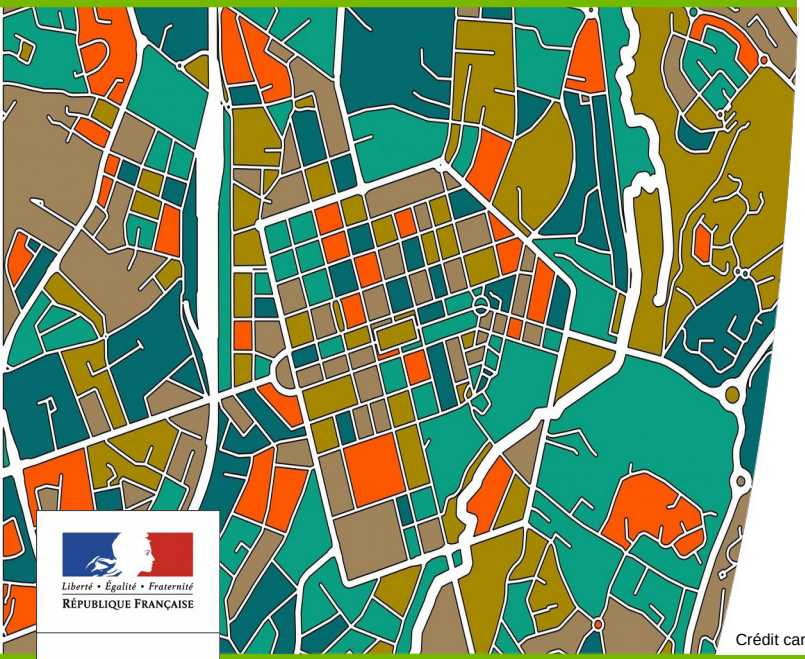
QGIS
Plugin Python
Création/modification

Décembre 2015



Crédit carte : DDTM85 olivier MAURY

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PRÉFET
DE LA VENDÉE

Ressources, territoires, habitats et logement
Energies et climat
Développement durable
Prévention des risques
Infrastructures, transports et mer

Présent pour l'avenir

Ministère de l'Écologie, du Développement durable,
des Transports et du Logement

Sommaire

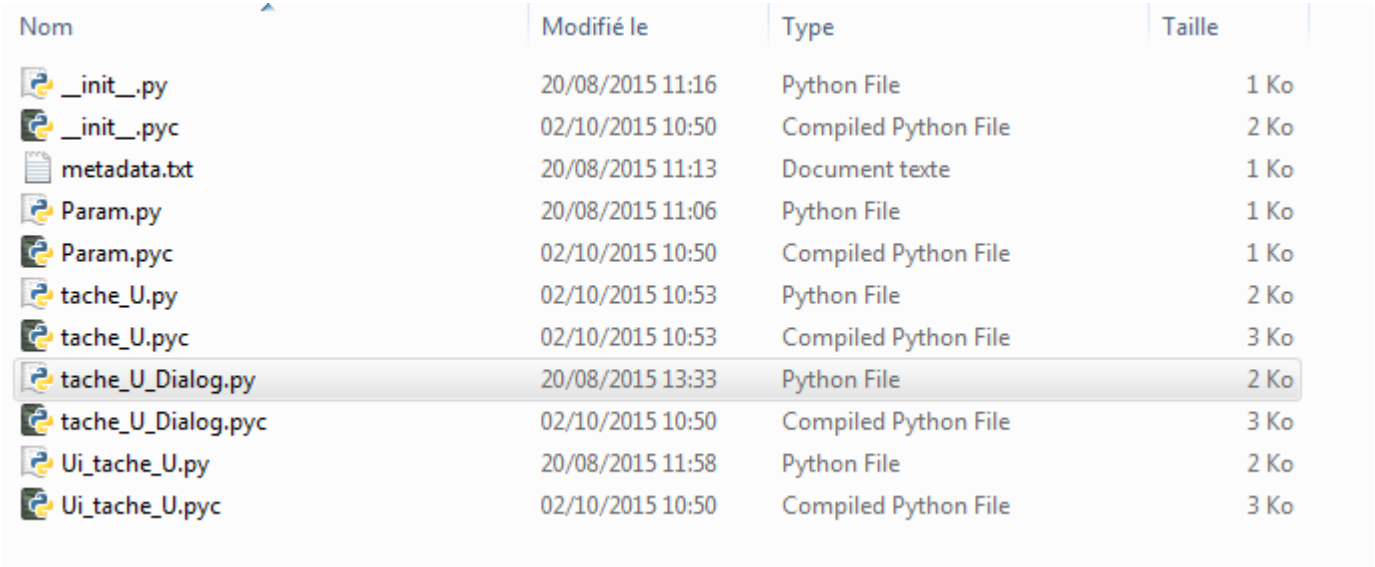
But analyser un petit plugin de recherche et affichage de données et voir comment l'adapter à ses données.

