Cfinclude vs Cfmodule and Other Things

Posted At : January 3, 2007 3:42 PM | Posted By : Mark Kruger
Related Categories: Coldfusion Tips and Techniques

> *Muse Reader Asks:*
> What is the difference between cfmodule, cfinclude? Which one should be used. We have two application.cfm file one at local folder and the other one at the common folder. Which application.cfm will be used when we try to execute Coldfusion template from local directory.

This is really 2 questions so let's take them one at a time. First, let's talk about the difference between Cfmodule and Cfinclude. The first thing to recognize is that that these 2 tags are *quite different*.

Cfinclude

Cfinclude allows you to include a block of code as the request is being processed and *share the same scope* as the page doing the including. If I have the following page called "mainPage.cfm":

```
<cfset x = 10/>
    <cfinclude template="incPage.cfm"/>
    <cfoutput>#x#</cfoutput>
```

...with the code on the "incPage.cfm"...

```
<cfset x = 20>
```

The result of the output will be 20. This is because the *variables* scope (the scope that means "this process") is a part of both templates.

The way includes are often used is to separate out chunks or snippets of code that are belong together. For example, while handling a form submission you might have a block of code that "validates" the form inputs and another block that "updates" the database. You could use Cfinclude to easily separate the code out and still program in a "procedural" manner as in:

```
<cfset isError = 0/>
    <!--- this template will set
            isError to 1 if the form
            fails validation
     --->
    <cfinclude template="formValidate.cfm"/>
    <!--- If the form is ok update db --->
    <cfif NOT isError>
        <cfinclude template="updateDb.cfm"/>
    </cfif>
```

Cfmodule

Cfmodule is really a *custom tag* call. It is, in fact, a way to call a custom tag without needing to place it in the "customtags" folder, or in the same folder as the calling process. In our example above, assuming both mainpage.cfm and incPage.cfm where in the same folder you could actually use either the module or the custom tag syntax:

Module syntax

```
<cfset x = 10/>
    <cfmodule template="incPage.cfm">
    <cfoutput>#x#</cfoutput>
```

Custom Tag Sytax

```
<cfset x = 10/>
    <cf_incPage>
    <cfoutput>#x#</cfoutput>
```

But the result of using cfmodule or the custom tag syntax would *not* be the same as using a cfinclude. Why? Because unlike a cfinclude, custom tags (or modules) do not share the same scope. They have their own scope called "attributes". Anything available to them must be "passed in" to be manipulated. The output result of the module code (or custom tag code) above would be 10, not 20. Here's how you would produce a result of 20:

```
<cfset x = 10/>
    <cfmodule template="incPage.cfm" x="#x#">
    <cfoutput>#x#</cfoutput>
```

...with the code on the "incPage.cfm"...

```
<!--- Attributes.x has been passed in
            and contains 10. Let's set it
            to 20
      --->
    <cfset attributes.x = 20/>
    <!--- To get it back out to the calling
            template we use the "caller" scope
      --->
     <cfset caller.x = attributes.x/>
```

Now obviously in this silly example I could have skipped the whole "attributes scope" and set caller.x to 20 and be done with it. What is the point of a custom tag? It's designed to make reusable code in a black box. The idea is that you "encapsulate" a routine and make rules for the "attributes" it requires. You pass those attributes in and you either get something back, or the tag "outputs" something to the buffer.

If you are concerned about abstraction and code re-use, and you are programming in CF 5 (or earlier - yikes) then a custom tag can be a "poor-man's OOP" (or "POOP" as we like to call it). You can mimic object oriented programming by creating an interface. You pass something in, you get something back.

In my opinion, if you are using CF7 there are few cases where a custom tag is an "ok" (meaning "so-so") solution. One case that passes muster with me is in the case of layout. You can use a custom tag or cfmodule to bracket your content and set up the header and footer - or to manipulate the generated content buffer. A good example of this approach is Ray's "scope cache" tag which is used by this very blog. The syntax looks like this:

```
<cfmodule
        template="tags/scopecache.cfm"
        cachename="#application.applicationname#"
        scope="application"
```

```
        disabled="#disabled#"
        timeout="#application.timeout#">
    ... all the blog content and articles go here....
    </cfmodule>
```

This ingenious tag creates caches generated content in the session or application scope with a timeout value. When a page is called the tag serves the data from memory. It's one of the reasons why blogCFC loads very fast. Other than examples like this (or when working with legacy code) you should use CFC's when you are trying to encapsulate and abstract.

As for includes, I use them freely for things like headers and footers. I also use them when building procedural utility scripts or when I want to separate the logic block (cfif this then do that else do this) from a script that does "heavy lifting" like queries and file manipulation. Why? So that when I'm troubleshooting I can work my way through the flow of logic without having to become embroiled in the other code.

Which Application File is called

Coldfusion will execute whichever application file it finds first. For example, if your code structure looks like this...

```
/folder/application.cfm
    /folder/subfolder/application.cfm
    /folder/subfolder/test.cfm
```

...and you run the file "test.cfm", the file /subfolder/application.cfm will be *the only one that is executed*. To take it a step further. What if my "test.cfm" file looked like this?

```
<cfset x = 10/>
    <cfinclude template="../test2.cfm"/>
```

Since I'm including a file in the /folder/ directory does that mean that /folder/application.cfm is going to run? No. The rule is that the *implicit* execution of application.cfm can only take place *at the start of a request*. Neither cfmodule calls nor cfinclude calls occur at the start of a request.

If your concern is to find a way to get both Application.cfm files called here's a couple of tricks.

- **Call A from B** - in the example above you could add the following to the /subfolder/application.cfm file

  ```
  <cfinclude template="../application.cfm"/>
  ```

  The result would be that both application.cfm files would run procedurally. Keep in mind that this technique does not work for Application.cfc - only for Application.cfm
- **Share the Application Scope using the Name** - If all you are really after is for both application to share the same application variables then you can do this readily by naming them both the same.

  ```
  inside of...
  /folder/application.cfm
  <cfapplication name="bob" ...>
  <cfset application.x = 'harry'/>

  inside of ....
  /folder/subfolder/application.cfm
  ```

```
<cfapplication name="bob" ...>
<cfset appliction.y = 'tom'/>
```

The end result here will be that application.x and appliction.y will exist for you whether you are executing a template inside of /folder *or* /subfolder. That's pretty neat, but it is also a cause for concern. It means that code in /folder/subfolder/ can overwrite variables set in /folder and vice versa. Use this technique with great care.

Hopefully this rundown will help you solve your issues and provide some good background for the future. Happy Coding!