Introduzione a Qt

Marco Martin

Cos'è Qt

- Insieme modulare di librerie in C++
- Framework per la costruzione di UI
- Cross platform: Linux, Mac, Windows, Symbian, Android...
- Cross device: desktop e mobile
- Iniziato da Trolltech, comprato da Nokia nel 2008
- Giunto alla varsione 4.7 (4.8 fine anno, Qt5 fine 2012)

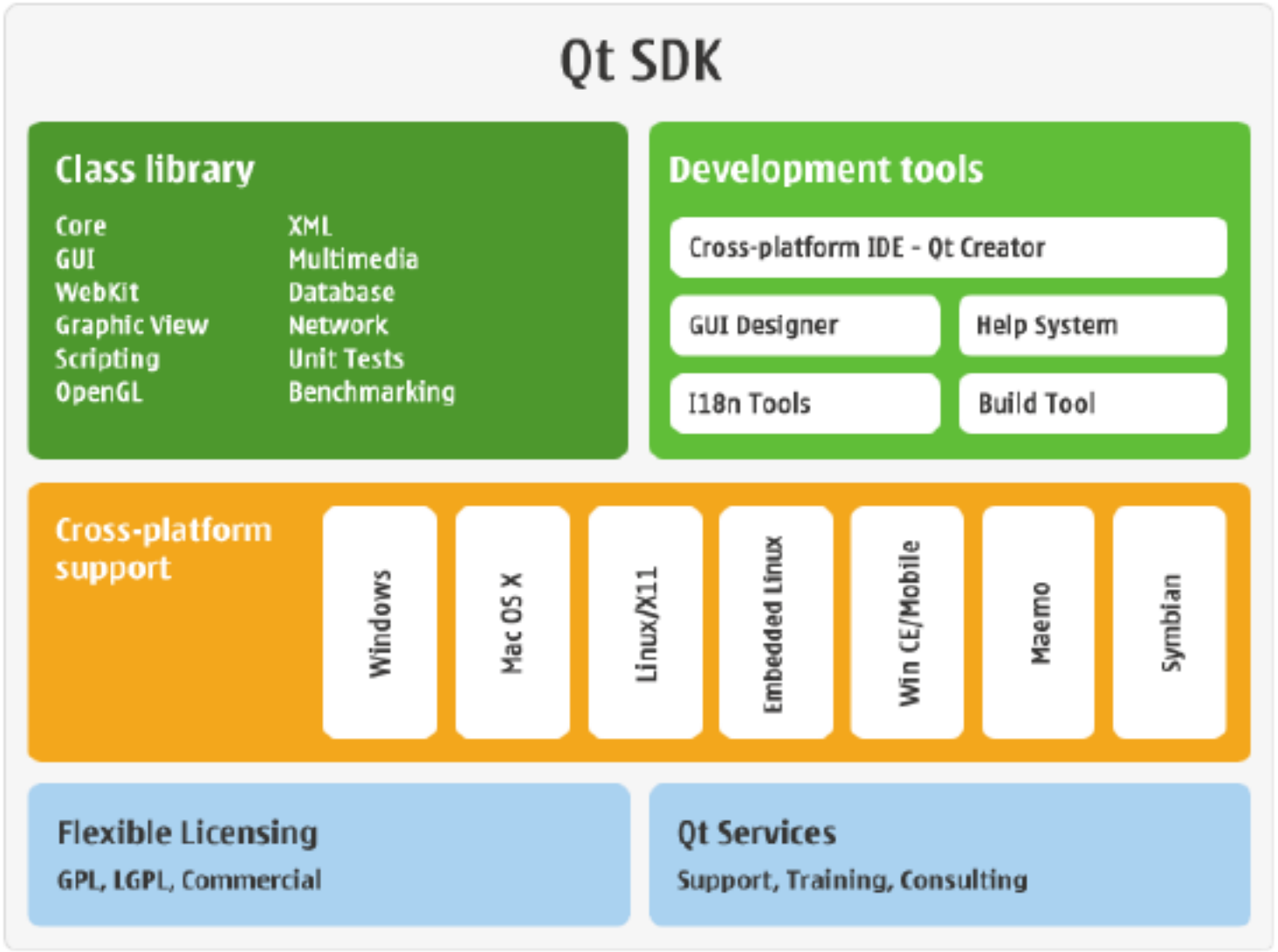
Esempi

- Opensource
  - KDE (e tutte le sue applicazione come Amarok, Okular, Kontact...)
  - Scribus
  - VLC
  - ...
- Commerciali
  - Google Earth
  - Skype
  - Autodesk Maya
  - Guitar pro
  - Opera
  - ...

