Qt Introduction

A cross-platform application and UI framework

Red Hat
Jaroslav Reznik
February 10, 2011

Part I
Introduction

redhat.

Section 1

Welcome and introduction

redhat.

Agenda

redhat.

About me

- Software Engineer @Red Hat Czech, Brno, responsible for:
  - Qt and KDE maintenance and development,
  - System configuration tools and Matahari
- Active in several open source communities.
  - Fedora (Board Member)
  - KDE
  - Linux v Brně
  - Openmobility
- Certified to be Qt ;-)

Section 2

Qt and Trolltech goes Nokia...

redhat.

What is Qt

Qt is

- a cross-platform,
- application and UI framework developed by Nokia,
- dual licensed under LGPL v2.1 (with exceptions) and Qt Commercial Developer License,
- and (mostly) C++ based (with bindings for Java, Ruby, Python...).

redhat.

Brief history

- Current version is Qt 4.7 (4.0 released June 28, 2005)
- Developed by Trolltech, now by Nokia
- FreeQt - QPL - GPL v2 - LGPL v2.1 (and GPL v3) with optional Qt Commercial Developer License.

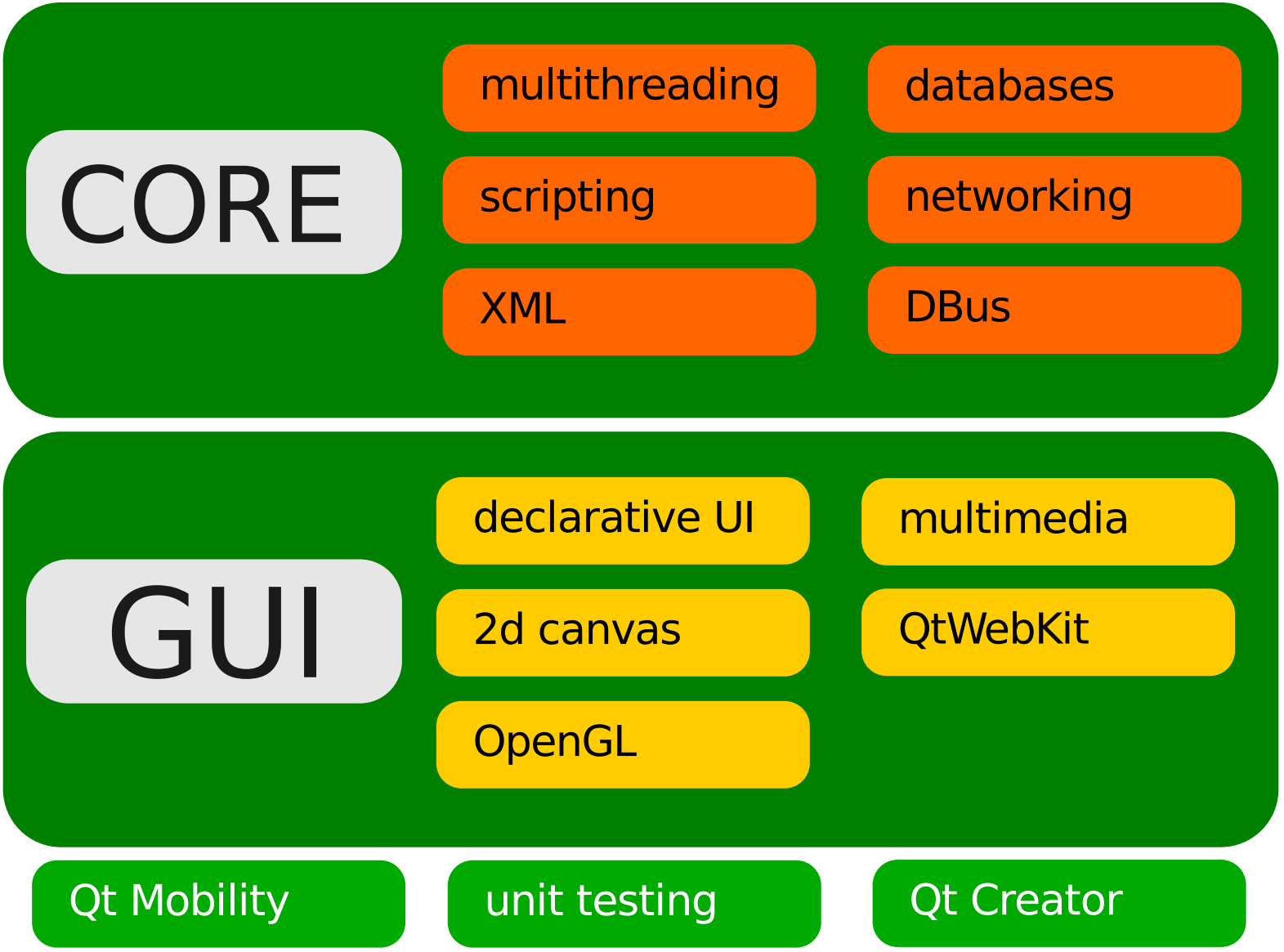
redhat.

