Article
Sean Connelly · Aug 21, 2017 2m read

Warning: Stale Persistent Objects created by $THIS

tl;dr I have discovered that using $THIS in a very specific way will make persistent objects go stale.

I found that it only happens when using $get on an array that contains the OID reference of $THIS. I assume that this is an unwanted feature and have raised a WRC. The problem can be recreated following the steps below, I have also posted the workaround that I am currently testing and looks to be working fine.

1. Create a persistent class...
```
Class Test.Store Extends %Persistent
{

Property SomeValue As %String;

Method SomeMethod()
{
  //this will cause $THIS to go stale in this process
  if $get(seen($THIS)) quit
}

ClassMethod Initialise() As %Status
{
  set store=..%New()
  set store.SomeValue=$zh
  quit store.%Save()
}

ClassMethod Update() As %Status
{
  set store=..%OpenId(1)
  set store.SomeValue=$zh
  quit store.%Save()
}

}
```

2. Open up a terminal and...
```
do ##class(Test.Store).Initialise()
```

3. Open a second terminal and...
```
set obj=##class(Test.Store).%OpenId(1)
zw obj
do obj.SomeMethod()
kill obj
```

Note, it's when calling SomeMethod() that this object will become stale.

4. In the first terminal...
```
do ##class(Test.Store).Update()
set obj=##class(Test.Store).%OpenId(1)
```

zw obj

5. In the second terminal...
set obj=##class(TestStore).%OpenId(1)
zw obj

You will see that the second terminal still has a stale version of the persistent object.

These stale objects are also apparent in CSP sessions.

Workaround

Thanks to Timothy for a simpler workaround...

if $get(seen(""_$THIS))

Appending an empty string prevents the side effect.

~~Instead of using the OID value of $THIS directly, create a separate variable and create a stringy value of the OID...~~

~~set oid=$lg($THIS."%%OID",1)_"@ "$lg($THIS."%%OID",2)~~

~~Using the oid variable instead of $THIS makes the problem go away.~~

Sean.

#Object Data Model #Terminal #Caché