Artículo
Muhammad Waseem · Abr 18 Lectura de 2 min

Python Embebido, usando parámetros de salida

Antecedentes

En las versiones de InterSystems IRIS >=2021.2 podemos usar irispython para escribir directamente código python encima de nuestras instancias IRIS. Esto nos permite usar paquetes de python, llamar a métodos, hacer consultas SQL y hacer casi cualquier cosa en Objectscript excepto pythonic.

Por ejemplo, a continuación compruebo si hay un *namespace*:

```
#!/usr/irissys/bin/irispython
import iris
# call arbitrary class methods
result = iris.cls('%SYS.Namespace').Exists('USER')
if result == 1:
    print(f"Namespace USER is present")
```

Pero, ¿ qué pasa si mi método en IRIS tiene parámetros especiales como Output y ByRef?
¿ Cómo podemos usar los parámetros de Salida en irispython?

Por ejemplo, Ens.Director tiene muchos parámetros de salida en sus métodos, ¿ cómo puedo usar estos parámetros en python?

Ens.Director:GetProductionStatus

```
// Un código auxiliar de método de ejemplo de Ens.Director
ClassMethod GetProductionStatus(Output pProductionName As %String, Output pState As %
Integer...
```

Probando los métodos de variables normales

A primera vista, puedes intentar lo siguiente:

```
import os
# Establecer el namespace de forma manual
os.environ['IRISNAMESPACE'] = 'DEMONSTRATION'

import iris

# PRUEBA 1 con variables de salida
productionName, productionState = None, None
status = iris.cls('Ens.Director').GetProductionStatus(productionName, productionState
)
```

```
print("Success? -- {}".format(productionState != None))
```

¡ Pero ambas pruebas no devolverán las variables de salida! Puedes probar esto por ti mismo en cualquier namespace de Ensemble

Usando iris.ref

La utilidad irispython iris.ref se puede utilizar para capturar las variables Output y ByRef.

1. Crea un objeto iris.ref()
2. Llama a tu Método ObjectScript
3. Utiliza la variable .value para obtener el resultado de ese parámetro

```
import os
# Establecer el namespace de la manera difícil
os.environ['IRISNAMESPACE'] = 'DEMONSTRATION'

import iris

# PRUEBA 2 con variables de salida
productionName, productionState = iris.ref('productionName'), iris.ref('productionSta
te')
status = iris.cls('Ens.Director').GetProductionStatus(productionName, productionState
)

print("Status: {}".format(status))
# see .value
print("Production: {}".format(productionName.value))
# see .value
print("Production State: {}".format(productionState.value))
```

code > text-fix.py > ...

```python
# Python
import os
# Set namespace the hard way
os.environ['IRISNAMESPACE'] = 'DEMONSTRATION'
import iris

# TEST 2 with output variables
productionName, productionState = iris.ref('productionName'), iris.ref('productionState')
status = iris.cls('Ens.Director').GetProductionStatus(productionName, productionState)

print("Status: {}".format(status))
# see .value
print("Production: {}".format(productionName.value))
# see .value
print("Production State: {}".format(productionState.value))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
$ /usr/irissys/bin/irispython /home/irisowner/code/text-fix.py
Status: 1
Production: Hospital.HospitalProduction
Production State: 1
$
```

#Python #Ensemble #InterSystems IRIS

URL de fuente: https://es.community.intersystems.com/post/python-embebido-usando-par%C3%A1metros-de-salida