PyQt for Desktop and Embedded Devices

David Boddie

<david.boddie@nokia.com>

PyCon Tre, Firenze, 8th–10th May 2009

About Qt

- Developed by Qt Software (Nokia)
- Cross-platform C++ framework
  - Linux, Windows, Mac OS X, other Unixes
  - Embedded Linux, Windows CE, Series 60
- Not just a widget toolkit – other features
- Available under the GPL (version 3)
- Available under the LGPL (version 2·1)
- Also available under a Commercial License

About PyQt

- Developed by Riverbank Computing Ltd.
- Set of Python bindings to Qt
  - Upcoming PyQt 4.5 will support Python 3
- Bindings are generated using SIP
- Includes most features of Qt
- Available under the GPL (version 2 and 3)
- Also available under a Commercial License

PyQt Modules

PyQt exposes many of Qt's 21 modules.

QtCore	QtDesigner	QtGui
QtHelp	QtNetwork	QtOpenGL
QtScript	QtSql	QtSvg
QtWebKit	QtXmlPatterns	Phonon

PyQt Modules

PyQt exposes many of Qt's 21 modules.

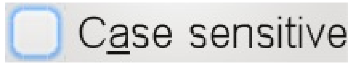
We'll take a quick look at these:

QtGui

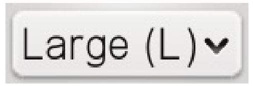
QtWebKit **QtXmlPatterns** **Phonon**

Graphical User Interfaces

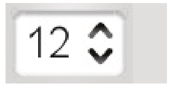
PyQt includes a comprehensive set of widgets:



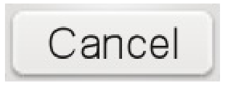
QCheckBox



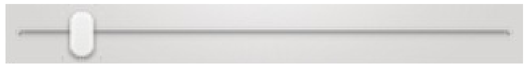
QComboBox



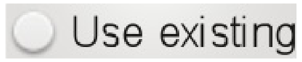
QSpinBox



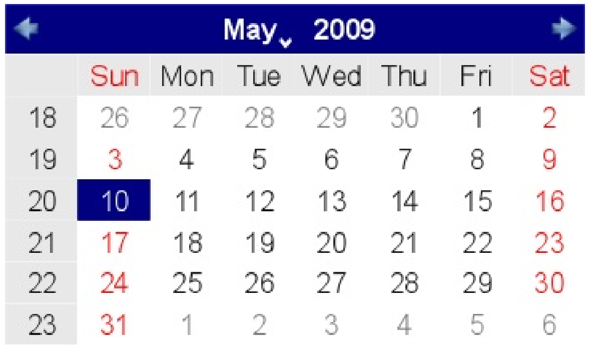
QPushButton



QSlider

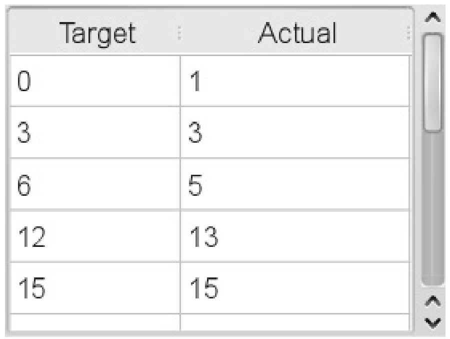


QRadioButton



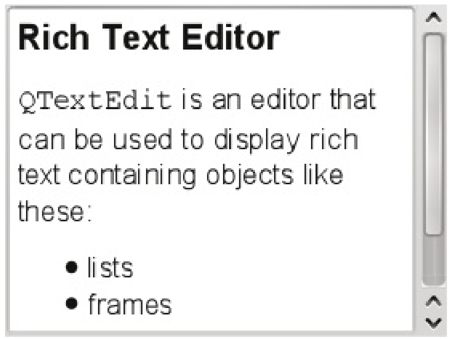
	Sun	Mon	Tue	Wed	Thu	Fri	Sat
18	26	27	28	29	30	1	2
19	3	4	5	6	7	8	9
20	10	11	12	13	14	15	16
21	17	18	19	20	21	22	23
22	24	25	26	27	28	29	30
23	31	1	2	3	4	5	6

QCalendarWidget



Target	Actual
0	1
3	3
6	5
12	13
15	15

QTableWidget



Rich Text Editor

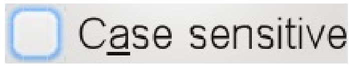
`QTextEdit` is an editor that can be used to display rich text containing objects like these:

