Table of content

Using QtWebKit and QML with PySide 1
WebKitView.py 1
Importing required modules 1
Send and receive helper functions 1
Putting it all together 1
WebKitView.qml 2
WebKitView.html 2
HTML header 2
JavaScript send and receive functions 2
Callbacks for our buttons 2
The HTML body contents 3
How the example app looks like 3

Using QtWebKit and QML with PySide

This PySide tutorial shows you how to integrate Python code and QtWebKit together with QML, so you can have HTML content and program logic inside a QML application while still being able to send messages back and forth between the JavaScript context inside the WebView and the Python world. It uses JSON, alert() and evaluateJavaScript() to exchange arbitrary data structures (values, lists, dictionaries) between Python and JavaScript in the WebView.

WebKitView.py

Importing required modules

The sys module is used to get command line arguments, time is used to get the current time and json is used to en- and decode data structures to/from JSON.

```
import sys
import time
import json

from PySide import QtGui, QtDeclarative
```

Send and receive helper functions

These two functions send data to the WebView and receive data from the WebView. sendData simply transforms the data into JSON and calls receiveJSON (as defined in the HTML file) via the evaluateJavaScript method to put the data there. receiveData is connected to the alert signal of the WebView object, and gets a string as parameter, which is decoded as JSON and processed. When we want to set the rotation, we simply set the QML property “rotation” on the WebView object, which will rotate it in our view.

```
def sendData(data):
    global rootObject
    print 'Sending data:', data
    json_str = json.dumps(data).replace('"', '\\"')
    rootObject.evaluateJavaScript('receiveJSON("%s")' % json_str)

def receiveData(json_str):
    global rootObject

    data = json.loads(json_str)
    print 'Received data:', data

    if len(data) == 2 and data[0] == 'setRotation':
        rootObject.setProperty('rotation', data[1])
    else:
        sendData({'Hello': 'from PySide', 'itsNow': int(time.time())})
```

Putting it all together

Instantiate a new QApplication, a new QDeclarativeView and load the QML file. We also set the render hints to SmoothPixmapTransform, which makes the rotated WebView content look nicer (if you want more performance, remove this line). We then set the url property of our WebView to our HTML file, and finally make sure that we connect to the alert signal of the WebView, which gets emitted when the JavaScript code inside the WebView executes alert().

```
app = QtGui.QApplication(sys.argv)

view = QtDeclarative.QDeclarativeView()
view.setRenderHints(QtGui.QPainter.SmoothPixmapTransform)
view.setSource(__file__.replace('.py', '.qml'))
rootObject = view.rootObject()
rootObject.setProperty('url', __file__.replace('.py', '.html'))
rootObject.alert.connect(receiveData)
view.show()

app.exec_()
```

WebKitView.qml

This one is relatively unspectacular. Simply import QtWebKit (for this you have to install the libqtwebkit4-declarative package) and create a new WebView. The settings.javascriptEnabled part here is key – without it, JavaScript inside the WebView will not be executed.

```
import QtWebKit 1.0

WebView { settings.javascriptEnabled: true; width: 400; height: 280 }
```

WebKitView.html

HTML header

Here's how we start out with the HTML file:

```
<html>
    <head>
        [removed]
```

JavaScript send and receive functions

The sendJSON function will be called by our code (see below), the receiveJSON function will be called from Python by executing JavaScript code on the WebView.

```
            function sendJSON(data) {
                alert(JSON.stringify(data));
            }
            function receiveJSON(data) {
                element = document.getElementById('received');
                element[removed] += "\n" + data;
            }
```

Callbacks for our buttons

We'll be using the sendJSON method to send data when the buttons are clicked. Here are the functions that generate the messages and then send the data to Python:

```
            function setRotation() {
                element = document.getElementById('rotate');
                angle = parseInt(element.value);
                message = ['setRotation', angle];
                sendJSON(message);
            }
            function sendStuff() {
                sendJSON([42, 'PySide', 1.23, true, {'a':1,'b':2}]);
            }
```

