QGIS-Plugin-Programmierung

mit Python
unter Windows
für Einsteiger
(eigene Erfahrungen)

Uwe Brengelmann,
VuKV
Regionaldirektion Oldenburg-Cloppenburg
Katasteramt Wildeshausen

Agenda

(lateinisch für „das zu Tuende, was getan werden muss“) steht für: Tagesordnung

Sinnvolle Programme

Ui-Datei und resources.qrc kompilieren

Pythoncode: Stil

Pythoncode: Welche Dateien gehören zu einem plugin

Pythoncode: Aufbau der Haupt-py-Datei

Sinnvolle Programme:

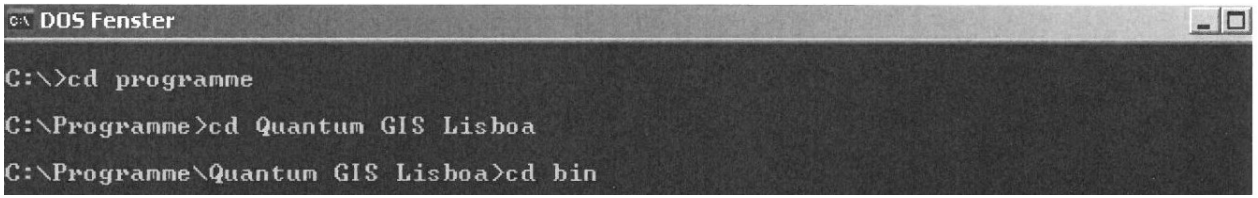
- Ein Programm zum Kreieren eines Icons; z.B. CorelDraw. Das Icon muss mit dem Programm auch als png- oder jpg-Datei exportierbar sein. Die Dateigröße der exportierten Rasterdatei des Icons sollte bei rund 5 kB liegen.
- QtDesigner-Portable zum Anlegen oder ergänzen eines Benutzermenüs (Schaltflächen, Dropdownlisten, Eingabefelder usw). Die Portable-Versionen erfordern keine Installation und können in jedem denkbaren Verzeichnis liegen.
- PyScripter-Portable zum Testen kleiner python Codeschnipsel, aber gößerer Importmodule. Das ist manchmal deutlich einfacher und schneller, wenn ganz neue python-Methoden ausprobiert werden sollen. Wenn die Funktionalität herausgearbeitet wurde, kann man die neue Methode dann auch in die eigentliche py-Datei einbauen.
- Die aktuelle LTR-Version von QGIS (LTR in der VuKV). Außerdem die plugins „Plugin-Builder“ und „Plugin-Reloader“. Mit dem Plugin-Builder kann ein neues Mini-Plugin erstellt werden, das dann nach und nach mit den anderen Hilfsmitteln erweitert wird. Mit dem Plugin-Reloader kann ein gerade erweitertes plugin in QGIS aktualisiert werden, ohne QGIS neu starten zu müssen.
- Notepad++ als Entwicklungsumgebung (IDE). Die py-Datei mit dem python-Code wird mit Notepad++ bearbeitet. Häufig können Codezeilen oder Blöcke durch copy and paste und Abänderung des kopierten Teiles ergänzt werden. Die Compare-Erweiterung, um das gleiche plugin in unterschiedlichen Versionen inhaltlich vergleichen zu können.

Ui-Datei und resources.qrc kompilieren:

- Mit dem plugin-Builder von QGIS ein neues leeres plugin anlegen, oder ein vorhandenes plugin in einen neuen Ordner kopieren.
- Ein neues icon mit Coreldraw kreieren, einen Namen vergeben (meist icon.png) und eventuell die Datei „resources.qrc“ mit Notepad++ anpassen.
- „resources.qrc“ nach „resources.py“ kompilieren. Dazu müssen folgende Dateien in einem Verzeichnis stehen, z.B. unter D:/

icon.png	30.11.2015 16:42	IrfanView PNG File	3 KB
pyrcc4.exe	11.08.2013 02:36	Anwendung	43 KB
QtCore4.dll	16.02.2014 22:04	Anwendungserweiteru...	2.900 KB
QtXml4.dll	09.02.2014 18:59	Anwendungserweiteru...	412 KB
resources.qrc	01.12.2015 07:15	QRC-Datei	1 KB
zlib1.dll	31.07.2013 21:27	Anwendungserweiteru...	76 KB

- Dann folgendes in ein DOS-Fenster eingeben: pyrcc4 –o resources.py resources.qrc



- Mit dem Programm „QtDesigner-Portable.exe“ ein neues Menüfenster kreieren und als ui-Datei abspeichern.

