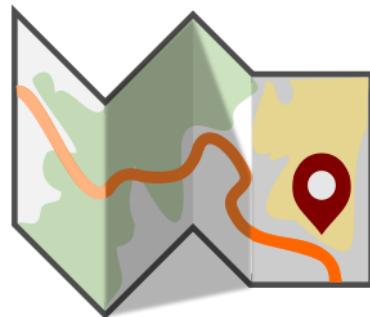


Building a Python Plugin

QGIS Tutorials and Tips



Author

Ujaval Gandhi

<http://google.com/+UjavalGandhi>

Translations by

Dick Groskamp

Een plug-in in Python bouwen

Plug-ins zijn een fantastische manier om de functionaliteit van QGIS uit te breiden. U kunt plug-ins schrijven met behulp van Python die kunnen variëren van een eenvoudige knop tot gesofisticeerde gereedschappen. Deze handleiding zal een overzicht van het proces geven voor het instellen van uw ontwikkelomgeving, de interface voor een plug-in ontwerpen en code schrijven voor interactie met QGIS. Bekijk de handleiding [Beginnen met programmeren in Python](#) om bekend te raken met de basisbeginselen.

Overzicht van de taak

We zullen een eenvoudige plug-in ontwikkelen, genaamd **Attributen opslaan** die gebruikers in staat zet een vectorlaag uit te zoeken en de attributen daarvan weg te schrijven naar een CSV-bestand.

De gereedschappen ophalen

Qt Creator

[Qt](#) is een framework voor softwareontwikkeling dat is gebruikt om toepassingen te ontwikkelen die kunnen worden uitgevoerd op Windows, Mac, Linux als ook op verschillende mobiele besturingssystemen. QGIS zelf is geschreven met behulp van het framework Qt. Voor de ontwikkeling van plug-ins zullen we een toepassing gebruiken, genaamd [Qt Creator](#) om de interface voor onze plug-in te ontwerpen.

Download en installeer de software van Qt Creator vanaf [SourceForge](#)

Python Bindings voor Qt

Omdat we de plug-in ontwikkelen in Python, moeten we de Python bindings voor Qt installeren. De methode voor het installeren hiervan is afhankelijk van het platform dat u gebruikt. Voor het bouwen van plug-ins hebben het gereedschap voor de opdrachtregel `pyrcc4` nodig.

Windows

Download de [OSGeo4W network installer](#) en kies Express Desktop Install. Installeer het pakket `QGIS`. Na die installatie zult u in staat zijn toegang te verkrijgen tot het gereedschap `pyrcc4` via de OSGeo4W Shell.

Mac

Installeer de [Homebrew](#) pakketbeheerder. Installeer het pakket `PyQt` door de volgende opdracht uit te voeren:

```
brew install PyQt
```

Linux

Afhankelijk van uw distributie, zoek en installeer het pakket `python-qt4`. Op Ubuntu en Debian-gebaseerde distributies kunt u de volgende opdracht uitvoeren:

```
sudo apt-get install python-qt4
```

Een tekstbewerker of een Python IDE

Elke soort softwareontwikkeling vereist een goede tekstbewerker. Als u al een favoriete tekstbewerker of een IDE (Integrated Development Environment) heeft, zou u die kunnen gebruiken voor deze handleiding. Anders biedt elk platform een grote variëteit van gratis of betaalde opties voor tekstbewerkers. Kies er een die aan uw wensen voldoet.

Deze handleiding gebruikt de bewerker Notepad++ op Windows.

Windows

[Notepad++](#) is een goede gratis tekstbewerker voor Windows. Download en installeer de tekstbewerker [Notepad++](#).

Note

Indien u Notepad++ gebruikt, zorg er dan voor dat Omzetten in spaties is geselecteerd in Instellingen › Voorkeuren › Tabs. Python is bijzonder gevoelig voor witruimte en deze instelling zal er voor zorgen dat tabs en spaties op de juiste manier worden behandeld.

plug-in Plugin Builder

Er is een nuttige plug-in voor QGIS, genaamd [Plugin Builder](#) die alle noodzakelijke bestanden en onderliggende code voor een plug-in maakt. Zoek en installeer de plug-in [Plugin Builder](#). Bekijk [Plug-ins gebruiken](#) voor meer details over hoe plug-ins te installeren.

plug-in Plugins Reloader

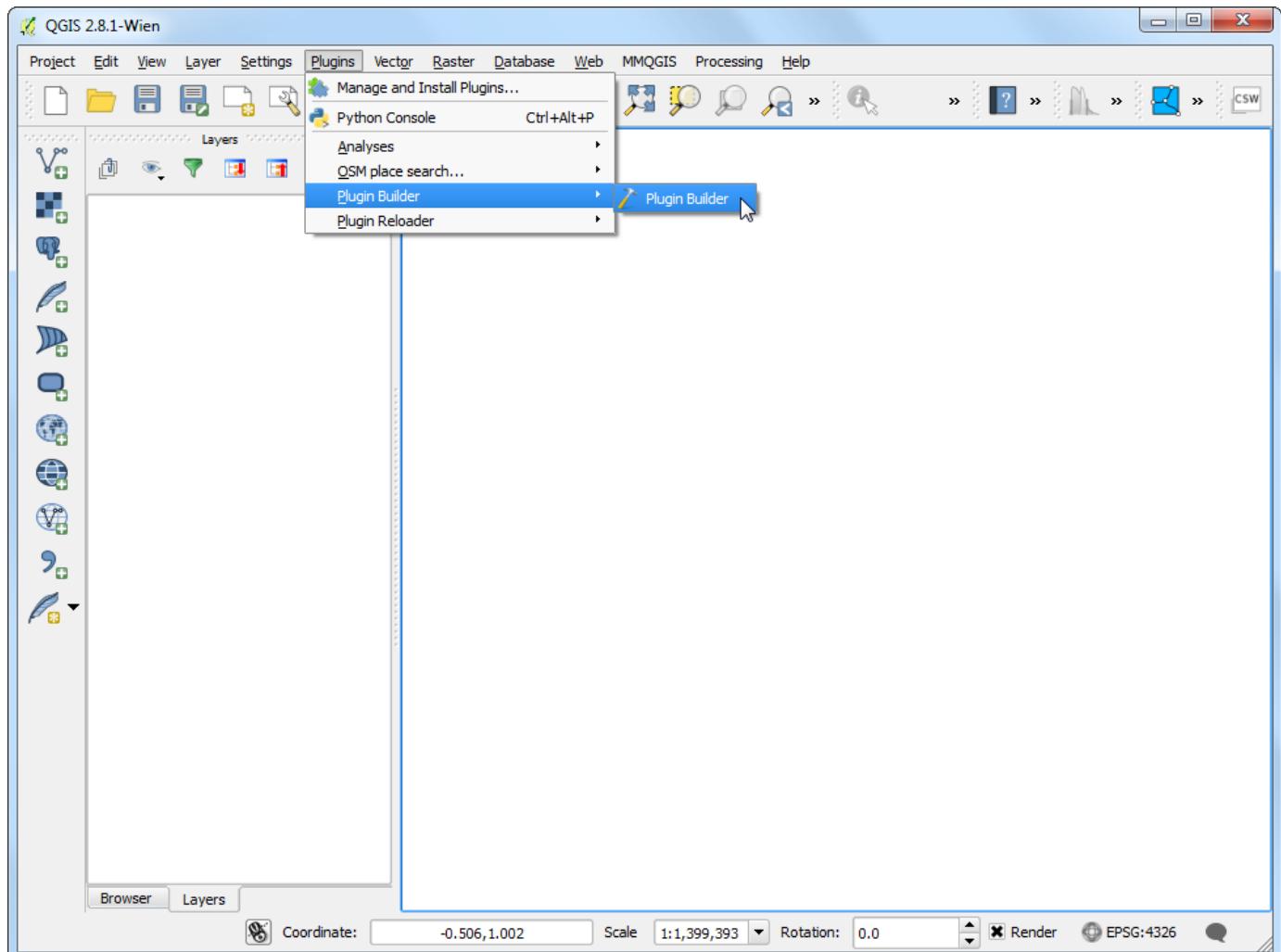
Dit is een andere nuttige hulpplug-in die het iteratief ontwikkelen van plug-ins mogelijk maakt. Met behulp van deze plug-in kunt u de code voor uw plug-in wijzigen en die hebben gereflecteerd in QGIS zonder dat u QGIS elke keer opnieuw moet starten. Zoek en installeer de plug-in [Plugin Reloader](#). Bekijk [Plug-ins gebruiken](#) voor meer details over hoe plug-ins te installeren.

Note

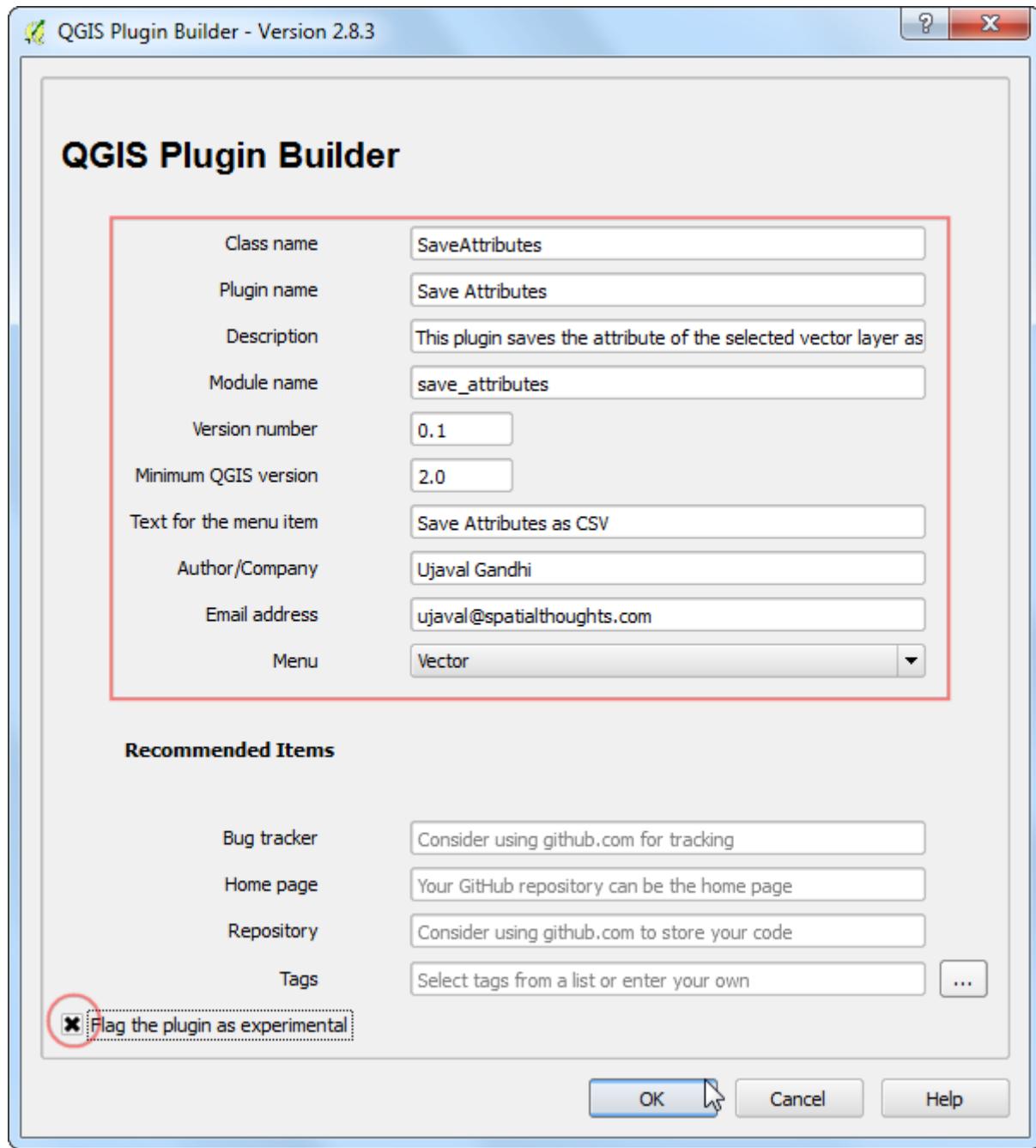
Plugin Reloader is een experimentele plug-in. Zorg er voor dat u ook Ook de experimentele plugins tonen in de instellingen van Plug-ins installeren en beheren te selecteren, als u hem niet kunt vinden.

Procedure

1. Open QGIS. Ga naar Plug-ins › Plugin Builder › Plugin Builder.



2. U zult het dialoogvenster zien van QGIS Plugin Builder met een formulier. U kunt het formulier vullen met details met betrekking tot onze plug-in. De Class name zal de naam zijn van de Python Class die de logica van de plug-in bevat. Dit al ook de naam zijn van de map die alle bestanden van de plug-in bevat. Voer **SaveAttributes** in als de naam van de klasse. De Plugin name is de naam waarmee uw plug-in zal verschijnen in Plug-ins beheren en installeren. Voer de naam in als **Attributen opslaan**. Voeg een beschrijving toe in het veld Description. De Module name zal de naam zijn van het belangrijkste Python-bestand voor de plug-in. Voer die in als **save_attributes**. Laat de versienummers zoals zij zijn. De waarde Text for menu item zal zijn hoe de gebruikers uw plug-in zullen vinden in het menu van QGIS. Voer het in als **Attributen opslaan als csv**. Voer uw naam en e-mailadres in in de toepasselijke velden. Het veld Menu bepaalt waar het item voor uw plug-in wordt toegevoegd in QGIS. Selecteer, omdat onze plug-in bestemd is voor vectorgegevens, **Vector**. Selecteer het vak Flag the plugin as experimental onderin. Klik op OK.