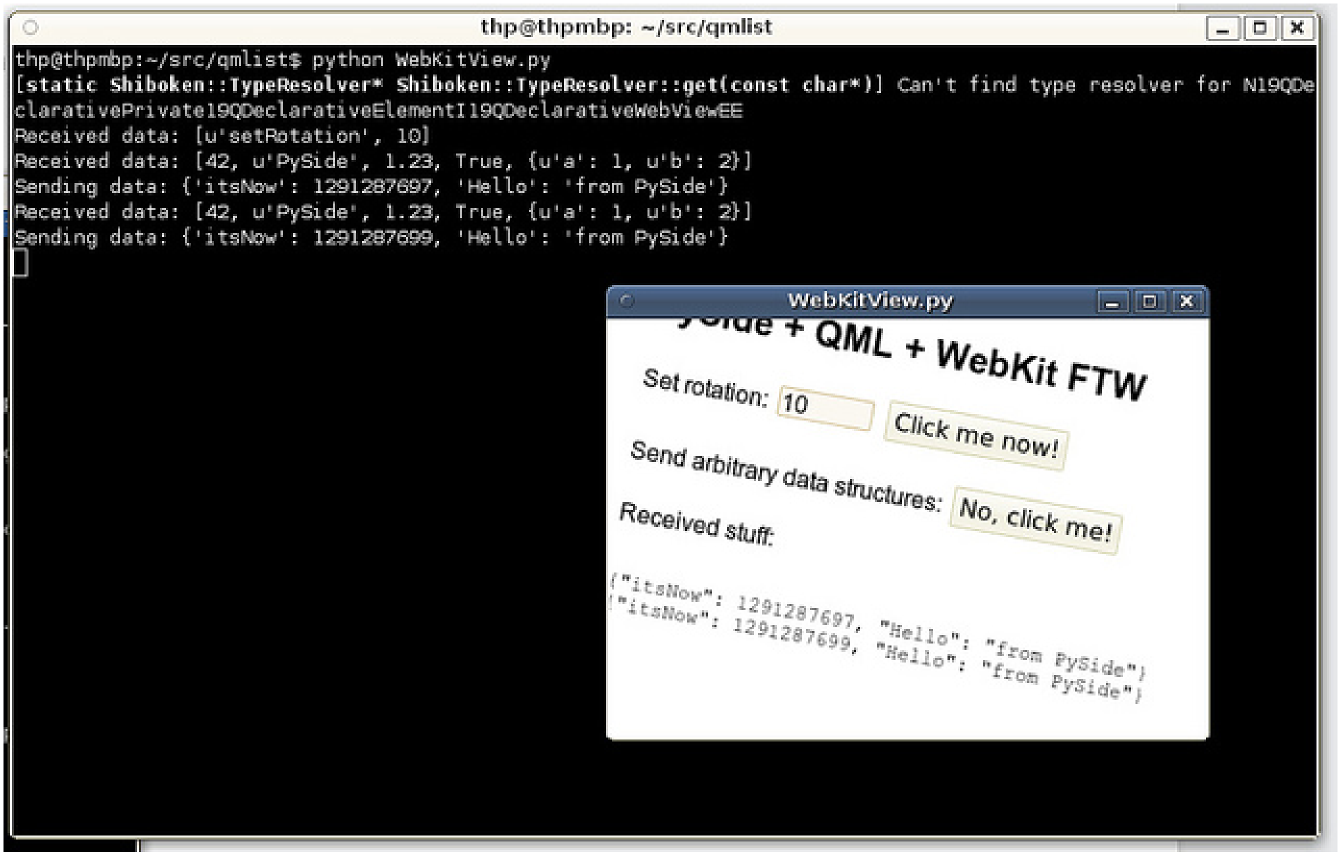
The HTML body contents

This is the rest of the HTML file, which simply gives some text, the input field, two buttons and a preformatted text element where we will insert received stuff from Python:

```
        [removed]
    </head>
    <body>
        [h2]PySide + QML + WebKit FTW[/h2]

        Set rotation:
        <input type="text" size="5" id="rotate" value="10"/>
        <button>Click me now!</button>


        Send arbitrary data structures:
        <button>No, click me!</button>

        Received stuff:
        <pre></pre>
    </body>
</html>
```

How the example app looks like

Once you have got all three files, start the example app like this: python WebKitView.py



```
thp@thpmbp:~/src/qmlist$ python WebKitView.py
[static Shiboken::TypeResolver* Shiboken::TypeResolver::get(const char*)] Can't find type resolver for N19QDe
clarativePrivate19QDeclarativeElementI19QDeclarativeWebViewEE
Received data: [u'setRotation', 10]
Received data: [42, u'PySide', 1.23, True, {u'a': 1, u'b': 2}]
Sending data: {'itsNow': 1291287697, 'Hello': 'from PySide'}
Received data: [42, u'PySide', 1.23, True, {u'a': 1, u'b': 2}]
Sending data: {'itsNow': 1291287699, 'Hello': 'from PySide'}
```

ySide + QML + WebKit FTW

Set rotation: 10 Click me now!

Send arbitrary data structures: No, click me!

Received stuff:

```
{"itsNow": 1291287697, "Hello": "from PySide"}
{"itsNow": 1291287699, "Hello": "from PySide"}
```