- lists
- frames

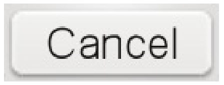
QTextEdit

Graphical User Interfaces

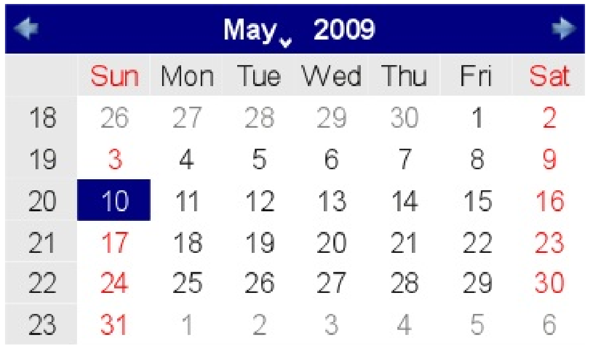
PyQt includes a comprehensive set of widgets:



QCheckBox



QPushButton



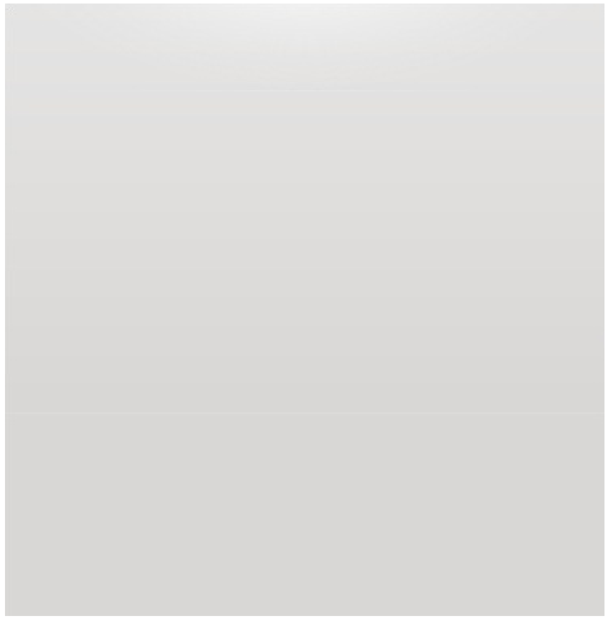
	Sun	Mon	Tue	Wed	Thu	Fri	Sat
18	26	27	28	29	30	1	2
19	3	4	5	6	7	8	9
20	10	11	12	13	14	15	16
21	17	18	19	20	21	22	23
22	24	25	26	27	28	29	30
23	31	1	2	3	4	5	6

QCalendarWidget

```
import sys
from PyQt4.QtGui import *

app = QApplication(sys.argv)

checkBox = QCheckBox("C&ase sensitive")
checkBox.show()

pushButton = QPushButton("Cancel")
pushButton.show()

calendar = QCalendarWidget()
calendar.show()

sys.exit(app.exec_())
```

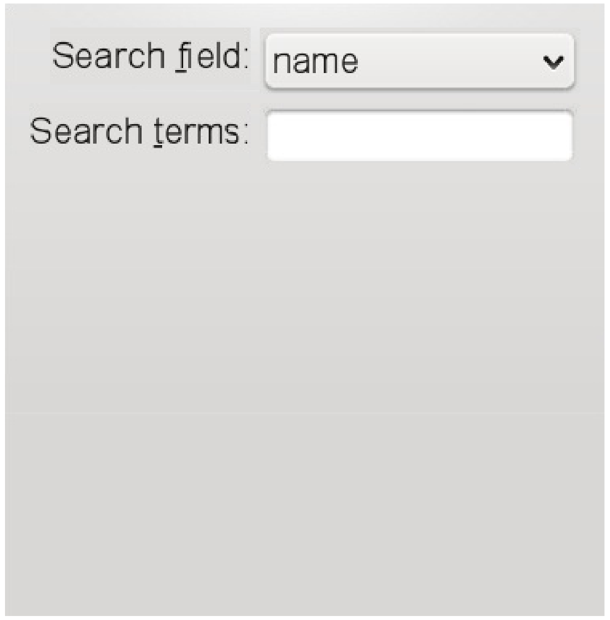
Widgets and Layouts

```
class PyPIWidget(QWidget):
  def __init__(self, parent = None):
    QWidget.__init__(self, parent)
```



