

Basic Introduction to Qt

David Doria

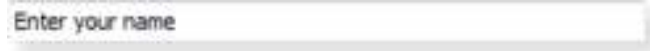
Qt

- Cute? Q-T?
- c++ library for creating user interfaces
- Professional looking GUIs very easily
- Cross-platform


“Widgets”

- Graphical “Things”/”objects”

- Text box

A rectangular text input field with a light gray border and a subtle drop shadow. The text "Enter your name" is displayed inside the field in a small, gray font.

- Label

A small, rectangular button with a light gray background and a subtle drop shadow. The text "Text Label" is centered on the button in a small, gray font.

- Button

A rectangular button with a light blue gradient background and a subtle drop shadow. The text "Cancel" is centered on the button in a small, gray font.

- Checkbox

A checkbox widget consisting of a small square box with a checkmark inside, followed by the text "Case sensitive". The entire widget has a light gray background and a subtle drop shadow.

- And dozens more

Signals and Slots

- Event driven system
- Similar to “throwing” and “catching” and event (or “invoking” and “handling” an event)
- Typical pair
 - “signal” = button was clicked
 - “slot” = a function to output “button was clicked”

Signals and Slots

- Connect objects to each other
 - Drive the text in a text box with a slider (automatically)
- Connect objects to custom functions

Program Structure

- `main()` is typically very short
- Simply creates the Form/`QMainWindow` object
- Must connect the signals to the slots that you want to handle them
- All of the code is driven by user events

Creating Objects Programmatically

- You can create objects like this:

```
QPushButton *button = new QPushButton;  
button->move(100, 100); // Position in the window  
button->show();
```

- This gets very hard to keep track of when the interface has more than a couple of widgets

Qt Designer

- A GUI to create your GUI!
- Demo

Using the .ui file (simple)

```
#include <QApplication>

#include "ui_Test.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QMainWindow *window = new QMainWindow;
    Ui::MainWindow ui;
    ui.setupUi(window);

    window->show();
    return app.exec();
}
```

Using the .ui file (normal workflow)

Main.cpp

```
#include <QApplication>

#include "testform.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    TestForm myform;

    myform.show();
    return app.exec();
}
```

Using the .ui file (normal workflow) cont.

testform.h

```
#ifndef MYFORM_H
#define MYFORM_H

#include "ui_myform.h"

class TestForm : public QWidget, private Ui::MyForm
{
    Q_OBJECT
public:
    TestForm(QWidget *parent = 0);

public slots:
    void pushButton_SetLabelText();
};

#endif
```

Using the .ui file (normal workflow) cont.

testform.cpp

```
#include "buttonform.h"
```

```
MyForm::MyForm(QWidget *parent)
    : QWidget(parent)
{
    setupUi(this);
    connect( this->pushButton, SIGNAL( clicked() ), this,
            SLOT(pushButton_SetLabelText()) );
}
```

```
void MyForm::pushButton_SetLabelText()
{
    this->label->setText("hello");
}
```


Connect Signals to Slots

```
connect(CallingObject , SIGNAL, ReceivingObject, SLOT);
```

```
connect( this->ui.pushButton, SIGNAL( clicked() ), this, SLOT(pushButton_SetLabelText()) );
```

Demos

- Text box (QLineEdit)




- Button (QPushButton)



- Check box (QCheckBox)



- Progress bar (QProgressBar) (marquee mode)



- File dialog box (QFileDialog) (save/open)



Documentation

- <http://doc.trolltech.com/4.5/qfiledialog.html>