- 1. Présentation d'un plugin ou extension python
  - Fichiers qui le composent
  - Emplacement des fichiers
- 2. Exemple de plugin de recherche
  - But et principe de fonctionnement
  - Présentation des codes et modifications
- 3. Possibilité de modification

1. Présentation d'un plugin et de ses fichiers

- Un pluging python est constitué d'un ensemble de fichiers stockés dans un répertoire attitré :

  C:\Program Files (x86)\QGIS\profil\python\plugins\nom_du_rep_du_plugin



Nom	Modifié le	Type	Taille
__init__.py	20/08/2015 11:16	Python File	1 Ko
__init__.pyc	02/10/2015 10:50	Compiled Python File	2 Ko
metadata.txt	20/08/2015 11:13	Document texte	1 Ko
Param.py	20/08/2015 11:06	Python File	1 Ko
Param.pyc	02/10/2015 10:50	Compiled Python File	1 Ko
tache_U.py	02/10/2015 10:53	Python File	2 Ko
tache_U.pyc	02/10/2015 10:53	Compiled Python File	3 Ko
tache_U_Dialog.py	20/08/2015 13:33	Python File	2 Ko
tache_U_Dialog.pyc	02/10/2015 10:50	Compiled Python File	3 Ko
Ui_tache_U.py	20/08/2015 11:58	Python File	2 Ko
Ui_tache_U.pyc	02/10/2015 10:50	Compiled Python File	3 Ko

On remarque dans cette liste, trois formats :

- .txt fichier texte
- .py fichier de code python
- .pyc fichier de code python compilé

Seul les fichiers txt et py sont éditables , les fichiers pyc sont créés lors du lancement de qgis.

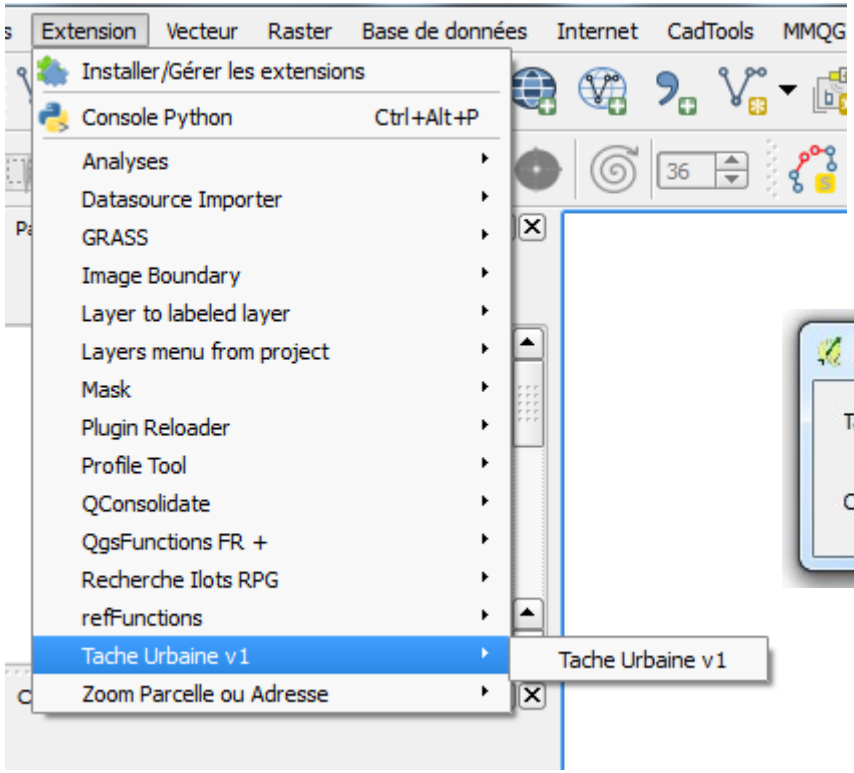
Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
PRÉFET DE LA VENDÉE

