Discussion

Aleksandar Kovacevic · Apr 26, 2020

Search within Globals (without class definitions)

So far, I found there are some interesting ways to search in global structure:

- through Query %Library.Global.Find
- through Query %Library.Global.Get

%Library.Global.Find

ObjectScript

```
 set statement=##class(%SQL.Statement).%New()
 set status=statement.%PrepareClassQuery("%Library.Global","Find")
 set resultset=statement.%Execute("USER","^Persons","Email")
// <Namespace>,<Global>,<Keyword>
 while resultset.%Next() {
   write !, resultset.%Get("Name"),", "
   write resultset.%Get("Value"),", "
   write resultset.%Get("Name Format"),", "
   write resultset.%Get("Value Format")
 }
```

Which would result in the following output:

```
^|"USER"|Persons(1,"Email","Home"), jsmith1234@gmail.com, 1, 1
^|"USER"|Persons(1,"Email","Work"), jsmith@somework.com, 1, 1
^|"USER"|Persons(2,"Email","Home"), mjones5678@email.com, 1, 1
^|"USER"|Persons(3,"Email","Home"), lstrait59@email.com, 1, 1
```

SQL

The same code can be directly used as a SQL call,

```
CALL %Library.Global_Find('USER','^Persons','Email')
```

Python

As such, this would also be able to be used in Python using JDBC,

```
import jaydebeapi
import pandas
import numpy
conn = jaydebeapi.connect(jdbc_javaDriverClass,
                          jdbc_link,
                          jdbc_auth,
                          jdbc_driver_jar,)
curs = conn.cursor()
query="CALL %Library.Global_Find('USER','^Persons','Email',1,1,1)"
curs.execute(query)
columns = [desc[0] for desc in curs.description] #getting column headers
df = pandas.DataFrame(curs.fetchall(),columns=columns)
curs.close()
conn.close()
```

Which would result in a nice Pandas DataFrame:

index	Name	Value	Name Format	Value Format
0	^¦"USER"¦Persons(1,"Email","Home")	jsmith1234@gmail.com	1	1
1	^¦"USER"¦Persons(1,"Email","Work")	jsmith@somework.com	1	1
2	^¦"USER"¦Persons(2,"Email","Home")	mjones5678@email.com	1	1
3	^¦"USER"¦Persons(3,"Email","Home")	lstrait59@email.com	1	1

%Library.Global.Get

The syntax is documented in the (InterSystems Reference API)[https://cedocs.intersystems.com/latest/csp/documatic/%25CSP.Documatic.cls?PAGE=CLASS&LIBRARY=%25SYS&CLASSNAME=%25Library.Global#AnchorQueries]

So what the most interesting part is that you can define search mask together with indices and at each specific level:

- To display a single node, use a complete global reference. For example: ^Sample.PersonD(9)
- To display a subtree, use a partial global reference without the right parenthesis. For example: ^%SYS("JOURNAL"
- To display all nodes that match a given subscript, include the desired subscript and leave other subscript fields empty.
    - For example: ^IRIS.Msg(,"en")
- To display all subtrees that match a given subscript, use a value as in the previous option but also omit the right parenthesis.
    - For example: ^IRIS.Msg(,"en"
- To display nodes that match a range of subscripts, use subscriptvalue1:subscriptvalue2 in the place of a subscript.
    - For example: ^Sample.PersonD(50:60)
- As with the previous option, if you omit the right parenthesis, the system displays the subtrees.

ObjectScript

```
set statement=##class(%SQL.Statement).%New()
set status=statement.%PrepareClassQuery("%Library.Global","Get")
set resultset=statement.%Execute("USER","^Persons(,""Email""")
  while resultset.%Next() {
    write !, resultset.%Get("Name"),", "
    write resultset.%Get("Value"),", "
    write resultset.%Get("Name Format"),", "
    write resultset.%Get("Value Format")
  }
```

which would result in:

```
^Persons(1,"Email","Home"), jsmith1234@gmail.com, 1, 1
^Persons(1,"Email","Work"), jsmith@somework.com, 1, 1
^Persons(2,"Email","Home"), mjones5678@email.com, 1, 1
^Persons(3,"Email","Home"), lstrait59@email.com, 1, 1
```

SQL

The same code can be directly used as a SQL call. Keep in mind that with InterSystems SQL; you should use single quote for strings instead of double-quotes.

```
CALL %Library.Global_Get('USER','^Persons(,"Email"','')
```

result:

Name	Value	Name Format	Value Format	Permissions
^Persons(1,"Email","Home")	jsmith1234@gmail.com	1	1	
^Persons(1,"Email","Work")	jsmith@somework.com	1	1	
^Persons(2,"Email","Home")	mjones5678@email.com	1	1	
^Persons(3,"Email","Home")	lstrait59@email.com	1	1	

Python

```
  import jaydebeapi
  import pandas
  import numpy
  conn = jaydebeapi.connect(jdbc_javaDriverClass,
                            jdbc_link,
                            jdbc_auth,
                            jdbc_driver_jar,)
  curs = conn.cursor()
  query="""
```

```
    CALL %Library.Global_Get('USER','^Persons(,"Email",)','',1,1,0)
  """
  curs.execute(query)
  columns = [desc[0] for desc in curs.description] #getting column headers
  df = pandas.DataFrame(curs.fetchall(),columns=columns)
  curs.close()
  conn.close()
```

You can notice there are some additional parameters at the end of the Get call. This is because when using JDBC, optional parameters are required.