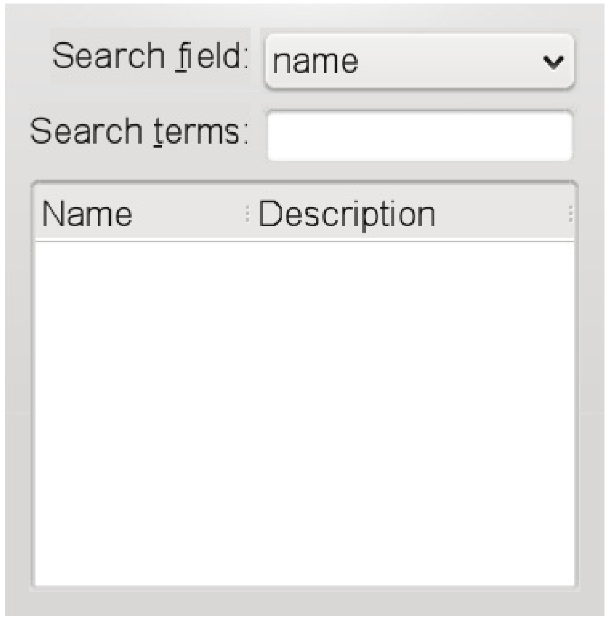
Widgets and Layouts

```
class PyPIWidget(QWidget):
  def __init__(self, parent = None):
    QWidget.__init__(self, parent)
    self.fieldCombo = QComboBox()
    self.fieldCombo.addItems(["name", "version",
        "author", "author_email", "maintainer",
        "maintainer_email", "home_page",
        "license", "summary", "description",
        "keywords", "platform", "download_url"])
    self.termsEdit = QLineEdit()
    layout = QFormLayout()
    layout.addRow(self.tr("Search &field:"),
                  self.fieldCombo)
    layout.addRow(self.tr("Search &terms:"),
                  self.termsEdit)
```



Widgets and Layouts

```
class PyPIWidget(QWidget):
  def __init__(self, parent = None):
    QWidget.__init__(self, parent)
    self.fieldCombo = QComboBox()
    self.fieldCombo.addItems(["name", "version",
        "author", "author_email", "maintainer",
        "maintainer_email", "home_page",
        "license", "summary", "description",
        "keywords", "platform", "download_url"])
    self.termsEdit = QLineEdit()
    layout = QFormLayout()
    layout.addRow(self.tr("Search &field:"),
                  self.fieldCombo)
    layout.addRow(self.tr("Search &terms:"),
                  self.termsEdit)
    self.treeWidget = QTreeWidget()
    self.treeWidget.setAlternatingRowColors(True)
    self.treeWidget.setRootIsDecorated(False)
    self.treeWidget.setHeaderLabels(
        [self.tr("Name"), self.tr("Description")])
    mainLayout = QVBoxLayout()
    mainLayout.addLayout(layout)
    mainLayout.addWidget(self.treeWidget)
```

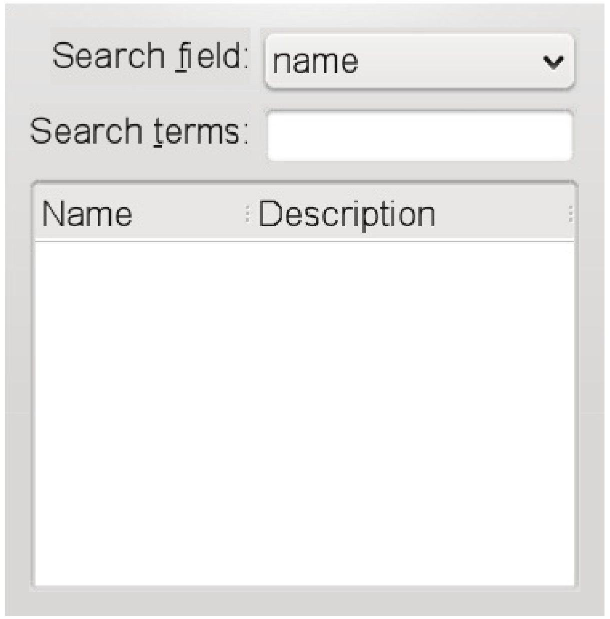


Search field: name

Search terms:

Widgets and Layouts

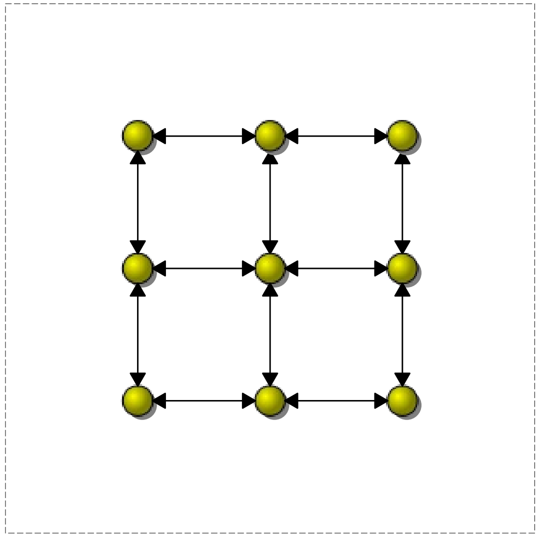
```
class PyPIWidget(QWidget):
  def __init__(self, parent = None):
    QWidget.__init__(self, parent)
    self.fieldCombo = QComboBox()
    self.fieldCombo.addItems(["name", "version",
        "author", "author_email", "maintainer",
        "maintainer_email", "home_page",
        "license", "summary", "description",
        "keywords", "platform", "download_url"])
    self.termsEdit = QLineEdit()
    layout = QFormLayout()
    layout.addRow(self.tr("Search &field:"),
                  self.fieldCombo)
    layout.addRow(self.tr("Search &terms:"),
                  self.termsEdit)
    self.treeWidget = QTreeWidget()
    self.treeWidget.setAlternatingRowColors(True)
    self.treeWidget.setRootIsDecorated(False)
    self.treeWidget.setHeaderLabels(
        [self.tr("Name"), self.tr("Description")])
    mainLayout = QVBoxLayout()
    mainLayout.addLayout(layout)
    mainLayout.addWidget(self.treeWidget)
    self.connect(self.termsEdit, SIGNAL("returnPressed()"), self.search)
```



Graphics View

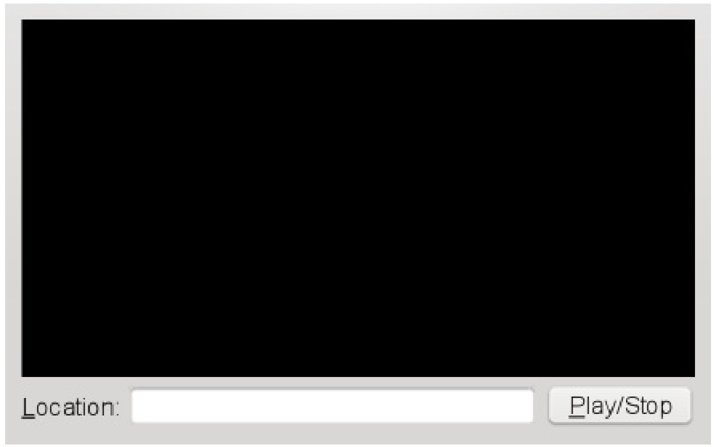
The Graphics View framework provides a canvas:

- **Interactive items (drag and drop)**
- **Nested items and groups**
- **Animation**
- **OpenGL rendering**
- **Embedded widgets**
- **Printing**



Multimedia Support

Phonon handles audio and video:



```
class Player(QWidget):
  def __init__(self, parent = None):
    QWidget.__init__(self, parent)
    self.player = VideoPlayer(VideoCategory)
    # ...

  def play(self):
    if self.player.isPlaying():
        self.player.stop()
    else:
        url = QUrl(self.urlEdit.text())
        self.player.play(MediaSource(url))
```