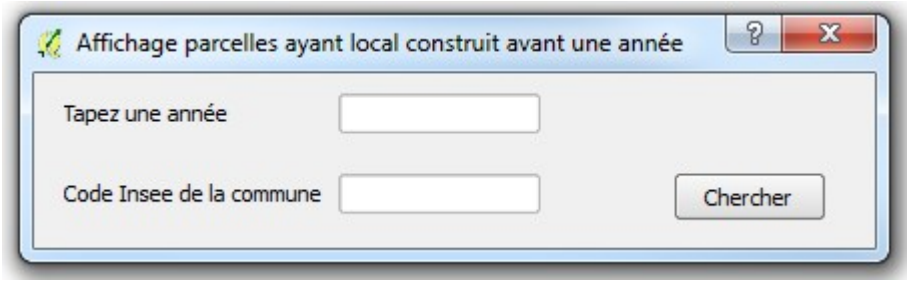
2. Exemple d'un plugin de recherche

- But et principe de fonctionnement

Le plugin que nous allons décortiqué dans le but de pouvoir être adapté à d'autres fins est un petit plugin de recherche et affichage.

On lance le plugin , un formulaire de saisie apparaît, on saisie une année et le code insee d'un commune. Le plugin se connecte à une base postgres, recherche et affiche les parcelles de la commune ayant un local construit avant la date saisie





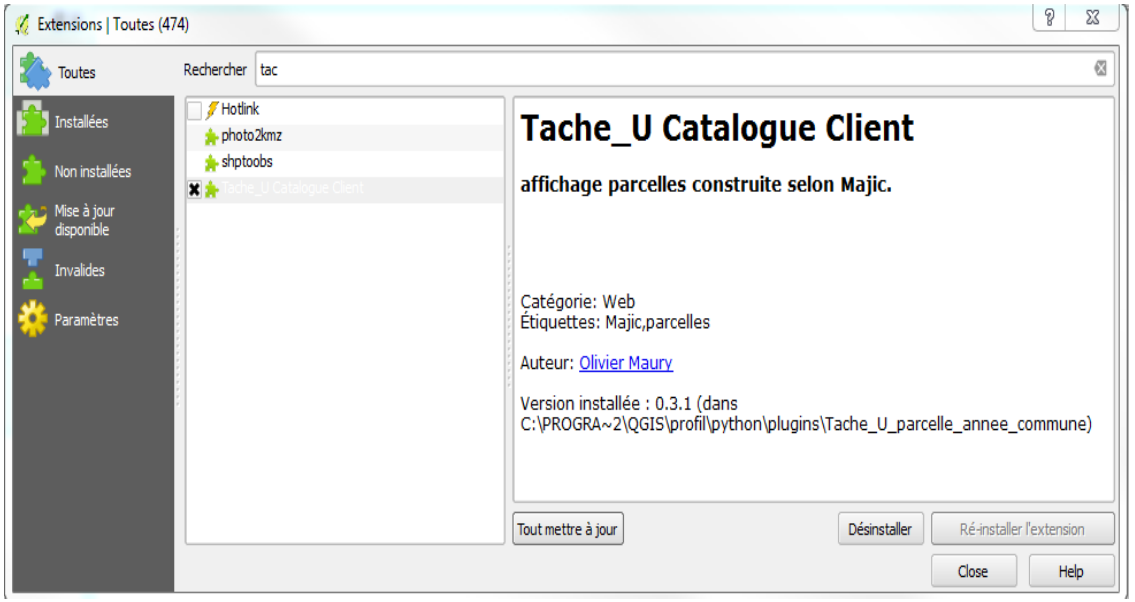
- Code des différents fichiers du plugin :

