

Programming with the Qt Framework

Saurabh Sood

October 15, 2014

Table Of Contents

- 1 Workshop on Qt
- 2 A bit of introduction
- 3 About Qt
- 4 Features
- 5 Lets Dig in...
- 6 Qt's event handling mechanism
- 7 Lets Code...
- 8 Some Helpful Links
- 9 Questions???

What is a framework?

What is a framework?

- A collection of tools and libraries to make certain tasks easier?

What is a framework?

- A collection of tools and libraries to make certain tasks easier?
 - Eg: A Dynamic Array, Hashtables etc (Collections)
 - Generic Algorithms to achieve a certain task

What is Qt?

What is Qt?

- Pronounced as 'Cute' ;)

What is Qt?

- Pronounced as 'Cute' ;)
- It is a *cross platform*, development framework
- Original Language is C++ with bindings in many languages (Ruby, Python etc)

What is Qt?

- Pronounced as 'Cute' ;)
- It is a *cross platform*, development framework
- Original Language is C++ with bindings in many languages (Ruby, Python etc)

Supported Platforms

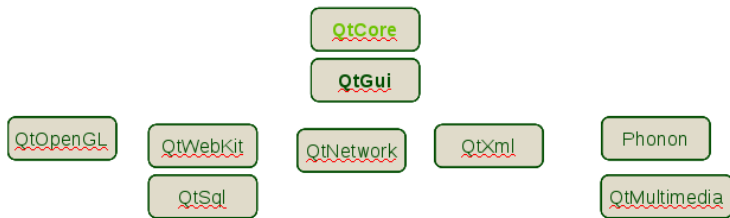
Supported Platforms

- *nix platforms
- Windows
- Android (through Necessitas)

Some features of Qt

Some features of Qt

- Modular Design



Macros and Introspection

Macros and Introspection

- Qt extends with *macros* and *introspection*

```
foreach (int value, intList) { ... }
```

```
QObject *o = new QPushButton;  
o->metaObject()->className(); // returns "QPushButton"
```

```
connect(button, SIGNAL(clicked()), window, SLOT(close()));
```

Meta Data and MOC

- Every *QObject* has a *meta object*
- The meta object knows about certain properties related to the class such as *class name, inheritance, signals and slots* etc
- The moc looks for macros like *signals, slots* etc
- Introspection refers to the class knowing about its own members at runtime (eg: class name, inheritance structure etc)

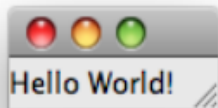
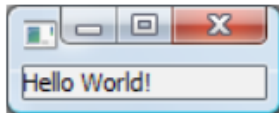
Applications using it...

Applications using it...

- KDE (The whole desktop environment along with the application suite)
- VLC Media Player
- Adobe Photoshop

Appease the Programming Gods...

Hello World!!!



Appease the Programming Gods...

Hello World!!!

```
#include <QApplication>
#include <QLabel>

int main( int argc, char **argv )
{
    QApplication app( argc, argv );
    QLabel l( "Hello World!" );
    l.show();
    return app.exec();
}
```

A Section by section dissection of the code...

```
#include <QApplication>
#include <QLabel>

int main( int argc, char **argv )
{
    QApplication app( argc, argv );
    QLabel l( "Hello World!" );
    l.show();
    return app.exec();
}
```

A Section by section dissection of the code...

```
#include <QApplication>
#include <QLabel>

int main( int argc, char **argv )
{
    QApplication app( argc, argv );
    QLabel l( "Hello World!" );
    l.show();
    return app.exec();
}
```

A Section by section dissection of the code...

```
#include <QApplication>
#include <QLabel>

int main( int argc, char **argv )
{
    QApplication app( argc, argv );
    QLabel l( "Hello World!" );
    l.show();
    return app.exec();
}
```

A Section by section dissection of the code...

```
#include <QApplication>
#include <QLabel>

int main( int argc, char **argv )
{
    QApplication app( argc, argv );
    QLabel l( "Hello World!" );
    l.show();
    return app.exec();
}
```


Signals and Slots

Signals and Slots

- Qt's *callback* mechanism

Signals and Slots

- Qt's *callback* mechanism
- Dynamically ties together events and state changes with reactions

Signals and Slots

- Qt's *callback* mechanism
- Dynamically ties together events and state changes with reactions
- What makes Qt tick!!! ;)

Slots

- *Callback* methods

Slots

- *Callback* methods
- Defined as normal methods but with the slot macro

Slots

- *Callback* methods
- Defined as normal methods but with the slot macro
- A number of signals can be connected to the same slot

Slots

- *Callback* methods
- Defined as normal methods but with the slot macro
- A number of signals can be connected to the same slot
- Can be called as an ordinary method

Signals

- Events, State Changes

Signals

- Events, State Changes
- Return void

Signals

- Events, State Changes
- Return void
- Must not be implemented. The *moc* provides an implementation










Signals

- Events, State Changes
- Return void
- Must not be implemented. The *moc* provides an implementation
- Can be emitted using the *emit* keyword

Making the connection

Signals

Slots

<code>rangeChanged(int,int)</code>		<code>setRange(int,int)</code>
<code>rangeChanged(int,int)</code>		<code>setValue(int)</code>
<code>rangeChanged(int,int)</code>		<code>updateDialog()</code>
<code>valueChanged(int)</code>		<code>setRange(int,int)</code>
<code>valueChanged(int)</code>		<code>setValue(int)</code>
<code>valueChanged(int)</code>		<code>updateDialog()</code>
<code>textChanged(QString)</code>		<code>setValue(int)</code>
<code>clicked()</code>		<code>setValue(int)</code>
<code>clicked()</code>		<code>updateDialog()</code>

Making the connection

Connection done through the *connect* method
connect(src, SIGNAL(sig()), dest, SLOT(doSomething()));

Getting our hands dirty...

Looking at some code...

Where I can learn some more...

- Qt project site - <http://qt-project.org/doc>
- Advanced Qt Programming - <http://www.qtrac.eu/aqpbook.html>

