# file2docx.py

# coding:utf-8  
  
from docx import Document  
import os  
import sys  
from PyQt5.QtWidgets import \*  
from PyQt5 import QtGui  
from GUI import Ui\_Dialog  
import time  
  
class UserWidgets(QMainWindow, Ui\_Dialog):  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(UserWidgets, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.pwd = os.getcwd()  
 self.files\_list = []  
 self.document = Document()  
 self.init\_menu()  
  
  
 self.setupUi(self)  
 self.config()  
 self.set\_events()  
 self.changes()  
  
 def init\_menu(self):  
 bar = self.menuBar()  
 stop = bar.addMenu('文件')  
 settings = bar.addMenu('设置')  
  
 stop.addAction('退出',exit)  
 filetype = QAction('文件类型',self)  
 filetype.setShortcut("Ctrl + F")  
 filetype.triggered.connect(self.get\_filetype)  
  
  
 settings.addAction(filetype)  
  
 def changes(self):  
 icon = QtGui.QIcon()  
 icon.addPixmap(QtGui.QPixmap("images/dict.jpg"), QtGui.QIcon.Normal, QtGui.QIcon.Off)  
 self.project.setIcon(icon)  
 self.output.setIcon(icon)  
  
 def config(self):  
 self.progressBar.setVisible(False)  
 if os.path.exists('cache/project.path'):  
 with open('cache/project.path', 'r') as file:  
  
 self.pro\_dict.insert(file.read())  
 file.close()  
  
 if os.path.exists('cache/output.path'):  
 with open('cache/output.path', 'r') as file:  
  
 self.output\_dict.insert(file.read())  
 file.close()  
  
 def set\_events(self):  
 self.project.clicked.connect(self.choose\_pro\_dir)  
 self.output.clicked.connect(self.choose\_output\_dir)  
 self.reset.clicked.connect(self.dialog\_reset)  
 self.generate.clicked.connect(self.generate\_docx)  
  
 def choose\_pro\_dir(self):  
 self.progressBar.setVisible(False)  
 self.pro\_dict.clear()  
 pro\_path = QFileDialog.getExistingDirectory(self, '请选择项目路径', './')  
 self.pro\_dict.insert(pro\_path)  
 self.write\_to\_file('project', 'path', pro\_path)  
  
 def choose\_output\_dir(self):  
 self.progressBar.setVisible(False)  
 self.output\_dict.clear()  
 output\_path = QFileDialog.getExistingDirectory(self, '请选择生成位置', './')  
 self.output\_dict.insert(output\_path)  
 self.write\_to\_file('output','path',output\_path)  
  
 def get\_filetype(self):  
 status = os.path.exists('cache/file.tp')  
 if status:  
 str = self.read\_file('cache/file.tp')  
 filetype = QInputDialog.getText(self, "文件类型","请输入文件类型(.\*)以分号隔开", QLineEdit.Normal,str)  
 else:  
 filetype = QInputDialog.getText(self, "文件类型", "请输入文件类型(.\*)以分号隔开", QLineEdit.Normal )  
 if filetype[1]:  
 self.write\_to\_file('file', 'tp', filetype[0])  
  
 def dialog\_reset(self):  
 self.pro\_dict.clear()  
 self.output\_dict.clear()  
 self.progressBar.setVisible(False)  
  
 def visitDir(self, path):  
 list\_all = os.listdir(path)  
  
 for item in list\_all:  
 name = os.path.join(path, item)  
 if not os.path.isfile(name):  
 self.visitDir(name)  
 else:  
 if name != os.getcwd() +'\\'+ os.path.basename(\_\_file\_\_):  
 # print(name)  
 self.files\_list.append(name)  
 return self.files\_list  
  
  
 def read\_file(self, path):  
 str = ''  
  
 if os.path.exists(path):  
 with open(path, 'r', encoding='utf-8') as file:  
 str = file.read()  
 file.close()  
  
 return str  
 def write\_to\_file(self,filename,type,text):  
 with open('cache/{0}.{1}'.format(filename,type), 'w+') as file:  
 file.write(text)  
 file.close()  
  
 def render\_txt(self):  
 # imp\_word=[]  
 pass  
  
 def render\_img(self):  
 pass  
  
 def generate\_docx(self):  
 # 1  
 self.file\_path = self.pro\_dict.text()  
 self.progressBar.setValue(1)  
 # 2  
 self.gen\_path = self.output\_dict.text()  
  
 # 3  
 if os.path.exists('cache/file.tp'):  
 with open('cache/file.tp', 'r') as file:  
 temp = file.read()  
 file.close()  
 # 4  
 file\_type = temp.split(';')  
 # 5  
 file\_type\_ok = True  
 if len(file\_type) == 0:  
 QMessageBox.warning(self, '错误', '文件类型不能为空', QMessageBox.Yes)  
 return  
 # 6  
 for item in file\_type:  
 if not item.startswith('.'):  
 QMessageBox.warning(self, '错误', '文件类型错误', QMessageBox.Yes)  
 file\_type\_ok=False  
 # 7  
 if not file\_type\_ok:  
 return  
  