- Ui-Datei kompilieren; folgendes in die Pythonkonsole von QGIS eingeben:

  - from PyQt4 import uic

  - import os

  - uic.compileUiDir(os.path.dirname(r‘C:\Program Files (x86)\VKV\QGIS_plugins\GebMan\ui_gebman.ui‘))

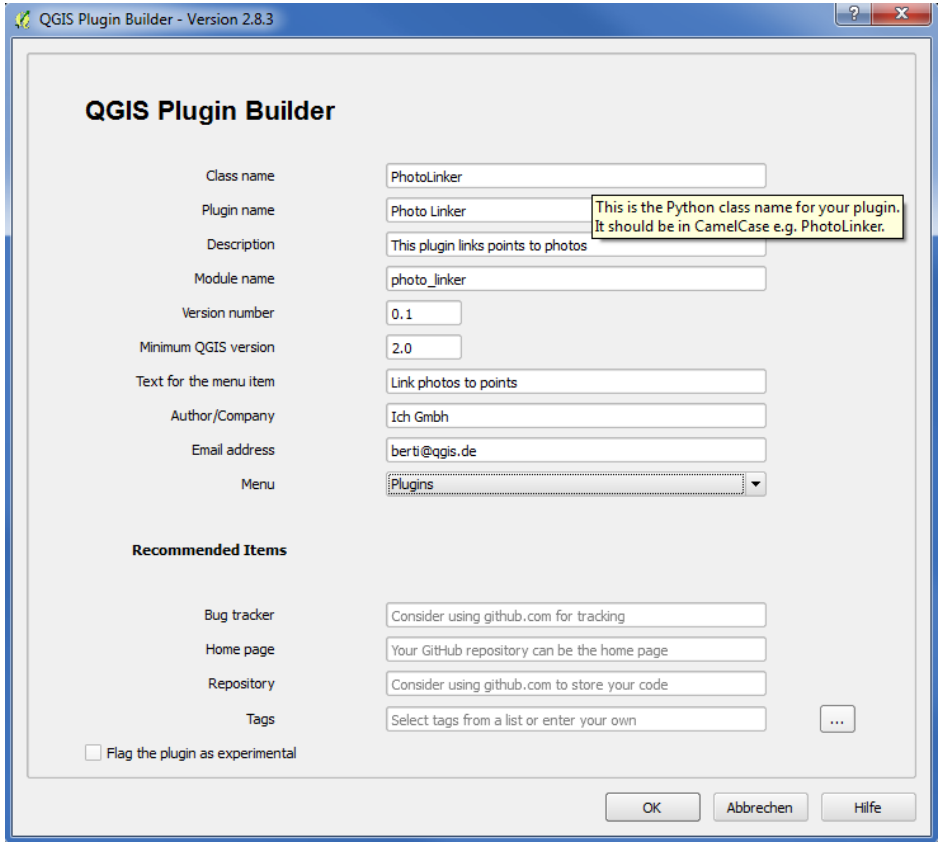
  Dabei wird die obige ui-Datei eingelesen und die py-Datei gleichen Namens angelegt.
  Anmerkung:
  QGIS selbst erzeugt aus den py-Dateien aus Performancegründen eine binäre pyc-Datei. Immer wenn an einer py-Datei etwas geändert wurde, dann wird beim nächsten Start von QGIS auch die pyc-Datei neu kompiliert.

Python: Code-Stil:

- Benutze als Einzug vier Leerstellen, keine Tabulatoren! Vier Leerstellen sind ein guter Kompromiss. Tabulatoren bringen Konfusion; bitte nicht benutzen.
- Begrenze die Zeilenlänge, so dass sie 170 Zeichen (~ 21 Zoll-Bildschirm) nicht überschreiten. Das vermeidet ständiges links- und rechts-Scrollen am Bildschirm.
- Benutze vier Leerzeilen, um Funktionen und Klassen voneinander zu trennen; auch bei längeren Blöcken innerhalb von Funktionen führen ein oder zwei Leerzeilen zu mehr Übersicht
- Wenn möglich, plaziere Kommentare auf die gleiche Linie. Bei längeren Kommentaren plaziere ihn vor dem Code mit einem Doppelpunkt.
- Setze Leerzeichen um Operatoren und nach Kommas; aber nicht direkt innerhalb von Klammerkonstrukten: *a = f(1, 2) + g(3, 4)*
- Benenne deine Klassen und Funktionen konsequent. Die Konvention ist,
  - *CamelCase* für Klassen zu benutzen, und
  - *lower_case_mit_unterstrichen* für Funktionen und Methoden.
- eigene Funktionen beginnen immer mit *"f_"* und stehen am Anfang der Klasse,
  eigene Methoden beginnen immer mit "m_" .
- Benutze immer "*self"* als Namen für das erste Argument einer Methode; nicht bei Funktionen.
- Benutze das ASCII-encoding; das ist international. Zum Schreiben von Daten in eine shape-Datei (=dbf) immer nur ASCII-Zeichen verwenden.

