

Inside Qt meta Object

Introduction

In this article we will discuss how and why we use **meta-object** in Qt.

The word **meta** indicates that the word prefixed is about itself. So a **meta-object** is an object describing the object. QT Meta-Object Compiler, moc, is the program that handles Qt's C++ extensions. The moc reads C++ source files, if it finds one or more class declarations that contain the Q_OBJECT macro, it produces another C++ source file which contains the meta object code for those classes and those are automatically included by build system.

QMetaObject Class

A **meta-object** contains information about a class that inherits QObject, e.g. class name, super class name, properties, signals and slots. Every QObject subclass that contains the Q_OBJECT macro will have a meta-object. Following code declares **MyQTCClass** which has a property declared by Q_PROPERTY macro. This will allow us to update the property in several ways.

```
class MyQTCClass : public QObject
{
    Q_OBJECT
    Q_PROPERTY(int Myage READ Age WRITE setAge)
public:
    MyQTCClass();
    ~MyQTCClass();
    void SomeFunction();
    int Age();
    void setAge(int aAge);
private:
    QString myName;
    int Myage;
    bool abool;
};

// test code
MyQTCClass *myinstance = new MyQTCClass;
QObject *object = myinstance;

myinstance->setAge(100); // using setter function
myinstance->setProperty("Myage", 100); //using property
int retage =myinstance->Age(); // retage should be 100

//Setting property
QVariant v(123);
bool b = object->setProperty("Myage", v); // b is true since we have a Myage property
b = object->setProperty("aVariable", v); // b is not true

QVariant test = object->property("Myage");
b = test.isValid(); // b is true
QVariant wrong = object->property("aVariabdfhdfgle");
b = wrong.isValid(); // b is not true

const QMetaObject *metaobject = object->metaObject();
QString classname(metaobject->className()); // classname should be "MyQTCClass"

int count = metaobject->propertyCount(); // number of Property
for (int i=0; i<count; ++i)
{
    QMetaProperty metaproperty = metaobject->property(i);
    const char *name = metaproperty.name(); // property name should come from MyQTCClass + QObject
    QString propertyName(name);
}
```

In the above code, at the very top of the class declaration we find the Q_OBJECT macro. It is important that this macro appears first in the body of the class declaration because it marks the class as a class that needs a **meta-object**. In the case of Qt, **meta-objects** are instances of the class. It means that the above code goes into a file called MyQTCClass.h. The implementation goes into MyQTCClass.cpp, and the moc generates another C++ file from the header file called moc_MyQTCClass.cpp. The contents from the generated file can change between Qt versions.

Example Applications

The full source code presented in this article is available here <File:InsidePro.zip>