3. Vervolgens zult u worden gevraagd om een map te kiezen voor uw plug-in. U moet naar de map voor de QGIS Python plug-ins bladeren op uw computer en Map selecteren selecteren. Normaal gesproken is een map `.qgis2/` aanwezig in uw eigen gebruikersmap. De locatie van de map `plugin` is als volgt afhankelijk van uw platform: (Vervang `username` door uw gebruikersnaam)

Windows

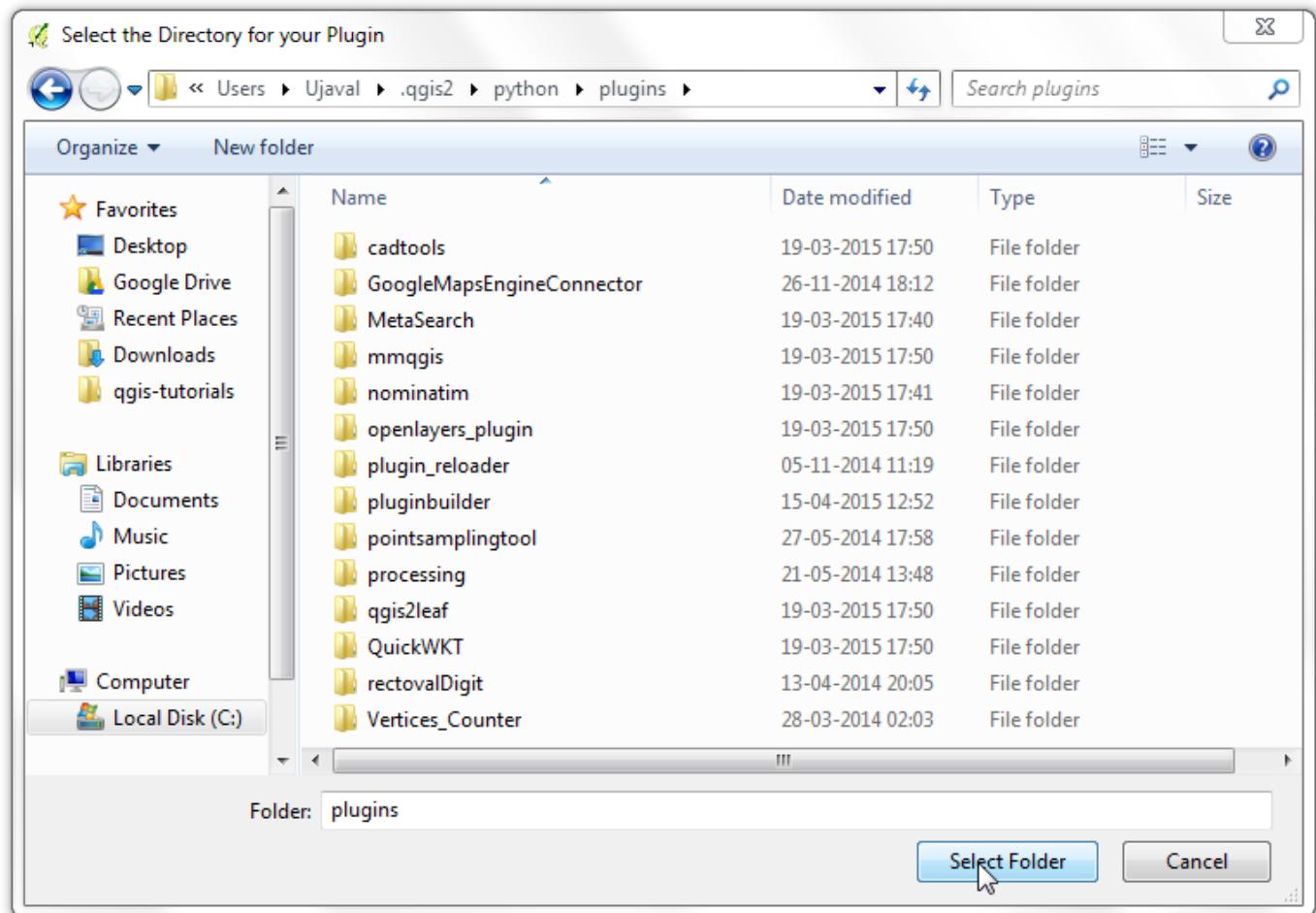
```
c:\Users\username\.qgis2\python\plugins
```

Mac

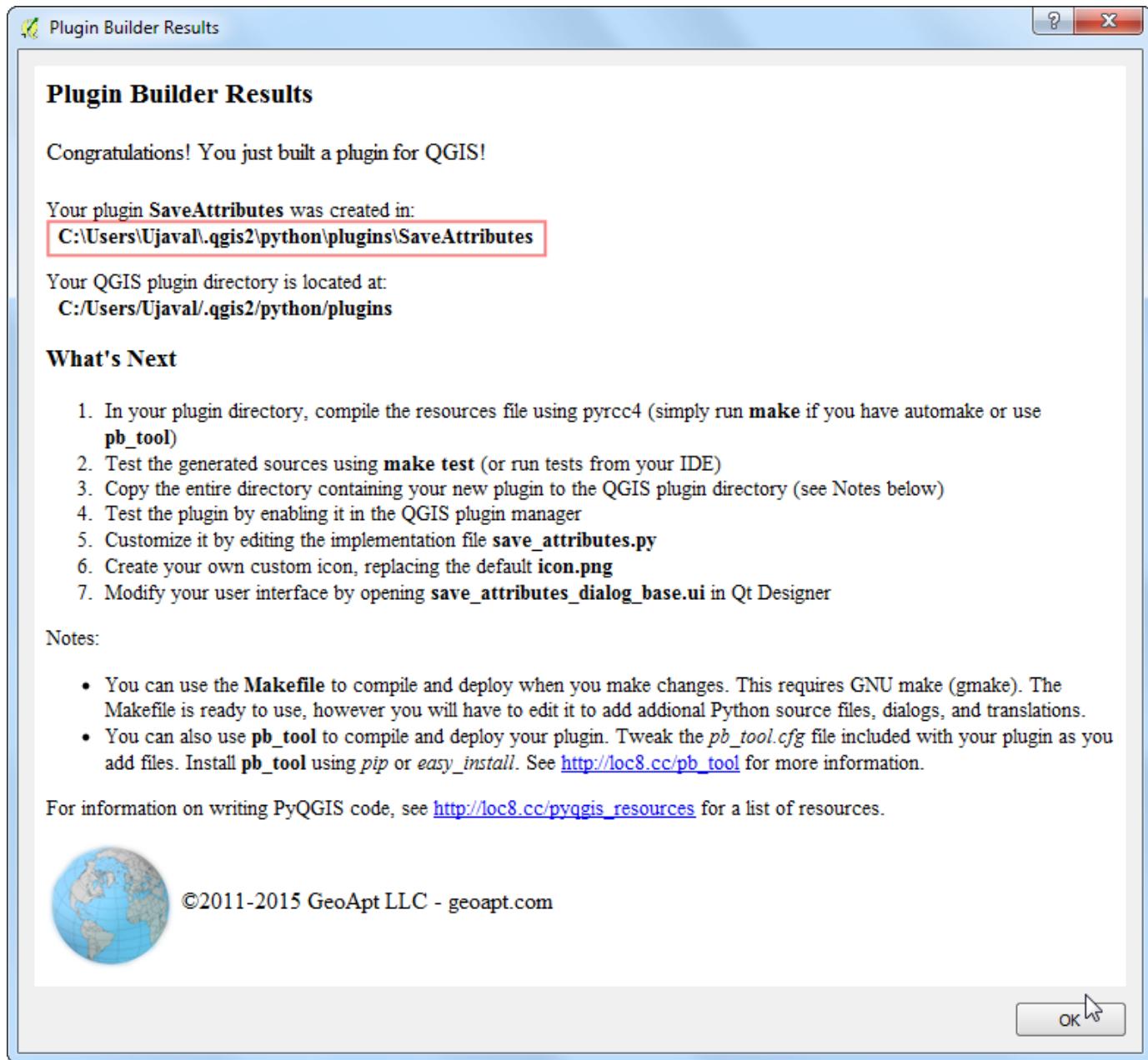
```
/Users/username/.qgis2/python/plugins
```

Linux

/home/username/.qgis2/python/plugins



4. U zult een dialoogvenster ter bevestiging zien als de sjabloon voor uw plug-in eenmaal is gemaakt. Onthoud het pad naar de map voor de plug-in.



5. Vóór we onze nieuw gemaakte plug-in kunnen gebruiken, moeten we het bestand **resources.qrc** , dat werd gemaakt door de Plugin Builder, compileren. Start de OSGeo4W Shell op Windows of een terminal op Mac of Linux.

```
OSGeo4W Shell
nircmdc      xmlpatterns
ogdi_import  xmlpatternsvalidator
ogdi_info    xmlwf
ogr2ogr      xxmklink

epsg_tr      gdal_fillnodata  ps2pdf12
esri2wkt    gdal_merge      ps2pdf13
gcps2vec    gdal_polygonize ps2pdf14
gcps2wld    gdal_proximity  ps2pdfxx
gdal2tiles   gdal_retile    pyuic4
gdal2xyz    gdal_sieve     qgis-browser
gdalchksum  grass64        qgis
gdalcompare gssetgs        rgb2pct
gdalident   make-bat-for-py setup-test
gdalimport  mkgraticule   setup
gdalmove    o-help          udig
gdal_auth   o4w_env
gdal_calc   pct2rgb
gdal_edit   ps2pdf

GDAL 1.11.2, released 2015/02/10
C:>_
```

6. Blader naar de map voor de plug-in waar de uitvoer van de *Plugin Builder* werd gemaakt. U kunt de opdracht *cd*, gevolgd door het pad naar de map, gebruiken.

```
cd c:\Users\username\.qgis2\python\plugins\SaveAttributes
```

```
OSGeo4W Shell
ogdi_info      xmlwf
ogr2ogr        xxmklink

epsg_tr        gdal_fillnodata  ps2pdf12
esri2wkt      gdal_merge      ps2pdf13
gcps2vec      gdal_polygonize ps2pdf14
gcps2wld      gdal_proximity  ps2pdfxx
gdal2tiles     gdal_retile    pyuic4
gdal2xyz      gdal_sieve     qgis-browser
gdalchksum    grass64        qgis
gdalcompare   gssetgs        rgb2pct
gdalident    make-bat-for-py setup-test
gdalimport   mkgraticule   setup
gdalmove     o-help          udig
gdal_auth    o4w_env
gdal_calc    pct2rgb
gdal_edit    ps2pdf

GDAL 1.11.2, released 2015/02/10
C:>cd Users\Ujaval\.qgis2\python\plugins\SaveAttributes
C:\Users\Ujaval\.qgis2\python\plugins\SaveAttributes>_
```

7. Als u eenmaal in de map bent, typ *make*. Dat zal de opdracht *pyrcc4* uitvoeren die we hadden geïnstalleerd als deel van de Qt bindings voor Python.

```
make
```

OSGeo4W Shell

```

epsg_tr          gdal_fillnodata  ps2pdf12
esri2wkt        gdal_merge       ps2pdf13
gcps2vec        gdal_polygonize ps2pdf14
gcps2wld        gdal_proximity  ps2pdfxx
gdal2tiles      gdal_retile     pyuic4
gdal2xyz        gdal_sieve      qgis-browser
gdalchksum      grass64        qgis
gdalcompare     gssetgs        rgb2pct
gdalident       make-bat-for-py setup-test
gdalimport      mkgraticule   setup
gdalmove        o-help         udig
gdal_auth       o4w_env        pct2rgb
gdal_calc        ps2pdf