Pythoncode – Welche Dateien gehören zu einem plugin:

→ plugin-Builder: erzeugt ein neues plugin



Plugin Builder Results

Plugin Builder Results

Congratulations! You just built a plugin for QGIS!

Your plugin **PhotoLinker** was created in:
C:\Program Files (x86)\VKV\QGIS_plugins\PhotoLinker

Your QGIS plugin directory is located at:
C:/Users/uwe.brengelmann/.qgis2/python/plugins

What's Next

1. In your plugin directory, compile the resources file using pyrcc4 (simply run **make** if you have automake or use **pb_tool**)
2. Test the generated sources using **make test** (or run tests from your IDE)
3. Copy the entire directory containing your new plugin to the QGIS plugin directory (see Notes below)
4. Test the plugin by enabling it in the QGIS plugin manager
5. Customize it by editing the implementation file **photo_linker.py**
6. Create your own custom icon, replacing the default **icon.png**
7. Modify your user interface by opening **photo_linker_dialog_base.ui** in Qt Designer

Notes:

- You can use the **Makefile** to compile and deploy when you make changes. This requires GNU make (gmake). The Makefile is ready to use, however you will have to edit it to add addional Python source files, dialogs, and translations.
- You can also use **pb_tool** to compile and deploy your plugin. Tweak the *pb_tool.cfg* file included with your plugin as you add files. Install **pb_tool** using *pip* or *easy_install*. See http://loc8.cc/pb_tool for more information.

For information on writing PyQGIS code, see http://loc8.cc/pyqgis_resources for a list of resources.

- Oct 9, 2014
- The Plugin Builder is a great tool for generating a working plugin project.
- One of the main tasks in the development cycle is deploying the plugin to the QGIS plugin directory for testing. Plugin Builder comes with a Makefile that can be used on Linux and OS X to aid in development. Depending on your configuration, the Makefile may work on Windows → Bei mir nicht!

Mit plugin-Builder erzeugte Dateien:

Computer ▸ (C:) OSIM_VOL ▸ Programme (x86) ▸ VKV ▸ QGIS_plugins ▸ PhotoLinker ▸

In Bibliothek aufnehmen ▾ Freigeben für ▾ Brennen Neuer Ordner

Name	Änderungsdatum	Typ	Größe
help	12.09.2016 10:32	Dateiordner	
i18n	07.04.2016 13:35	Dateiordner	
scripts	07.04.2016 13:35	Dateiordner	
test	12.09.2016 10:32	Dateiordner	
__init__.py	12.09.2016 10:32	PY-Datei	2 KB
icon.png	17.03.2015 15:48	IrfanView PNG File	2 KB
Makefile	12.09.2016 10:32	Datei	8 KB
metadata.txt	12.09.2016 10:32	TXT-Datei	1 KB
pb_tool.cfg	12.09.2016 10:32	CFG-Datei	3 KB
photo_linker.py	12.09.2016 10:32	PY-Datei	7 KB
photo_linker_dialog.py	12.09.2016 10:32	PY-Datei	2 KB
photo_linker_dialog_base.ui	12.09.2016 10:32	UI-Datei	2 KB
plugin_upload.py	17.03.2015 15:48	PY-Datei	4 KB
pylintrc	17.03.2015 15:48	Datei	9 KB
README.html	12.09.2016 10:32	HTML-Dokument	2 KB
README.txt	12.09.2016 10:32	TXT-Datei	1 KB
resources.qrc	12.09.2016 10:32	QRC-Datei	1 KB

Beispielplugin:

Computer ▸ (C:) OSIM_VOL ▸ Programme (x86) ▸ VKV ▸ QGIS_plugins ▸ GebMan ▸

In Bibliothek aufnehmen ▾ Freigeben für ▾ Diashow Brennen Neuer Ordner

Name	Änderungsdatum	Typ	Größe
fuer_QGIS	17.05.2016 13:42	Dateiordner	
__init__.py	16.08.2016 10:17	PY-Datei	2 KB
__init__.pyc	16.08.2016 10:17	Compiled Python ...	3 KB
gebman.py	09.09.2016 11:09	PY-Datei	265 KB
gebman.pyc	09.09.2016 13:08	Compiled Python ...	140 KB
Gebman_Handbuch.pdf	15.08.2016 15:31	Adobe Acrobat D...	3.311 KB
gebmandialog.py	29.01.2016 13:19	PY-Datei	2 KB
gebmandialog.pyc	19.04.2016 16:05	Compiled Python ...	3 KB
help.png	19.12.2011 10:34	IrfanView PNG File	2 KB
icon.cdr	12.07.2016 09:26	CorelDRAW X4 Gr...	22 KB
icon.png	07.06.2013 12:39	IrfanView PNG File	2 KB
metadata.txt	19.04.2016 12:40	TXT-Datei	1 KB
resources.py	07.06.2013 14:10	PY-Datei	8 KB
resources.pyc	12.07.2016 09:59	Compiled Python ...	3 KB
resources.qrc	19.04.2016 12:37	QRC-Datei	1 KB
ui_gebman.py	26.08.2016 11:22	PY-Datei	115 KB
ui_gebman.pyc	26.08.2016 11:22	Compiled Python ...	61 KB
ui_gebman.ui	26.08.2016 11:21	UI-Datei	102 KB
ui_gebman_gross.py	26.08.2016 11:22	PY-Datei	123 KB
ui_gebman_gross.pyc	24.08.2016 10:48	Compiled Python ...	64 KB
ui_gebman_gross.ui	26.08.2016 11:21	UI-Datei	110 KB

py beinhaltet den Python source code.

pyc beinhaltet den kompilierten bytecode des Python source code.

Pythoncode – Aufbau der Haupt-py-Datei:

- Anmerkungen zu Python selbst:
  - www: “Python ist einfach” ? (Möglichkeit des Imports externer Module).
  - Installation Eclipse sinnvoll ? (eher nein!)
  - Python Lehrgänge (z.B. VHS Heidelberg)
- photo_linker.py (vom plugin-Builder)
- gebman.py (eigenes plugin)
- QT-Designer-Portable (Beispiel: Zwei Seiten weiter)
- PyScripter-Portable (Beispiel: Drei Seiten weiter)

Zusammenspiel der Komponenten:

Python-Code der py-Datei verändert.

↓

Plugin-Reloader in QGIS klicken.

Menüoberfläche (ui-Datei) verändert.

↓

Bei neuen Menüpunkten den Slot in der py-Datei definiert.