  Le fichier metadata.txt :

  indispensable au chargement du plugin lors du lancement de QGIS

```
1  [general]
2  name=Tache_U Catalogue Client
3  description=affichage parcelles construite selon Majic.
4  category=Web
5  version=0.3.1
6  qgisMinimumVersion=2.0
7  icon=images/MetaSearch.png
8  author=Olivier Maury
9  email=olivier.maury@vendee.gouv.fr
10 tags=Majic,parcelles
11 experimental=False
12 deprecated=False
```

Il contient les informations affichées dans le gestionnaire d'extensions



Pour accéder au code python on utilise un éditeur de texte

Le fichier _init_.py :

```
"""
/***************************************************************************
Name         : Tache_U_V1
Description: parcelle Majic selon son annee construction local
Date         : 20/08/2015
Copyright    : (C) 2015 olivier MAURY/DDTM85 - Logiciel libre
email        : DDTM@gouv.fr
 ***************************************************************************/
"""
def name():
    return "Tache_U_V1"
def description():
    return "Affichage de donnees MAJIC"
def version():
    return "Version 0.2"
def qgisMinimumVersion():
    return "1.0"
def classFactory(iface):
    # load tache_U class from file tache_U
    from tache_U import tache_U
    return tache_U(iface)
```

Lignes de description du fichier

Lignes d'initialisation du plugin

Appel de la class Tache_U et lancement du fichier tache_U.py

- tache_U.py

```
# Import the PyQt and QGIS libraries
from PyQt4.QtCore import *
from PyQt4.QtGui import *
from qgis.core import *
# Import the code for the dialog
from tache_U_Dialog import tache_U_Dialog

class tache_U:

  def __init__(self, iface):
    # Save reference to the QGIS interface
    self.iface = iface

  def initGui(self):
    # Create action that will start plugin configuration
    self.action = QAction("TUrbaine v1", self.iface.mainWindow())
    # connect the action to the run method
    QObject.connect(self.action, SIGNAL("activated()"), self.run)

    # Add toolbar button and menu item
    self.iface.addToolBarIcon(self.action)
    self.iface.addPluginToMenu("&Tache Urbaine v1", self.action)

  def unload(self):
    # Remove the plugin menu item and icon
    self.iface.removePluginMenu("&Tache Urbaine v1",self.action)
    self.iface.removeToolBarIcon(self.action)

  # run method that performs all the real work
  def run(self):
    # create and show the dialog
    dlg = tache_U_Dialog()
    # show the dialog
    dlg.show()
    result = dlg.exec_()
    # See if OK was pressed
    if result == 1:
      # remplacer pass par votre code
      pass
```

