Build A Better ColdFusion App: Simple Conditionals

Posted At : January 3, 2014 9:26 AM | Posted By : Cutter
Related Categories: CFScript, Development, ColdFusion, Adobe

In really large apps, every line of processing can become critical. Every small process can add up, in terms of overall overhead. Especially in high traffic apps. Sometimes, the little things can really make a difference. Even your simple code might be...simpler. Take the following example:

```
<cfset mySelections = "1">
<cfif #mine# eq "0">
 <cfset mySelections = "0">
</cfif>
```

I saw something like this in code this morning. Right off, the first thing you see is that all of the variables are unscoped, forcing the server to do a scopeCheck() to find out which scope each variable is a part of. Then there's the unnecessary bits to go along with it, like setting a variable, then checking another to reset the variable based on a condition. There's also the bit where the one variable is unnecessarily hashed. On top of all of that, variable assignments are one of those things it just makes more sense to write in CFScript. Let's take a look at how this might be improved:

```
REQUEST.mySelections = (URL.mine) ? 1 : 0;
```

One line of code. Less keystrokes. Clearer references, and logic.

The "mySelections" variable, in this instance, is a part of the REQUEST scope (I upper case all variable scopes, to make them easier to spot when editing code).

In this situation, "mine" was a URL variable, param'd further up the page. We've used short-circuit boolean logic in our conditional statement. For those that don't understand that, any numeric value other than "0" is evaluated to true. If the value is "0", then it is false.

And speaking of numeric values, you don't need to quote them. Even though ColdFusion would figure it out, a quoted value is a string. Don't quote numeric, and don't quote Booleans.

We use the URL conditional inside of a ternary operator. Ternary operators were introduced in ColdFusion 9. The basics of this are, if the conditional is true, then use the value after the question mark. If the conditional is false, then use the value after the colon.

Finally, there's no need to hash the URL variable. You only need to hash variable references as part of output, or when they are used inside of quotation marks in code.

Now comes the really fun part. There really isn't any need for REQUEST.mySelections at all. Since the value or URL.mine was param'd, further up in the code, then the variable is always available here. Rather than copying that value to another variable (which takes up more memory), we can just reference URL.mine anywhere that REQUEST.mySelections was being used.

As you maintain your applications, it's always good to take some time and read through what's been written before. Refactoring little nuggets like this one, and a little testing, can eventually go a long way in preserving the life of your app and your server.