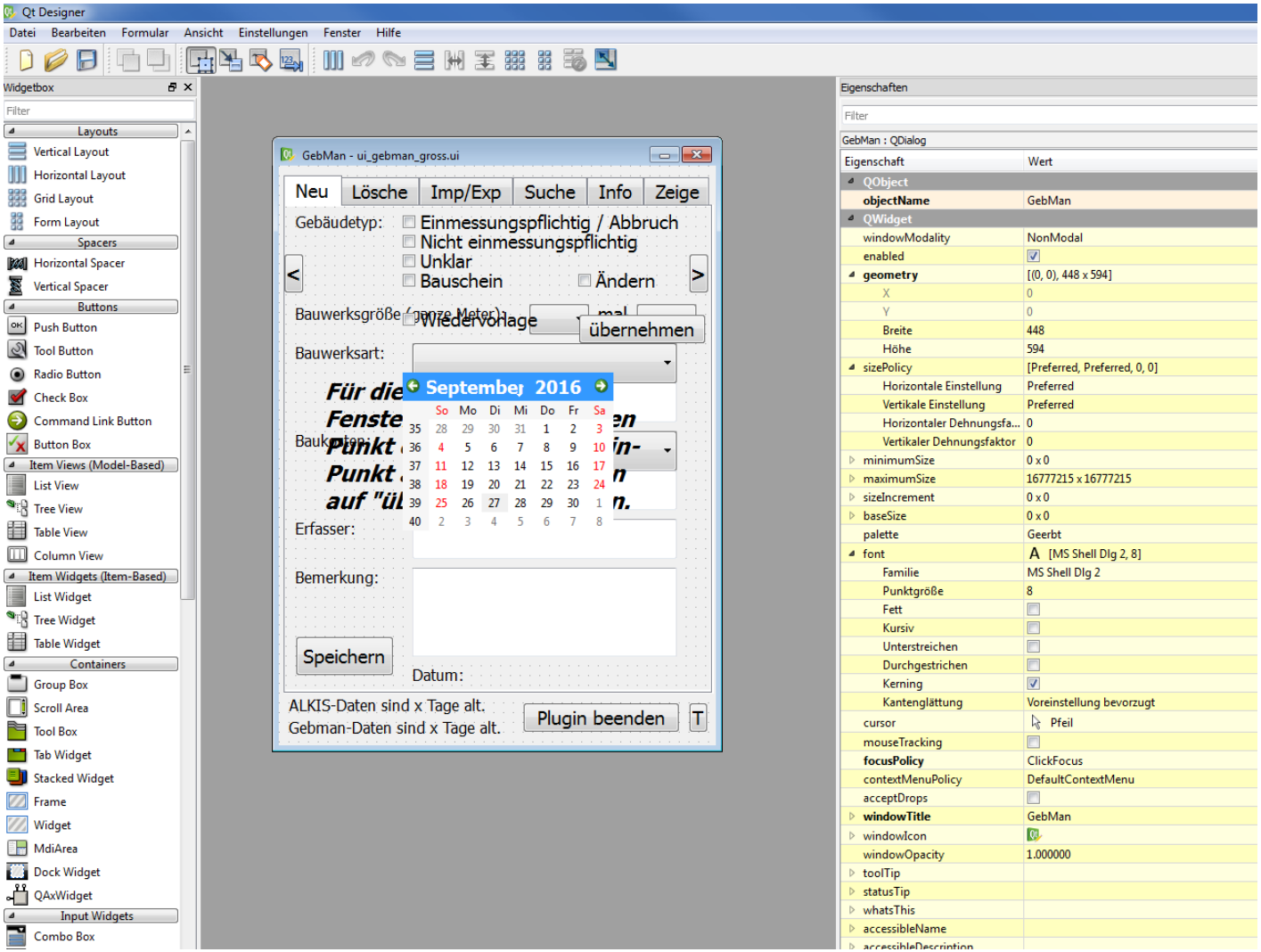
↓

ui-Datei in der Pythonkonsole von QGIS kompilieren.

↓

Plugin-Reloader in QGIS klicken.

QT-Designer: Beispiel



Py-Scripter: Beispiel

PyScripter - Q:\Dezernat-2\WDH\Marketing\GebMan\PortablePython2751_Daten\Rechnen_mit_Datum.py

File Edit Search View Project Run Tools Help

Find:

File Explorer

- Computer
  - (C:) OSIM_VOL
  - (D:) Lokaler Datenträger
  - (E:) DVD-Laufwerk
  - (F:) DVD-RW-Laufwerk
  - (H:) brengelmann.uwe (\\GLL-WDH-FS61\home)
  - (Q:) RD-OL (\\gll.ads.niedersachsen.de)

