Using onSessionEnd in Coldfusion 7's new framework

Posted At : June 21, 2005 10:38 AM | Posted By : Mark Kruger
Related Categories: Coldfusion MX 7, Coldfusion Tips and Techniques

The new Application.cfc file (as of Coldfusion MX 7) is a great "step up" for the framework. In case you are not familar with it, here's a rundown. You replace the venerable old "application.cfm" file in the root of your application with an "application.cfc" file. It works in a *similar* way, but there are some extra features *and* some gotchas. Basically there are 8 function calls that are made by 8 different events that are a part of any application. To put it another way, when certain things happen within the application it fires 8 possible events - which in turn call these functions.

p> The events are centered around the Application, the Session, the Request and the error handler. The one that interests me (at least today) is "onSessionEnd( )". Why? Because (like onApplicationEnd) it is called without user interaction. It's not called inside of a request. Rather, the server calls it separately as sessions expire. Try it, set your sessionExpiration to 10 seconds and then refresh the page. Wait more than 10 seconds and onSessionEnd() will be called. Oh... how do you know? Well, that's the rub. You can't *really* see what's going on inside of onSessionEnd( ). That's the first gotcha - debugging. There are 4 issues that you may run into using onSessionEnd( ).

onSessoinEnd( ) is Hard to Debug

Errors thrown by onSessionEnd( ) do not show even show up in the error log. Do yourself a favor. Don't write a big complex routine and put it in onSessionEnd( ) without testing it. Write your onSessionEnd( ) routine carefully - 1 step at a time. Here's my trick for onSessionEnd( ), use cflog, Cffile or cfmail to output items of interest from inside the onSessionEnd() function. Here's an easy way to do it.

```
<cfmail from="*email*" to="*email*" subject="on session end arguments" type="html">
      <cfdump var="#arguments#">
   </cfmail>
```

Having gone through all of that in debugging, I received a suggestion from **S. Isaac Dealey** (correction - the tip actually came from **Matt Walker**) that I try instantiating and invoking the application.cfc directly. This thought had not occurred to me. Since the function is designed to fire in response to particular event, I assumed I would need to fire the event to get it to work. That's not the case. You ***can invoke it directly***. The function takes 2 arguments. The first argument is the session scope and the second is the application scope. Here's how:

```
<cfscript>
   ap      =   createObject("component","Application");
   ap.onSessionEnd(session,application);
</cfscript>
```

This will throw any errors directly to the page. I wish I had known that before - but thanks Isaac.

Changing the Application Scope Inside of onSessionEnd( )

One of the things you might want to do inside the onSessionEnd() function is alter application scope information. For example, perhaps the application scope has a variable tracking the number of sessions or users. In "onSessionStart( )" you add 1 to

the variable and in onSessionEnd() you subtract 1 from the variable - easy, right? No, not exactly. You *cannot manipulate the application scope by name* from inside of onSessionEnd(). Unfortunately, if you write code like "Application.numSessions = Application.numSession + 1" and then test it using the *invocation* method (as in the createObject( ) sample), it will work! The problem is, when it's fired by the timeout of the session it *will not* work. So in effect, it will work in testing but not in production.

To manipulate the application scope you must use the reference passed as an argument to the function. In case you've never heard of such a thing, a reference is simple an alias or a pointer to another object. The application scope in this case is the second argument - so you can modify the application scope by modifying that argument. Like this:

```
<cffunction name="onSessionEnd">
   <!--- reference to the session scope --->
   <cfargument name="sessionScope" required="true"/>
<!--- reference to the application scope --->
   <cfargument name="ApScope" required="true"/>
   <!--- lock and update the appication scope --->
   <cflock name="AppLock" timeout="15" type="Exclusive">
      <!--- decrement the number of sessions --->
      <Cfset ApScope.numSess = ApScope.numSess - 1>

   </cflock>

</cffunction>
```

When you do this you are actually modifying the application scope. Remember, you *can* modify the application scope when invoking the function directly, but it *will not* work when the event fires. To modify the scope when the event fires you *must* use the reference. That means you must include the CFARGUMENT tags and give your reference a name inside the function (like "apScope" above).

Locking the application Scope

When you write to the application scope you, naturally, want to lock it - right? Well, here's another gotcha. You cannot use a scope lock here. In other words, you can't write something like this:

```
<cflock scope="APPLICATION" timeout="5" type="Exclusive">
      <!--- decrement the number of sessions --->
      <Cfset ApScope.numSess = ApScope.numSess - 1>

   </cflock>
```

Why? Because to do so would mean directly accessing the Application scope - and when the onSessionEnd( ) event fires, it won't be able to do that (Doh!). Since the errors are not written to the log, this code will die silently and you will wonder why it's not working. You will need to use a named lock instead.

```
<cflock name="AppLock" timeout="15" type="Exclusive">
      <!--- decrement the number of sessions --->
      <Cfset ApScope.numSess = ApScope.numSess - 1>

   </cflock>
```

Keep in mind that with a named lock, synchronization is up to you. If you are writing to this variable elsewhere, make sure and use the same named lock so the data stays synchronized.

Explicit Session Timeout

This tip comes from the inestimable **Ray Camden** (the author of this very blog software). He notes that if you do not explicitly set a timeout value for the session, the onSessionEnd() function will never fire. In other words, you cannot rely upon the server level default session timeout. You must set the sessiontimeout in the "this" scope of the application object. This is usually done at the top of the component with a series of set statements like so:

```
<Cfcomponent>
<cfset this.name = "test_ap_2_5">
<cfset this.applicationTimeout = createTimeSpan(2,0,0,0)>
<cfset this.clientManagement = true>
<cfset this.clientStorage = "registry">
<cfset this.loginStorage = "session">
<cfset this.sessionManagement = true>
<cfset this.sessionTimeout = createTimeSpan(0,0,18,0)>
<cfset this.setClientCookies = true>
<cfset this.setDomainCookies = false>
<cfset this.scriptProtect = false>
....event code here...
</CFCOMPONENT>
```

Since this is a new approach there is still quite a lot to learn. If you have found another tip you'd like to share, or you want to contradict some of my findings here, leave me a note. I'd love to uncover all the ins and outs of the Application.cfc object.