Result:

index	Name	Value	Name Format	Value Format	Permissions
0	^Persons(1,"Email","Home")	jsmith1234@gmail.com	1	1	None
1	^Persons(1,"Email","Work")	jsmith@somework.com	1	1	None
2	^Persons(2,"Email","Home")	mjones5678@email.com	1	1	None
3	^Persons(3,"Email","Home")	lstrait59@email.com	1	1	None

Iterate through globals

$Query

Performs a physical scan of a local or global array.

```
Class Selector.Globals Extends %RegisteredObject
{
 ClassMethod Find(FindWhat As %String, Root As %String) {
  Set node = $Query(@Root)
  While (node '= "") {
   // write that depth-
first node with that containing values concatenated with (concatenate operator is _)
   // result of $locate function which tells 1 (true) or 0 (false) if regular express
ion matches string
   Write node_":"_$locate(node,FindWhat,,,tMatch),!
   // get next node
   Set node = $Query(@node)
   }
    }
}
```

When you use it,

```
 set a=##class(Selector.Globals).%New()
 do a.Find(".*\""Address\"".*","^Persons")
```

Note that the quotes are escaped in string with double quotes, so original regular expression for it is: .*"Address".*

This could be the way to search through globals using the regular expressions.

```
^Persons(1):0
^Persons(1,"Address","City"):1
^Persons(1,"Address","State"):1
^Persons(1,"Address","Street"):1
^Persons(1,"Address","Zip"):1
^Persons(1,"Email","Home"):0
^Persons(1,"Email","Work"):0
^Persons(1,"Name"):0
^Persons(2):0
^Persons(2,"Address"):1
^Persons(2,"Email","Home"):0
^Persons(2,"Name"):0
^Persons(2,"Phone",1,"Number"):0
^Persons(2,"Phone",1,"Type"):0
^Persons(2,"Phone",2,"Number"):0
^Persons(2,"Phone",2,"Type"):0
^Persons(3,"Address","City"):1
^Persons(3,"Address","State"):1
^Persons(3,"Address","Street"):1
^Persons(3,"Address","Zip"):1
^Persons(3,"CellPhone"):0
^Persons(3,"Email","Home"):0
^Persons(3,"Name"):0
^Persons(4,"Contact","Email","Home"):0
```

Appendix: Test Data for this example

ObjectScript

```
set ^Persons(1,"Name")="John Smith"
set ^Persons(1,"Email","Home")="jsmith1234@gmail.com"
set ^Persons(1,"Email","Work")="jsmith@somework.com"
set ^Persons(1,"Address","Street")="123 High St."
set ^Persons(1,"Address","City")="Cambridge"
set ^Persons(1,"Address","State")="MA"
set ^Persons(1,"Address","Zip")="02138"
set ^Persons(2,"Name")="Mary Jones"
set ^Persons(2,"Email","Home")="mjones5678@email.com"
set ^Persons(2,"Address")="67 Bennett Ave., Boston, MA 02111"
set ^Persons(2,"Phone",1,"Type")="Cell"
set ^Persons(2,"Phone",1,"Number")="333-333-3333"
set ^Persons(2,"Phone",2,"Type")="Business"
set ^Persons(2,"Phone",2,"Number")="111-111-1111"
set ^Persons(2,"Phone",2,"Type")="Home"
set ^Persons(2,"Phone",2,"Number")="555-555-5555"
set ^Persons(3,"Name")="Lena Strait"
set ^Persons(3,"Email","Home")="lstrait59@email.com"
set ^Persons(3,"Address","Street")="124 Main St."
set ^Persons(3,"Address","City")="Syracuse"
```

```
set ^Persons(3,"Address","State")="NY"
set ^Persons(3,"Address","Zip")="13211"
set ^Persons(3,"CellPhone")="444-444-4444"
```

Python

```
import irisnative
# create database connection and IRIS instance
connection = irisnative.createConnection("localhost",
        51773,
        "USER",
        "_SYSTEM",
        "SYS")
iris = irisnative.createIris(connection)
iris.set("John Smith","Persons","1","Name")
iris.set("jsmith1234@gmail.com","Persons","1","Email","Home")
iris.set("jsmith@somework.com","Persons","1","Email","Work")
iris.set("123 High St.","Persons","1","Address","Street")
iris.set("Cambridge","Persons","1","Address","City")
iris.set("MA","Persons","1","Address","State")
iris.set("02138","Persons","1","Address","Zip")
iris.set("Mary Jones","Persons","2","Name")
iris.set("mjones5678@email.com","Persons","2","Email","Home")
iris.set("67 Bennett Ave., Boston, MA 02111","Persons","2","Address")
iris.set("Cell","Persons","2","Phone","1","Type")
iris.set("333-333-3333","Persons","2","Phone","1","Number")
iris.set("Business","Persons","2","Phone","2","Type")
iris.set("111-111-1111","Persons","2","Phone","2","Number")
iris.set("Home","Persons","2","Phone","2","Type")
iris.set("555-555-5555","Persons","2","Phone","2","Number")
iris.set("Lena Strait","Persons","3","Name")
iris.set("lstrait59@email.com","Persons","3","Email","Home")
iris.set("124 Main St.","Persons","3","Address","Street")
iris.set("Syracuse","Persons","3","Address","City")
iris.set("NY","Persons","3","Address","State")
iris.set("13211","Persons","3","Address","Zip")
iris.set("444-444-4444","Persons","3","CellPhone")
connection.close()
```

Question

The question here is, are there any other ways to search for subscript/value within Global?
What would be other ways to do it?
How would the code look like if you want to optimize it with indexing (another global)?

#Globals #Caché