FREE eBook

LEARNING

pyqt

Free unaffiliated eBook created from
Stack Overflow contributors.

#pyqt

Table of Contents

About 1

Chapter 1: Getting started with pyqt 2

Remarks 2

Examples 2

Installation of PyQt4 2

A basic application 3

Hello world 3

A Simple Drag & Drop Sample 4

Chapter 2: Using threads with PyQt 8

Remarks 8

Examples 8

The worker model 8

Credits 10

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: pyqt

It is an unofficial and free pyqt ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official pyqt.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with pyqt

Remarks

PyQt is a Python binding to the popular cross-platform Qt application framework commonly used to make graphical applications. PyQt4 supports Qt4 and PyQt5 supports Qt5. It runs on all platforms supported by Qt (Windows, OS X, Linux, iOS and Android). The bindings are implemented as a set of Python modules and classes.

For more information see the PyQt website.

Examples

Installation of PyQt4

Suggested Install Method

Windows: Download and run the binary setup file.

Linux(Debian): Run this command in your command line:

```
$ apt-get install python-qt4 pyqt4-dev-tools qt4-designer
```

OS X : Run this command in your command line:

```
$ brew install pyqt
```

Install Manually

You can also download the source code manually from here and then install and configure it yourself.

Test your installation

If pyqt is installed correctly, you will be able to run the `pyuic4` command. If it is installed correctly, you will see the following error:

```
$ pyuic4
Error: one input ui-file must be specified
```

Installation Complete

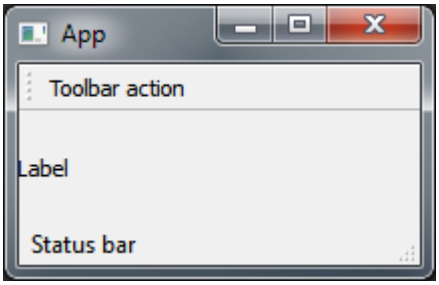
You have now installed the PyQt4 library. Two useful applications have also been installed along side PyQt4:

- Qt Designer: An application for 'drag & drop' design of graphical interfaces (creates `.ui` files),

- pyuic4: A command line application that can convert `.ui` files into Python code.

A basic application

The following example shows a basic main GUI window with a label widget, a toolbar, and a status bar using PyQt4.




```
import sys
from PyQt4 import QtGui


class App(QtGui.QApplication):
    def __init__(self, sys_argv):
        super(App, self).__init__(sys_argv)
        self.build_ui()

    def build_ui(self):
        # build a main GUI window
        self.main_window = QtGui.QMainWindow()
        self.main_window.setWindowTitle('App')
        self.main_window.show()

        # add a label to the main window
        label = QtGui.QLabel('Label')
        self.main_window.setCentralWidget(label)

        # add a toolbar with an action button to the main window
        action = QtGui.QAction('Toolbar action', self)
        toolbar = QtGui.QToolBar()
        toolbar.addAction(action)
        self.main_window.addToolBar(toolbar)

        # add a status bar to the main window
        status_bar = QtGui.QStatusBar()
        status_bar.showMessage('Status bar')
        self.main_window.setStatusBar(status_bar)


if __name__ == '__main__':
    app = App(sys.argv)
    sys.exit(app.exec_())
```

Hello world

This basic code will launch a "Hello world" GUI window using PyQt4:

```
import sys
```

```
from PyQt4 import QtGui

# create instance of QApplication
app = QtGui.QApplication(sys.argv)

# create QLabel, without parent it will be shown as window
label = QtGui.QLabel('Hello world!')
label.show()

# start the execution loop of the application
sys.exit(app.exec_())
```

This is the same code using PyQt5.

