# -\*- coding: utf-8 -\*-  
*"""  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 image  
 A QGIS plugin  
 tpds  
 Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-Builder/  
 -------------------  
 begin : 2024-05-08  
 git sha : $Format:%H$  
 copyright : (C) 2024 by dounia  
 email : @gmil.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 qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication  
from qgis.PyQt.QtGui import QIcon  
from qgis.PyQt.QtWidgets import QAction  
from sentinelsat import SentinelAPI, read\_geojson, geojson\_to\_wkt  
import datetime  
  
# Initialize Qt resources from file resources.py  
from .resources import \*  
# Import the code for the dialog  
from .tp\_dialog import imageDialog  
import os.path  
  
  
class image:  
 def update\_percentage(self, value):  
 *"""Met à jour l'affichage du pourcentage."""* # Convertit la valeur du slider en pourcentage  
 percentage = value / self.dlg.horizontalSlider.maximum() \* 100  
 # Met à jour l'affichage du pourcentage dans votre interface utilisateur  
 self.dlg.label\_7.setText(f"{percentage:.2f}%")  
 """QGIS Plugin Implementation."""  
  
   
  
 # Connecte le signal sliderMoved du horizontalSlider à la fonction update\_percentage  
   
  
 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',  
 'image\_{}.qm'.format(locale))  
  
 if os.path.exists(locale\_path):  
 self.translator = QTranslator()  
 self.translator.load(locale\_path)  
 QCoreApplication.installTranslator(self.translator)  
  
 # Declare instance attributes  
 self.actions = []  
 self.menu = self.tr(u'&Tp02')  
  
 # Check if plugin was started the first time in current QGIS session  
 # Must be set in initGui() to survive plugin reloads  
 self.first\_start = None  
  
 # 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('image', 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:  
 # Adds plugin icon to Plugins toolbar  
 self.iface.addToolBarIcon(action)  
  
 if add\_to\_menu:  
 self.iface.addPluginToMenu(  
 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/tp/icon.png'  
 self.add\_action(  
 icon\_path,  
 text=self.tr(u''),  
 callback=self.run,  
 parent=self.iface.mainWindow())  
  
 # will be set False in run()  
 self.first\_start = True  
  
  
 def unload(self):  
 *"""Removes the plugin menu item and icon from QGIS GUI."""* for action in self.actions:  
 self.iface.removePluginMenu(  
 self.tr(u'&Tp02'),  
 action)  
 self.iface.removeToolBarIcon(action)  
  
 class imageDialog(QDialog):  
 def \_\_init\_\_(self):  
 QDialog.\_\_init\_\_(self)  
  
 # Créer un widget de calendrier pour sélectionner une date  
 self.calendarWidget = QCalendarWidget()  
  
 # Créer un bouton de téléchargement  
 self.downloadButton = QPushButton('Télécharger')  
 self.downloadButton.clicked.connect(self.downloadImage)  
  
 # Disposition des éléments dans la boîte de dialogue  
 layout = QVBoxLayout()  
 layout.addWidget(self.calendarWidget)  
 layout.addWidget(self.downloadButton)  
  
 self.setLayout(layout)  
  
 def downloadImage(self):  
 # Fonction pour télécharger l'image satellite pour la date sélectionnée  
 selectedDate = self.calendarWidget.selectedDate()  
 # Ici, vous pouvez écrire le code pour télécharger l'image satellite pour la date sélectionnée  
 print("Télécharger l'image pour la date:", selectedDate.toString("yyyy-MM-dd"))  
 # Connexion à l'API Copernicus  
  
 api = SentinelAPI('Dounia Aoulad Allouch', 'j#-EMPUR7LC9jN9', 'https://scihub.copernicus.eu/dhus')  
  
 def download\_image(area\_of\_interest, start\_date, end\_date, cloud\_cover=10):  
 # Convertir la zone d'intérêt en format WKT  
 footprint = geojson\_to\_wkt(area\_of\_interest)  
  
 # Recherche d'images disponibles dans la zone d'intérêt et la plage de dates spécifiée  
 products = api.query(footprint,  
 date=(start\_date, end\_date),  
 platformname='Sentinel-2',  
 cloudcoverpercentage=(0, cloud\_cover))  
  
 # Téléchargement des images  
 for product in products:  
 print(f"Téléchargement de l'image: {product['title']}")  
 api.download(product['uuid'])  
  
 # Exemple d'utilisation  
 if \_\_name\_\_ == '\_\_main\_\_':  
 # Zone d'intérêt au format GeoJSON  
 area\_of\_interest = {  
 "type": "Polygon",  
 "coordinates": [[[5.0, 45.0], [6.0, 45.0], [6.0, 46.0], [5.0, 46.0], [5.0, 45.0]]]  
 }  
  
 # Dates de début et de fin de la période de recherche  
 start\_date = datetime.datetime(2024, 1, 1)  
 end\_date = datetime.datetime(2024, 1, 31)  
  
 # Télécharger l'image pour la zone d'intérêt et la période spécifiées  
 download\_image(area\_of\_interest, start\_date, end\_date)  
  
  
 def run(self):  
 *"""Run method that performs all the real work"""* # Create the dialog with elements (after translation) and keep reference  
 # Only create GUI ONCE in callback, so that it will only load when the plugin is started  
 if self.first\_start == True:  
 self.first\_start = False  
 self.dlg = imageDialog()  
 self.dlg.horizontalSlider.sliderMoved.connect(self.update\_percentage)  
  
 # 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.  
 pass