Perché Qt

- Facilità di sviluppo, più semplice rispetto ad altre librerie C++ (STL, Boost)
- Write once, run everywhere
- LGPL 2/3
- Supporto commerciale

Architettura



Setup SDK

- Linux
  - Pacchettizzato dalla maggior parte delle distribuzioni (LSB)
  - Per lo sviluppo installare pacchetti *-devel*
- Windows/MAC
  - http://qt.nokia.com/downloads
  - Installer con librerie/header/documentazione/QtCreator

Moduli

- QtCore: modulo fondamentale, **non** ha funzioni grafiche
  - Container (liste,hash,set..)
  - QObject (base dell'object model di Qt)
  - Event loop
- QtGui: widget, model/view, QgraphicsView...
- QtNetwork: astrazione di protocolli di rete come HTTP

Moduli (continua)

- QtSql: accesso database
- QtScript: binding di oggetti Qt in Javascript
- QtDeclarative: libreria per costruire UI per dispositivi mobili a partire da un linguaggio declarative (QML) e JavaScript
- QtXML, QtOpenGl,QtSvg ...

Hello world

```
#include <QApplication>
  #include <QPushButton>

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

      QPushButton hello("Hello world!");
      hello.resize(100, 30);
      QObject::connect(&hello, SIGNAL(clicked()), &app, SLOT(quit()));
      hello.show();
      return app.exec();
  }
```

QMake

- Semplice tool per gestire il build
- Gestito automaticamente da QtCreator
- Per progetti piu' complessi puo' essere necessario un tool piu' completo, come Cmake

```
CONFIG += qt
 HEADERS += hello.h
 SOURCES += hello.cpp
 SOURCES += main.cpp


qmake hello.pro
make
```

Classi di base

- Stringhe
  - QString è una classe (più gestibile di char*)
  - Metodi di convenienza come ::length(), ::arg(), ::replace()...
  - Operatori: assegnazione (=) e concatenazione (+)
- Qurl: url, astrae cose come protocollo, dominio, fragment
- Container: Qlist, Qhash Qmap
  - classi template Qlist<int>
  - Forniscono iteratori stile STL e Java

Q(Core)Application e event loop

- I tipi primitivi come QList possono essere utilizzati ovunque
- Per poter utilizzare tipi QObject bisogna essere dentro ad un event loop
- Gestito da QCoreApplication (testuale) o QApplication (GUI)
- Completamente trasparente, gestisce il dispatching di eventi di input e output(paint) oltre all'invocazione degli slot
- Event loop possono essere annidati
- Ogni thread ha il proprio event loop

QObject

- Base dell'object model di Qt
- Piu' feature rispetto a plain C++
  - signal e slot
  - Proprietà
  - Eventi
  - Gestione memoria
  - ...
- Usa il meta object system (codice generato)
- Non visuale (su Linux QtCore **NON** dipende da X11)

Object tree

- Le sottoclassi di QObject sono organizzate secondo una gerarchia padre/figlio
- QObject(QObject *parent = 0)
  - Il parent aggiunge il nuovo oggetto ai suoi figli
- Quando un oggetto Qobject viene distrutto (delete o ::deleteLater()) esso distruggera' tutti i suoi figli (se nell'heap)
- **NON** è ereditarietà

Eventi

virtual void mousePressEvent (QMouseEvent * event)

- Ad un evento, il loop invoca automaticamente il metodo corrispondente dell'oggetto a cui verra' consegnato (quale widget È sotto il mouse?)
- Metodi di gestione protected virtual, possono essere reimplementati
- Da fare solo se non c'e' un segnale utile per quello che si vuole fare

Signal e slot

- Rispondere nella classe B agli eventi che capitano nella classe A
- Qt fornisce il meccanismo dei Signal e Slot:
- Classi Qobject possono possedere metodi ti tipo Q_SIGNALS e metodi di tipo Q_SLOTS.
- Un segnale puo' essere connesso ad uno o piu' slot
  - connect(Object1 *, SIGNAL(foo), Object2 *, SLOT(bar()))
- Signal e slot devono avere la stessa signature per poter essere connessi (es. valueChanged(int) e manageValue(int))
- Emit valueChanged(3) avra' l'effetto di un'invocazione immediata di manageValue(3) (e tutti gli altri eventuali slot collegati)

