

Artículo

[Robert Cemper](#) · 7 feb, 2022 Lectura de 1 min[Open Exchange](#)

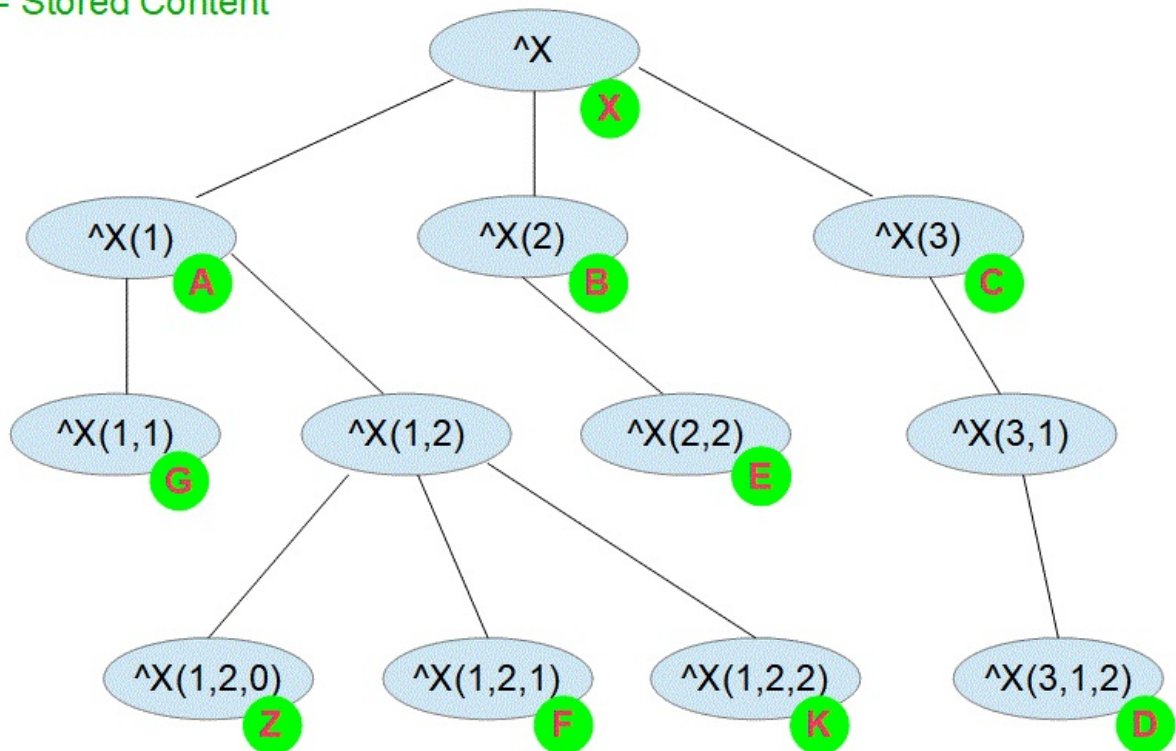
## GlobalToJSON-embedded-Python

Este es un paquete para exportar un Global a un archivo de objeto JSON y volver a crearlo recargando desde este archivo

embeddedPython se refiere a las nuevas tecnologías disponibles. Debe entenderse como un ejercicio de aprendizaje de

cómo manejar las diferentes interfaces. Solo los nodos de Global que contienen datos se presentan en el archivo JSON generado.

