Artigo
Danusa Calixto · Set. 14, 2022 3min de leitura

Chamando classmethods com a API Native para o Python

O SDK Nativo para Python da InterSystems é uma interface leve de APIs do InterSystems IRIS que antes estavam disponíveis somente por ObjectScript.

Estou especialmente interessado na capacidade de chamar métodos ObjectScript ou class methods, para ser preciso. Funciona muito bem, mas, por padrão, as chamadas só são compatíveis com argumentos escalares: strings, booleanos, inteiros e floats.

No entanto, se você quiser:
- Transmitir ou retornar estruturas, como dicionários ou listas
- Transmitir ou retornar streams

Você precisará escrever glue code ou usar este projeto (instalação com pip install edpy). O pacote edpy fornece uma simples assinatura:

```
call(iris, class_name, method_name, args)
```

que permite chamar qualquer método ObjectScript e receber resultados de volta.

Faça a importação assim:

```
from edpy import iris
```

call aceita 4 argumentos obrigatórios:
- iris — uma referência a um objeto IRIS estabelecido
- classname — classe IRIS para chamar
- methodname — método IRIS para chamar
- args — lista de 0 ou mais argumentos

Argumentos

Cada argumento pode ser um destes:

- string (qualquer comprimento, se for maior do que $$$MaxStringLength or 3641144 símbolos, é automaticamente convertida em um stream)
- booleano
- inteiro
- float
- dict (convertido em um objeto dinâmico)
- lista ou tupla (convertidas em array dinâmica)

os argumentos dicionário, lista e tupla podem conter recursivamente outros dicionários, listas e tuplas (enquanto houver memória).

Valor de retorno

Em retorno, esperamos um objeto/array dinâmico ou um stream/string JSON. Nesse caso, edpy primeiro converteria em uma string em Python e, se possível, interpretaria como um dicionário ou uma lista em Python. Caso contrário, o resultado seria retornado ao autor da chamada da mesma maneira.

É basicamente isso, mas veja alguns exemplos de métodos ObjectScript e como chamá-los usando essa função em Python.

Exemplo 1: Pong

```
ClassMethod Test(arg1, arg2, arg3) As %DynamicArray
{
    return [(arg1), (arg2), (arg3)]
}
```

Chame com:

```
>>> iris.call(iris_native, "User.Py", "Test", [1, 1.2, "ABC"])
[1, 1.2, 'ABC']
```

Nenhuma surpresa aqui. Os argumentos são colocados de volta em uma array e retornados ao autor da chamada.

Exemplo 2: Propriedades

```
ClassMethod Test2(arg As %DynamicObject) As %String
{
    return arg.Prop
}
```

Chame desta maneira:

```
>>> iris.call(iris_native, "User.Py", "Test2", [{"Prop":123}])
123
```

Agora para uma invocação mais incorporada:

```
>>> iris.call(iris_native, "User.Py", "Test2", [{"Prop":{"Prop2":123}}])
{'Prop2': 123}
```

Se uma propriedade for muito longa, também não tem problema — os streams serão usados para enviá-la ao IRIS e/ou de volta:

```
ret = iris.call(iris_native, "User.Py", "Test2", [{"Prop":"A" * 10000000}])
>>> len(ret)
10000000
```

Se você precisar de streams garantidos no lado do InterSystems IRIS, você pode usar %Get:

```
set stream = arg.%Get("Prop",,"stream")
```

Se o stream for codificado em base64, você pode decodificá-lo com:

```
set stream = arg.%Get("Prop",,"stream<base64")
```

Exemplo 3: String ou Stream

```
ClassMethod Test3(arg As %Stream.Object) As %String
{
    set file = ##class(%Stream.FileCharacter).%New()
    set file.TranslateTable = "UTF8"
    set filename = ##class(%File).ManagerDirectory() _ "test.txt"
    do file.LinkToFile(filename)
    if $isObject(arg) {
        set sc = file.CopyFromAndSave(arg)
    } else {
        do file.Write(arg)
        set sc = file.%Save()
    }
    if $$$ISERR(sc) {
        set jsonret = {"status":0, "payload":($system.Status.GetErrorText(sc))}
    } else {
        set jsonret = {"status":1}
    }
    quit jsonret.%ToJSON()
}
```

Aqui escrevemos uma string ou um stream para <mgr>test.txt.

```
>>> iris.call(iris_native, "User.Py", "Test3", ["&#x1f60a;"])
{'status': 1}
```

Observação: em todas as amostras de código, "&# x1f642;" é inserido como .

E se eu abrir o arquivo e ver um e não dois ?? — então, a codificação é preservada.

```
>>> iris.call(iris_native, "User.Py", "Test3", ["&#x1f642;" * 10000000])
{'status': 1}
```

Omitirei o resultado do arquivo para ser breve, mas ele existe.

Por fim, ao transmitir uma array ou um objeto dinâmico dentro, você consegue evitar completamente a dicotomia string/stream, mesmo se não souber se a propriedade seria mais curta ou longa do que o limite da string. Nesse caso, você sempre pode obter a propriedade suspeita como um stream.

Exemplo 4: Retornar streams

```
ClassMethod Test4(arg As %DynamicArray) As %String
{
    return arg.%Get(0)
}
```

Veja como isso funciona:

```
>>> ret = iris.call(iris_native, "User.Py", "Test4", [["&#x1f60a;" * 10000000]])
>>> len(ret)
10000000
>>> ret[:5]
'&#x1f60a;&#x1f60a;&#x1f60a;&#x1f60a;&#x1f60a;'
```

Mais uma coisa

Também há uma função getiris(ip="localhost", port=1972, namespace="USER", username="SYSTEM", password="SYS") para obter um objeto IRIS que funciona. Veja um exemplo completo, se quiser tentar por conta própria:

Primeiro, carregue a classe User.Py e instale a biblioteca Python edpy:

```
pip install edpy
```

Em seguida, na chamada do Python:

```
from edpy import iris
iris_native = iris.get_iris()
iris.call(iris_native, "User.Py", "Test",  [1, 1.2, "ABC"])
iris.call(iris_native, "User.Py", "Test2", [{"Prop":123}])
iris.call(iris_native, "User.Py", "Test2", [{"Prop":{"Prop2":123}}])
ret2 = iris.call(iris_native, "User.Py", "Test2", [{"Prop":"A" * 10000000}])
iris.call(iris_native, "User.Py", "Test3", ["&#x1f60a;"])
iris.call(iris_native, "User.Py", "Test3", ["&#x1f60a;" * 10000000])
ret4 = iris.call(iris_native, "User.Py", "Test4", [["&#x1f60a;" * 10000000]])
```

Conclusão

O SDK Nativo para Python é uma ferramenta poderosa, que fornece acesso completo e irrestrito ao InterSystems IRIS. Com esse projeto, espero que seja possível poupar tempo com o empacotamento das chamadas do InterSystems IRIS. Alguma combinação de argumentos de métodos não é compatível? Se não for, compartilhe nos comentários como você faz a chamada.

Links

- Documentos do SDK Nativo
- Classe User.Py
- Repositório

#Python #InterSystems IRIS

URL de origem:https://pt.community.intersystems.com/post/chamando-classmethods-com-api-native-para-o-python