Platforms support

- Official support for
  - Linux/X11 and embedded devices (even Wayland in progress),
  - Windows, Windows CE/mobile,
  - Symbian,
  - Mac OS X.
- Experimental for
  - Android,
  - WebOS,
  - Amazon Kindle,
  - OpenSolaris,
  - Haiku,
  - OS/2,
  - and even iOS (iPhone).

redhat

Modularity



Part II
Qt Development

Section 3

Main concepts

redhat.

Qt as framework

- Qt is not just a library!
  - Great documentation.
  - STL replacement.
  - Modules for nearly everything, not GUI only!
  - Support.
- Nokia Qt SDK vs Qt SDK ;-)

Best practices

Try to avoid mixing Qt code with STL one as Qt has a great offering for most common operations and containers! (But it's still possible).

redhat.

Qt as framework

> *"It's fast, easy to understand, the syntax is clear, and you'll never regret it! You think about something, simply imagine what the object may look like, and you can be sure it exists in the Qt SDK with that name! It's pretty awesome. :D"*

Vincent Bénony, Arobas Music as quoted in "How Qt can turn you into a guitar maestro" by David Stone; June 2010

redhat.

MOC and C++ extensions

- MOC stands for meta object compiler and handles Qt's C++ extensions like
  - signals and slots mechanism for inter-object communication,
  - runtime type information,
  - and the dynamic property system.
- MOC is preprocessor.
- Introduces new keywords - emit (Q_EMIT), foreach (Q_FOREACH)
- Parses C++ source codes and for every class with Q_OBJECT macro definition prepares C++ source code that's finally included into original code.

Best practices

Use QMake buildsystem or use automoc4 for CMake based projects.

redhat.

QObject and Q_OBJECT macro

- The QObject class is the base class of all Qt objects.
- QObjects are organized in object trees:
  - set a parent objects (calls insertChild() of parent, visible in parent's childrens() list,
  - parent takes ownership (children object is automatically deleted in parent's destructor).
- Every object has an object name(), can report its className() and class that inherits().
- QObject can receive and filter events, has basic timer support.

Best practices

- Do not use Q_OBJECT macro if you don't need it (signals, properties etc.).
- It's strongly recommended to use QObject together with Q_OBJECT macro.

redhat.

Example Q_OBJECT

Class inherits QObject

```
class MyClass : public QObject
 {
     Q_OBJECT

 public:
     MyClass(QObject *parent = 0);
     ~MyClass();

 signals:
     void mySignal(int);

 public slots:
     void mySlot(int);
 };
```

redhat.

Signals and slots

- Communication between objects.
- MOC and Q_OBJECT macro.
- Signals and slots are typesafe (in opposit to callbacks)!
- Signal is emited (emit, Q_EMIT).
- Slot is a method called as reaction to emited signal.
  - Access rights - private, protected, public.

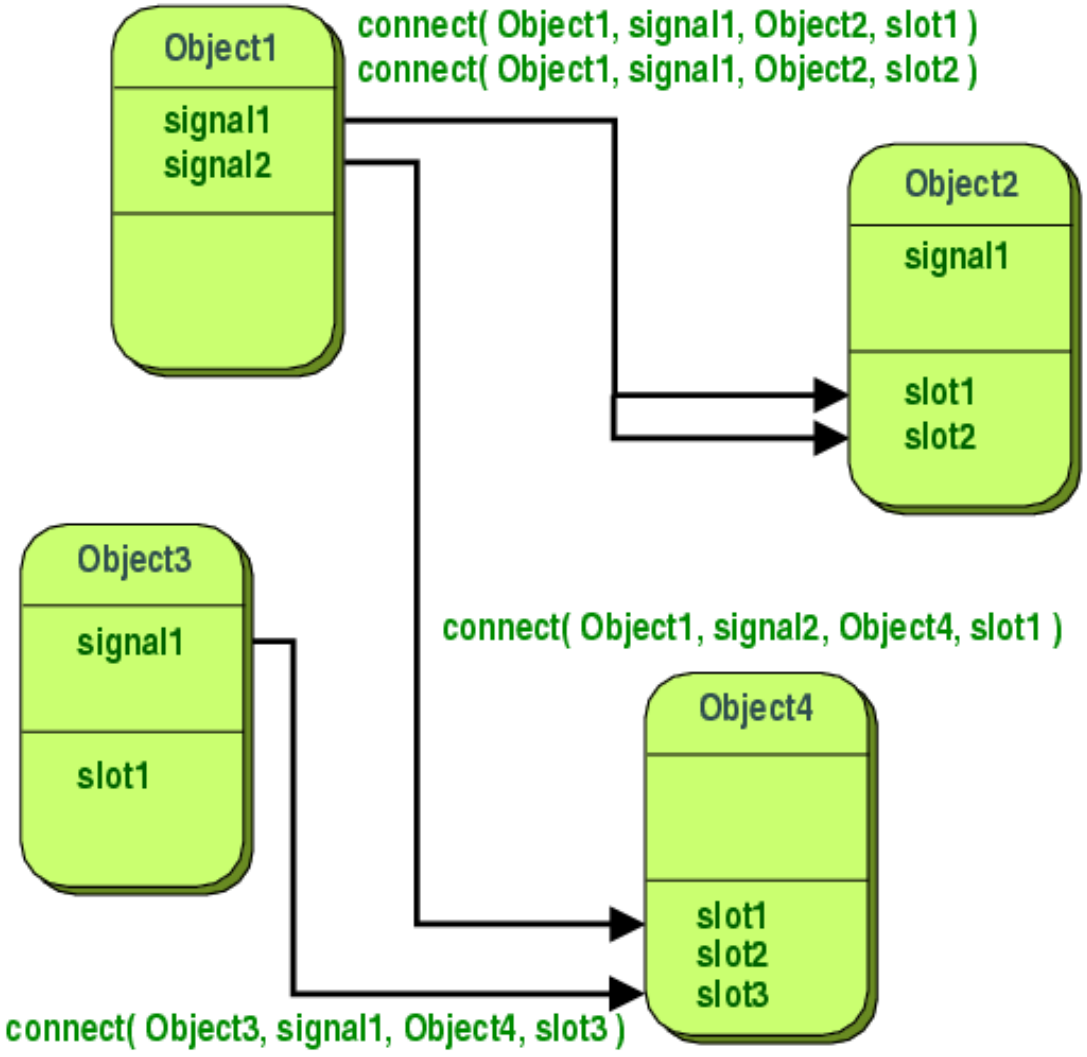
Connect signal and slot

```
// connect signal and slot
connect(&first, SIGNAL(mySignal(int)),
        &second, SLOT(mySignal(int)));

// emit signal
emit mySignal(some_value);
```