Import des fonctions de l'API

Appel du code du fichier tache_U_Dialog

Création et paramètrage du menu du plugin dans Qgis

Appel de la procédure run et définition de celle ci

Appel du fichier tache_U_dialog

Si besoin

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PRÉFET
DE LA VENDÉE

Tache_U_Dialog.py

```
from PyQt4.QtCore import *
from PyQt4.QtGui import *
from qgis.core import *
from qgis.gui import QgsMapCanvas
import qgis.utils
from PyQt4 import QtCore, QtGui
from Ui_tache_U import Ui_tache_U

import Param

Gu=""
Hote=Param.Hote
Port=Param.Port
basebd=Param.basebd
nomdeconnexion=Param.nomdeconnexion
mdpasse=Param.mdpasse
schema=Param.schema
tablededonnees=Param.tablededonnees
champ=Param.champ
# create the dialog
class tache_U_Dialog(QtGui.QDialog):
  def __init__(self):
    QtGui.QDialog.__init__(self)
    # Set up the user interface from Designer.
    self.ui = Ui_tache_U ()
    self.ui.setupUi(self)
    self.connect(self.ui.BtacheU,SIGNAL('clicked()'),self.affichage_tacheU)

  def affichage_tacheU (self):
      annee=self.ui.ZonSaitacheU.text()
      insee_commune=self.ui.ZonSaitacheU2.text()
      insee=insee_commune[-3:]
      nomdelacouche='Construction_avant_'+annee
      Critere_recherche="jannatmin<="+"'"+annee+"' and jannatmin>'0' and ccocom="+"'"+insee+"'"
      print Critere_recherche
      uri = QgsDataSourceURI()
      uri.setConnection(Hote, Port, basebd, nomdeconnexion, mdpasse)
      uri.setDataSource(schema, tablededonnees, champ, Gu+Critere_recherche+Gu)
      vlayer = QgsVectorLayer(uri.uri(), nomdelacouche, "postgres")
      QgsMapLayerRegistry.instance().addMapLayer(vlayer)
      qgis.utils.iface.zoomToActiveLayer()
```

Appel et chargement code de Ui_tache_U

Appel des données du fichier Param.py : il comprend les code d'accès à une base postgres

Affectation des variables nécessaire à l'accès à postgres

Initialisation du formulaire de dialogue définit dans le fichier Ui_tache_U.py si BtacheU=chercher est validé alors on lance la procédure affichage_tacheU

On récupère les zones de saisie du formulaire dans deux variables

critères de recherche dans postgres

Ouverture de la table avec le critère de recherche et paramètre de connexion Postgres

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

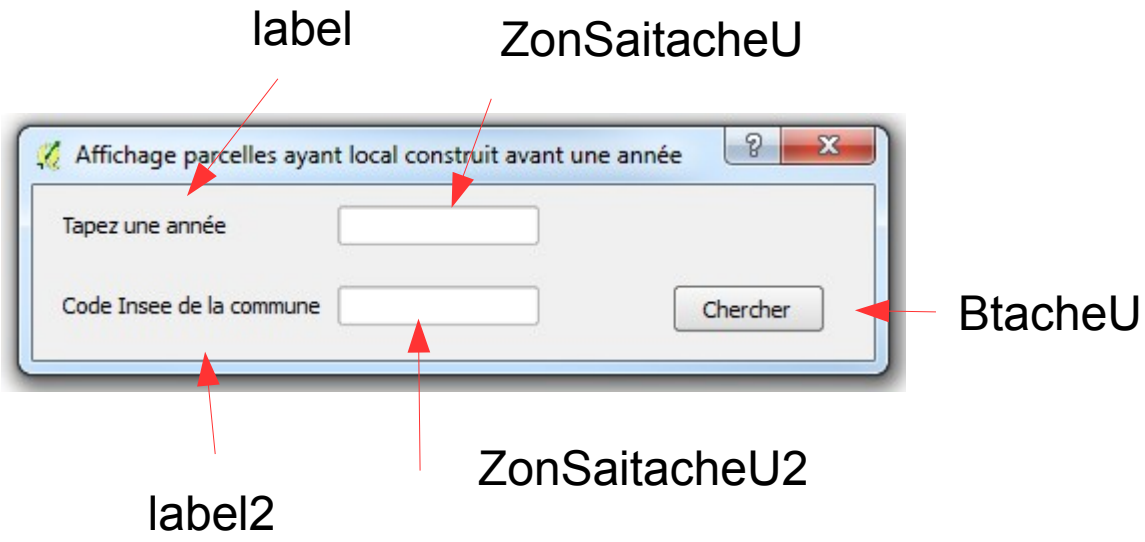
PRÉFET
DE LA VENDÉE

- Param.py

```
# -*- coding: ISO8859-1 -*-
Hote='localhost'
Port='5432'
basebd='fichiersfonciers'
nomdeconnexion='postgres'
mdpasse='postgres'
schema='ff_d85_2013'
tablededonnees='d85_2013_pnb10_parcelle'
champ='geompar'
```

Paramètres nécessaire à la connexion à Postgres
Ici base local sur poste individuel (localhost)