```

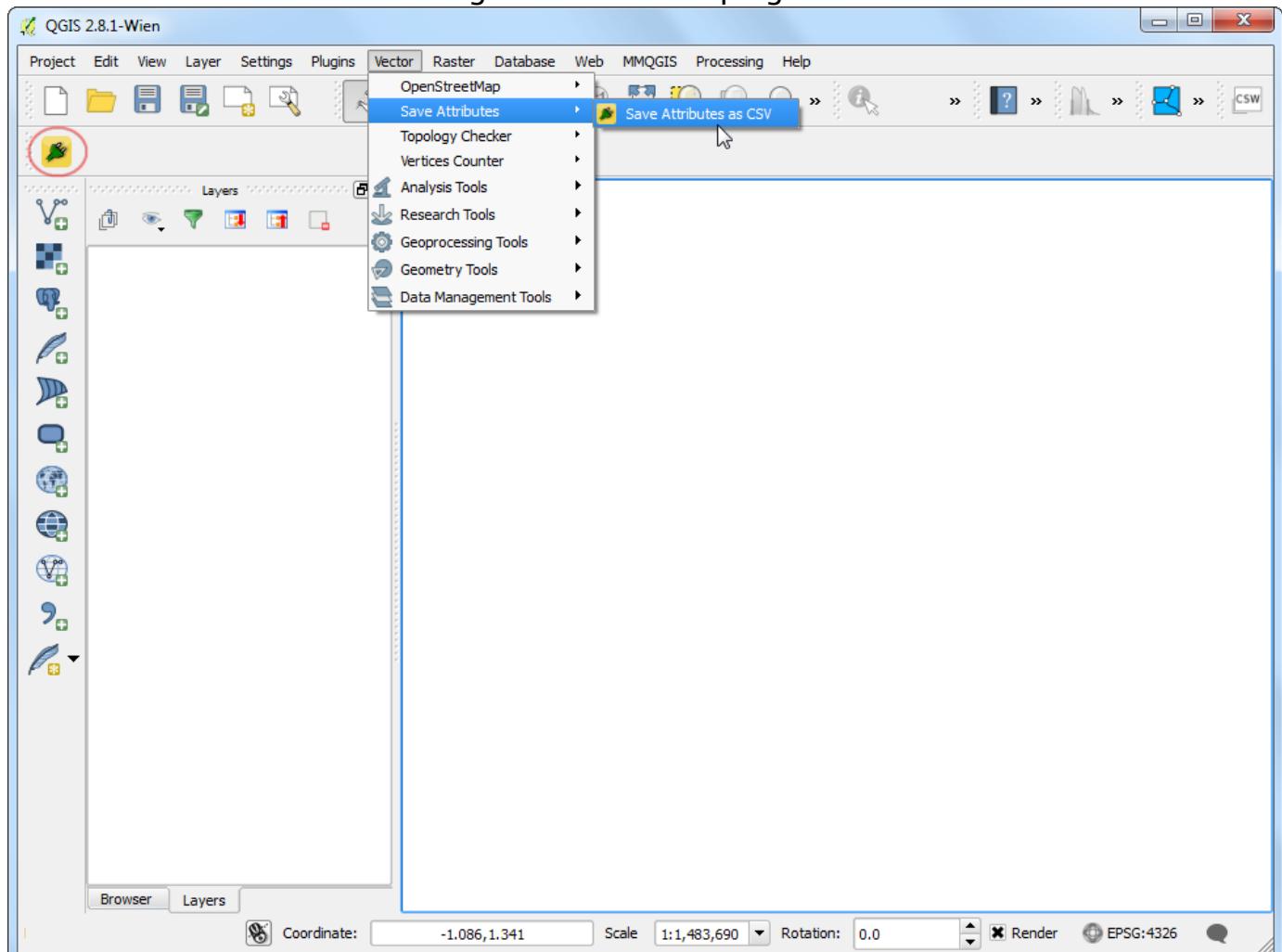
GDAL 1.11.2, released 2015/02/10

C:\>cd Users\Ujaval\.qgis2\python\plugins\SaveAttributes

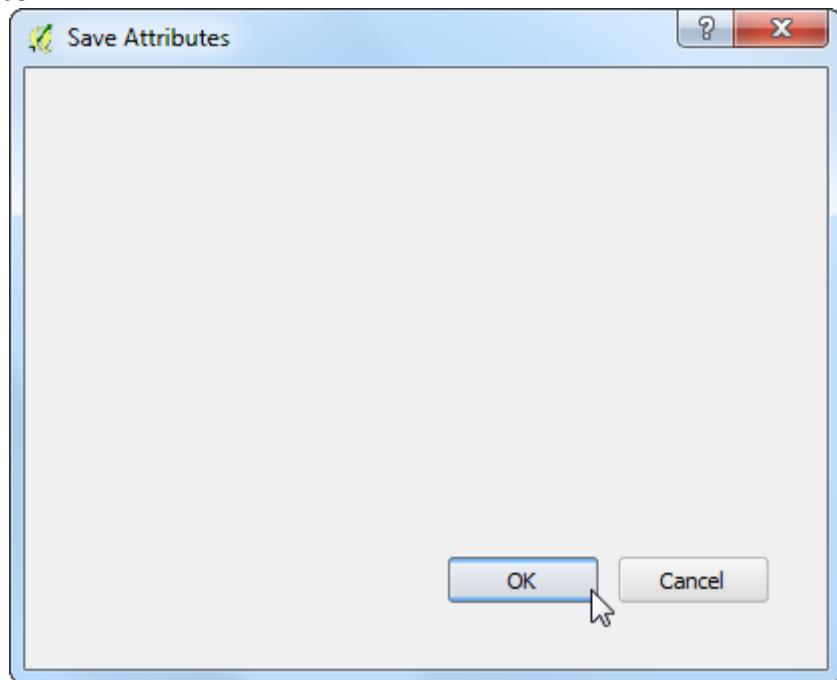
C:\Users\Ujaval\.qgis2\python\plugins\SaveAttributes>make
pyrcc4 -o resources_rc.py resources.qrc

C:\Users\Ujaval\.qgis2\python\plugins\SaveAttributes>_

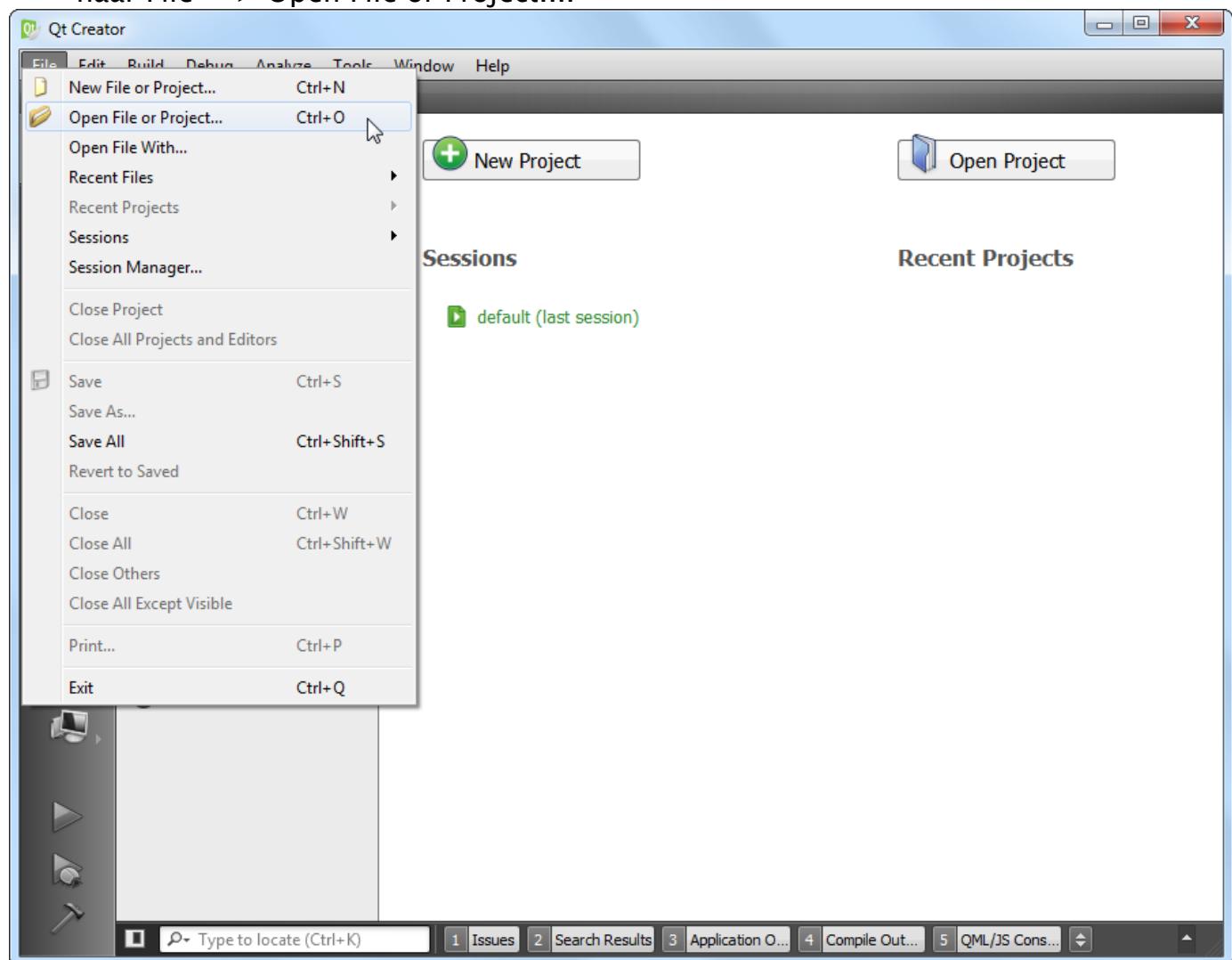
8. Nu zijn we klaar om een eerste blik te werpen op de brandnieuwe plug-in die we hebben gemaakt. Sluit QGIS en start het opnieuw. Ga naar Plug-ins > Plug-ins beheren en installeren en schakel de plug-in **Attributen opslaan** in op de tab Geïnstalleerd. U zult zien dat er een nieuw pictogram op de werkbalk staat en een nieuw menuitem onder Vector > Attributen opslaan > Attributen opslaan als CSV` . Selecteer die om het dialoogvenster voor de plug-in te starten.



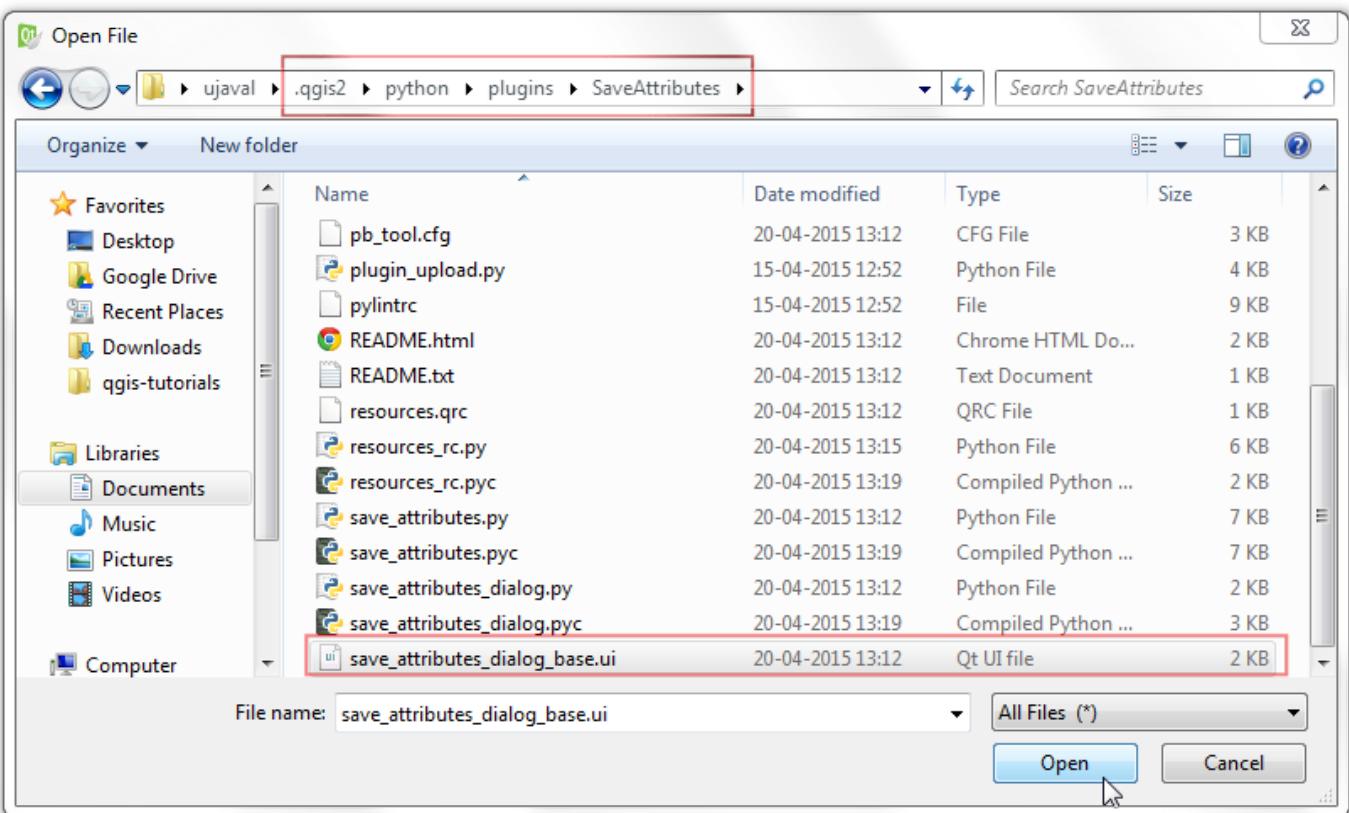
9. U zult een nieuw blanco dialoogvenster zien, genaamd **Attributen opslaan**. Sluit dit dialoogvenster.



