CFCs, the Variables Scope, and the Application Scope

Posted At : June 16, 2008 6:21 PM | Posted By : Mark Kruger
Related Categories: Coldfusion Tips and Techniques, Coldfusion 8

It's pretty common to use the application scope to cache components. If your component is a collection of methods or data access functions it's often faster to put them into the application scope than it is to create an instance with each request. Now you probably know that you should quality all of the variables in a function with the "var" key word. This insures that the variable exists inside the "scope" of the function call. This allows multiple function calls to be made to the same instance without one set of variables over writing the other.

One of the areas where this can be difficult to manage is when using a ColdFusion tag that creates its own scope. Take CFHTTP as an example.

The Problem

Consider this "test.cfc" component.

```
<cfcomponent>
    <cffunction name="test">
        <cfhttp url="http://www.yahoo.com"/>
    </cffunction>

    <cffunction name="test2">
        <cfhttp url="http://www.cfwebtools.com"/>
    </cffunction>

    <cffunction name="dumpthis">
        <h4>This scope</h4>
        <cfdump var="#this#"/>
        <h4>Variables Scope</h4>
        <cfdump var="#variables#"/>
    </cffunction>

</cfcomponent>
```

It doesn't do much. It simply makes a CFHTTP call to 2 different web sites within 2 different functions - test() and test2(). As you probably already know when you create a CHTPP call Coldfusion automatically creates a structured variable called "cfhttp" for you to work with. The results "page" or "content" of the HTTP call is stored in "filecontent" - so "cfhttp.filecontent" contains the results you are after. So far so good.

The function "dumpthis( )" dumps out 2 scopes that exist *within* the scope boundaries of the component - "this" and "variables".

Now consider this test framework.

```
<cfif NOT isDefined('application.test')
    OR isDefined('url.refresh')>
<!--- caching our test component --->
<cfset application.test = createobject("component", "test")/>
<!--- running one of the functions --->
<cfset application.test.test()/>

</cfif>
```

```
<h1>Dump the this and variables scopes</h1>
<cfset application.test.dumpthis()/>

<!--- run the 2nd call --->
<cfset application.test.test2()/>

<h1>Dump the new content</h1>
<cfset application.test.dumpthis()/><br>
<!--- local variables empty --->
<cfdump var="#variables#"/>
```

This code instantiates the "test" CFC in the application scope if it does not already exist and then runs a call to "test" (the function retrieving the yahoo page). It dumps out the “this” scope and “variables” scope that belong to the component. If you run the test you will notice that a cfhttp variable containing the Yahoo content is a part of the "variables" scope inside the CFC.

Finally, the code runs the function "test2()", which retrieves the cfwebtools page. The dumpthis() functions shows that now the variables scope contains a cfhttp variable that contains the cfwebtools content.

The implications

Maybe you are saying, “So what ... that's what I would expect”. But the implications reach a bit farther than that. What is actually going on here is that the *variables* scope that is local to the CFC is *persisting* in the application scope. In fact, once you have run the code above put the following code on a separate script within the same application.

```
<cfset application.test.dumpthis()/>
```

You will notice that, even though you have not called either test() or test2(), the variables scope *still contains a cfhttp variables* and the "filecontent" key still contains the content of the last cfhttp call made. How would this a problem? Let's say that 2 requests arrive simultaneously - one designed to return the results of "test()" (the yahoo page) and the other designed to return the results of test2() (the CF Webtools page). One request could overwrite the other and they could *both* get the same content even though they both fired different functions from entirely different request threads. Consider the implications for a web service, stock quotes, tracking information etc. You could cause yourself some real data headaches.

The fix

How to fix it? Luckily there is a useful attribute called "result" that you can use for your CFHTTP call. The fix is to var the result variable and then use it as an attribute to your CFHTTP tag:

```
<cffunction name="test">
        <cfset var rs = ''/>
        <cfhttp url="http://www.yahoo.com" result="rs"/>
    </cffunction>

    <cffunction name="test2">
        <cfset var rs = ''/>
        <cfhttp url="http://www.cfwebtools.com" result="rs"/>
    </cffunction>
```

This keeps the CFHTTP results encapsulated inside the function scope. Rerun the test script and you'll find that there is no longer a CFHTTP variable in the CFC's variable scope.

Of course in addition to CFHTTP there are other Coldfusion tags that create special scopes (cffile with the "upload" action for example). As a rule of thumb when you are caching CFCs in the application scope, take an extra look at any calls to external resources made from within a function.

And now, dear readers, I'm sure that some of you will be tempted to comment on whether you should or should not be using the application scope in this way. Please understand that the muse believes there are folks on both sides of the issue that have a legitimate point of view - and keep the comments charitable.