- Ui_tache_U.py



```python
# -*- coding: utf-8 -*-

from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    _fromUtf8 = lambda s: s

class Ui_tache_U(object):
    def setupUi(self, tache_U):
        tache_U.setObjectName(_fromUtf8("tache_U"))
        tache_U.resize(420, 87)
        self.BtacheU = QtGui.QPushButton(tache_U)
        self.BtacheU.setGeometry(QtCore.QRect(320, 50, 75, 23))
        self.BtacheU.setObjectName(_fromUtf8("BtacheU"))
        self.ZonSaitacheU = QtGui.QLineEdit(tache_U)
        self.ZonSaitacheU.setGeometry(QtCore.QRect(152, 10, 101, 20))
        self.ZonSaitacheU.setObjectName(_fromUtf8("ZonSaitacheU"))
        self.ZonSaitacheU2 = QtGui.QLineEdit(tache_U)
        self.ZonSaitacheU2.setGeometry(QtCore.QRect(152, 50, 101, 20))
        self.ZonSaitacheU2.setObjectName(_fromUtf8("ZonSaitacheU2"))
        self.label = QtGui.QLabel(tache_U)
        self.label.setGeometry(QtCore.QRect(15, 10, 150, 20))
        self.label.setObjectName(_fromUtf8("label"))
        self.label2 = QtGui.QLabel(tache_U)
        self.label2.setGeometry(QtCore.QRect(15, 50, 150, 20))
        self.label2.setObjectName(_fromUtf8("label2"))

        self.retranslateUi(tache_U)
        QtCore.QMetaObject.connectSlotsByName(tache_U)

    def retranslateUi(self, tache_U):
        tache_U.setWindowTitle(QtGui.QApplication.translate("tache_U", "Affichage parcelles ayant local construit avant une année", None, QtGui.QApplication.UnicodeUTF8))
        self.BtacheU.setText(QtGui.QApplication.translate("tache_U", "Chercher", None, QtGui.QApplication.UnicodeUTF8))
        self.label.setText(QtGui.QApplication.translate("tache_U", "Tapez une année ", None, QtGui.QApplication.UnicodeUTF8))
        self.label2.setText(QtGui.QApplication.translate("tache_U", "Code Insee de la commune", None, QtGui.QApplication.UnicodeUTF8))
```

