Question
Arturo Masero · Feb 17, 2022

Help with indirection and variable scope

Hello fellows, I need you wisdom.

In my organization we code in ObjectScript and .int everyday. When I joined nobody knew barely nothing about classes and their use was almost only for storage definition purpose. During my self-learning in caché I discovered the object oriented programming and class developing was possible in caché and started to code in .cls . Being something self-taught, I may have some basic doubts I'd have missed in my documentation readings.

With this scenario in mind, it is common that we use the @ operator to use wildcards for storing same data structures in different named tables, or playing with indexes. And relating to this I have found a problem in classes I don't know how to get rid off.

I'll try to put an example as generic and understandable as I can:

In OS .int we use this:

```
Indirection()
N (evenTotal)

set TABLES(0)="EVEN"
set TABLES(1)="ODD"

for i=1:1:100 {
   set table = TABLES((i#2)) set @(table_"("_i_")")=i
   // Xecute alternative, same issue
   ; set cmd = "set "_table_"("_i_")="_i
   ; x cmd
}

set evenTotal=0

set i="" for { set i=$O(EVEN(i)) Q:i=""
   set evenTotal = evenTotal+EVEN(i)
}
zw evenTotal
Q
```

It works and when finished no variables are in memory after execution (besides evenTotal, included in NEW).

But when I do this in .cls:

```
 ClassMethod Indirection()
{
set TABLES(0)="EVEN"
set TABLES(1)="ODD"
```

```
for i=1:1:100 {
   set table = TABLES((i#2))
   set @(table_"("_i_")")=i

   // Xecute alternative, same issue
   ; set cmd = "set "_table_"("_i_")="_i
   ; x cmd
}

set evenTotal=0
set i="" for { set i=$ORDER(EVEN(i)) QUIT:i=""
   set evenTotal = evenTotal+EVEN(i)
}
zwrite evenTotal

}
```

I get evenTotal zero. Inside the method I can zwrite EVEN and I see the values, but the ORDER goes empty and if I do 'write EVEN(2)' I get UNDEFINED, although I see the value though a zwrite. It's very confusing.

A side effect I get using @ is EVEN and ODD jumps in memory after the method execution. I don't want them in memory.

Playing with the PublicList keyword has not worked for me.

How can I make the 'ObjectScript .int' example work properly in .cls?

Nowadays I try to evade this practice in cls, and some rare times I add the variables to the publicList and try to KILL them before the end of the method. Has worked so far.

#ObjectScript #Caché
Product version: Caché 2018.1
$ZV: Cache for Windows (x86-64) 2018.1.2 (Build 309521269) Fri Nov 26 2021 11:37:39 EST