Artículo
Luis Angel Pére... · 16 ene, 2023 Lectura de 4 min

Open Exchange

Cómo serializar objetos Python en globals

Motivación

Empecé en este proyecto pensando en cómo permitir que el código Python trabaje de forma natural con el almacenamiento escalable y el eficiente mecanismo de recuperación de datos ofrecido por los *globals* de IRIS, a través de Python Embebido.

Mi idea inicial era crear algo como un diccionario de implementación de Python usando *globals*, pero pronto me di cuenta de que antes debía ocuparme de la abstracción de objetos.

Así que empecé creando algunas clases en Python que pudieran envolver objetos Python, almacenando y recuperando sus datos en *globals*, es decir, serializar y deserializar objetos Python en *globals* IRIS.

¿Cómo funciona?

Como ObjectScript %DispatchGetProperty(), %DispatchSetProperty() y %DispatchMethod(), Python tiene métodos para delegar las propiedades de los objetos y las llamadas a los métodos.

Cuando haces set o get sobre una propiedad del objeto, el intérprete de Python te permite interceptar esta operación a través de métodos setattr(self, name, value) y getattr(self, name).

Podemos comprobarlo con este sencillo ejemplo:

```
>>> class Test:
...         def __init(self, prop1):
...                 self.prop1 = prop1
...         def __setattr__(self, name, value):
...                 print(f"setting property {name} to value {value}")
...         def __getattr__(self, name):
...                 print(f"getting property {name}")
...
>>> obj = Test()
>>> obj.prop1 = "test"
setting property prop1 to value test
>>> obj.prop1
getting property prop1
>>> obj.prop2
getting property prop2
>>> obj.prop2
getting property prop2
>>>
```

Ten en cuenta que los métodos setattr() y getattr() fueron llamados indirectamente por las operaciones set y get sobre objetos de la clase Test - que implementa esos métodos. Incluso una propiedad no-declarada, como prop2 en este ejemplo, lanza llamadas hacia ellos.

Ese mecanismo es el nucleo de la prueba de serialización que realicé en mi proyecto python-globals-serializer-example. Con él, se pueden interceptar operaciones set/get y almacenar/recuperar datos desde los *globals* de IRIS.

Modelo de objetos

Los *globals* ofrecen una estructura altamente personalizable. Al usar su modelo jerárquico para acceder a la información, que es bastante similar a los objetos JSON y diccionarios Python, podemos almacenar los datos y metadatos de las propiedades de los objetos.

Así es como usé los *globals* para crear un modelo de objetos sencillo para serializar objetos Python:

Para cada objeto serializado, su nombre de clase es serializado en un nodo etiquetado como " clase " :

```
^test(1,"class")="<class 'employee.SalaryEmployee'>"
```

El primer índice es un número incremental que se usa como referencia de este objeto en el modelo. Así que, en el ejemplo anterior, un objeto de clase employee.SalaryEmployee es almacenado con el valor de referencia 1.

Para propiedades de tipos de datos primitivos, se almacenan su tipo y valor. Por ejemplo:

```
^test(1,"name","type")="<class 'str'>"
^test(1,"name","value")="me"
```

Esa estructura es interpretada como el objeto referido por el índice 1, tiene una propiedad llamada name, con valor igual a ' me ' .

Para referenciar propiedades de tipo objeto el modelo es ligeramente diferente, porque a diferencia de los objetos JSON o los diccionarios Python, los *globals* están pensados para almacenar solo tipos de datos primitivos. Así que se crea otro nodo " class " para este objeto referenciado y su nodo índice (es decir, su referencia) es almacenado en el nodo propiedad:

```
^test(1,"company","oref")=2
^test(1,"company","type")="<class 'iris_global_object.IrisGlobalObject'>"

^test(2,"class")="<class 'employee.Company'>"
^test(2,"name","type")="<class 'str'>"
^test(2,"name","value")="Company ABC"
```

Esas estructuras significan que el objeto 1 tiene una propiedad llamada company, cuyos valores se almacenan en el índice 2 - ten en cuenta el valor para ^test(1,"company","oref").

Proceso de Serialización/Deserialización

Cuando creas un envoltorio para serializar o deserializar objetos Python, necesitas definir el nombre del *global* que almacena el objeto.

Tras ello, el proceso de serialización se completa cuando se ejecuta una operación set. El método setattr()

establece el valor y tipo de propiedad en el *global* definido para almacenar el objeto, usando el modelo de objeto simple explicado anteriormente.

En la dirección contraria, una deserialización es realizada por el método getattr, cuando se lleva a cabo una operación get

Para tipos de datos primitivos, este proceso es sencillo: solo coge el valor almacenado en el *global* y lo devuelve.

Pero para objetos, el proceso necesita instanciar sus tipos de datos de clase y definir todas sus propiedades también, y de esta manera se podría usar un objeto Python recuperado, incluyendo llamadas a sus métodos.

Trabajo futuro

Como comenté al principio de este artículo, este proyecto nació como una simplificación de una manera de permitir que el código Python use el *global* como un motor de almacenamiento natural, y solo pretende ser una prueba de concepto.

Serializar/deserializar objetos es solo el primer paso para alcanzar este objetivo. Hace falta mucho trabajo para que esta idea madure.

Espero que el artículo os permita entender el propósito de mi trabajo en este proyecto, y puede que os inspire a pensar en nuevas formas de usar los *globals* de IRIS para que Python esté aún más cerca de IRIS.

#Embedded Python #Globals #Modelo de datos #Python #InterSystems IRIS #Open Exchange
Ir a la aplicación en InterSystems Open Exchange