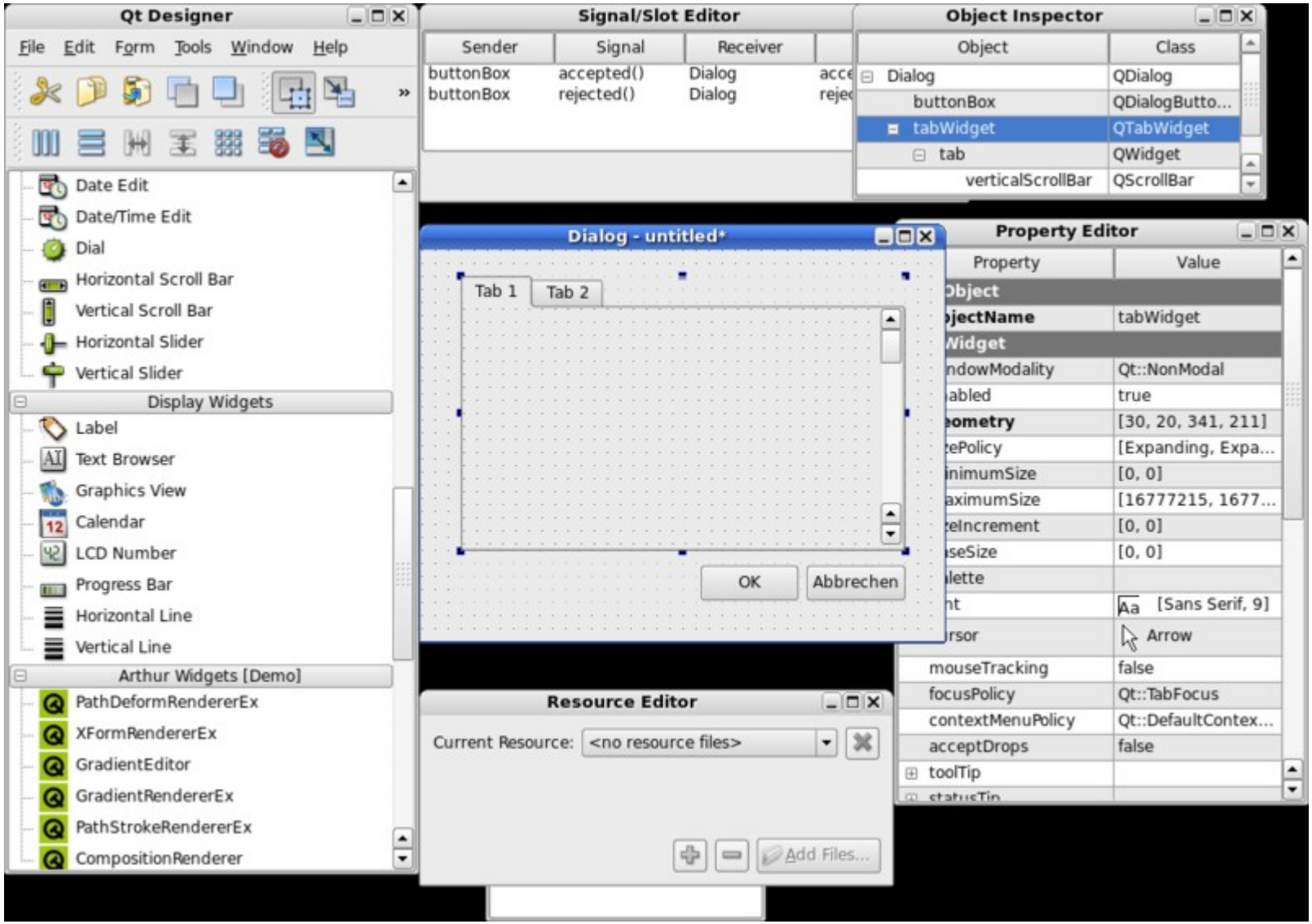
Signal e Slot

```
#include <QApplication>
    #include <QFont>
    #include <QLCDNumber>
    #include <QPushButton>
    #include <QSlider>
    #include <QVBoxLayout>
    #include <QWidget>

    class MyWidget : public QWidget
    {
    public:
        MyWidget(QWidget *parent = 0);
    };

    MyWidget::MyWidget(QWidget *parent)
        : QWidget(parent)
    {
        QPushButton *quit = new QPushButton(tr("Quit"));
        quit->setFont(QFont("Times", 18, QFont::Bold));

        QLCDNumber *lcd = new QLCDNumber(2);
        lcd->setSegmentStyle(QLCDNumber::Filled);

        QSlider *slider = new QSlider(Qt::Horizontal);
        slider->setRange(0, 99);
        slider->setValue(0);

        connect(quit, SIGNAL(clicked()), qApp,
SLOT(quit()));
        connect(slider, SIGNAL(valueChanged(int)),
                lcd, SLOT(display(int)));

        QVBoxLayout *layout = new QVBoxLayout;
        layout->addWidget(quit);
        layout->addWidget(lcd);
        layout->addWidget(slider);
        setLayout(layout);
    }

    int main(int argc, char *argv[])
    {
        QApplication app(argc, argv);
        MyWidget widget;
        widget.show();
        return app.exec();
    }
```