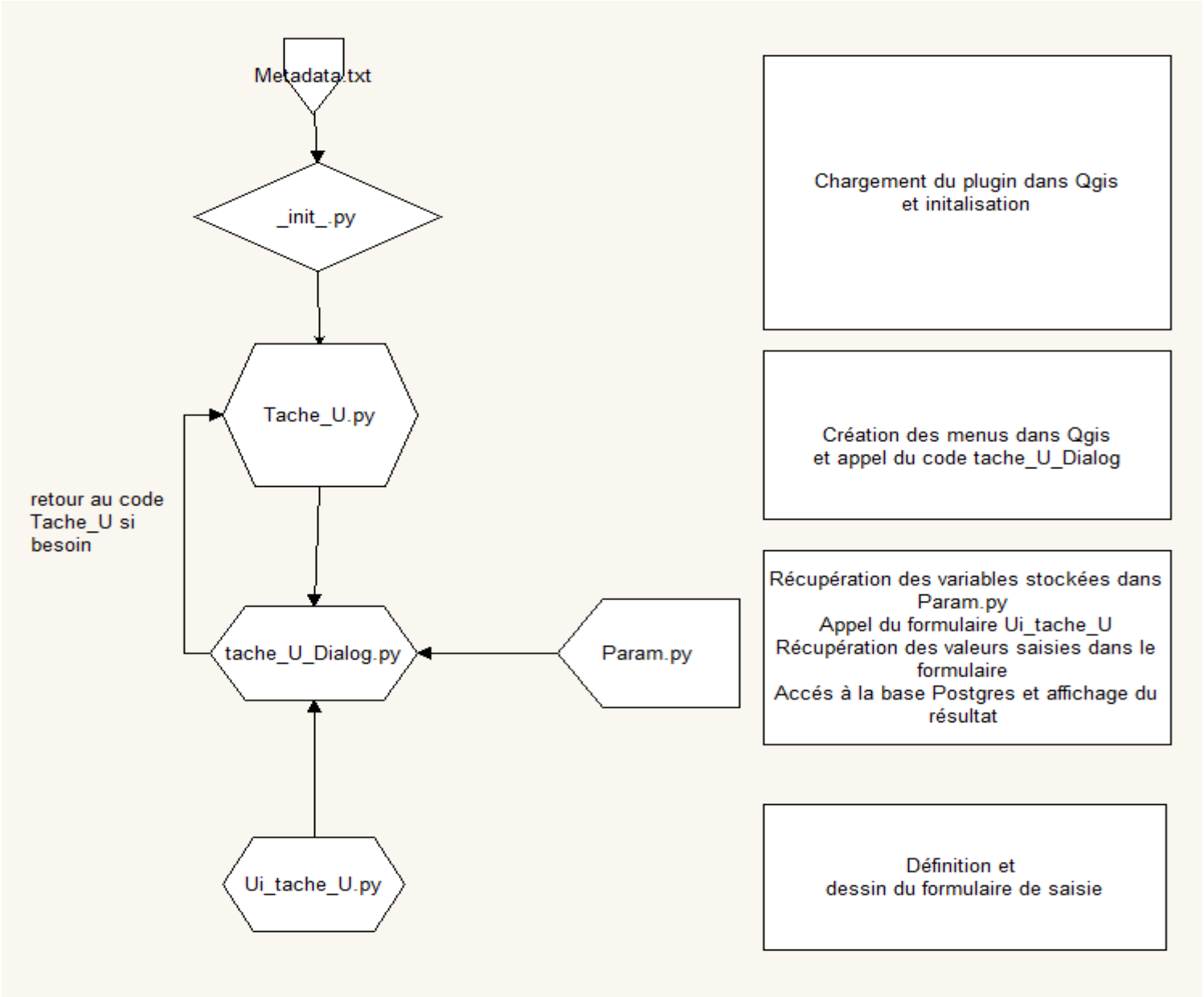
Définition et dessin du formulaire (On peut aussi créer le code du formulaire en s'aidant de Qtcreator)

Mise en forme

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PRÉFET
DE LA VENDÉE

Schéma de fonctionnement



3. Modification et adaptation

- Vous pouvez utiliser directement ce pluging en modifiant le fichier Param.py : remplacer les coordonnées de connexion à la base Postgres, vérifier le reste des données (nom de la table et champ geom affiché), et vous pourrez vous connecter à vos fichiers fonciers MAJIC dans Postgres pour les même critères de recherche.

- Puis vous pouvez changer les critères de recherche dans Tache_U_dialog.

  En rouge : champs présents dans le fichier Majic d85_2013_pnb10_parcelle

  En vert : variables définies par le formulaire python Ui_tache_U.py

  Critere_recherche="jannatmin<="+"'"+annee+"' and jannatmin>'0' and ccocom="+"'"+insee+"'"

- Le chargement de données peut se faire via un fichier à plat

  Code à substituer dans le fichier tache_U_Dialog.py après la définition de la variable critère de recherche (à adapter aussi)

  Critere_recherche="pacage="+"'"+pacage+"'"