XML Processing

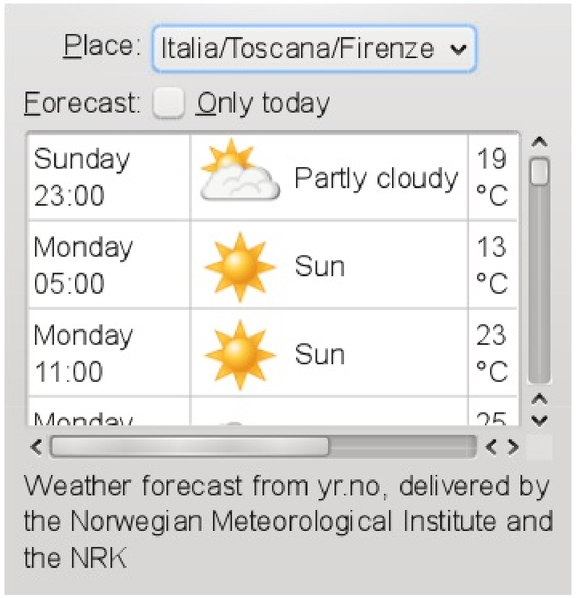
Use XPath, XQuery and XSLT to process XML:

```
<sun rise="2009-05-01T05:32:17" set="2009-05-01T20:23:58" />
<forecast>
  <tabular>
    <time from="2009-05-01T06:00:00" to="2009-05-01T12:00:00" period="1">
      <symbol number="2" name="Fair" />
      <precipitation value="0.0" />
      <windDirection deg="114.6" code="ESE" name="East-southeast" />
      <windSpeed mps="0.8" name="Light air" />
      <temperature unit="celcius" value="8" />
      <pressure unit="hPa" value="1022.2" />
    </time>
    ...
```

```
declare variable $url external;
(
string(doc($url)//sun/@rise),
string(doc($url)//sun/@set),

for $time in doc($url)//tabular/time
order by $time/@from
return (string($time/@from),
        string($time/symbol/@name), string($time/symbol/@number),
        string($time/temperature/@value), string($time/temperature/@unit))
)
```

XML Processing

Use XPath, XQuery and XSLT to process XML:



Sunday 23:00	Partly cloudy	19 °C
Monday 05:00	Sun	13 °C
Monday 11:00	Sun	23 °C
Monday		25

Weather forecast from yr.no, delivered by the Norwegian Meteorological Institute and the NRK

```
def fetchForecast(self, place):
  url = "http://www.yr.no/place/" + \
        place + "/forecast.xml"
  b = QBuffer()
  b.setData(query_string)
  b.open(QBuffer.ReadOnly)
  query = QXmlQuery()
  query.bindVariable("url",
                     QXmlItem(QVariant(url)))
  query.setQuery(b)
  b.close()
  if query.isValid():
      self.results.clear()
      query.evaluateTo(self.results)
      self.updateTable()
```

Could be useful if combined with GPS...

