Application Variables On Ice

Posted At : August 19, 2005 3:24 PM | Posted By : Mark Kruger
Related Categories: Coldfusion Tips and Techniques, Coldfusion Troubleshooting

Getting a handle on Application variables can be difficult. Even folks who know what they are sometimes use them incorrectly. There are 2 common forms of *simple* application variables (we'll leave the discussion of objects and CFCs for another time). There is the "set once read many" variable. Usually these are "settings" for the application - things like paths and data source names - Application.dsn, or Application.imagePath.

Then there is the "global tally" variable. This is less common, but it is usually some variable that is incremented or updated throughout the life of the application. For example, it might be a variable to track the number of logged in users. This variable is actually set (sometimes frequently) as the application is used. The main thing to remember about an application variable is that it *lives beyond the request or session*. It's something that is put in place as a variable and stays put for *every user* of the application. Some call it persistence - but the OO purist pee their pants when they hear that so I will refrain (too bad too, it's a good word for it). Let's focus on that first example of "set once read many".

Take this example:

```
<cfif NOT isDefined('Application.dsn')OR IsDefined('url.forceAp')>
    <cflock scope="APPLICATION" timeout="1">

        <Cfset Application.dsn = 'myDatasource'>
     <cfset Application.imgPath = expandPath('./images')>
    </cflock>

</cfif>
```

It's not hard to see what's going on here. If there is no application variable defined, the application sets the variable, otherwise this code never runs. If you change the data source or path, you will have to either restart CF, change the application name or append a "?forceAp=true" to your URL to get the new setting to be valid.

When Do I Use Application Variables

It's a trickier question than you might think. The answer is, if you are using simple variables and no custom tags or CFCs and your code organization is easy then you *may not* need these variables as application variables. You might just as well use:

```
<cfset dsn = 'mydatasource'>
<cfset imgPath = expandPath('./images')>
```

You *probably* won't see any difference in performance - especially if you are setting a few simple variables as above. Of course the application scope keeps you from accidentally overwriting the local variables, but that could be a minor issue. The other thing to keep in mind is that Application variables fit a "framework" of sorts for you code. It makes sense to put "settings" that apply to your whole application in a scope titled "application". It's a judgment call - with no harm no foul. Just remember, the more complex your application, the more you need the separation that scopes provide.

Something You May Not Know

Here's a tricky *gotcha* that will drive you crazy the first time you see it. Developers are used to dealing with an Application as a collection of files. Usually these files are part of the same folder structure (though not always). So, for example, we start a site or application with a "root" for the application, and add files and folders into this folder or sub-folders. For Example:

```
/application.cfm

/index.cfm

    /forms/userEdit.cfm

    /display/userList.cfm



    ... and so on...
```

We know that when we access the file "userEdit.cfm" in the above example, that the "application.cfm" file is run first - and any variables are calls made in that file will be available inside of the included file.

What you *may* not know is that the Application variables are not tied to the folder structure. They are tied to the "name" of the application. Remember the "name" part of the cfapplication tag?

```
<cfapplication name="Muses_cool_application" ... >
```

Or in Application.cfc...

```
<cfset this.name ="muses_cool_cfmx_application" ...>
```

All of the variables in the application are bundled under the "name" of that application. If you change the name, a *new* application is instantiated in memory and the old one (stll valid) hangs around until it times out.

Why is this important? If you are like me, you do a lot of code reuse. You might, for example, copy an application.cfm file from another application because it did basically what you wanted and just needed some minor adjustments. If you don't change the *name* of the application, the 2 application.cfm files will share the same scope. So, for example, if you change the data source name and then reset the application variables, your first application will break because it's now pointed to the wrong data source

Try this experiment. Create folder A and folder B. In both folders create a cfapplication tag with the same "name". Then add a set statement like this.

```
<!--- folder A's application.cfm file --->
<cfif NOT IsDefined('application.testVar') OR isDefined('url.forceAp')>
    <cfset testVar ="I'm Application A, aren't I?">
</cfif>

<!--- folder B's application.cfm file --->
<cfif NOT IsDefined('application.testVar') OR isDefined('url.forceAp')>
    <cfset testVar = "I'm Application B, aren't I?">
</cfif>
```

Then Add an index.cfm file that looks like this:

```
<cfdump var="#application#">
```

Open A/index.cfm. You will see the message from folder A. Now open B/index.cfm. You will see the message from ***folder A***. Now open B/index.cfm?forceAp=true. The message changes to the message from B. Now change the name of A's application and refresh A/index.cfm. You will see A's message. Go back to B and you will see that B's message has remained the same.

Real World Example

A friend of mine developed a "franchise" application. He sold web sites to franchise members for the parent company. When a franchise signed on his plan was to create a new folder, copy in his franchise site code, customize the application variables, and point the user to the admin page for setting up his data. When the program launched he had 8 or 10 franchises sign up right away. He followed his plan and sent them all emails the first day. Each user would sign on and edit his information. You guessed it. Because they were all sharing the same scope, one user was overwriting the information of another. They competed all day to get their individual sites updated, and they were frustrated that it kept reverting to some other franchise's data. The fix? Go into each site code and change the name of the application. That's it.