

好久没有新的博文啦，期待有新的作品

发表评论

昵　　称 登录 快速注册

验证码： 请点击后输入验证码 博客过2级，无需填写验证码

内　　容