Bamboozled by Pound Sign Decisions

Posted At : February 10, 2011 6:36 PM | Posted By : Mark Kruger
Related Categories: ColdFusion

If you want to greatly vex the Muse and his merry band of developers, all you have to do is send them sample code with a liberal sprinkling of superfluous pound signs and cfoutputs. ColdFusion is a very forgiving language, so you are free to do all sorts of nails-on-the-chalkboard things that make me cringe - and you will get away with it too. I began thinking about this topic because I'm privileged in my current work to review code created by others. This is double edged sword of course. It's easy to sit back and be cheeky about someone else's approach when you are in the "expert" seat. Still, I have recently reviewed code from more than 10 different developers. I've come to the conclusion that many (perhaps most) sort of *guess* at when to use pound signs. Once they figure out the approaches that work they stick with that.

The result is that pound signs are like Tribbles on the star trek enterprise. They are fairly innocuous but they annoying show up everywhere. So... having never seen a definitive post on the topic I thought I would jump in and put my stamp on when to use and when not to use pound signs. That's right; the Muse is not making room for other opinions this time. I'm going to lay down the law! So put your seat belts on 'cause it's going to be a bumpy ride.

The Output Buffer

So the first thing to know is that there are conditions that *always* require pound signs. To understand this let me introduce you to the concept of the "output buffer". I've not heard a lot of folks talk about this and there may be other monikers for it, but the Muse has adopted "output buffer" so that's what we are going with. Consider the following dialogue between ColdFusion and your request.

- **Request:** Yo CF I have some stuff to execute here.
- **CF:** All right let me create a little memory space to return to the web server. Let's see... type string buffer, yada yada yada.... ok, ready to go.
- **Request:**Ok... first we got some html stuff and html, header and body tag. Oh... and script tag that references an external JS file.
- **CF:** JavaScript file... hehe... oh you crazy kid. Ok, let me throw all that into my buffer. What's next.
- **Request [checking clipboard]:** Let's see.... ok now he wants to run this query - *select * from users*. Got that?
- **CF:** Yeah yeah I got it keep your threads on. [whispers to DB driver] ... ok, you are all set. I have the query.
- **Request:** Cool. Ok now we have a couple of table rows, then a cfoutput, then he outputs a bunch of columns from the DB into a table.
- **CF:** Got it. Anything else?
- **Request:** standard stuff... closing table tags, disclaimer and end tags. That about wraps it up.
- **CF:** Okey dokey. Here's a reference to your output buffer. Get this packed up and back to the browser.
- **Request:** Will do. take care - I'll see in you a few miliseconds.
- **CF:** Yeah... and don't forget - we have a party at the maintenance window at 2:00 am. I hear Britney the JDBC driver is showing off her new build.

- **Request:** I wouldn't miss it for the world....

This silly little dialogue illustrates the old "output buffer" idea. ColdFusion assumes by default that your request will result in output. It's a web server platform after all. So each request starts with an empty buffer. As things happen in a request the content of the buffer is populated with the results of your work.

It may not be obvious at first, but you can manage this buffer. If you have good code separation where you do all your component calls and heavy lifting first you can bracket your controller code like this:

```
<cfsetting enablecfoutputonly="yes"/>
        <cfinclude template="myController.cfm"
    <cfsetting enablecfoutputonly="no"/>
```

This effectively *stops* ColdFusion from sending anything to the output buffer unless you specify it using cfoutput. Why would you do this? To get rid of the copious white space at the top of the page. You see, CF will see line breaks spaces and tabs in your code as output even if it's just part of a query or logic. That's why so many CF sites have lots and lots of white space at the top.

The Always Rule

Now that I have you thinking about the output buffer let's get back to our pound signs. Here's my first rule:

> *If the contents of a variable are headed for the output buffer then pound signs are definitely needed.*

This is the first most obvious rule. It's also never broken because breaking it generates errors or means that you don't get what you want on the page. But it's important to state it because it tells you exactly what 99% of pound signs are used for. They are used for outputting variables to be seen by the user. If you remember that one single rule then you will likely *not* fall into some of the traps I'm about to lay out for you.

