IM Trust - Credicorp Capital
Administradora General de Fondos
Mesa de Gestión de Inversiones
Diego Steinsapir

9 de enero de 2017

Generación de Excel para Reporte de Seguimiento

1. Introducción

Este documento detalla paso a paso cómo se compiló la versión actual de Generador.exe. Su compilación es necesaria para poder distrubuir el programa de manera sencilla sin la necesidad de instalar librerías ni programas de forma que su uso sea lo más simple posible. El método de compilación aquí plasmado no es el único posible y cualquier otra forma de generar el archivo ejecutable a partir del código original es igualmente válida. Sin embargo, debido a las dificultades que hubo a la hora de compilar, este manual puede ser de gran ayuda para quien desee generar una nueva versión del archivo ejecutable.

2. Materiales

Para la compilación se necesita cumplir con las siguientes condiciones:

- Equipo corriendo Windows 10 (Con Windows 7 la librería no funcionó)
- Python 3.5.2 instalado
- PyInstaller Versión 3.2 para Python 3.5
- Todas las dependencias de 'Libreria_Fdo' instaladas
- Código Fuente

IMPORTANTE: Si se hace la compilación con Anaconda instalado (y probablemente con otros frameworks también), el compilador incluye muchas librerías innecesarias y hacen más peasdo el programa. Además, resulta más complicado hacerlo funcionar, pero eventualmente se debiese poder de igual manera.

3. Compilación Paso a Paso

1. En generador.py, borrar la siguiente linea:

   ```
   15        sys.path.insert(0, '../libreria/')
   ```

2. En carpeta generador_presentacion_gestionactivos (o donde se encuentre el código fuente)crear una copia de 'Libreria_Fdo' [1]

3. En la copia generada de 'Libreria_Fdo', se debe borrar la siguiente linea:

   ```
   8         from pptx import Presentation
   ```

4. Abrir consola y navegar hasta directorio de generador.py.

[1] 05-01-2016: Este paso debiese cambiar cuando se afine el proceso de compilación

5. En consola escribir el siguiente comando:

```
pyinstaller --onefile .\generador.py
```

6. Si no hay problemas, PyInstaller ejecutará hasta generar dos carpetas y un archivo llamado "generador.spec." en la misma carpeta.

7. Borrar carpetas 'build' y 'dist' generadas y abrir archivo 'generador.spec' y modificar la siguiente linea:

```
5       a = Analysis([ ...
                    hiddenimports=[],
                        ...
                        )
```

a:

```
5       a = Analysis([ ...
                  hiddenimports=['_mssql'],
                        ...
                        )
```

8. Después del objeto Analysis, agregar el siguiente código (sacado de esta pregunta)

```
def extra_datas(mydir):
    def rec_glob(p, files):
        import os
        import glob
        for d in glob.glob(p):
            if os.path.isfile(d):
                files.append(d)
            rec_glob("%s/*" %d, files)
    files = []
    rec_glob("%s/*" %mydir, files)
    extra_datas = []
    for f in files:
        extra_datas.append((f, f, 'DATA'))

    return extra_datas
###############################################

# append the 'data' dir
a.datas += extra_datas('queries_generador')
```

9. Ejecutar ahora en consola el siguiente comando:

```
pyinstaller --onefile .\generador.spec
```

Notar que se utiliza la terminacion '.spec' en lugar de '.py'

10. Terminada la ejecución, navegar por consola a carpeta dist y ejecutar el programa con el siguiente comando:

```
.\generador.exe
```

11. El programa intentará ejecutar. Si no ejecuta, el error más probable es del tipo ImportError: no module named 'XXXXX'", que implica que el compilador no utilizó esa dependencia al ejecutar y por lo tanto el programa no puede correr. En estos casos se debe modificar la misma sección de antes del archivo generador.spec de la siguiente manera:

```
a = Analysis([ ...
              hiddenimports=['_mssql'],
                  ...
                  )
```

a:

```
a = Analysis([ ...
              hiddenimports=['_mssql', 'XXXXX'],
                  ...
                  )
```

12. Borrar nuevamente carpetas "dist" "build". Repetir los últimos 3 pasos hasta que el programa ejecute de manera correcta

4. Anexo: generador.spec

A continuación, se adjunta un código de generador.spec generado por PyInstaller:

```
# -*- mode: python -*-

block_cipher = None

a = Analysis(['generador.py'],
             pathex=['C:\\Users\\diego\\Desktop\\IM Trust\\
                      generador_presentacion_gestionactivos'],
             binaries=None,
             datas=[],
             hiddenimports=['_mssql'],
             hookspath=[],
             runtime_hooks=[],
             excludes=[],
             win_no_prefer_redirects=False,
             win_private_assemblies=False,
             cipher=block_cipher)

def extra_datas(mydir):
   def rec_glob(p, files):
       import os
       import glob
       for d in glob.glob(p):
           if os.path.isfile(d):
               files.append(d)
           rec_glob("%s/*" %d, files)
   files = []
   rec_glob("%s/*" %mydir, files)
   extra_datas = []
   for f in files:
```

```
        extra_datas.append((f, f, 'DATA'))

    return extra_datas
##############################################

# append the 'data' dir
a.datas += extra_datas('queries_generador')

# a.datas += [("benchmarkName.sql", "C:\\Users\\diego\\Desktop
\\IM Trust\\generador_presentacion_gestionactivos\\queries_generador
\\benchmarkName.sql", "DATA")]

pyz = PYZ(a.pure, a.zipped_data,
             cipher=block_cipher)
exe = EXE(pyz,
          a.scripts,
          a.binaries,
          a.zipfiles,
          a.datas,
          name='generador',
          debug=False,
          strip=False,
          upx=True,
          console=True )
```