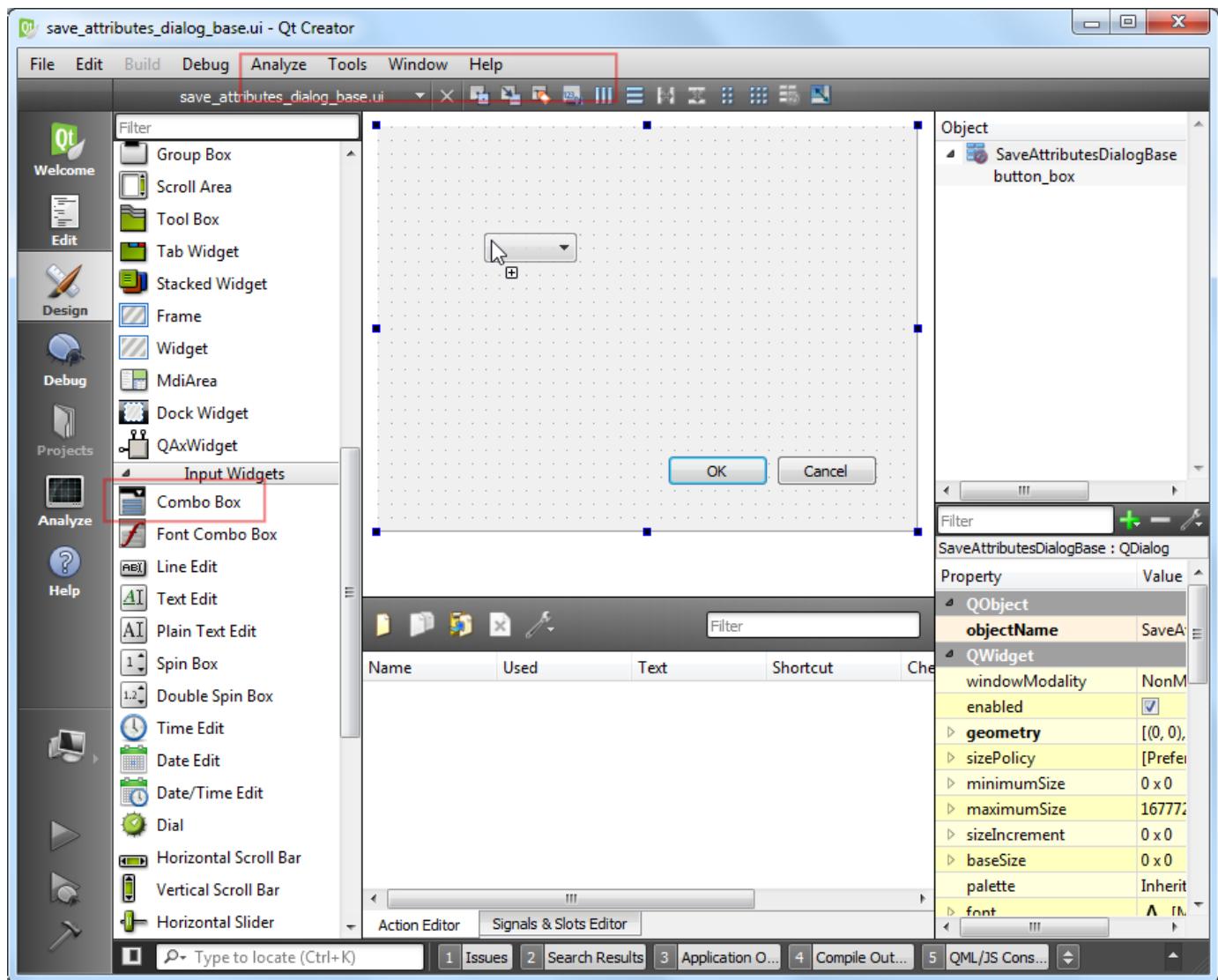
10. We zullen nu ons nieuw dialoogvenster ontwerpen en enkele elementen voor de gebruikersinterface er aan toevoegen. Open het programma **Qt Creator** en ga naar **File --> Open File or Project....**



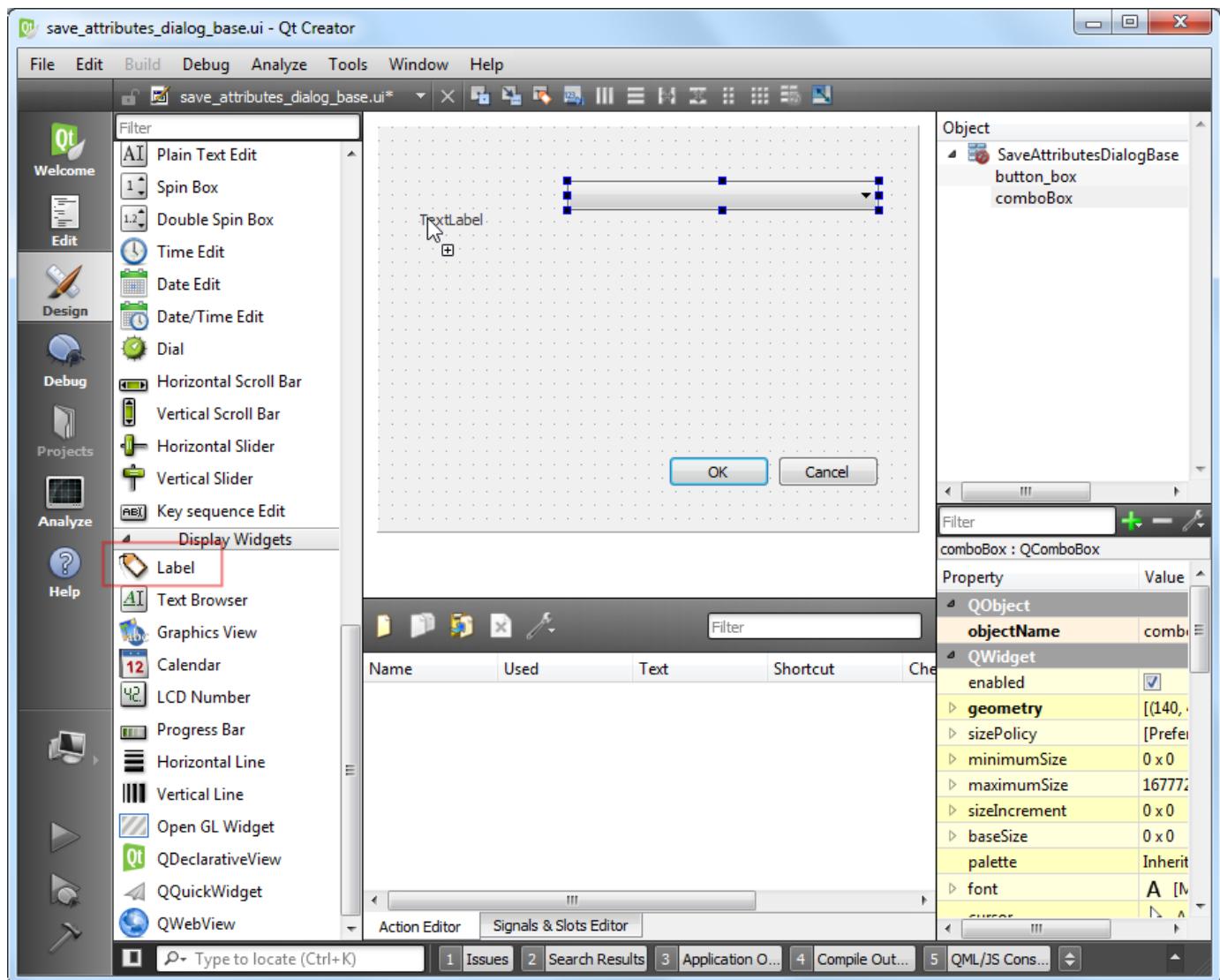
11. Blader naar de map met de plug-in en selecteer het bestand **save_attributes_dialog_base.ui** file. Klik op Openen.



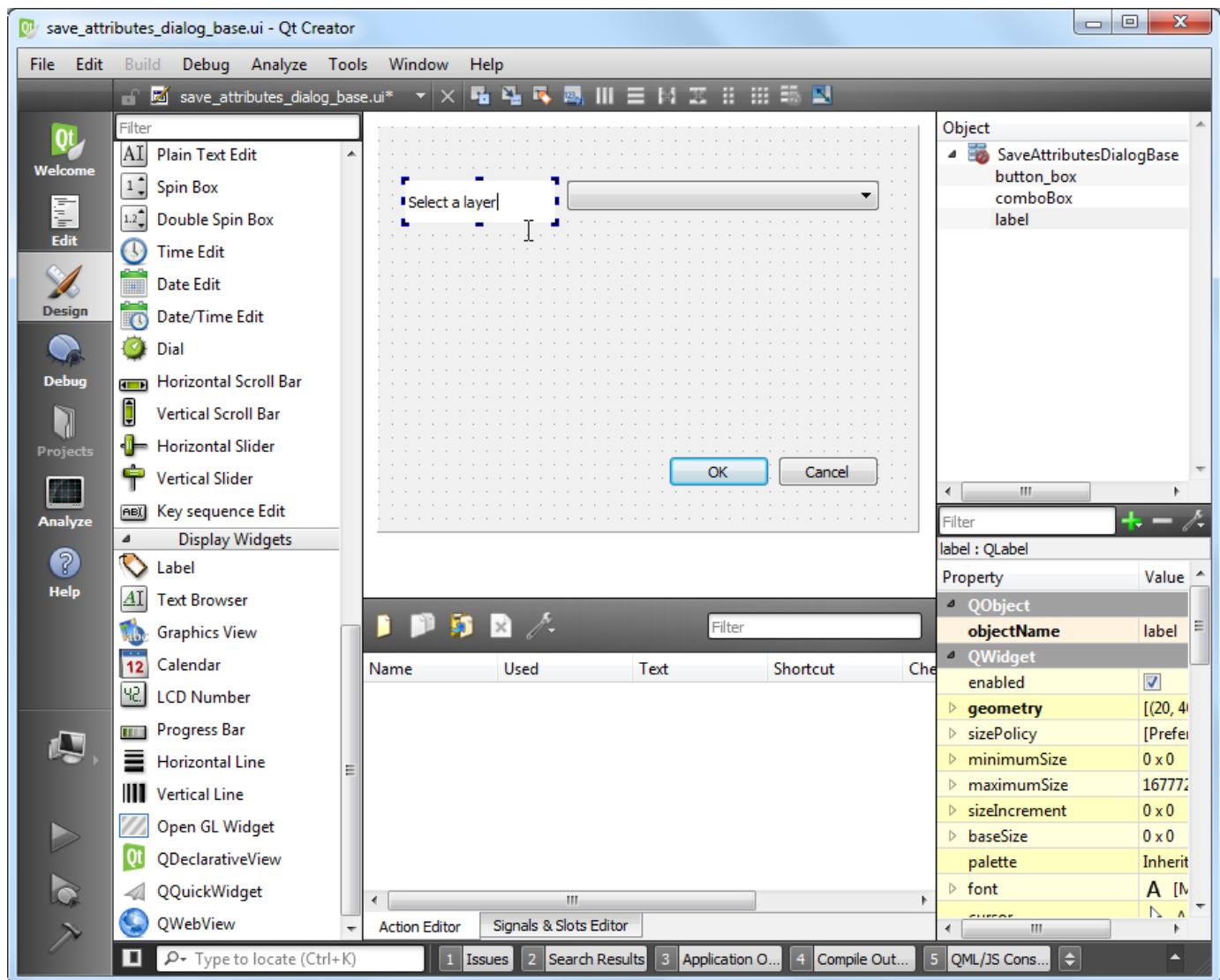
12. U zult een blanco dialoogvenster uit de plug-in zien. U kunt met slepen en neerzetten elementen verplaatsen vanuit het paneel aan de linkerkant naar het dialoogvenster. We zullen een type Combo Box van Input Widget toevoegen. Sleep het naar het dialoogvenster van de plug-in.