redhat.

Signals and slots



redhat.

Buildsystem

- QMake is recommended and default Qt build system.
  - Generates Makefiles, Visual Studio projects...
- Qt is very well supported in CMake.
- Classic Makefiles - moc, linkage etc.

Section 4

UI design

redhat.

Widgets

- All Qt widgets inherit QObject.
- Widgets base class is QWidget.
- Widget receives events (mouse, keyboard etc.) and paints itself.
- Widget without parent widget is window (QMainWindow, QDialog etc...).
- Styles using stylesheets (CSS inspired).

redhat.

Layouts

- Layouts of child widgets.
  - Positioning of child widgets.
  - Sensible sizes (default, minimum) for windows.
  - Resize handling.
- Horizontal, vertical and grid layouts.

- Custom layout managers.
- Qt Designer.

redhat.

Graphics View Framework

- 2D graphical items and view framework.
- BSP (Binary Space Partitioning) tree
  - Large scenes visualisation.
- Model-view design patter design.
  - QGraphicsScene as a model of QGraphicsItems objects. It acts as controller too.
  - QGraphicsView as a view widget to visualize scene.
- Support for:
  - Zooming and rotation, printing, drag&drop, cursors, animations, OpenGL rendering.
- Even widgets and dialogs can be embedded (QGraphicsWidget or native widgets!).

redhat.

Qt Quick

- Declarative UI design
- QML as a language
  - Analogy in HTML + CSS or Edje
- Logic in embedded JavaScript or C++ (Qt Declarative)
  - Uses Qt C++ reflection, properties bindings
- Designers can mockup UI directly in QML
- Rich interfaces using Animation Framework and State Framework
- Supported in Qt 4.7 and latest Qt Creator (designer)

Qt Quick Example

Hello Qt Quick (qmlviewer hello.qml)

```
import Qt 4.7

Rectangle {
    id: myRectangle
    width: 500
    height: 400

    Text {
        text: "<h1>Hello Qt Quick</h1>"
        color: "white"
        x: 100; y: 100
    }

    color: "blue"
}
```

Part III
Tools

IDEs

- Qt Creator by Nokia
- KDevelop

Support tools

- Qt Assistant - documentation viewer system.
- Qt Linguist - internationalization tools.
- Qt Designer - GUI designer.
- Qt Simulator - emulates Maemo/MeeGo/Symbian phones.

Part IV
Qt in use

KDE

- Multi-platform desktop environment written mostly in C++ and Qt

MeeGo

- Mobile platform
  - Netbooks, handhelds, cell phones, in-vehicle device, TVs User Experiences
- by Nokia and Intel under the Linux Foundation hood
- Qt based MeeGo Touch Framework
  - but platform contains Gtk+ and Clutter too
- Based on Fedora (RPM) with OpenSUSE Build Service in use
  - but optimized for embedded devices, small footprint
- First MeeGo devices expected this year??? (N9?)
- QtMobility (Symbian supported)

Other open source software

- VLC
- MythTV
- Arora browser
- Amarok
- Psi

Proprietary/commercial software

- Lot of commercial software is developed in Qt.
- Skype for Linux
- Google Earth
- Autodesk Maya
- Guitar Pro 6

Links

- http://qt.nokia.com/

The end.

Thanks for listening.