Article
Daniel Kutac · Jul 18, 2016 3m read

Caché tricks - capture write output to variable

Cache tricks

Several years ago, long before Developer Community Portal was launched, I published a series of Caché tricks at one of Czech web sites. In this article, I ’ m posting translated version of one of them.

Capturing output of someone else ’ s methods or routines

Suppose you, or someone else created a useful method or routine, that was producing some computation that you ’ d like to benefit from, but the routine was writing output to process principal device.

You would like to use the data, but you need it not written to a device, but assigned to a variable. And, for any reason, you can ’ t modify the code. What can you do?

Well, there is a solution to it. A bit tricky, but works very well.

Let ’ s take an example from not so far past. In Caché 2013 we introduced enhancement to ZEN framework, a class %ZEN.Auxiliary.jsonProvider, that allowed to pass JSON objects between client (browser) and server and vice versa. Unfortunately, the %ObjectToJSON() method was implemented in just that way - it was writing JSON serialized object representation to output device (the TCP / HTTP channel).

So, how to deal with the problem? There is something we call mnemonic space, these spaces are available with Caché IO devices. We can uses these mnemonic spaces to open IO devices. However, online documentation is very concise about mnemonic spaces and is not providing much practical details of use.

Let ’ s have a look at the following code, that demonstrates one use case.

```
        set pObject=##class(%ZEN.proxyObject).%New()        // any class can be here
        set pObject.error="my-error-1"
        set pObject.description="Error number one"

        Set tInitIO = $IO
        // we MUST use %ISCJSONStream variable used by mnemonic space ^%ZEN.Auxiliary
.jsonProvider.1 as output redirection container
        set %ISCJSONStream=##class(%Stream.TmpCharacter).%New()
        use tInitIO::("^%ZEN.Auxiliary.jsonProvider.1") // this routine – here as mne
monic space – is created during %ZEN.Auxiliary.jsonProvider class compilation
        do ##class(%Library.Device).ReDirectIO(1)
        $$$THROWONERROR(tSC,##class(%ZEN.Auxiliary.jsonProvider).%ObjectToJSON(pObjec
t))
        If ##class(%Library.Device).ReDirectIO(0) Use tInitIO
        do %ISCJSONStream.Rewind()
        // display result
        w %ISCJSONStream.Read()
        // result: { "error":"my-error-1", "description":"Error number one" }
```

You can, alternatively, write your own redirection routine

```
        // save as MAC routine, e.g. test.mac
        // launch by from terminal: d run^test()
run()
        Set tInitIO = $IO
        // we MUST use variable called %Stream that is used by a mnemonic space creat
ed by our routine
        set %Stream=##class(%Stream.TmpCharacter).%New()
        use tInitIO::("^"_$zname)
        do ##class(%Library.Device).ReDirectIO(1)
        // call routine that we want to redirect
        do demo()
        If ##class(%Library.Device).ReDirectIO(0) Use tInitIO
        do %Stream.Rewind()
        // show result
        set res=%Stream.Read() write res
        // result: Hi there!
        q

demo() public
{
        w !,"Hi there!"
}


redirects()
#; Public entry points for redirection
wstr(s) Do %Stream.Write(s) Quit
wchr(a) Do %Stream.Write($char(a)) Quit
wnl Do %Stream.Write($char(13,10)) Quit
wff Do %Stream.Write($char(13,10,13,10)) Quit
wtab(n) New chars Set $piece(chars," ",n+1)="" Do %Stream.Write(chars) Quit
rstr(len,time) Quit ""
rchr(time) Quit ""
```

Please note that redirects() procedure is not using procedural block and uses public variable %Stream that needs to be initialized before calling code to what we apply redirection.

Hope you find this helpful.

Dan

PS: When I searched community portal, I saw an article that was referring to the redirection, too. However, it was in a different context and was not explaining the power of such redirection.

#Code Snippet #ObjectScript #Caché

Source URL:https://community.intersystems.com/post/cach%C3%A9-tricks-capture-write-output-variable