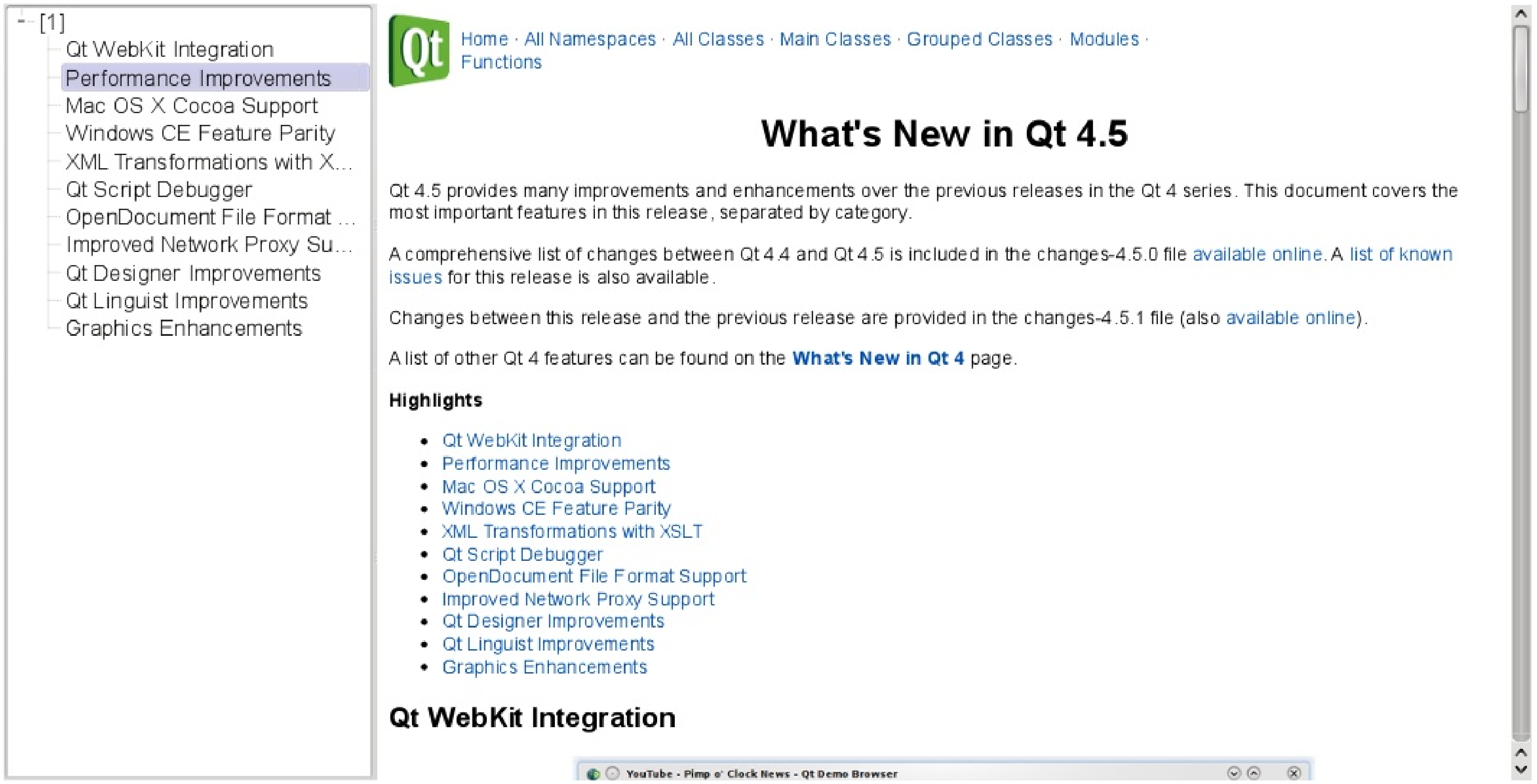
Web Browser Engine

WebKit is integrated into Qt:

- Web browser widget
- JavaScript, SVG, CSS, SSL, etc.
- Control over browser settings and history
- Support for Netscape and native Qt plugins
- Client-side storage
- Support for in-page editing
- Python/C++ objects can be added to pages

Web Browser Engine

WebKit is integrated into Qt:


[1]
Qt WebKit Integration
Performance Improvements
Mac OS X Cocoa Support
Windows CE Feature Parity
XML Transformations with X...
Qt Script Debugger
OpenDocument File Format ...
Improved Network Proxy Su...
Qt Designer Improvements
Qt Linguist Improvements
Graphics Enhancements

Home · All Namespaces · All Classes · Main Classes · Grouped Classes · Modules · Functions

What's New in Qt 4.5

Qt 4.5 provides many improvements and enhancements over the previous releases in the Qt 4 series. This document covers the most important features in this release, separated by category.

A comprehensive list of changes between Qt 4.4 and Qt 4.5 is included in the changes-4.5.0 file available online. A list of known issues for this release is also available.

Changes between this release and the previous release are provided in the changes-4.5.1 file (also available online).

A list of other Qt 4 features can be found on the **What's New in Qt 4** page.

Highlights

- Qt WebKit Integration
- Performance Improvements
- Mac OS X Cocoa Support
- Windows CE Feature Parity
- XML Transformations with XSLT
- Qt Script Debugger
- OpenDocument File Format Support
- Improved Network Proxy Support
- Qt Designer Improvements
- Qt Linguist Improvements
- Graphics Enhancements

Qt WebKit Integration

YouTube - Pimp o' Clock News - Qt Demo Browser

Intermission

These features are nice on the desktop!

We can also use them on embedded hardware.

Embedded Platforms

- Qt runs on Embedded Linux, Windows CE, Series 60
- Python runs on Embedded Linux, Windows CE, Series 60
- PyQt runs on Embedded Linux
  - Windows CE?, Series 60?

Embedded Platforms

Disadvantages:

- Small screens (≈ 240 × 320)
- Low memory (≈ 64 MB)
- Slow processors (≈ 300 MHz)
- Different architectures
- Limited storage (≈ 128MB)
- Cut down environments

Advantages:

- Can be portable
- Accelerometers
- Touch screens
- GPS
- GSM, Wi-Fi, Bluetooth
- Cameras

Embedded Platforms

Where is Embedded Linux used:

- Phones, media players
- GPS devices, Web tablets
- Set top boxes
- Routers, plug computing
- Handhelds, toys, kit computing

`http://www.linuxdevices.com/`

Embedded Python

Nice things about Python (2.x):

- Portable C implementation
- Fairly small (compared to all the Qt libraries)
- Few dependencies
- Batteries included

Embedded Python

Not so nice things about Python (2.x):

- Annoying to cross-compile (despite a good foundation)
- Relies on a native interpreter at various points
  - Runs setup.py using the built interpreter
  - Needs a native interpreter to build a parser generator
- Package-specific checks
  - OpenSSL, Curses, pyexpat, Tkinter
- Batteries included (even old ones)

Embedded Python

Ways to build Python for Linux Devices:

- **OpenEmbedded**
- **Scratchbox**
- **Buildroot**
- **Crosstool**
- **Distribution packages (e.g., Debian)**

  ...