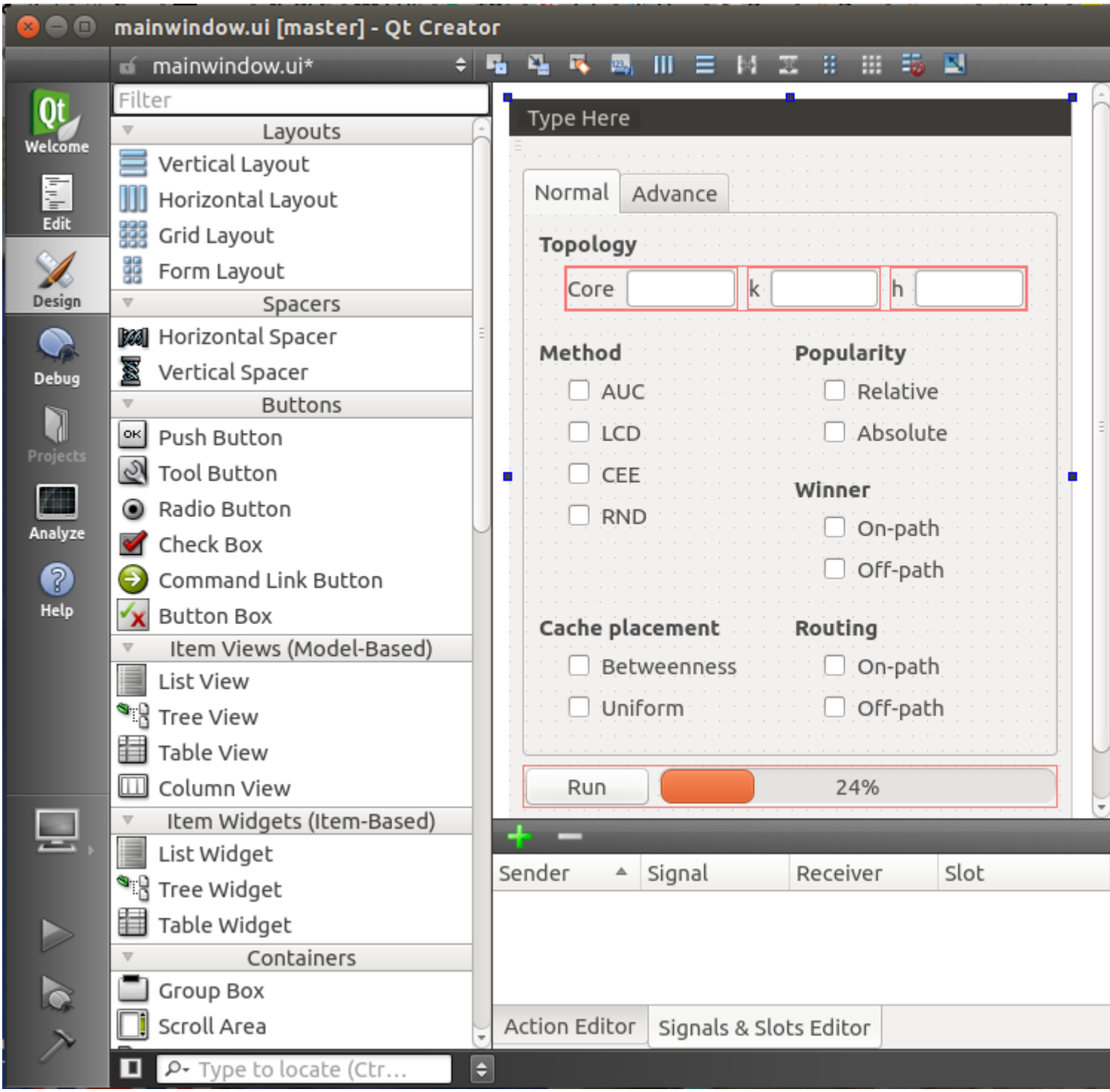
```
import sys
from PyQt5 import QtWidgets

# create instance of QApplication
app = QtWidgets.QApplication(sys.argv)

# create QLabel, without parent it will be shown as window
label = QtWidgets.QLabel('Hello world!')
label.show()

# start the execution loop of the application
sys.exit(app.exec_())
```

A Simple Drag & Drop Sample

Make a simple GUI application in 3 easy steps.

1. Design

Open `Qt Creator`, create a new project and make your design. Save your result as `.ui` file (here: `mainwindow.ui`).



2. Generate corresponding .py file

Now you can create a .py file from your .ui file that you generated in the previous step. Enter the following into your command line:

```
$ pyuic4 mainwindow.ui -o GUI.py
```

If the above line is run successfully a `GUI.py` file is created.