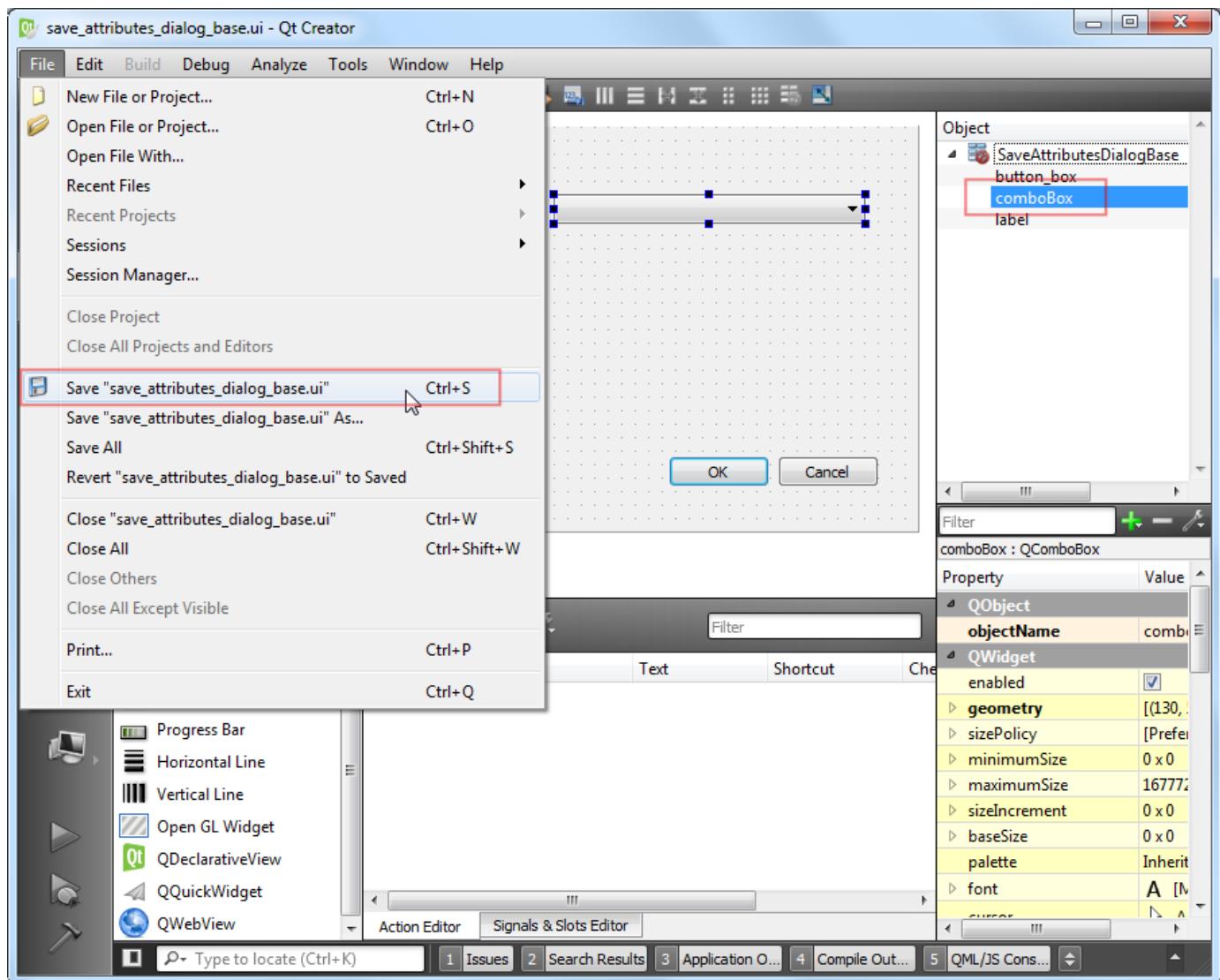
13. Maak het combinatievak op maat en pas de grootte aan. Sleep nu een type Label van Display Widget in het dialoogvenster.



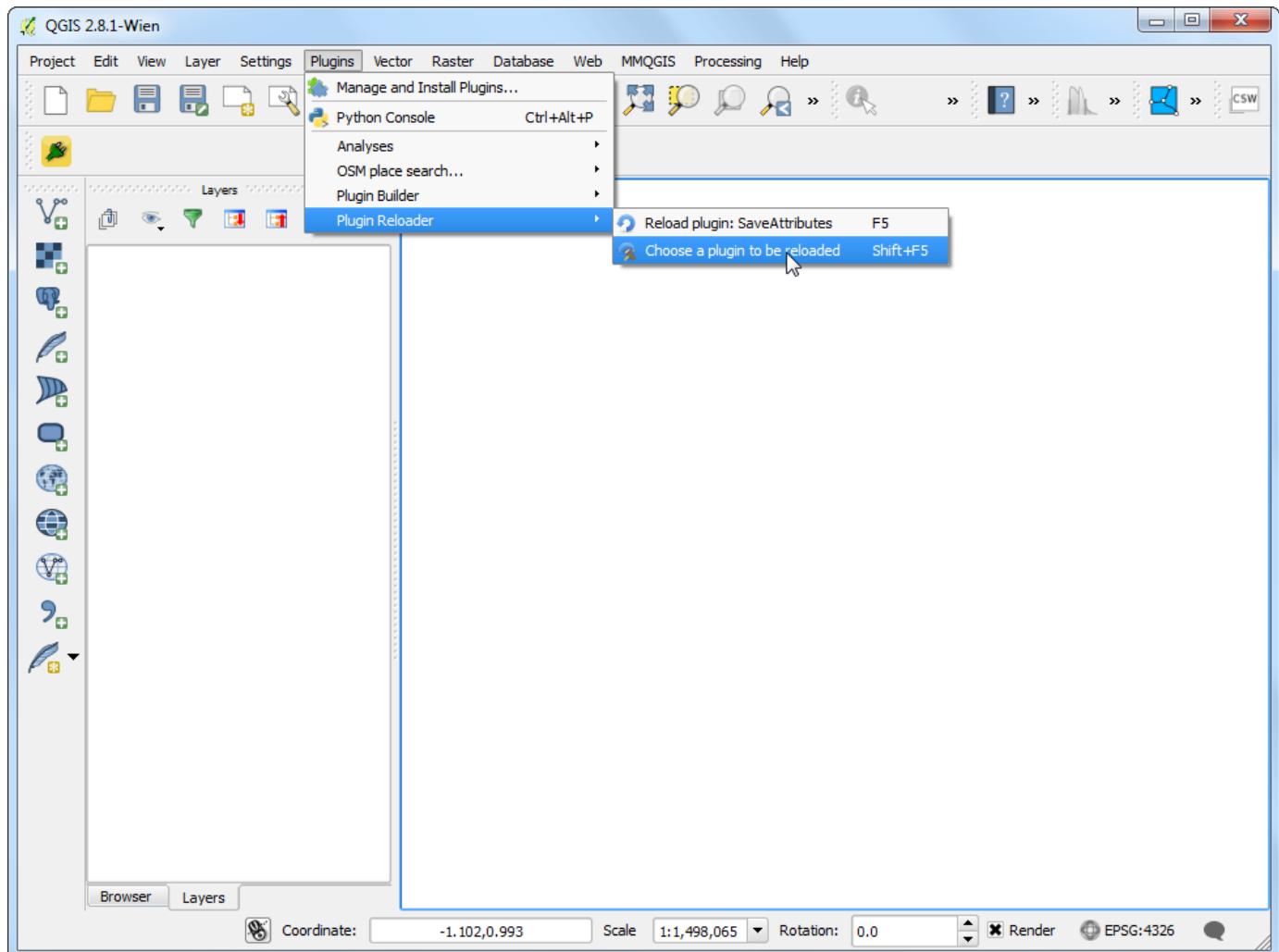
14. Klik op de labeltekst en voer in **Selecteer een laag**.



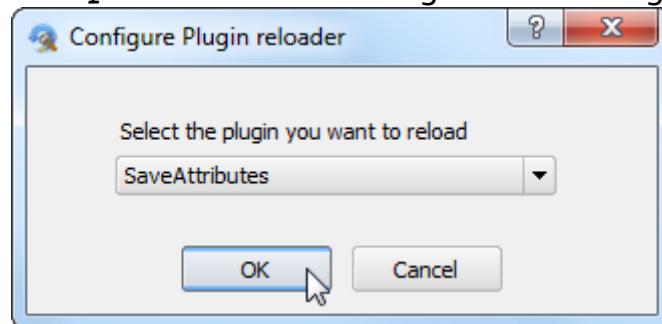
15. Sla dit bestand op door te gaan naar File > Save save_attributes_dialog_base.ui. Onthoud dat de naam van het object combinatievak **comboBox** is. We zullen er met die naam naar moeten verwijzen om interacties uit te kunnen voeren met behulp van code voor Python.



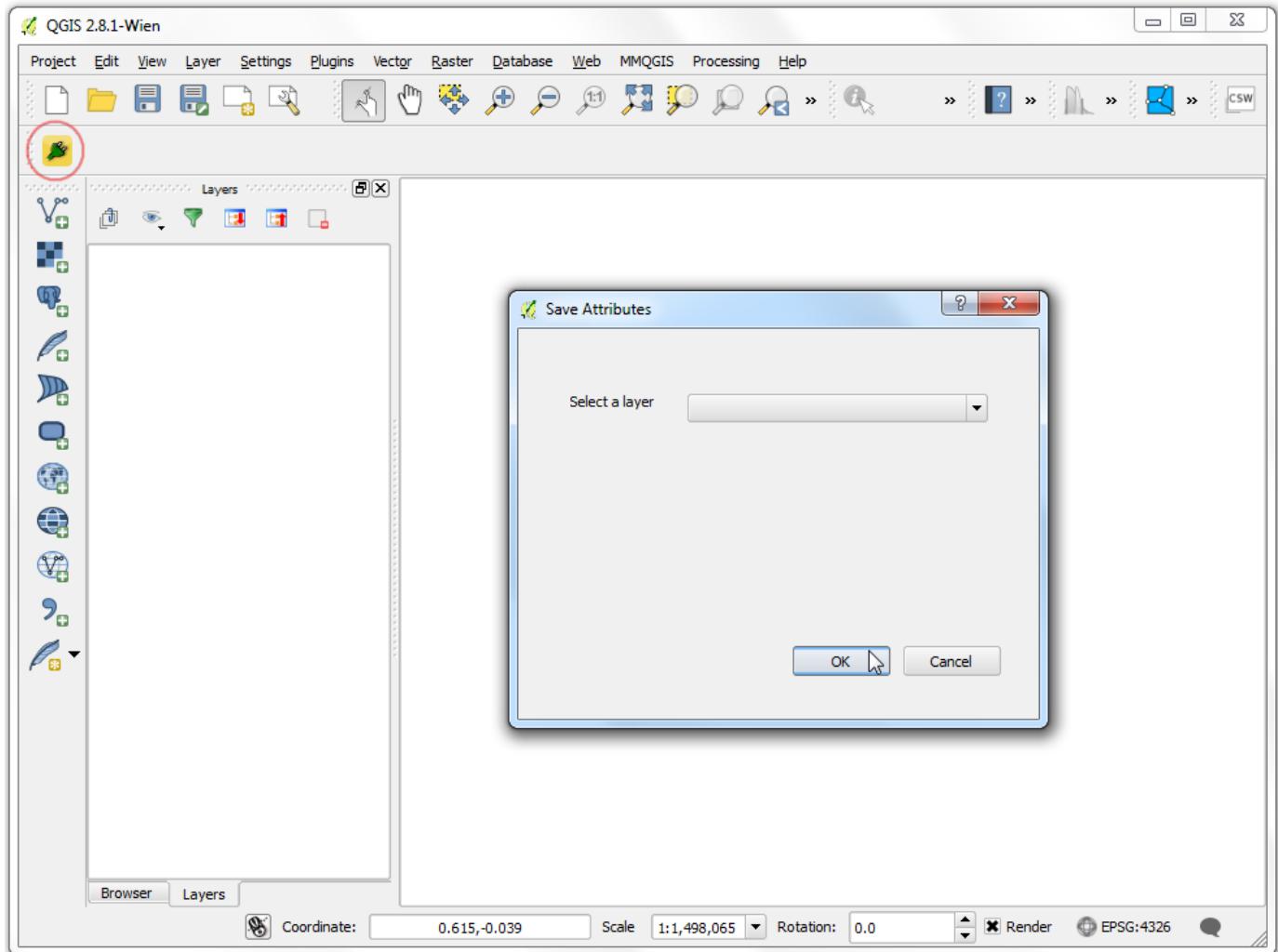
16. Laten we onze plug-in opnieuw laden zodat we de wijzigingen in het venster van het dialoogvenster kunnen zien. Ga naar Plug-ins > Plugin Loader > Choose a plugin to be reloaded.