  champ variable

  Chemin accès à la table

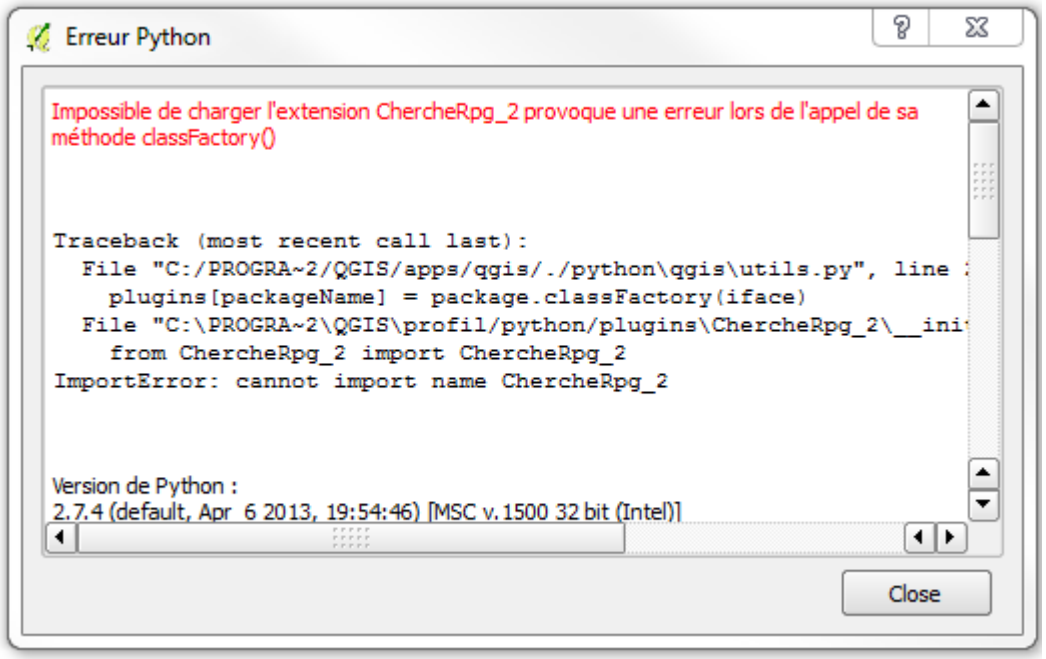
```
Cible = 'O:/agriculture_peche/RPG_S2_2014.TAB'
CRpg=QgsVectorLayer(Cible, "PACAGE_2014",'ogr')
CcRpg=CRpg.setSubsetString (Gu+Critere_recherche+Gu)
QgsMapLayerRegistry.instance().addMapLayer(CRpg)
qgis.utils.iface.zoomToActiveLayer()
```

Nom affiché dans le gestionnaire de couche
Pour éviter qu'il y ai plusieurs couches nommée PACAGE_2014 , il faut insérer ici la variable pacage
Remplacer "PACAGE_2014" par "PACAGE_2014_"+pacage
Ce qui donnera pour une recherche sur le pacage : 85000003
La couche : PACAGE_2014_85000003 dans le gestionnaire de couche

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
PRÉFET DE LA VENDÉE

- Test dans Qgis :

L'ensemble des fichiers du plugin sont placés dans le répertoire définitif (voir chapitre1). Au lancement de Qgis les fichiers sont compilés (des messages d'erreurs peuvent s'afficher).



Pour ne pas fermer et relancer à chaque test Qgis, il y a un plugin adapté « Plugin Reloader » : il permet de recharger (compiler) un plugin sans relancer qgis donc gain de temps. On choisit dans une liste le plugin à relancer puis après il suffit de cliquer sur le bouton associé.

Pour la rédaction des fichier Python j'utilise un éditeur de texte comme notepad (il colorise la syntaxe et aide à la rédaction)

3. approfondir le sujet :

- Livre de recettes PYGIS : python pour Qgis.

  http://docs.qgis.org/2.0/fr/docs/pyqgis_developer_cookbook/index.html

  http://docs.qgis.org/2.8/pdf/fr/QGIS-2.8-PyQGISDeveloperCookbook-fr.pdf

- Conception d'un plugin Python pour Qgis

  http://geoinformations.metier.e2.rie.gouv.fr/fichier/pdf/CETENP_2011-10-27_Concevoir_Plugin_Python_QGIS_cle1f6963.pdf?arg=177828825&cle=b663023619b5e68c2fbe7f3d7dedbb239e70f484&file=pdf/CETENP_2011-10-27_Concevoir_Plugin_Python_QGIS_cle1f6963.pdf

- Utilisation plus simple du python sans passer par le plugin

  http://geoinformations.metier.e2.rie.gouv.fr/qgis-utilisation-des-actions-par-olivier-maury-a3265.html

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PRÉFET
DE LA VENDÉE

A vous de coder !!