The Never Rule

> *If a variable is being checked, manipulated, assigned or used in logic you never need pound signs.*

This is the one that's violated constantly. You don't need pound signs inside of CFIF statements or inside of cfset statements or to use a variable in a function (although you may need them *outside* the function if the results of the function are intended as output).

The Wrong Way

```
<cfif #isDefined("form.blah")# OR #action# Is "foo">
    <cfset blah = #form.blah#/>
</cfif>
```

The Right Way

```
<cfif isDefined("form.blah") OR action IS "foo">
```

```
    <cfset blah = form.blah/>
</cfif>
```

The Query Rule

> *The names of queries used in tags never need pound signs.*

If you tried to use pound signs it would throw an error so this one is never violated. It does, however illustrate a little inconsistency since pound signs are needed for tags in other places as we shall see. In any case, query names are used with no pound signs inside of tags - like so:

```
<cfloop query="qry">
    ...
</cfloop>
<Cfoutput query="qry">
    ...
</Cfoutput>
<cfmail query="qry">
    ...
</cfmail>
```

The Loop Rule

CFLOOP is one of those areas where pound signs don't feel like they *should* be needed, but in fact they usually are. Take arrays as an example. You can't do Array="arrLoop". Instead you have to do Array="#arrLoop#". Collections (used for objects or structs) suffer the same fate. Doing Collection="MyCollection" is incorrect. Instead you have to do Collection="#myCollection#". I'm sure this probably has to do with by reference / by value type decision making under the hood.

The Concatenation Rule

Ok.. the muse will give some leeway here. When I concatenate a string with variables I tend to do the following:

```
<cfset msg = "Welcome back " & cookie.firstname/>
```

This is more of a personal preference. I often see string concatenation that looks like this:

```
<cfset msg = "Welcome back #cookie.firstname#"/>
```

I don't have an overt objection to using pound signs in assignment this way (it *is* readable) as part of a larger string. But in general my campaign is to minimize the use of pound signs so I don't do it this way as a matter of principle.

The Usually Rule

> *Variables inside of CF Tags usually need pound signs.*

When working with a ColdFusion tag (not logic or evaluation or assignment) you are usually going to need the pound sign to pass in values. Here are 2 good examples.

CFSWITCH: When using cfswitch the "expression" is expecting a primitive *value* (a string, number, etc) for comparison. So it needs the contents of the variable - not the variable name or reference. So this would be correct:

```
<cfswitch expression="#form.action#">
        ...
    </cfswitch>
```

While this would be *incorrect:*

```
<cfswitch expression="form.action">
        ...
    </cfswitch>
```

Why does it work this way? Because cfswitch accepts a string. When you pass simply "form.action" it sees what it expects - the *string* "form.action". It's not going to evaluate form.action for its contents. It won't ever throw an error because syntactically there is nothing wrong. That's why you need the pound signs. You need to pass in the *contents* of form.action. So in a way this one makes sense - at least to the muse.

CFMAIL: Another example, Cfmail, requires pound signs around variables *except* for query (see the Query Rule). This leads to an odd result when working with a query. The name of the query is *not* bracketed by pound signs while columns from the query *must* be bracketed. This makes sense if you think of a query driven cfmail as version of a query driven cfoutput. So this would NOT be correct:

```
<cfmail query="getUser" from="email" to="email" subject="test this">
    ...
</cfmail>
```

Instead - this would be correct.

```
<cfmail query="getUser" from="#email#" to="#email#">
    ...
</cfmail>
```

So again the variables passed into a tag need pound signs. Look at it closely and you can see why. The "From" and "To" attributes *also take a string*. How is the tag to discern that a string is a column name? The only way it could would be to force the "to" and "from" address in a query driven cfmail tag to be *required* to be a column - and of course that would severely limit the use of the tag. Other tags like Cfchart, cfftp, cfschedule, cfzip, cffile etc - all require pound signs when a variable is passed in.

Conclusion

I guess if I had something I would like folks to get out of this post it is to stop using pound signs "when in doubt". Now Muse readers, feel free to set me straight, but let's keep those comments clean and civil.