文章
Hao Ma · 六月 4, 2021 阅读大约需 3 分钟

JSON in IRIS (1) - Dynamic Object

之所以称为Dynamic，是说这个对象在代码编译的时候可以不定义对象的属性和结构，在runtime时才根据装入的数据来产生对象定义。IRIS里用Dynamic Object来处理JSON数据。简单说: 先定义一个Dynamic Object, 把JSON数据装进去，然后用对象的方式处理JSON文档。

让我们看看是它是怎么工作的。

创建一个Dynamic Object很简单, 标准而且啰嗦的写法是：

```
set dynObject1 = ##class(%DynamicObject).%New()
```

大家通常用简单的写法，像这样用一个{}来定义Dynamic Object:

```
DEMO>set dynObject1 = {}
DEMO>zw dynObject1
dynObject1={}  ; <DYNAMIC OBJECT>
```

字符串，流到DynamicObject的导入导出

把JSON数据从字符串或者流导入DynamicObject被称作Deserializing；反之，把DynamicObject里的JSON导出来到String或者Stream叫Serializing。在类%DynmicObject中用的是%FromJSON()和%ToJSON()两个方法，一个是类方法，一个是实例方法：

```
//?????????????%DynmicObject?str???????stream, ??????URL
ClassMethod %FromJSON(str)

//?%DynmicObject??????(??)????(outStrm);????????method???
Method %ToJSON(outStrm As %Stream.Object) As %String
```

实际使用的例子是这样：

```
DEMO>set str="{""aNumber"":42,""aDate"":""2021-06-04""}"
DEMO>set dynObject2=##class(%DynamicObject).%FromJSON(str)
DEMO>zw dynObject2
dynObject2={"aNumber":42,"aDate":"2021-06-04"}  ; <DYNAMIC OBJECT>
```

上面的例子很傻。既然字符串是定义好的，就没必要先赋值给一个变量，直接用下面的代码就创建了一个Dynamic Object, 省掉了赋值字符串变量里用到的来escape符号"。

```
set dynObject2 = {"aNumber":42,"aDate":"2021-06-04"}
```

接着， 我们来看看真正体现Dynamic的对象定义。下面的代码中传入一个String消息，那么就需要用%FromJSON()来把JSON装入DynamicObject

```
method example1(pInput As %Ens.StringRequest) As %Status
{   //?????JSON?????DynamicObject
    set dynObject2=##class(%DynamicObject).%FromJSON(pInput.StringValue)

    //?DynamicObject??????
    set myString= dynObject2.%ToJSON()
    //?DynamicObject????
    set myStream=##class(%Stream.GlobalCharacter).%New()
    do dynObject2.%ToJSON(myStream)

    //????
    w dynObject2.%ClassName(),!
    w mystring, !
    w myStream.Read(),!
}
```

以上代码的执行结果是：

```
%DynamicObject
{"aNumber":42,"aDate":"2021-06-04"}
{"aNumber":42,"aDate":"2021-06-04"}
```

用DynamicObject处理JSON

把JSON数据装入DynamicObject是为了方便用对象的方式处理JSON，比如取值：

```
w dynObject2.aName,!
w dynObject.aDate,!
```

或者，给对象的属性赋值

```
set dynObject2.aNumber=43
set dynObject2.aDate=$ZD($H)
```

或者，也可以使用%Get(), %Set()方法，根据key,得到或者修改value,比如

```
set tempVariable = dynObject2.%Get("aNumber")
do dynObject2.%Set("aNumber",44)
```

关于%Get(),%Set()还有些复杂的用法，请参考在线文档。

下面是最重要的， 遍历整个DynamicObject

```
set itr = dynObject2.%GetIterator()
while itr.%GetNext(.key,.val)
```

```
{
    write key_", "
    write val_", "
    write dynObject2.%GetTypeOf(key),!
}
```

其他的方法包括： %Remove(), %IsDefined(), 请自己查阅文档。

DynamicObject的错误处理

Dynamic Object不使用的%Status来做错误处理，因此我们必须用try-catch来捕捉错误，比如下面的代码：

```
TRY {
 set invalidObject = {}.%FromJSON("{:}")
 }
 CATCH errobj {
 write errobj.Name_", "  errobj.Location", error code "_errobj.Code,!
 RETURN
 }
```

其他的几点注意

有了上面的知识，已经足以在代码中处理JSON文档了。还有几点请注意：

- %Library.DynamicArray

和DynamicObject的区别是这是一个Array, 用法很类似，请各位自己阅读文档了解。

- Dynamic Object可以嵌套， 比如这样
  set objectStr = {"a":12,"b":"some string","c":{"x":1,"y":2}}
- 定义Dynamic Object时可以使用ObjectScirpt表达式，比如

```
set dynObj = { "Date":($ZD($H,3)) }
```

下篇文章我会介绍%JSON.Adapter

#JSON #新手 #InterSystems IRIS

源 URL:https://cn.community.intersystems.com/post/json-iris-1-dynamic-object