Creating input masks for QGIS using Python, PyQt, Qt Designer

FOSS4G, Boston, 18 August 2017

Numa Gremling
Some Basics
What is a GUI?

- A graphical user interface

- Basically: an interface that lets you interact with more complex tools in the background
When do you need a GUI?

- **An example:**
  - You have written an awesome Python script and you want others to use it
  - But those who should work with it do not know any Python
  - **Solution:** you create a graphical user interface so people can access your tools: by clicking buttons, not by writing code
Do I need a GUI in QGIS?

- Well... QGIS in itself consists of one huge GUI

- So why would you bother creating even more GUIs?

- There are a few reasons why you might decide to add your own GUI to QGIS

- The main reason: QGIS plugins!
What do I need to create a GUI in QGIS?

Qt & PyQt
What is Qt?

- QGIS uses the Qt framework for many of its visual components
- Basically, anything visual in QGIS is made using Qt
- Menus, toolbars, tool’s interfaces: they are all based on Qt

- Qt is very extensive and offers endless possibilities
- But... Qt is based on C++
What is PyQt?

- PyQt is a Python binding for the Qt framework

- You can do anything you do in Qt but... you write Python code 😊
How do I create GUIs for QGIS?

- There are two ways to create GUIs in QGIS:
  - Write from scratch
  - Use Qt Designer
How do I get started with PyQt?
How do I create a GUI?

- There are some tutorials out there, but the truly best way to get to know Qt is to use a software that comes with any QGIS installation: Qt Designer

- Yes: you can get started by omitting the `Py`
Qt Designer

- Qt Designer is a WYSIWYG editor and you generally do two things:
  - Drag and drop elements to the correct position
  - Adjust these elements
The **objectName**!

- Do you want to communicate with an element later on?
- For example:
  - What text was entered?
  - What value was selected in the dropdown?
  - Was the checkbox clicked?
  - Find out when the user clicks the button
**objectName**

- Any element you decide to communicate with later on should have an **objectName**
- It is assigned by default... but the naming is usually not very telling, so you should change it
I saved my file, and now?

- Qt Designer saves files with the ending .ui
- You can open these files in any text editor
- You will notice the content is pure XML:

```xml
<ui version="1.0" encoding="UTF-8">
    <class>Dialog</class>
    <widget class="QDialog" name="Dialog">
        <property name="geometry">
            <rect>
                <x>0</x>
                <y>0</y>
                <width>217</width>
                <height>198</height>
            </rect>
        </property>
        <property name="windowTitle">
            <string>Save Attributes</string>
        </property>
        <widget class="QPushButton" name="pushButton">
            <property name="geometry">
```
Compiling with pyuic4

- You can compile your .ui file to a .py file
- The OSGeo Shell includes the command pyuic4
pyuic4

pyuic4 originalFile.ui –o newFile.py
Example

`pyuic4 attributes.ui -o attributes.py`
Let’s look at the file

```python
from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName(_fromUtf8("Dialog"))
        Dialog.resize(217, 198)
        self.pushButton = QtGui.QPushButton(Dialog)
        self.pushButton.setGeometry(QtCore.QRect(40, 157, 111, 23))
        self.pushButton.setObjectName(_fromUtf8("pushButton"))
        self.lineEdit = QtGui.QLineEdit(Dialog)
        self.lineEdit.setGeometry(QtCore.QRect(70, 20, 121, 20))
        self.lineEdit.setObjectName(_fromUtf8("lineEdit"))
        self.comboBox = QtGui.QComboBox(Dialog)
        self.comboBox.setGeometry(QtCore.QRect(70, 70, 121, 22))
        self.comboBox.setObjectName(_fromUtf8("comboBox"))
```
Do not learn PyQt the hard way!

- Build your GUI
- Compile it
- Look at the Python commands
- Most of the things you want to know you can find out that way

```python
self.txt_NameInput.setObjectName(_fromUtf8("txt_NameInput"))
```
Find out how to read elements

- Reading text from a line edit:
  ```java
  ui.LineEditName.text()
  ```

- Reading selected item in dropdown (combo box):
  ```java
  ui.comboBoxName.currentText()
  ```

- Finding out if checkbox checked:
  ```java
  ui.radioButtonName.isChecked()
  ```
Find out how to set elements

- Reading text from a line edit:
  
  ```python
  ui.lineEditName.setText('Text')
  ```

- Reading selected item in dropdown (combo box):
  
  ```python
  ui.comboBoxName.setCurrentIndex(index)
  ```

- Check a checkbox:
  
  ```python
  ui.checkboxName.setChecked(True)
  ```
Before you get started with PyQt it is highly recommended that you look into object-oriented programming (OOP) and at least know its basics.

Technically you can use PyQt without OOP but you will hit its limits soon.

You need to know some OOP anyways to develop QGIS plugins.

So: go ahead and learn it.

It is not that hard but it can be daunting and confusing at first.
<table>
<thead>
<tr>
<th>User of</th>
<th>Do I need OOP?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyQt</td>
<td>No</td>
</tr>
<tr>
<td>Loading a GUI in QGIS</td>
<td>No</td>
</tr>
<tr>
<td>Loading a GUI in a QGIS plugin</td>
<td>Yes</td>
</tr>
</tbody>
</table>
So...

- If you simply want to play with PyQt and create user interfaces, you do not necessarily need to learn OOP.

- But if you want to work with QGIS plugins you need to learn it.
4 ways to load a UI in QGIS
How can I load this file in QGIS?

- There are a few ways to load the file you just created:
  - Attach it to a dataset in QGIS
  - Load it in the Python console
  - Load it in a plugin (compiled)
  - Load it in a plugin (uncompiled)

- Let’s look at all four of these ways!
1. Loading the UI from a dataset
2. Loading the UI in the console / editor

```python
import sys
sys.path.insert(0, r'D:\PythonTesting\Qt\gui')
from PyQt4.QtGui import QApplication, QDialog
from myGui import Ui_Dialog

window = QDialog()
ui = Ui_Dialog()
ui.setupUi(window)

window.show()
```
3. Loading the (compiled) GUI in a plugin

- There is an easier way... let’s skip to that one
- You can - **but you do not have to** – compile your .ui to a .py to display it in QGIS
- Recommendation: compile it to learn PyQt but load the uncompiled file (.ui)
4. Loading the (uncompiled) GUI in a plugin

```python
pluginPath = os.path.split(os.path.dirname(__file__))[0]
WIDGET, BASE = uic.loadUiType(os.path.join(pluginPath, 'ui', 'attributes.ui'))

try:
    _encoding = QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QApplication.translate(context, text, disambig)

curr_path = os.path.dirname(os.path.abspath(__file__))[0]

class attributes(BASE, WIDGET):
    def __init__(self, iface, parent=None):
        super(attributes, self).__init__(parent)
        self.setupUi(self)
```
Learn PyQt... soon!

https://learn.gis-trainer.com/
console.log('Thank you! :-)!');

numa.gremling@geosysnet.de

http://geosysnet.com
@geoSYSnet

https://gis-trainer.com
https://learn.gis-trainer.com
@gis_trainer
gistrainer