Getting Started with PyQt4

David Boddie
david@boddie.org.uk

7th February 2009

Overview

PyQt is a set of Python bindings to the Qt application framework, providing access to features which include:

- Widgets and other graphical user interface controls
- Database management and querying
- XML handling and processing
- Graphics and multimedia
- Web browser integration and networking

PyQt is available under the GNU General Public License.

Installing Qt and PyQt

Both Qt and PyQt are available as standard in many GNU/Linux distributions:

- **Debian, Ubuntu:** `python-qt4` and `pyqt4-dev-tools`
- **openSUSE:** `python-qt4`
- **Mandriva:** `python-qt4`
- **Fedora:** `PyQt4`
- **Pardus:** `PyQt4`

The PyQt download page contains:

- Binary installers for Windows
- Source packages for Mac OS X, Windows, Unix and GNU/Linux

Checking the Installation

Start a Python session in a terminal and type:

```
from PyQt4.QtCore import QT_VERSION_STR
print QT_VERSION_STR
```

The version of Qt in use should be printed to the terminal.

```
>>> from PyQt4.QtCore import QT_VERSION_STR
>>> print QT_VERSION_STR
4.3.2
```

The version reported will depend on your PyQt installation.

Hello World in PyQt

Here's a simple PyQt application:

```
import sys
from PyQt4.QtGui import QApplication, QPushButton
app = QApplication(sys.argv)
button = QPushButton("Hello world!")
button.show()
sys.exit(app.exec_())
```

This does four things:

1. Creates an application object
2. Creates a button
3. Shows the button
4. Runs the event loop

PyQt Concepts – The Event Loop

The application communicates with the system via events.

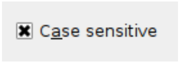
- We do not dispatch events ourselves.
- We ask the QApplication object to run an event loop.
- The event loop controls the execution of the application.
- We give up control when it starts running, but it calls functions and methods that we provide.

The application won't work unless we start an event loop by calling the QApplication object's exec_() method.

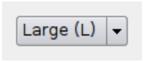
Widgets

Widgets are graphical elements that the user interacts with.

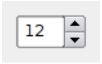
```
checkbox = QCheckBox("C&ase sensitive")
combobox = QComboBox()
combobox.addItem("Large (L)")
spinbox = QSpinBox()
```



QCheckBox



QComboBox

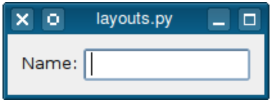


QSpinBox

Widgets and Layouts

Widgets can be used as containers to hold other widgets.

```
window = QWidget()
label = QLabel("Name:")
editor = QLineEdit()
layout = QHBoxLayout(window)
layout.addWidget(label)
layout.addWidget(editor)
window.show()
```



- The label, line edit and window are created in the same way.
- The layout puts the label and line edit inside the widget.
- The window is the parent of both the label and line edit.

PyQt Concepts – Parents and Children

Container widgets are parents of the widgets inside them:

- Windows have no parents.
- Child widgets can have their own children.
- Children of a widget are only visible if their parent is.

Parent widgets "own" their children:

```
def create_window():
    window = QWidget()
    editor = QLineEdit()
    layout = QHBoxLayout(window)
    layout.addWidget(editor)
    return window
```

The editor and layout objects go out of scope but are not deleted.