GUI: QWidget

- Gli oggetti visualizzati a schermo sono sottoclassi di QWidget
- È la classe base che riceve gli eventi dall'utente (mouse, keyboard, touch) ed effettua il paint a schermo della grafica
- Se un qwidget non ha parent (0) È una finestra top level, se il parent e' un altro qwidget, apparira' all'interno di quest'ultimo
- I widget sono posizionati e ridimensionati dentro ad un altro attraverso dei *layout* (QboxLayout, QGridLayout)

Qt Designer


Qt Designer
File Edit Form Tools Window Help
Date Edit
Date/Time Edit
Dial
Horizontal Scroll Bar
Vertical Scroll Bar
Horizontal Slider
Vertical Slider
Display Widgets
Label
Text Browser
Graphics View
Calendar
LCD Number
Progress Bar
Horizontal Line
Vertical Line
Arthur Widgets [Demo]
PathDeformRendererEx
XFormRendererEx
GradientEditor
GradientRendererEx
PathStrokeRendererEx
CompositionRenderer
Signal/Slot Editor
Sender Signal Receiver
buttonBox accepted() Dialog
buttonBox rejected() Dialog
Object Inspector
Object Class
Dialog QDialog
buttonBox QDialogButto...
tabWidget QTabWidget
tab QWidget
verticalScrollBar QScrollBar
Dialog - untitled*
Tab 1
Tab 2
OK
Abbrechen
Property Editor
Property Value
Object
objectName tabWidget
Widget
windowModality Qt::NonModal
enabled true
geometry [30, 20, 341, 211]
sizePolicy [Expanding, Expa...
minimumSize [0, 0]
maximumSize [16777215, 1677...
sizeIncrement [0, 0]
baseSize [0, 0]
palette
font [Sans Serif, 9]
cursor Arrow
mouseTracking false
focusPolicy Qt::TabFocus
contextMenuPolicy Qt::DefaultContex...
acceptDrops false
toolTip
statusTip
Resource Editor
Current Resource: <no resource files>
Add Files...

Paint

- Il disegno dei widget avviene nel metodo virtuale reimplementato paintEvent()
- Un paint deve essere brevissimo per assicurare fluidita'
- Classe base: QPainter
- Esegue paint su sottoclassi di QPaintDevice: widget, pixmap, immagini, pdf, svg, stampante

QPainter

- Funzioni per varie primitive, come linee, ellissi, rettangoli, immagini
- Es:

```
void Lines::paintEvent(QPaintEvent *event)
{
  QPen pen(Qt::black, 2, Qt::SolidLine);

  QPainter painter(this);

  painter.setPen(pen);
  painter.drawLine(20, 40, 250, 40);

  pen.setStyle(Qt::DashLine);
  painter.setPen(pen);
  painter.drawLine(20, 80, 250, 80);
  painter.setBrush(QBrush("#9e4757"));
  painter.drawRect(250, 105, 90, 60);
...
```

QImage vs QPixmap

- QImage: allocata in memoria, solo software rendering, possibilita' di manipolare singoli pixel, salvare e caricare da disco
- QPixmap: allocata nella scheda video, hardware rendering, possibile solo eseguire paint, in X11 È gestita direttamente dal server
- Possibile convertire da immagini a pixmap, costoso (fare solo quando necessario e fuori dal paint event)

QML

- Linguaggio dichiarativo per costruire interfacce grafiche per dispositivi mobili
- Parte imperativa in JavaScript (WebCore, prossimamente V8)
- Opera con qualsiasi Qobject
- Slegato da QWidget
- In Qt5 sara' basato su una scena OpenGL (Scene-graph)
- A lungo termine sostituira' QWidget

Esempio

```
import QtQuick 1.0

 Rectangle {
     width: 200
     height: 200
     color: "blue"

     Image {
         source: "pics/logo.png"
         anchors.centerIn: parent
     }
 }
```

Link

- Download: http://qt.nokia.com/downloads
- Training: http://qt.nokia.com/developer/learning/online/trainin
- Documentazione API: http://doc.qt.nokia.com/