3. Python codes

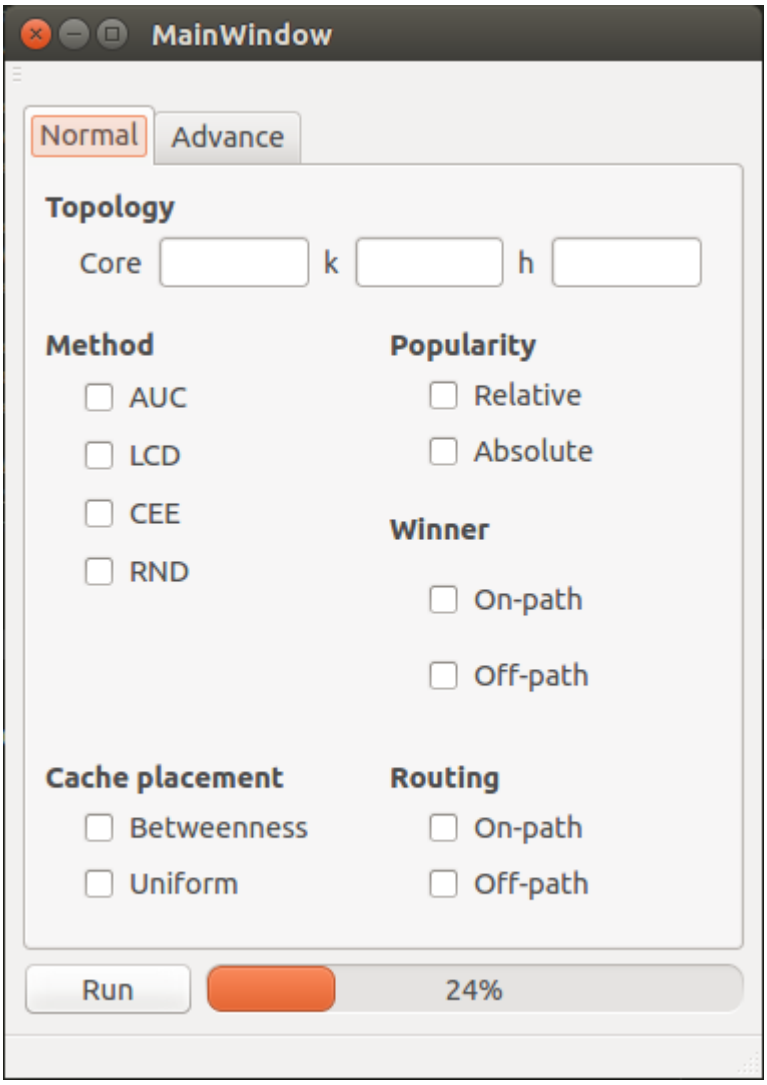
You can add your own code (e.g. signals and slots) in the `GUI.py` file but it's better to add them in a

new file. If you ever want to make changes to your GUI, the `GUI.py` file will be overwritten. That's why using another file to add functionality is better in most cases.

Let's call the new file `main.py`.

```
from PyQt4 import QtGui
import sys
import GUI # Your generated .py file


class MyApp(QtGui.QMainWindow, GUI.Ui_MainWindow):
    def __init__(self, parent=None):
        super(ExampleApp, self).__init__(parent)
        self.setupUi(self)

        # Connect a button to a function
        self.btn_run.clicked.connect(self.run)

    def run(self):
        # Write here what happens after the button press
        print("run")


if __name__ == '__main__':
    app = QtGui.QApplication(sys.argv)
    form = ExampleApp()
    form.show()
    app.exec_()
```

Now you can run `main.py` and see your GUI.



Read Getting started with pyqt online: https://riptutorial.com/pyqt/topic/1715/getting-started-with-pyqt

Chapter 2: Using threads with PyQt

Remarks

While some parts of the Qt framework are thread safe, much of it is not. The Qt C++ documentation provides a good overview of which classes are reentrant (can be used to instantiate objects in multiple threads). The following rules are the most widely sought:

- You cannot create or access a Qt GUI object from outside the main thread (e.g. anything that subclasses `QWidget` or similar).
- Even if the Qt class is reentrant, you cannot share access to a Qt object between threads unless the Qt documentation for that class explicitly states that instances are thread safe.
- You can use `QObject.moveToThread()` if you need to move a Qt object from one thread to another (does not apply to Qt GUI objects which must always remain in the main thread). But note that the object must not have a parent.

As per this Stack Overflow QA, it is not recommended to use Python threads if your thread intends to interact with PyQt in any way (even if that part of the Qt framework is thread safe).

Examples

The worker model

```
# this method can be anything and anywhere as long as it is accessible for connection
@pyqtSlot()
def run_on_complete():

    pass

# An object containing methods you want to run in a thread
class Worker(QObject):
    complete = pyqtSignal()

    @pyqtSlot()
    def a_method_to_run_in_the_thread(self):
        # your code

        # Emit the complete signal
        self.complete.emit()

# instantiate a QThread
thread = QThread()
# Instantiate the worker object
worker = Worker()
# Relocate the Worker object to the thread
worker.moveToThread(thread)
# Connect the 'started' signal of the QThread to the method you wish to run
thread.started.connect(worker.a_method_to_run_in_the_thread)
# connect to the 'complete' signal which the code in the Worker object emits at the end of the
method you are running
worker.complete.connect(run_on_complete)
```

```
# start the thread (Which will emit the 'started' signal you have previously connected to)
thread.start()
```

Read Using threads with PyQt online: https://riptutorial.com/pyqt/topic/2775/using-threads-with-pyqt

Credits

S. No	Chapters	Contributors
1	Getting started with pyqt	101, Achayan, Community, Ian, Makoto, mehD, three_pineapples, Trilarion
2	Using threads with PyQt	ederag, ekhumoro, three_pineapples