 # 8  
 if os.path.exists(self.file\_path) and os.path.exists(self.gen\_path):  
 self.progressBar.setVisible(True)  
 # 3  
 self.files\_list = self.visitDir(self.file\_path)  
 file\_type.append('Makefile')  
 for item in self.files\_list:  
 filename = os.path.splitext(item)[0]  
 suffix = os.path.splitext(item)[1]  
 head = filename.split('\\')[-1]  
 print(head)  
  
 if suffix in file\_type:  
 data=self.read\_file(item)  
 self.document.add\_heading(head+suffix)  
 self.document.add\_paragraph(data)  
 if head == 'Makefile':  
 data = self.read\_file(item)  
 self.document.add\_heading(head)  
 self.document.add\_paragraph(data)  
 value = 0  
 for i in range(8):  
 time.sleep(0.05)  
 value += 12.5  
 self.progressBar.setValue(value)  
 print(self.gen\_path)  
 self.document.save('{0}/{1}.docx'.format(self.gen\_path, self.file\_path.split('/')[-1]))  
 else:  
 QMessageBox.warning(self, '错误', '路径不能为空', QMessageBox.Yes)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 win = UserWidgets()  
 win.setWindowTitle('代码文档生成器')  
 win.show()  
 sys.exit(app.exec\_())

# GUI.py

# -\*- coding: utf-8 -\*-  
  
# Form implementation generated from reading ui file 'GUI.ui'  
#  
# Created by: PyQt5 UI code generator 5.13.0  
#  
# WARNING! All changes made in this file will be lost!  
  
  
from PyQt5 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_Dialog(object):  
 def setupUi(self, Dialog):  
 Dialog.setObjectName("Dialog")  
 Dialog.resize(520, 164)  
 self.reset = QtWidgets.QPushButton(Dialog)  
 self.reset.setGeometry(QtCore.QRect(410, 40, 91, 41))  
 self.reset.setObjectName("reset")  
 self.label = QtWidgets.QLabel(Dialog)  
 self.label.setGeometry(QtCore.QRect(20, 40, 61, 16))  
 self.label.setObjectName("label")  
 self.label\_2 = QtWidgets.QLabel(Dialog)  
 self.label\_2.setGeometry(QtCore.QRect(10, 80, 71, 20))  
 self.label\_2.setObjectName("label\_2")  
 self.pro\_dict = QtWidgets.QLineEdit(Dialog)  
 self.pro\_dict.setGeometry(QtCore.QRect(90, 40, 271, 21))  
 self.pro\_dict.setObjectName("pro\_dict")  
 self.output\_dict = QtWidgets.QLineEdit(Dialog)  
 self.output\_dict.setGeometry(QtCore.QRect(90, 80, 271, 20))  
 self.output\_dict.setObjectName("output\_dict")  
 self.generate = QtWidgets.QPushButton(Dialog)  
 self.generate.setGeometry(QtCore.QRect(410, 110, 91, 41))  
 self.generate.setObjectName("generate")  
 self.project = QtWidgets.QPushButton(Dialog)  
 self.project.setGeometry(QtCore.QRect(360, 40, 21, 21))  
 self.project.setText("")  
 icon = QtGui.QIcon()  
 icon.addPixmap(QtGui.QPixmap("C:/Users/JessenHua/Pictures/dict.jpg"), QtGui.QIcon.Normal, QtGui.QIcon.Off)  
 self.project.setIcon(icon)  
 self.project.setObjectName("project")  
 self.output = QtWidgets.QPushButton(Dialog)  
 self.output.setGeometry(QtCore.QRect(360, 80, 21, 21))  
 self.output.setText("")  
 self.output.setIcon(icon)  
 self.output.setObjectName("output")  
 self.progressBar = QtWidgets.QProgressBar(Dialog)  
 self.progressBar.setEnabled(True)  
 self.progressBar.setGeometry(QtCore.QRect(110, 130, 251, 23))  
 self.progressBar.setProperty("value", 0)  
 self.progressBar.setTextVisible(True)  
 self.progressBar.setObjectName("progressBar")  
  
 self.retranslateUi(Dialog)  
 QtCore.QMetaObject.connectSlotsByName(Dialog)  
  
 def retranslateUi(self, Dialog):  
 \_translate = QtCore.QCoreApplication.translate  
 Dialog.setWindowTitle(\_translate("Dialog", "文档生成器"))  
 self.reset.setText(\_translate("Dialog", "重置"))  
 self.label.setText(\_translate("Dialog", "项目文件夹"))  
 self.label\_2.setText(\_translate("Dialog", "文档生成位置"))  
 self.generate.setText(\_translate("Dialog", "生成"))