17. Selecteer **AttributenOpslaan** in het dialoogvenster Configure Plugin reloader.

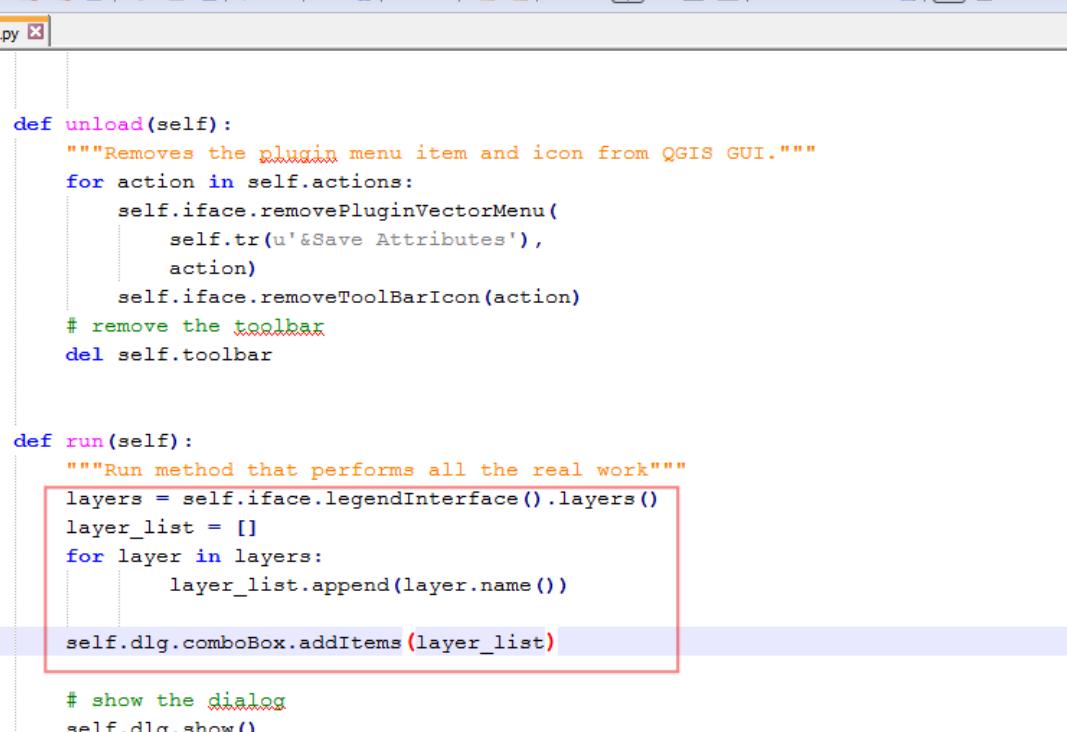


18. Klik nu op de knop Attributen opslaan als CSV. U zult het nieuw ontworpen dialoogvenster zien.



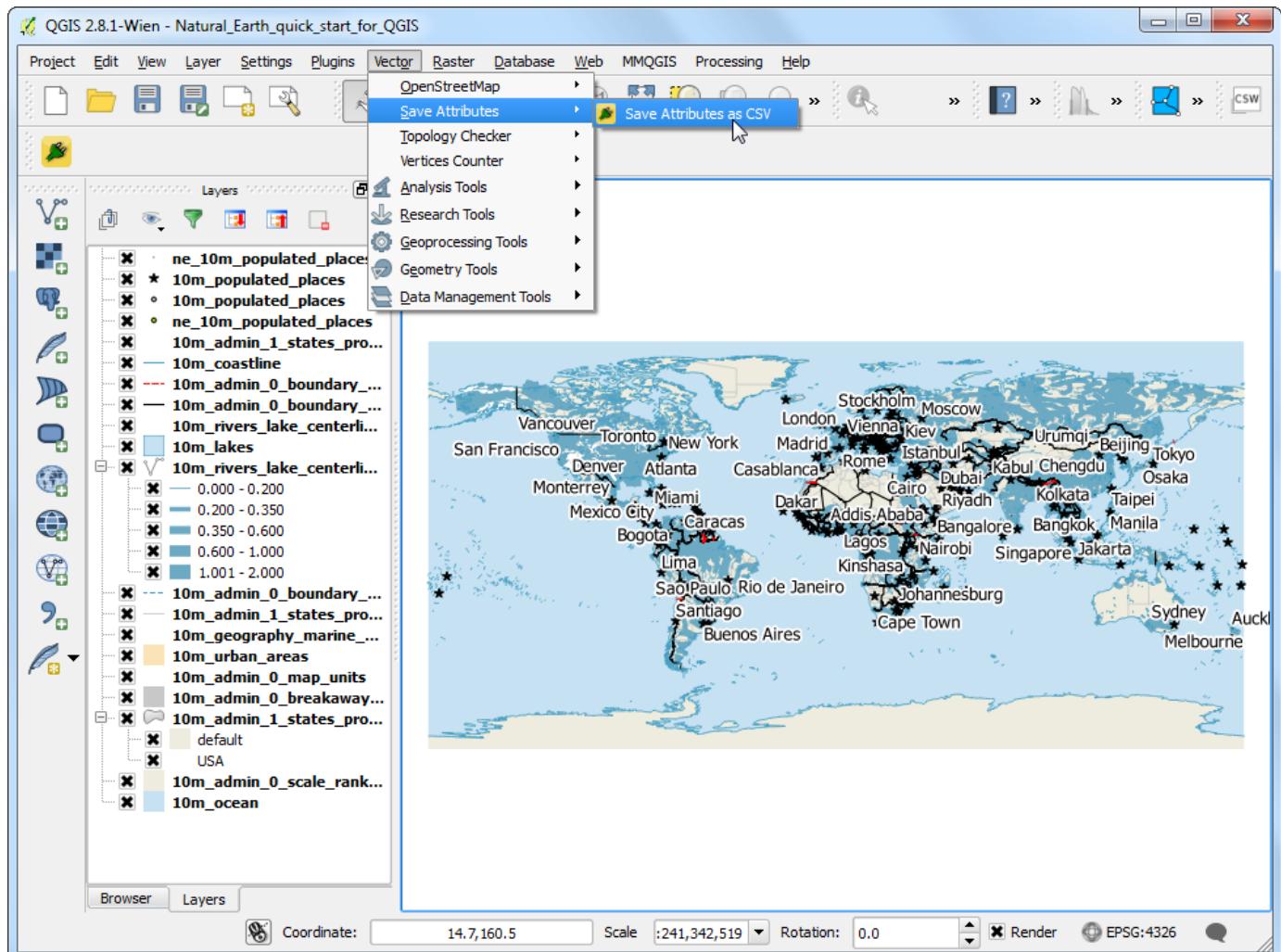
19. Laten we enige logica toevoegen aan de plug-in die het combinatievak zal vullen met de in QGIS geladen lagen. Ga naar de map van de plug-in en laadt het bestand `save_attributes.py` in een tekstbewerker. Scroll naar beneden en zoek naar de methode `run(self)`. Deze methode zal worden aangeroepen wanneer u op de knop van de werkbalk klikt of het menuitem van de plug-in selecteert. Voeg de volgende toe aan het begin van de methode. Deze code haalt de geladen lagen in QGIS op en voegt die toe aan het object `comboBox` van het dialoogvenster van de plugin.

```
layers = iface.legendInterface().layers()
layer_list = []
for layer in layers:
    layer_list.append(layer.name())
    self.dlg.comboBox.addItems(layer_list)
```

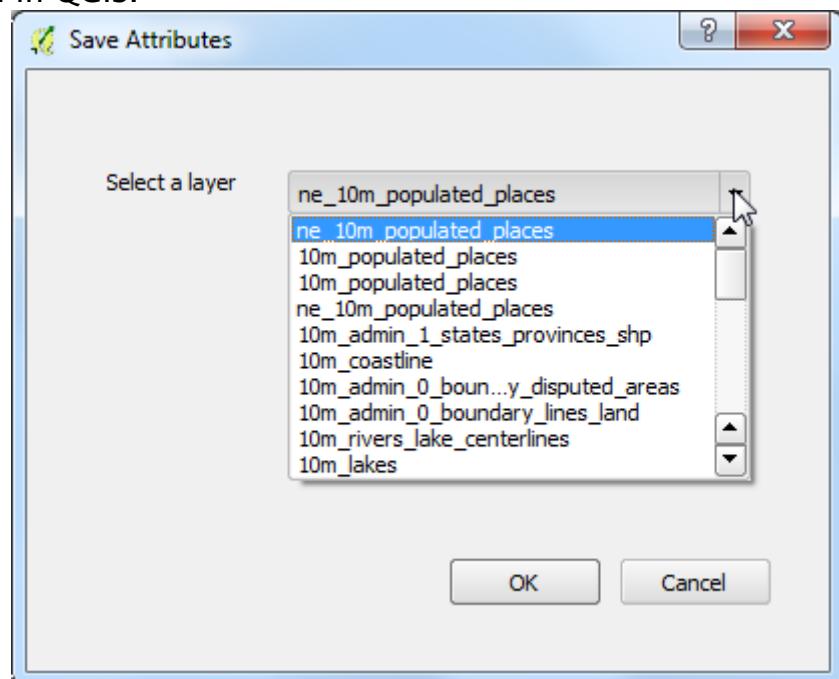


```
C:\Users\Ujava\qgis2\python\plugins\SaveAttributes\save_attributes.py - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
save_attributes.py
169
170
171     def unload(self):
172         """Removes the plugin menu item and icon from QGIS GUI."""
173         for action in self.actions():
174             self.iface.removePluginVectorMenu(
175                 self.tr(u'&Save Attributes'),
176                 action)
177             self.iface.removeToolBarIcon(action)
178         # remove the toolbar
179         del self.toolbar
180
181
182     def run(self):
183         """Run method that performs all the real work"""
184         layers = self.iface.legendInterface().layers()
185         layer_list = []
186         for layer in layers:
187             layer_list.append(layer.name())
188
189         self.dlg.comboBox.addItems(layer_list)
190
191         # show the dialog
192         self.dlg.show()
193         # Run the dialog event loop
```

20. Laadt, terug in het hoofdvenster van QGIS, de plug-in opnieuw door te gaan naar **Plug-ins** → **Plugin Loader** → **Reload plugin: SaveAttributes**. Als alternatief kunt u eenvoudigweg op F5 drukken. We moeten enkele lagen in QGIS laden om deze functionaliteit te testen. Nadat u enkele gegevens hebt geladen, start de plug-in door te gaan naar **Vector** → **Attributen opslaan** → **Attributen oposlaan als CSV**.