File Explorer | Project Explorer | Code Explorer

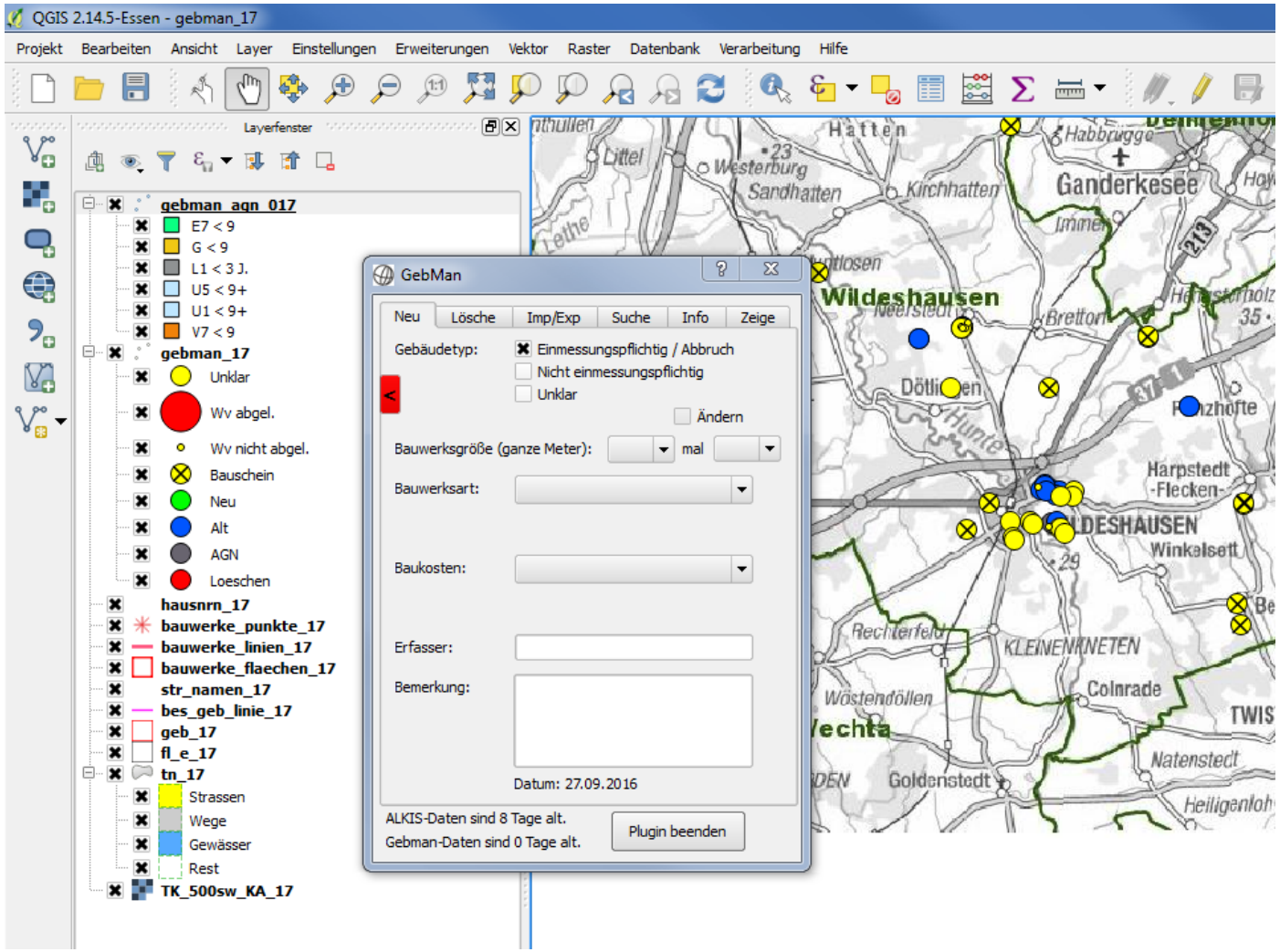
```python
# -*- coding: utf-8 -*-
#-------------------------------------------------------------------------
# Name:         module1
# Author:       uwe.brengelmann
# Erstellt:     03.12.2014
# Copyright:    (c) uwe.brengelmann 2014
# Licence:      <GNU-GPL>
#-------------------------------------------------------------------------

import os
import time
from time import *


datum1 = '2015-10-28'
tupDatum1 = [int(s) for s in datum1.split("-")]
tupDatum1.extend([0, 0, 0, 0, 0, 0])
print tupDatum1

datum2 = '15.01.2016'
tupDatum2 = [int(s) for s in datum2.split(".")]
tupDatum2.reverse()
tupDatum2.extend([0, 0, 0, 0, 0, 0])
print tupDatum2

damals = mktime(tupDatum1)
print damals # [sec]
heute = mktime(tupDatum2)
print heute

# Tagesdifferenz berechnen:
print "Tagesdifferenz:", int((heute - damals) /86400) # 60 * 60 * 24 = 86400


import datetime
print str(datetime.datetime.now())
print 'Millisekunden:', str(datetime.datetime.now())[11:].replace(".", ":").split(":")[3]
```

Rechnen_mit_Datum.py

Python Interpreter

```
*** Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32. ***
*** Remote Python engine  is active ***
>>>
*** Remote Interpreter Reinitialized  ***
>>>
[2015, 10, 28, 0, 0, 0, 0, 0, 0]
[2016, 1, 15, 0, 0, 0, 0, 0, 0]
1445986800.0
1452812400.0
Tagesdifferenz: 79
2016-09-27 07:59:53.377000
Millisekunden: 377000
>>>
```

Plugin-Oberfläche in QGIS: Beispiel



Feedback please:

interessant, aber im Moment nicht mein Thema?
ähnliche Fragen stelle ich mir auch gerade?
alles alter Kram?