I Never Break Encapsulation (almost)

Posted At : March 3, 2009 7:30 AM | Posted By : Steve
Related Categories: ColdFusion, OO Principles, com.sebtools

One of the larger systems that on which I work runs multiple sites from one code base. Each site needs to have some data that is specific to that site. For example, each site should use a different from email address and mail server when sending email.

During the initial development of the system, I decided to give each site its own application scope. After that, I don't have to worry about which data is site-specific and which isn't. Despite a continually growing number of sites, this has worked pretty well for more than five years.

Then the use of one of my own open source projects revealed a problem with this approach.

One of the features of the system is that the administrator can send email to a subset of the users on the system. I have it set up to use **Scheduler.cfc** to throttle email.

```
<cfif qMessages.RecordCount GTE variables.MaxRecords>
   <cfinvoke component="#variables.Scheduler#" method="setTask">
      <cfinvokeargument name="Name" value="Send Messages">
      <cfinvokeargument name="ComponentPath" value="sys.pub.Alerts">
      <cfinvokeargument name="Component" value="#this#">
      <cfinvokeargument name="MethodName" value="sendMessages">
      <cfinvokeargument name="Interval" value="once">
      <cfinvokeargument name="Args" value="#arguments#">
   </cfinvoke>
</cfif>
```

So, it has a sendMessages method that causes itself to be run again in an hour if it reaches the throttle limit.

This approach has worked very well on other sites. Because the Scheduler is used to call the method only once, it is effectively scheduled to run every hour until all of the email is sent.

In this case, however, I hit a snag. I only have one actual ColdFusion scheduled task (one of the advantages of Scheduler.cfc, after all). The scheduled task hits a URL on Site A.

This time, however, the administrator decided to send the email from Site B. Since the two sites each have their own Application scope, they each have their own instance of Scheduler. So, the Scheduler being called from the scheduled task didn't know about it should call the sendMessages, because it wasn't the instance that was told this.

This problem uncovered a larger issue with how I had decided to handle site specific data. I really only had a few choices (if not breaking encapsulation):

- Pass in site specific data on initialization (requiring separate components for every site)
- Pass in site specific data on every method call where it is needed (requiring me to know everywhere it is needed and to potentially have to pass that information through several method calls in an event chain)

Now, I could try to do some sort of hybrid whereby I have some components in Server scope (those without site specific data) and others in Application scope), but it turns out that most of the components use components that need site specific data (meaning that they effectively need that data themselves).

Think of the site specific data as a marble and each component as a box. When a component composites (or is passed) another component, it is a box that holds a box. So, if a component holds a component that has site-specific data (even if it doesn't, itself, need site specific data) then it indirectly needs that that data and so is site specific.

I found none of the available solutions attractive, so I finally have found what I consider to be an exception to the nearly inviable rule of encapsulation.

In this case I created a component called "EvilDecapsulation.cfc" which I pass to any component that needs site specific data. EvilDecapsulation.cfc itself looks in request scope for site specific data, breaking encapsulation.

This allowed me to use Application scope only once for the entire system, instead of having a separate Application scope for each site (freeing up some RAM as well).

The advantage of this approach is that while I am breaking encapsulation, I am doing it in a very limited way. If I later need to run tests or take some other activity whereby I don't want to have access to request data, I can create an alternate EvilDecapsulation.cfc to pass to my components for that purpose.

So, despite the fact that I have broken encapsulation in this case, I still believe that I don't (generally) do so. It took much consideration before I decided that the benefits were worth the cost in this case. Even then, I tried to contain the evil of breaking encapsulation and name my component something that would draw attention to the fact that rules were being broken.