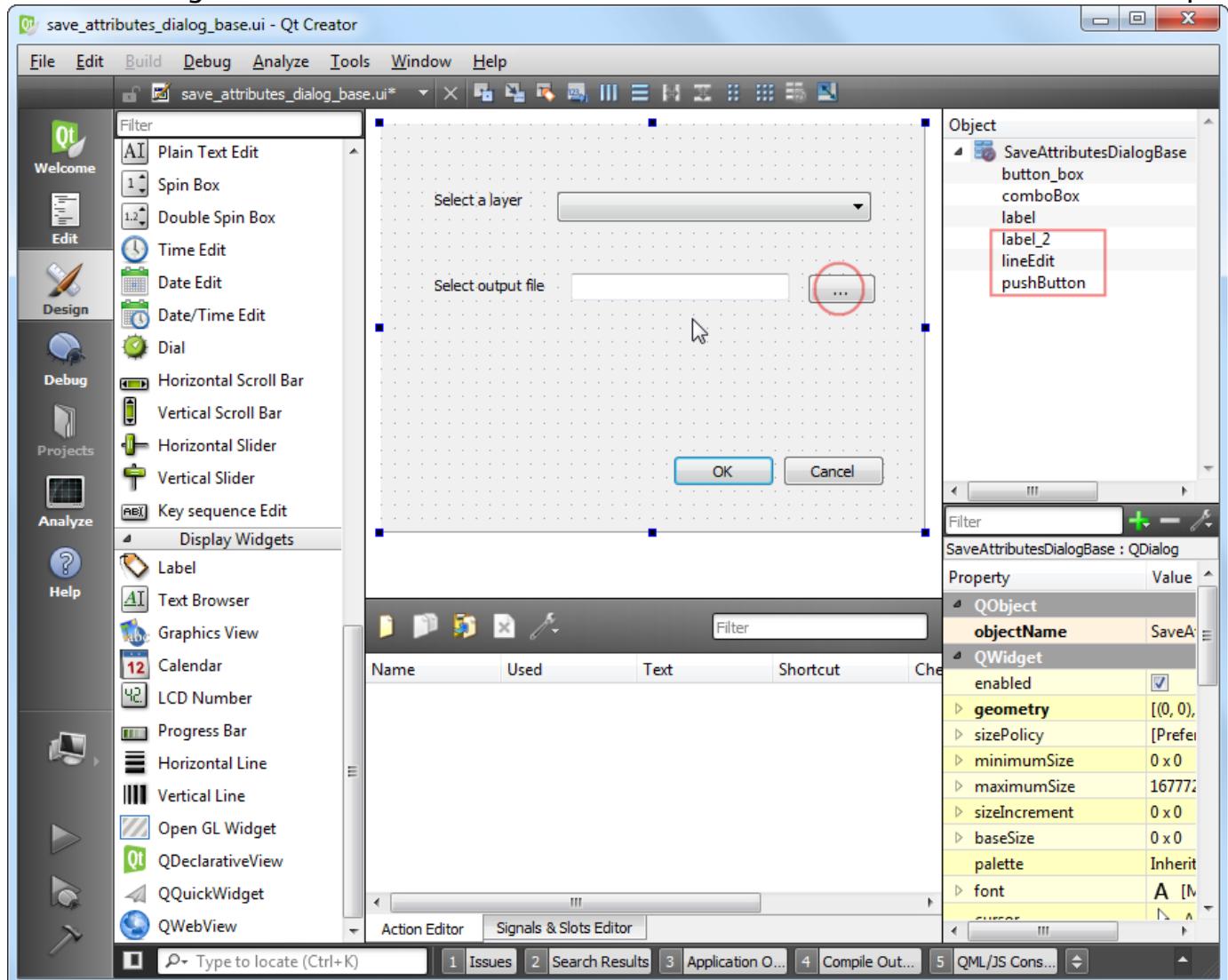


21. U zult zien dat ons combinatievak nu is geladen met de namen van de lagen die zijn geladen in QGIS.

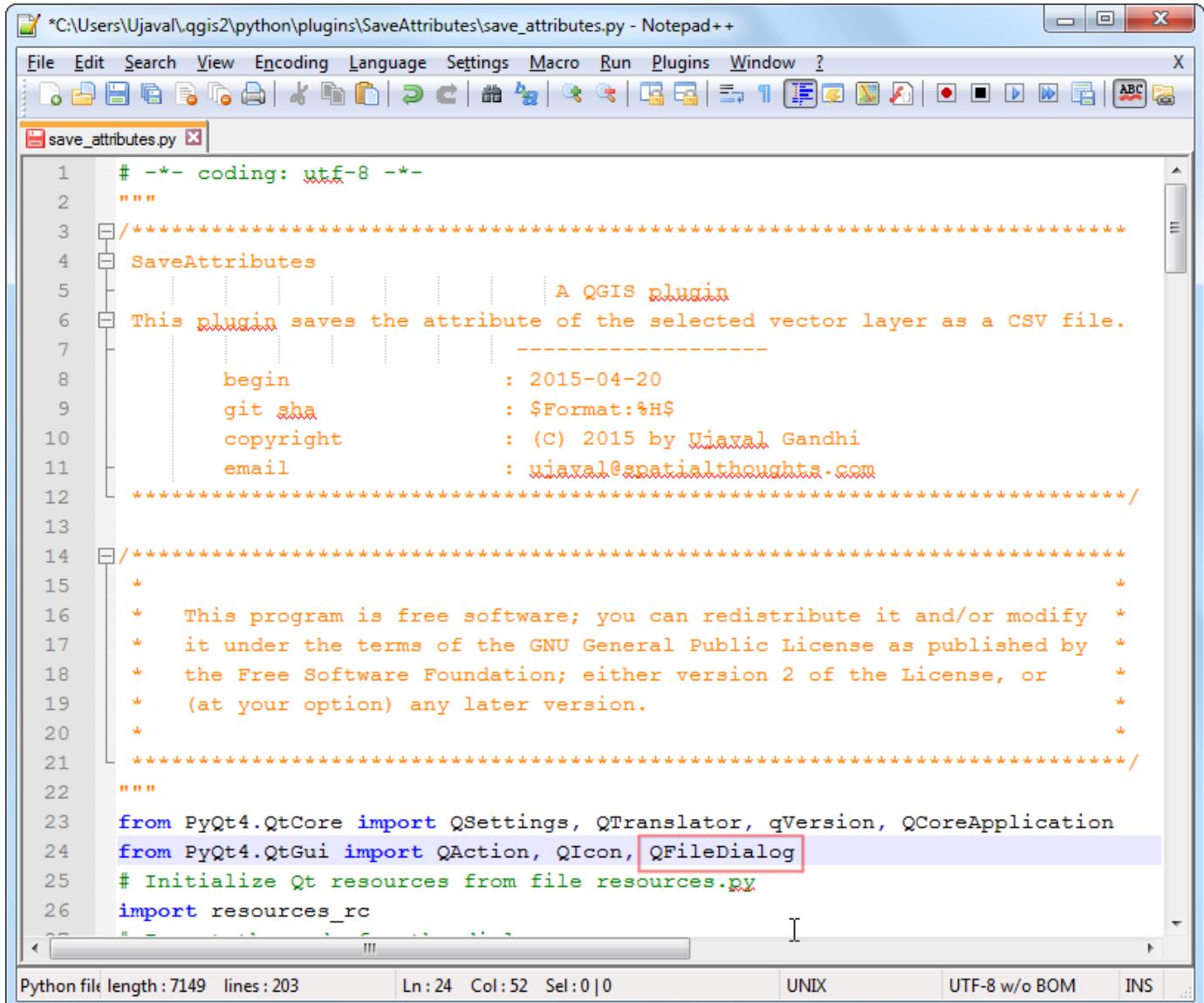


22. Laten we nog wat andere elementen van de gebruikersinterface toevoegen. Schakel terug naar *Qt Creator* en laadt het bestand *save_attributes_dialog_base.ui*. Voeg een *Label* Display Widget toe en

wijzig de tekst naar *uitvoerbestand selecteren*. Voeg een type *LineEdit* van Input Widget toe dat het pad naar het uitvoerbestand laat zien dat de gebruiker heeft gekozen. Voeg vervolgens een type *Push Button* van Button toe en wijzig het label van de knop naar Onthoud de objectnamen van de widgets die we moeten gebruiken om er interacties mee uit te kunnen voeren. Sla het bestand op.



23. We zullen nu code voor Python toevoegen om een dialoogvenster voor bestandsselectie te openen als de gebruiker op de knop ... drukt en het geselecteerde pad weergeven in het widget line edit. Open het bestand *save_attributes.py* in een tekstbewerker. Voeg *QFileDialog* toe aan onze lijst voor import aan het begin van het bestand.



```

1  # -*- coding: utf-8 -*-
2  """
3  *****
4  SaveAttributes
5  |          A QGIS plugin
6  |  This plugin saves the attribute of the selected vector layer as a CSV file.
7  |  -----
8  |      begin          : 2015-04-20
9  |      git sha        : $Format:%H$
10 |      copyright      : (C) 2015 by Ujaval Gandhi
11 |      email          : ujaval@spatialthoughts.com
12  *****
13
14  *****
15  *
16  *  This program is free software; you can redistribute it and/or modify
17  *  it under the terms of the GNU General Public License as published by
18  *  the Free Software Foundation; either version 2 of the License, or
19  *  (at your option) any later version.
20  *
21  *****
22  """
23  from PyQt4.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
24  from PyQt4.QtGui import QAction, QIcon, QFileDialog
25  # Initialize Qt resources from file resources.py
26  import resources_rc

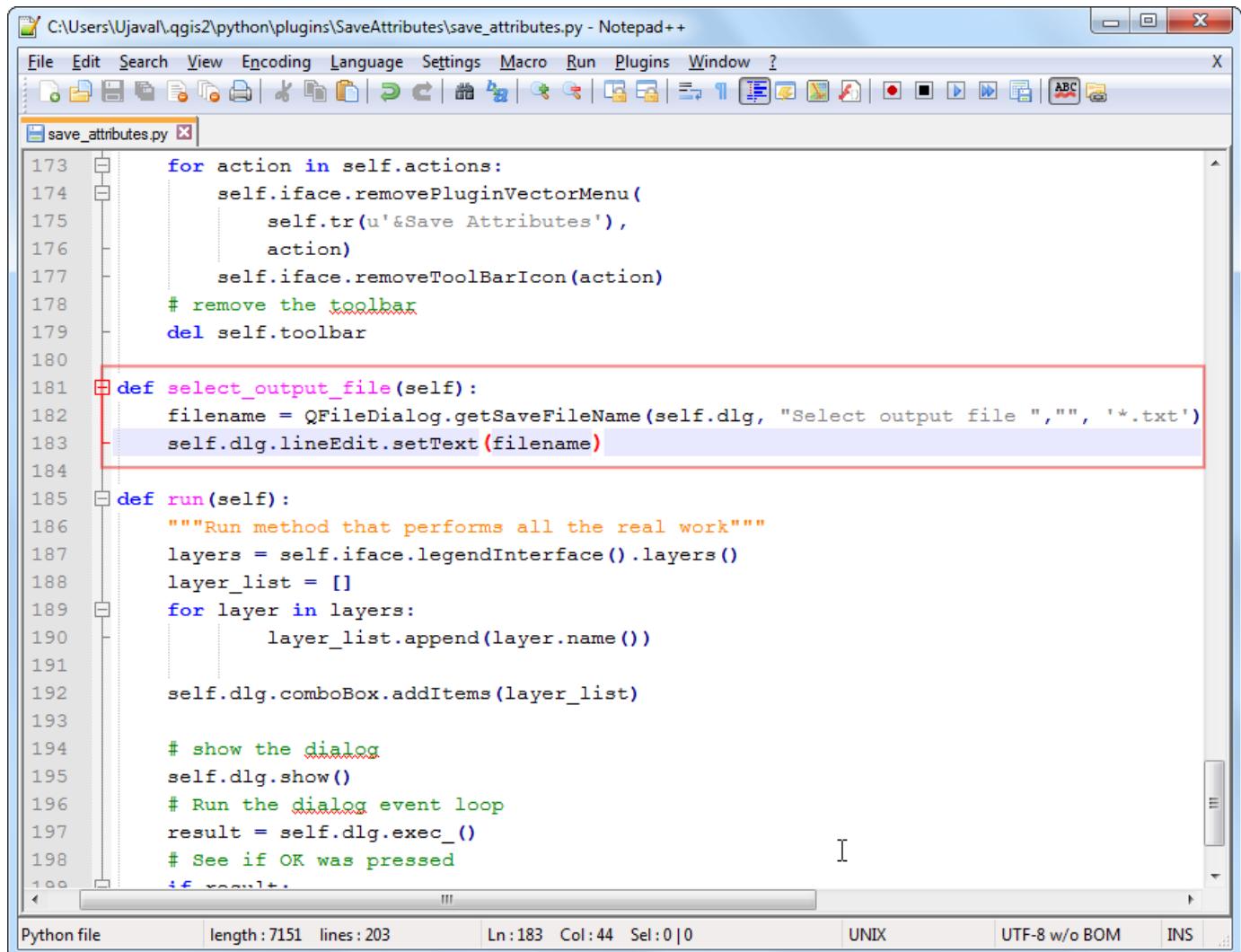
```

24. Voeg een nieuwe methode, genaamd `select_output_file` toe, met de volgende code. Deze code zal een dialoogvenster voor bestandsselectie openen en het widget line edit vullen met het pad van het bestand dat de gebruiker heeft gekozen.

```

def select_output_file(self):
    filename = QFileDialog.getSaveFileName(self.dlg, "Select output file","", "*.txt")
    self.dlg.lineEdit.setText(filename)

```



```

173     for action in self.actions:
174         self.iface.removePluginVectorMenu(
175             self.tr(u'&Save Attributes'),
176             action)
177         self.iface.removeToolBarIcon(action)
178         # remove the toolbar
179         del self.toolbar
180
181     def select_output_file(self):
182         filename = QFileDialog.getSaveFileName(self.dlg, "Select output file","", "*.txt")
183         self.dlg.lineEdit.setText(filename)
184
185     def run(self):
186         """Run method that performs all the real work"""
187         layers = self.iface.legendInterface().layers()
188         layer_list = []
189         for layer in layers:
190             layer_list.append(layer.name())
191
192         self.dlg.comboBox.addItems(layer_list)
193
194         # show the dialog
195         self.dlg.show()
196         # Run the dialog event loop
197         result = self.dlg.exec_()
198         # See if OK was pressed
199         if result...

```

25. Nu moeten we code toevoegen zodat, als op de knop ... wordt geklikt, de methode `select_output_file` wordt aangeroepen. Scroll naar boven naar de methode `__init__` en voeg aan de onderzijde daarvan de volgende regels toe. Deze code zal de eerder geladen tekst opruimen (indien aanwezig) in het widget line edit en ook de methode `select_output_file` verbinden met het signal `clicked` van het widget knop.

```

self.dlg.lineEdit.clear()
self.dlg.pushButton.clicked.connect(self.select_output_file)

```

C:\Users\Ujava1\qgis2\python\plugins\SaveAttributes\save_attributes.py - Notepad++

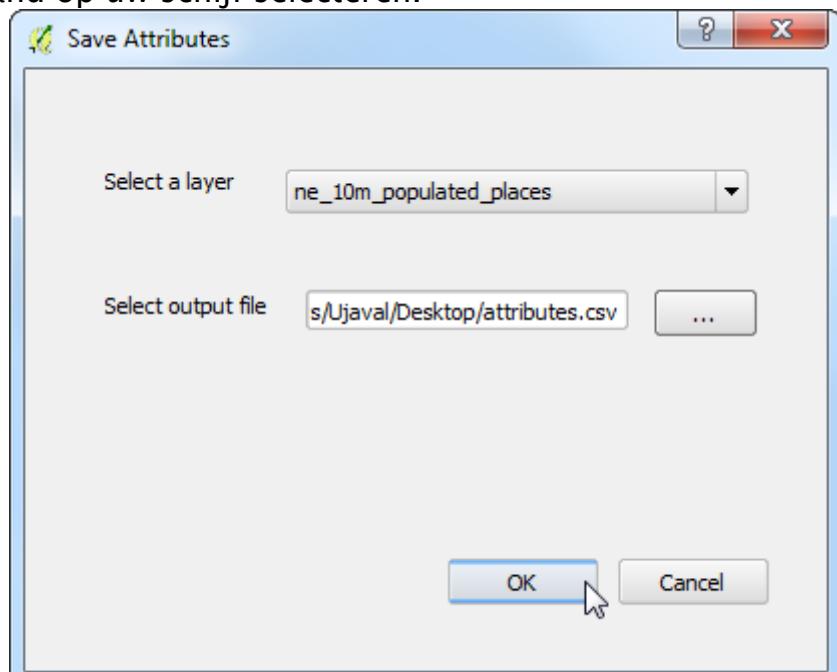
```
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
```

save_attributes.py

```
58     if qVersion() > '4.3.3':  
59         QCoreApplication.installTranslator(self.translator)  
60  
61     # Create the dialog (after translation) and keep reference  
62     self.dlg = SaveAttributesDialog()  
63  
64     # Declare instance attributes  
65     self.actions = []  
66     self.menu = self.tr(u'&Save Attributes')  
67     # TODO: We are going to let the user set this up in a future iteration  
68     self.toolbar = self iface.addToolBar(u'SaveAttributes')  
69     self.toolbar.setObjectName(u'SaveAttributes')  
70  
71     self.dlg.lineEdit.clear()  
72     self.dlg.pushButton.clicked.connect(self.select_output_file)  
73  
74  
75 # noinspection PyMethodMayBeStatic  
76 def tr(self, message):  
77     """Get the translation for a string using Qt translation API.  
78  
79     We implement this ourselves since we do not inherit QObject.  
80  
81     :param message: String for translation.  
82     :type message: str, QString  
83  
84     :return: translated version of message  
85     """
```

Python file | length: 7272 lines: 207 | Ln: 69 Col: 35 Sel: 0 | 0 | UNIX | UTF-8 w/o BOM | INS

26. Laadt, terug in QGIS, de plug-in opnieuw en open het dialoogvenster Attributen opslaan. Als alles goed ging kunt u nu op de knop ... drukken en een uitvoerbestand op uw schijf selecteren.

