Sneaky Coldfusion - Creating CF Tags Out of HTML Tags

Posted At : May 10, 2006 2:24 PM | Posted By : Mark Kruger
Related Categories: Coldfusion Tips and Techniques

This is quite possibly the neatest trick ever invented for Coldfusion - so hang on to your Cf_hat. I had nearly forgotten about it until someone on **CF-Talk** mentioned a problem they were having. It seems they were struggling with a CMS system where users were entering hyperlinks that were incorrectly formatted (they lacked URL encoding). The dilemma was how to fix it without requiring action from the user.

The choices were slim and none of them seamless.

- On submitting a new page, parse out the HREF's and encode the values.
- Provide tools an instructions on encoding along with a table of html encode values.
- Teach users how to "wrap" the HREF query string in the #urlencodedformat( )# function

Only the first option provides a solution that is seamless to the user, but it comes with it's own set of problems. There is however, a third option. It used to be called "adaptive tags". When I did a search for "cfimport" and "adaptive tags" I came back with my own previous blog post referencing adaptive tags that I learned from Tim Buntel, and an article by Charlie Arehart in sys-con from 2002. That was pretty much all I found. I wondered if the technique would even work on a CFMX 7 server, but when I tested it, it worked just dandy.

Adaptive Tags Can Render HTML Tags as Coldfusion

Here's the scoop. You probably already know that you can use CFIMPORT to create a library of custom tags (JSP or CF). You do this by designating a folder containing all the tags and a prefix which subsequently becomes a part of your call to the tag. For example, if I had a directory called "foo" with a custom tag called "bar.cfm" in it, I could do the following:

```
<!--- import all the tags in foo --->
<cfimport taglib="foo" prefix="callme"/>
<!--- call bar.cfm using the prefix --->
<callme:bar attribute1="foo.bar has been called"/>
```

So far so good. We actually end up with a tried and true JSP type syntax. But what happens if we forget the prefix? Can we still call "bar.cfm"?

```
<!--- import all the tags in foo --->
<cfimport taglib="foo" prefix=""/>
<!--- call bar.cfm using the prefix --->
<bar attribute1="foo.bar has been called"/>
```

As it turns out, *this works*! Now take a moment and let that sink in. How can we use this new found knowledge to fix the problem above? Simple! We create a directory called "htm" (or whatever) and place a tag called "a.cfm" in it. Then we import "htm" without a prefix. When the scripting engine comes across <a it's going to call "A.cfm". All the attributes that exist in the anchor tag are going to be passed to the custom tag in the "attribute" scope. So the following code:

```
<cfimport taglib="htm" prefix=""/>

<a href="http://blah.cfm?name=Joe J. Smith">Joe J. Smith</a>
```

...now outputs the following link....

```
<a HREF="http://blah.cfm?nameJoe%20J%2E%20Smith">Joe J. Smith</a>
```

Here's a Sample A.cfm Tag

In my test I did something else. I checked for the existence of a "style" attribute. If I didn't find one, I set the style to font-size: 14pt. Here's the sample tag.

```
<cfsetting enablecfoutputonly="Yes">
<cfif thistag.executionmode IS 'Start'>
<!---enforce a style attribute --->
<cfset isStyle = false/>
<!--- begin the tag --->
<cfoutput><a </cfoutput>
<Cfloop collection="#attributes#" item="aItem">
   <!--- check for style --->
   <cfif aItem IS 'style'>
      <cfset isStyle = true/>
   </cfif>
   <!--- encode the values of the href attr. --->
   <cfif aItem IS 'href' AND listlen(attributes[aItem],'?') IS 2>
      <!--- front part --->
      <cfset qString = listlast(attributes[aItem],'?') />
      <!--- string to hold new qstring --->
      <cfset newQString = "">
      <!--- loop through the qString and encode JUST the values --->
      <cfloop list="#qString#" index="vItem" delimiters="&">
         <cfset newQstring = ListAppend(newQString,listFirst(vItem,'=') &
urlencodedformat(listlast(vItem,'=')),'&')/>
      </cfloop>
      <!--- reset the value of this attr. --->
      <cfset attributes[aItem] = listfirst(attributes[aItem],'?') & '?' & newQstring/>
   </cfif>
   <!--- output the attribute to the <a> tag --->
   <cfoutput>#aItem#="#attributes[aItem]#" </cfoutput>

</CFLOOP>
<!--- if no style exists set one --->
<cfif NOT isStyle>
   <cfoutput>style="font-size: 14pt;" </cfoutput>
</cfif>

<cfoutput> ></cfoutput>

</cfif>
<!--- tack on the end tag --->
<cfif thistag.executionmode IS 'End'>
   <cfoutput></a></cfoutput>
</cfif>


<cfsetting enablecfoutputonly="no">
```

If you want to test it, simply create a directory called "htm" in the same folder as your test script and run this code:

```
<cfimport taglib="htm" prefix=""/>

<a href="http://blah.cfm?name=Joe J. Smith">Joe J. Smith</a>
```

Other Possibilities

Obviously you could use this technique to great effect. You could enforce styles and classes. You could create a click-handler that all URLs would be forwarded through. You could rewrite boldface, italics, font tags - virtually any tag with attribute type syntax. Obviously some tags would be problematic (like the body tag). Before you go hog-wild you should consider whether this technique is actually necessary. I suspect in many or most cases it is not. Don't use it just because it's neat. It definitely results in something of a management problem with your code. Now, when you look at a page, you can no longer separate out what belongs to CF and what doesn't. That makes it difficult to maintain code using this technique. Still, it might save the day in a few cases.

Additional Resources on Adaptive Tags in Coldfusion

Muse's Old Blog (Oct. 2002)
Charlie Arehart's "hidden gems" article (Aug. 2002)