Artículo

[Luis Angel Pére...](#) · 2 ene, 2023 · Lectura de 5 min

[Open Exchange](#)

Plantilla de Python Embebido

¡Hola desarrolladores!

Comparto con vosotros una [plantilla de Python Embebido](#) básica, que recomiendo como punto de partida para cualquier proyecto general con InterSystems IRIS que use Python Embebido.

Funcionalidades:

- Python Embebido configurado;
- 3 ejemplos con difentes formas de desarrollar con Python Embebido;
- VSCode configurado;
- Docker disponible;
- Demo online disponible;
- Desarrollo ZPM First disponible.

Vamos a comentar estas funcionalidades.

Primero vamos a hablar de Python Embebido. Esta funcionalidad viene con InterSystems IRIS 2021.2 y permite desarrollar soluciones usando InterSystems IRIS con Python. IRIS 2021.2 y posteriores permiten la ejecución de scripts de Python en un entorno de memoria compartida con InterSystems IRIS que ofrece opciones únicas a los desarrolladores Python al usar la bases de datos en la que el código está cerca de los datos.

3 formas de desarrollar con Python Embebido

Invocación a librerías de Python desde ObjectScript

Esto es posible por la clase [%SYS.Python](#), que permite importar librerías de python y llamar a Python a través de ObjectScript. [Documentación](#), [Ejemplo](#). Aquí podéis ver un ejemplo de código:

```
ClassMethod Today() As %Status
{
    Set sc = $$$OK
    Set dt = ##class(%SYS.Python).Import("datetime")
    write dt.date.today().isoformat()
    Return sc
}
```

Escribir métodos de clase de ObjectScript en Python

De hecho, ahora los desarrolladores pueden usar la etiqueta [Language=python] en la declaración del método y codificar en Python puro. Hay también una librería de ayuda en Python "iris" que se puede utilizar para hacer referencia a las clases y globals de ObjectScript. [Documentación](#), [Ejemplo](#), código de muestra:

```
ClassMethod CreateRecordPython(propValue As %VarString, ByRef id As %Integer) [ Language
```

```
age = python ]
{
    import iris
    obj=iris.cls(__name__)._New()
    obj.Test=propValue
    sc=obj._Save()
    id=obj._Id()
    return sc
}
```

Codificar las soluciones de InterSystems IRIS en Python puro

Esta es la tercera opción para que los desarrolladores puedan trabajar con IRIS. Aquí el script de Python necesita ser conectado a IRIS, y esto se puede hacer a través de variables de entorno y el servicio CallIn configurado a "On". Podéis ver los detalles más abajo. Una vez configurado el script de Python es ejecutado en la memoria compartida junto con IRIS. Aquí la librería "iris" es muy útil también. [Documentación](#), [Ejemplo](#).

```
def create_rec(var):
    obj=iris.cls('dc.python.PersistentClass')._New()
    obj.Test=var
    obj._Save()
    id=obj._Id()
    return id

# test record creation
from datetime import datetime
now=str(datetime.now())
print("Creating new record in dc.python.PersistentClass")
print(create_rec(now))

## run SQL and print data
def run_sql(query):
    rs=iris.sql.exec(query)
    for idx, row in enumerate(rs):
        print(f"[{idx}]: {row}")

query="Select * from dc_python.PersistentClass"
print("Running SQL query "+query)
run_sql(query)
```

Docker configurado

El repositorio de la plantilla ejecuta IRIS en un contenedor y configura todo lo necesario para trastear con Python Embebido.

Variables de entorno. Python Embebido necesita configurar ciertas variables de entorno para conectarse a IRIS y ejecutar los scripts de Python. Esta es la configuración que lo posibilita en [dockerfile](#):

```
# init Python env
ENV PYTHON_PATH=/usr/irissys/bin/irispthon
ENV SRC_PATH=/irisrun/repo
ENV IRISUSERNAME "SuperUser"
ENV IRISPASSWORD "SYS"
```

```
ENV IRISNAMESPACE "USER"
```

También, Python Embebido necesita el servicio CallIn "on". Esto se configura en el [iris.script](#) durante la fase del build del docker:

```
; enabling callin for Embedded Python
do ##class(Security.Services).Get("%Service_CallIn", .prop)
set prop("Enabled")=1
set prop("AutheEnabled")=48
do ##class(Security.Services).Modify("%Service_CallIn", .prop)
```

Además, tu solución puede necesitar algunas librerías de Python a mayores. Esto es proporcionado mediante el archivo [requirements.txt](#) en la raíz del repositorio y una llamada pip3 en el [dockerfile](#):

```
pip3 install -r requirements.txt && \
```

VSCoide configurado

Es muy recomendable desarrollar en VSCode usando docker. Si quieres desarrollar IRIS con Python Embebido en Docker, VSCode necesita ser configurado a [modo Devcontainer](#). Para ello, introduce el archivo [devcontainer.json file](#) en la carpeta .devcontainer folder. Describe qué servicio docker necesita para trabajar con (iris en nuestro caso) y esto ayuda a ejecutar los scripts de Python en VSCode que serán servidos por el motor de Python utilizado por IRIS que se está ejecutando en el contenedor. El archivo devcontainer.json también tiene una sección con una descripción de qué [extensiones](#) son necesarias en modo contenedor:

```
"extensions": [
    "ms-python.python",
    "ms-python.vscode-pylance",
    "intersystems-community.vscode-objectscript",
    "intersystems.language-server",
    "intersystems-community.servermanager",
    "ms-vscode.docker"
],
```

Instalar Python Embebido con ZPM

Esta plantilla está configurada como un repositorio de desarrollo "ZPM first". Esto significa que todo el código desarrollado está descrito en module.xml y es instalado como un módulo ZPM cada vez que la imagen del docker es desplegada por el desarrollador mediante la siguiente línea de código en iris.script:

```
zpm "load /home/irisowner/irisbuild/ -v":1:1
```

Y el código de Python Embebido también está descrito en el módulo ZPM - siendo instalado a través de [FILECOPY](#):

```
<FileCopy Name="python/" Target="${libdir}python/" />
```

Con esta expresión indicamos que queremos empaquetar todos los scripts de Python bajo la carpeta /python en el repositorio e instalarla en la carpeta python/ en un directorio de librerías de la instalación IRIS de destino. Si los scripts de Python son copiados bajo la carpeta \${libdir}python/, estarán disponibles para ser invocadas desde ObjectScript o desde Python en la máquina IRIS de destino.

¡Ojo! Asegúrate de que los nombres de tus carpeta tienen nombres únicos para los scripts de Python para no invalidar otro código Python accidentalmente.

Espero que esta plantilla os resulte útil. Comentarios y especialmente pull requests son bienvenidos :D

[#Embedded Python](#) [#Entorno de desarrollo](#) [#InterSystems IRIS](#) [#VSCode](#)
[Ir a la aplicación en InterSystems Open Exchange](#)

URL de fuente: <https://es.community.intersystems.com/post/plantilla-de-python-embebido>