# 第四章

本章节将会介绍如何使用Qt进行图形界面的开发，以及Qt的元对象系统、信号和槽的介绍和使用和一些基本控件的介绍。

**Qt元对象系统**

**Qt元对象系统的介绍**

介绍Qt的元对象系统前，需要先介绍一下C++的运行时类型信息(Runtime type information，RTTI)，RTTI可以通过typeid以及dynamic\_cast关键字实现运行时类型判断，但是这个功能太弱了，C++还不支持反射机制，无法做到运行时获取对象的全部信息。

Qt通过扩展C++的内省机制，但是并没有使用C++的RTTI，而是采用了更强大的元对象系统(Meta Object System)来实现内省。基于内省机制，可以列出对象的方法和属性列表，并且能获取对象的全部信息，在灵活性上优于RTTI。

**Qt元对象系统包含的功能**

Qt中的元对象系统是一个基于标准C++的扩展，为Qt提供了信号与槽机制、实时类型信息、动态属性系统。元对象系统基于QObject类、Q\_OBJECT宏、元对象编译器MOC实现。

* QObject类

每个需要使用元对象系统的类所需要的基类。

* Q\_OBJECT宏

定义每一个类的私有段，用来启用动态属性、信号和槽。

在每个QObject类或者派生类中，如果没有启用Q\_OBJECT宏，类的metaobject对象就不会被生成，调用对象的metaObject()返回的就是父类的metaobject对象，导致从类实例中获取的都是父对象的信息，从而类中定义的信号和槽都无法使用，所以任何基于QObject类，都需要启用Q\_OBJECT宏。

* MOC实现

MOC会分析C++源文件，如果发现头文件中定义了Q\_OBJECT宏，就会动态的生成一个moc文件，源文件中包含Q\_OBJECT的实现代码。

元对象系统除了提供信号和槽的功能外，还提供了运行时的方法，用来在运行时对QObject对象进行属性的读写：

|  |  |
| --- | --- |
| 方法 | 方法介绍 |
| QObject::metaObject() | 获取与一个类相关的metaObject。 |
| QMetaObject::className() | 运行时获取对象的类名，不需要编译的RTTI支持。 |
| QObject::inherits() | 用来判断一个对象是不是从一个特定的类继承的，必须是在QObject类直接或间接的派生类中。 |
| QObject::tr() | 为软件的国际化提供支持。 |
| QObject::trUtf8() | 为软件的国际化提供支持。 |
| QObject::setProperty() | 为QObject对象设置一个属性。 |
| QObject::property() | 获取一个QObject对象的属性值。 |
| qobject\_cast() | 可以在QObject类之间提供动态转换，qobject\_cast类似于C++的dynamic\_cast，但是qobject\_cast不需要RTTI的支持。 |

可以看出，Qt通过MOC而不是RTTI实现了很多标准C++不支持的功能，提供了强大的运行时对对象操作的功能。

**Qt元对象系统的实现**

Qt Meta Object的所有数据和方法都封装在QMetaQObject类中，它包含一个Qt类的meta信息，并提供查询功能。

1. 信号表: 与对应Qt类相关的定义和自定义signal的名字
2. 槽表: 与对应Qt类相关的定义和自定义slot的名字
3. 类信息表: Qt类的类型信息
4. 属性表: 与对应类中所有的属性名字
5. 指向parent meta object的指针

Q\_OBJECT宏定义如下：

/\* qmake ignore Q\_OBJECT \*/

#define Q\_OBJECT \

public: \

QT\_WARNING\_PUSH \

Q\_OBJECT\_NO\_OVERRIDE\_WARNING \

static const QMetaObject staticMetaObject; \

virtual const QMetaObject \*metaObject() const; \

virtual void \*qt\_metacast(const char \*); \

virtual int qt\_metacall(QMetaObject::Call, int, void \*\*); \

QT\_TR\_FUNCTIONS \

private: \

Q\_OBJECT\_NO\_ATTRIBUTES\_WARNING \

Q\_DECL\_HIDDEN\_STATIC\_METACALL static void qt\_static\_metacall(QObject \*, QMetaObject::Call, int, void \*\*); \

QT\_WARNING\_POP \

struct QPrivateSignal {}; \

QT\_ANNOTATE\_CLASS(qt\_qobject, "")

staticMetaObject是静态常量，metaObject()是获取该对象指针的方法，所有的QObject对象都会共享staticMetaObject常量，来完成信号和槽的功能。

struct { // private data

const QMetaObject \*superdata;

const QByteArrayData \*stringdata;

const uint \*data;

typedef void (\*StaticMetacallFunction)(QObject \*,QMetaObject::Call, int, void \*\*);

StaticMetacallFunction static\_metacall;

const QMetaObject \* const \*relatedMetaObjects;

void \*extradata; //reserved for future use

} d;

以上是QMetaObject所定义的全部成员，记录了所有的signal、slot、property和class等信息。

const QMetaObject\* superdata，这个变量指向与之对应的QObject类的父类对象，或者祖先类的QMetaObject对象，每一个QObject类或者派生类都有可能有一个父类或者父类的父类，superdata指向的是最接近祖先类的QMetaObject对象，如果没有父类，该值为nullptr。

const QByteArrayData\* stringdata，这是一个指向string data的指针，在连续的内存中保存了若干字符串，里面保存的是MetaSystem的核心功能——属性。

Const unit \*data，这个指针指向一个无符号整形数组，该数组是一个预定义的复合结构数据，由QMetaObjectPrivate类提供管理，保存了类的基本信息和一些索引值。在不同的QObject对象中的数据长度是不一样长的，这取决于定义了多少的signal、slot和property。有一部分指出了前一个变量stringdata中不同字符串的索引值，但是需要注意的是，这里面的数值并不是直接标明了每一个字符串的索引值，这个数组还需要通过一个相应的算法计算后，才能获取正确的字符串索引值。

const void\* extradata，这是一个指向QMetaObjectExtraDat数据结构的指针。