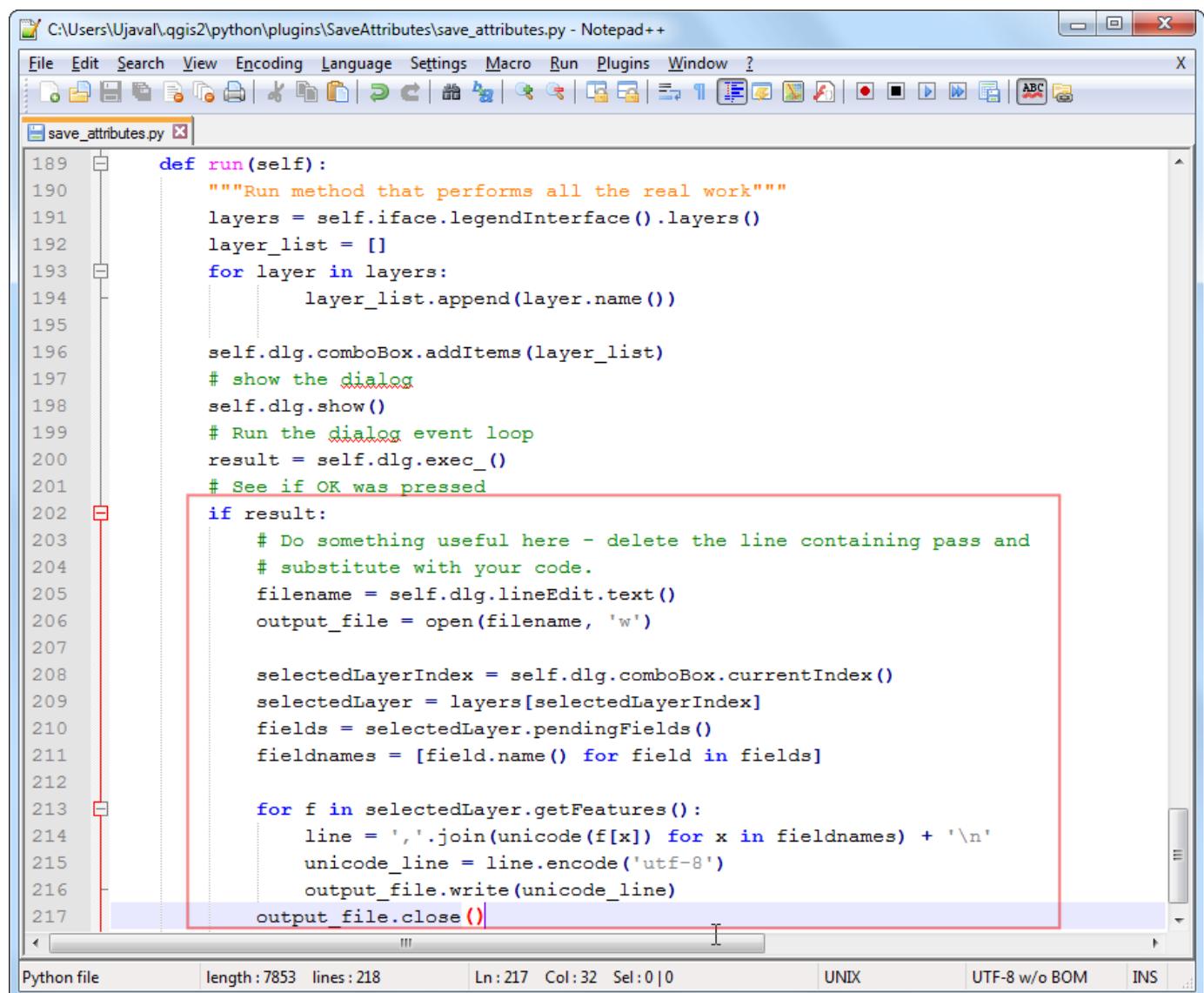


27. Als u op OK klikt in het dialoogvenster van de plug-in, gebeurt er niets. Dat komt omdat we de logica om de informatie over de attributen uit de laag op te halen en weg te schrijven naar het tekstbestand nog niet hebben toegevoegd. We hebben nu alle stukken op hun plaats om nu precies dat te gaan doen. Zoek naar de plek in de methode `run` waar `pass` staat. Vervang dat door de code die hieronder staat. De verklaring van deze code kan worden gevonden in [Beginnen met programmeren in Python](#).

```
filename = self.dlg.lineEdit.text()
output_file = open(filename, 'w')

selectedLayerIndex = self.dlg.comboBox.currentIndex()
selectedLayer = layers[selectedLayerIndex]
fields = selectedLayer.pendingFields()
fieldnames = [field.name() for field in fields]

for f in selectedLayer.getFeatures():
    line = ','.join(unicode(f[x]) for x in fieldnames) + '\n'
    unicode_line = line.encode('utf-8')
    output_file.write(unicode_line)
output_file.close()
```



```
189     def run(self):
190         """Run method that performs all the real work"""
191         layers = self iface.legendInterface().layers()
192         layer_list = []
193         for layer in layers:
194             layer_list.append(layer.name())
195
196         self.dlg.comboBox.addItems(layer_list)
197         # show the dialog
198         self.dlg.show()
199         # Run the dialog event loop
200         result = self.dlg.exec_()
201         # See if OK was pressed
202         if result:
203             # Do something useful here - delete the line containing pass and
204             # substitute with your code.
205             filename = self.dlg.lineEdit.text()
206             output_file = open(filename, 'w')
207
208             selectedLayerIndex = self.dlg.comboBox.currentIndex()
209             selectedLayer = layers[selectedLayerIndex]
210             fields = selectedLayer.pendingFields()
211             fieldnames = [field.name() for field in fields]
212
213             for f in selectedLayer.getFeatures():
214                 line = ','.join(unicode(f[x]) for x in fieldnames) + '\n'
215                 unicode_line = line.encode('utf-8')
216                 output_file.write(unicode_line)
217             output_file.close()
```

28. Nu is onze plug-in klaar. Laadt de plug-in opnieuw en probeer hem uit. U zult zien dat het tekstbestand voor de uitvoer dat u kiest de attributen van de vectorlaag bevat. U kunt de map voor de plug-in zippen en delen met uw gebruikers. Zij kunnen de inhoud uitpakken naar hun eigen map voor de plug-in en uw plug-in uitproberen. Als dit een echte plug-in zou zijn zou u die uploaden naar de [QGIS Plugin Repository](#) zodat alle gebruikers van QGIS hem kunnen vinden en uw plug-in kunnen downloaden.

Note

Deze plug-in is alleen ter demonstratie. Publiceer deze plug-in niet en upload hem ook niet naar de opslagplaats van plug-ins voor QGIS.

Hieronder staat, als referentie, het volledige bestand `save_attributes.py`.

```
# -*- coding: utf-8 -*-
"""
*****
SaveAttributes
    A QGIS plugin
This plugin saves the attribute of the selected vector layer as a CSV file.

-----
begin          : 2015-04-20
git sha        : $Format:%H$
copyright      : (C) 2015 by Ujaval Gandhi
email          : ujaval@spatialthoughts.com
*****"""

"""
*   This program is free software; you can redistribute it and/or modify
*   it under the terms of the GNU General Public License as published by
*   the Free Software Foundation; either version 2 of the License, or
*   (at your option) any later version.
*
*****"""

from PyQt4.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
from PyQt4.QtGui import QAction, QIcon, QFileDialog
# Initialize Qt resources from file resources.py
import resources_rc
# Import the code for the dialog
from save_attributes_dialog import SaveAttributesDialog
import os.path

class SaveAttributes:
    """QGIS Plugin Implementation."""

    def __init__(self, iface):
        """Constructor.

        :param iface: An interface instance that will be passed to this class
                      which provides the hook by which you can manipulate the QGIS
                      application at run time.
        :type iface: QgsInterface
        """

        # Save reference to the QGIS interface
```

```

self.iface = iface
# initialize plugin directory
self.plugin_dir = os.path.dirname(__file__)
# initialize locale
locale = QSettings().value('locale/userLocale')[0:2]
locale_path = os.path.join(
    self.plugin_dir,
    'i18n',
    'SaveAttributes_{}.qm'.format(locale))

if os.path.exists(locale_path):
    self.translator = QTranslator()
    self.translator.load(locale_path)

if qVersion() > '4.3.3':
    QCoreApplication.installTranslator(self.translator)

# Create the dialog (after translation) and keep reference
self.dlg = SaveAttributesDialog()

# Declare instance attributes
self.actions = []
self.menu = self.tr(u'&Save Attributes')
# TODO: We are going to let the user set this up in a future iteration
self.toolbar = self.iface.addToolBar(u'SaveAttributes')
self.toolbar.setObjectName(u'SaveAttributes')

self.dlg.lineEdit.clear()
self.dlg.pushButton.clicked.connect(self.select_output_file)

# noinspection PyMethodMayBeStatic
def tr(self, message):
    """Get the translation for a string using Qt translation API.

    We implement this ourselves since we do not inherit QObject.

    :param message: String for translation.
    :type message: str, QString

    :returns: Translated version of message.
    :rtype: QString
    """
    # noinspection PyTypeChecker,PyArgumentList,PyCallByClass
    return QCoreApplication.translate('SaveAttributes', message)

def add_action(
    self,
    icon_path,
    text,
    callback,
    enabled_flag=True,
    add_to_menu=True,
    add_to_toolbar=True,
    status_tip=None,
    whats_this=None,
    parent=None):
    """Add a toolbar icon to the toolbar.

    :param icon_path: Path to the icon for this action. Can be a resource

```

```

    path (e.g. ':/plugins/foo/bar.png') or a normal file system path.
:type icon_path: str

:param text: Text that should be shown in menu items for this action.
:type text: str

:param callback: Function to be called when the action is triggered.
:type callback: function

:param enabled_flag: A flag indicating if the action should be enabled
    by default. Defaults to True.
:type enabled_flag: bool

:param add_to_menu: Flag indicating whether the action should also
    be added to the menu. Defaults to True.
:type add_to_menu: bool

:param add_to_toolbar: Flag indicating whether the action should also
    be added to the toolbar. Defaults to True.
:type add_to_toolbar: bool

:param status_tip: Optional text to show in a popup when mouse pointer
    hovers over the action.
:type status_tip: str

:param parent: Parent widget for the new action. Defaults None.
:type parent: QWidget

:param whats_this: Optional text to show in the status bar when the
    mouse pointer hovers over the action.

:returns: The action that was created. Note that the action is also
    added to self.actions list.
:rtype: QAction
"""

icon = QIcon(icon_path)
action = QAction(icon, text, parent)
action.triggered.connect(callback)
action.setEnabled(enabled_flag)

if status_tip is not None:
    action.setStatusTip(status_tip)

if whats_this is not None:
    action.setWhatsThis(whats_this)

if add_to_toolbar:
    self.toolbar.addAction(action)

if add_to_menu:
    self iface.addPluginToVectorMenu(
        self.menu,
        action)

self.actions.append(action)

return action

def initGui(self):
    """Create the menu entries and toolbar icons inside the QGIS GUI."""

```

```

icon_path = ':/plugins/SaveAttributes/icon.png'
self.add_action(
    icon_path,
    text=self.tr(u'Save Attributes as CSV'),
    callback=self.run,
    parent=self iface mainWindow( ))

def unload(self):
    """Removes the plugin menu item and icon from QGIS GUI."""
    for action in self.actions():
        self iface.removePluginVectorMenu(
            self.tr(u'&Save Attributes'),
            action)
        self iface.removeToolBarIcon(action)
    # remove the toolbar
    del self.toolbar

def select_output_file(self):
    filename = QFileDialog.getSaveFileName(self.dlg, "Select output file", "", "*.txt")
    self.dlg.lineEdit.setText(filename)

def run(self):
    """Run method that performs all the real work"""
    layers = self iface.legendInterface().layers()
    layer_list = []
    for layer in layers:
        layer_list.append(layer.name())

    self.dlg.comboBox.addItems(layer_list)
    # show the dialog
    self.dlg.show()
    # Run the dialog event loop
    result = self.dlg.exec_()
    # See if OK was pressed
    if result:
        # Do something useful here - delete the line containing pass and
        # substitute with your code.
        filename = self.dlg.lineEdit.text()
        output_file = open(filename, 'w')

        selectedLayerIndex = self.dlg.comboBox.currentIndex()
        selectedLayer = layers[selectedLayerIndex]
        fields = selectedLayer.pendingFields()
        fieldnames = [field.name() for field in fields]

        for f in selectedLayer.getFeatures():
            line = ','.join(unicode(f[x]) for x in fieldnames) + '\n'
            unicode_line = line.encode('utf-8')
            output_file.write(unicode_line)
        output_file.close()

```