Used to create toolchains and/or whole systems.

Embedded Python

Ways I built Python for Linux Devices:

- Scratchbox
  - Used to try PyQt on Maemo
  - There are packages available now
- Crosstool
  - Used to try PyQt on a Greenphone

Qt and PyQt on Linux Devices

Which graphics system to use?

X11 (Qt for X11):

- Fairly standard procedure for building Qt and PyQt
- Not all that common to cross-compile Qt for X11
- Qt works better with X extensions (Render)
- Develop using PyQt for X11

QWS (Qt for Embedded Linux):

- Fairly easy to build Qt for Embedded Linux
- PyQt needs patching for cross-compilation
- Qt uses the framebuffer
- Develop using PyQt derived from PyQt for X11

Qt and PyQt on Linux Devices

Developing and Simulating

X11-based devices:

- Use Xephyr (nested X server) to simulate a small screen
- Run or simulate the device's window system
- The Maemo SDK emulates the device environment
- Possible to use system PyQt to prototype applications
  - Beware of version differences

```
Xephyr -ac -extension Composite -screen 800x480 :1
DISPLAY=:1 python application.py
```

Qt and PyQt on Linux Devices

QWS-based devices:

- Use a virtual framebuffer or VNC to simulate the screen
- Run or simulate the device's window system
- No need for a dedicated window manager
- Need to build your own libraries for desktop and device

```
python application.py -qws -display VNC:0:size=240x320
vncviewer :0
```

Qt and PyQt on Linux Devices

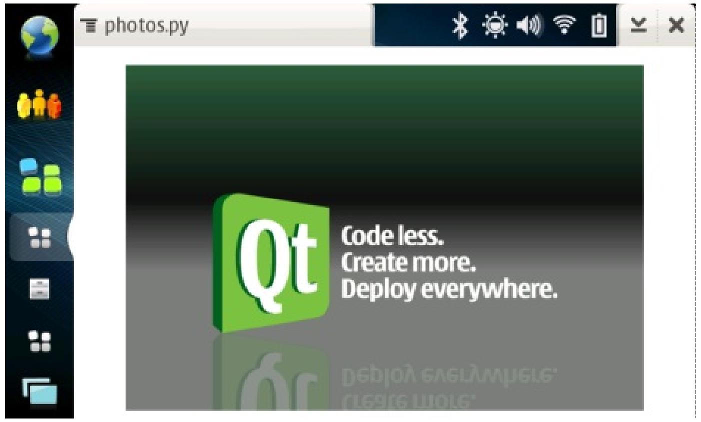
Common approach:

- Applications can be developed on desktops using PyQt
- Obviously, look and feel may be a bit different
- The APIs should be the same
- Care must be taken with the widgets used:
  - Input widgets without a keyboard...
  - Scrollbars on a small screen...

Demonstrations

Two devices:

Nokia N800:



CPU: 400 MHz OMAP2420

Screen: 800 × 480

Greenphone:

CPU: 312 MHz Intel PXA270

Screen: 240 × 320

At this point we show some demonstrations...

Cutting Out Features

On desktops, we want as many features as possible...
...but...
...on embedded devices, we might not want everything.

Cutting Out Features

Inappropriate features:

- Classic dialogs too large for small screen devices
- Menu bars, dock windows follow the wrong paradigm

Unnecessary features:

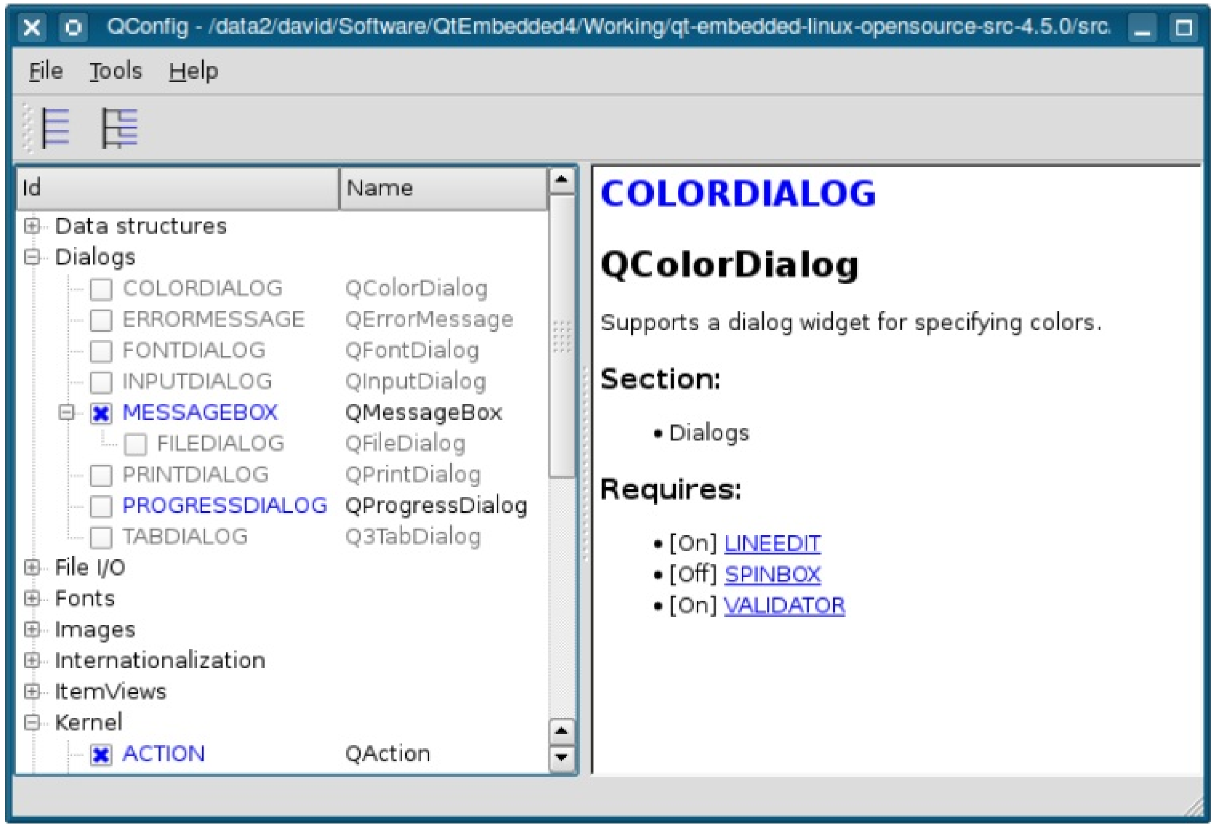
- Devices with touch screens don't need cursors
- Specialized displays don't need all the widget styles

Redundant features:

- We don't really need:
  - four sets of XML classes,
  - two JavaScript engines,
  - two ways to access networked resources

Cutting Out Features

Configuring Qt for Embedded Linux:



QConfig - /data2/david/Software/QtEmbedded4/Working/qt-embedded-linux-opensource-src-4.5.0/src

Id	Name
Data structures	
Dialogs	
☐ COLORDIALOG	QColorDialog
☐ ERRORMESSAGE	QErrorMessage
☐ FONTDIALOG	QFontDialog
☐ INPUTDIALOG	QInputDialog
☒ MESSAGEBOX	QMessageBox
☐ FILEDIALOG	QFileDialog
☐ PRINTDIALOG	QPrintDialog
☐ PROGRESSDIALOG	QProgressDialog
☐ TABDIALOG	Q3TabDialog
File I/O	
Fonts	
Images	
Internationalization	
ItemViews	
Kernel	
☒ ACTION	QAction

COLORDIALOG

QColorDialog

Supports a dialog widget for specifying colors.

Section:

- Dialogs

Requires:

- [On] LINEEDIT
- [Off] SPINBOX
- [On] VALIDATOR

QConfig lets you remove features from an embedded build

Finishing Up

PyQt makes it possible to

- **Write code that works on different (embedded) platforms**
- **Despite allowances for differences between devices**
  - **You get to work around those at a high level**
- **Prototype applications on the desktop**

Finishing Up

PyQt makes it possible to

- Write code that works on different (embedded) platforms
- Despite allowances for differences between devices
  - You get to work around those at a high level
- Prototype applications on the desktop

Python makes it possible to

- Ignore problems like cross-compiling
  - Build on top of existing pure-Python code
- Write portable, deployable applications
- Take advantage of interactivity to prototype on devices

Resources

Qt

http://www.qtsoftware.com/

PyQt

http://www.riverbankcomputing.com/

PyQt and PyKDE Wiki

http://www.diotavelli.net/PyQtWiki/

Develer

http://www.develer.com/

Rapid GUI Programming with Python and Qt
by Mark Summerfield