 = Stored Content



Exportamos este Global

## View global in namespace USER:

Global Search Mask: ^dc.MultiD		Display	Cancel
Search History: ^dc.MultiD		Maximum Rows: 100	<input type="checkbox"/> Allow Edit
1:	^dc.MultiD	= 5	
2:	^dc.MultiD(1)	= \$lb("Braam,Ted Q.",51353)	
3:	^dc.MultiD(1,"mJSON")	= "{}"	
4:	^dc.MultiD(2)	= \$lb("Klingman,Uma C.",62459)	
5:	^dc.MultiD(2,2,"Multi","a")	= 1	
6:	^dc.MultiD(2,2,"Multi","rob",1)	= "rcc"	
7:	^dc.MultiD(2,2,"Multi","rob",2)	= 2222	
8:	^dc.MultiD(2,"Multi","a")	= 1	
9:	^dc.MultiD(2,"Multi","rob",1)	= "rcc"	
10:	^dc.MultiD(2,"Multi","rob",2)	= 2222	
11:	^dc.MultiD(2,"mJSON")	= {"A":"ahahah","Rob":"VIP","Rob2":1111,"Rob3":true}	
12:	^dc.MultiD(3)	= \$lb("Goldman,Kenny H.",45831)	
13:	^dc.MultiD(3,"mJSON")	= "{}"	
14:	^dc.MultiD(4)	= \$lb("","")	
15:	^dc.MultiD(4,"mJSON")	= {"rcc":122}	
16:	^dc.MultiD(5)	= \$lb("","")	
17:	^dc.MultiD(5,"mJSON")	= "{}"	
Total: 17 [End of global]			

Este es el contenido del archivo.

```

1  [ "gbl": [
2    { "node": "^dc.MultiD", "val": "5" }
3    { "node": "^dc.MultiD(1)", "val": "$lb(\"Braam,Ted Q.\",51353)" }
4    { "node": "^dc.MultiD(1,\"mJSON\")", "val": "{}" }
5    { "node": "^dc.MultiD(2)", "val": "$lb(\"Klingman,Uma C.\",62459)" }
6    { "node": "^dc.MultiD(2,2,\"Multi\",\"a\")", "val": "1" }
7    { "node": "^dc.MultiD(2,2,\"Multi\",\"rob\",1)", "val": "rcc" }
8    { "node": "^dc.MultiD(2,2,\"Multi\",\"rob\",2)", "val": "2222" }
9    { "node": "^dc.MultiD(2,\"Multi\",\"a\")", "val": "1" }
10   { "node": "^dc.MultiD(2,\"Multi\",\"rob\",1)", "val": "rcc" }
11   { "node": "^dc.MultiD(2,\"Multi\",\"rob\",2)", "val": "2222" }
12   { "node": "^dc.MultiD(2,\"mJSON\")", "val": "{\"A\":\"ahahah\",\"Rob\":\"VIP\",\"Rob2\":1111,\"Rob3\":true}" }
13   { "node": "^dc.MultiD(3)", "val": "$lb(\"Goldman,Kenny H.\",45831)" }
14   { "node": "^dc.MultiD(3,\"mJSON\")", "val": "{}" }
15   { "node": "^dc.MultiD(4)", "val": "$lb(\"\",\"\")" }
16   { "node": "^dc.MultiD(4,\"mJSON\")", "val": "{\"rcc\":122}" }
17   { "node": "^dc.MultiD(5)", "val": "$lb(\"\",\"\")" }
18   { "node": "^dc.MultiD(5,\"mJSON\")", "val": "{}" }
19 ] ]

```

El Loader relacionado crea exactamente el mismo Global

El ejemplo es una clase en la que se mezclan InterSystems Object Script y Embedded Python.

Así que es un código híbrido. InterSystems Object Script se usa para iterar a través de Global con \$QUERY(). También escribe datos con una estructura \$LISTBUILD(). El objeto JSON corresponde así al modelo Eficiente.

Embedded Python se utiliza para escribir y leer el archivo de datos. Y también se genera el objeto JSON y su resolución se realiza con Embedded Python.

Una tarea especial es la conversión correcta de una estructura de subíndice convencional (sub,sub,sub,...) en una lista de Python [sub,sub,sub, ...]. Esta lista de Python es obligatoria para cualquier acceso directo a los nodos de Global.

¡Y al nodo superior del Global, que por definición no tiene subíndice, se accede con esta lista especial [None]!

Puede ser beneficioso mirar el código en detalle.

[Vídeo \(en inglés\)](#)

[#Globals #JSON #InterSystems IRIS](#)

[Ir a la aplicación en InterSystems Open Exchange](#)

---

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