Method Composition Works Externally But Not Internally - reEscape(), reUnescape(), and reEscape2()

Posted At : December 1, 2007 6:30PM | Posted By : Matt Quackenbush
Related Categories: General, ColdFusion

What is reEscape()?

Often times, when you pass a string through ColdFusion's built-in regular expression (regEx) functions [reFind()/reFindNoCase()/reReplace()/reReplaceNoCase()], the string needs to have regEx special characters escaped. reEscape() is a tidy little method that I wrote to do just that.

So, let's see reEscape() in action, and point out its good and its bad.

```
1. <cfset foo = "+$[(.){}" />
2. <cfset foo2 = "$3.95 + $5.97" />
```

#foo# and #foo2# are just simple strings, but we've loaded them with several regEx special characters. Let's run each thru the reEscape() method.

```
1. #reEscape(foo)#: +$[(.)\{}
2. #reEscape(foo2)#: $3.95 + $5.97
```

Now these strings are both safe to pass into those built-in regEx functions.

Now then, on occasion our code needs to pass a string off to other methods. Both our original method and the secondary method(s) perform regEx operations on the string, so each method needs the string to be escaped. Here we will mimmick this by passing our variables through reEscape() twice.

```
1. #reEscape(reEscape(foo))#: \+\$\[\(\.\)\\\{}
2. #reEscape(reEscape(foo2))#: \$3\.95 \+ \$5\.97
```

This does ***not*** give the proper result. While the resulting strings may be safe for regEx operations, the strings are not correct. Therefore, any parsing of them will most likely not end in the desired result.

reEscape() + reUnescape()

So, to fix this problem, I wrote an reUnescape() method. We'll mimmick the multiple pass-thrus as above, but this time we'll pass it through reUnescape() in the middle.

```
1. #reEscape(reUnescape(reEscape(foo)))#: +$[(.)\{}
2. #reEscape(reUnescape(reEscape(foo2)))#: $3.95 + $5.97
```

Note that the result is proper, just like our first example above (a single pass-thru). It works like a charm, and everyone is happy.

So what is the problem then? Well, nothing really. But I like to have as few lines of code as possible when I can, so I decided that I would eliminate the need for calling the reUnescape() method every time I make a subsequent call to reEscape().

"Easy enough," I thought, "I'll just add the reUnescape() call into the reEscape() method." (See reEscape2() below .)

reUnescape() Inside of reEscape() [a.k.a. reEscape2()]

Nope, not so fast. For some reason that I cannot for the life of me figure out, no can do. Well, I can do it, but the results are incorrect.

```
1. #reEscape2(foo)#: +$[(.){}
2. #reEscape2(foo2)#: $3.95 + $5.97
```

"What do you mean, Quack?" you ask perplexed. "They both look just fine to me!"

Yeah, that's what I too thought at first. #foo2# is just fine. But #foo# is not. Notice that we included a backslash () in that string when we set it above. A backslash must be escaped by another backslash (\). In the reEscape() examples above, it is properly escaped. But when we wrapped reUnescape() up into the reEscape2() method, the result simply deletes the backslash from the string.

So my question is: Why on earth does the very same code work perfectly when called externally, but once placed inside the other method, it fubar's things? This has me quite puzzled. Any thoughts or suggestions?

<update date="(12/02/2007)">

Sean showed me yesterday that I was calling methods in a different order externally than I was calling them internally. When I took his suggestion and applied it, I achieved the desired results. The suggestion was to alter reEscape2() so that it would first reEscape() the provided string.

That is, instead of

```
1. var rtn = reUnescape(arguments.strIn);
```

using this

```
1. var rtn = reUnescape(reEscape(arguments.strIn));
```

While that achieved the desired results in this specific example, it did not achieve the desired results in other more "real world" examples. And it bugged the hell out of me that I was going to have two methods instead of one. As a result I kept looking at it and just now figured out what the problem is...

If you notice, in setting the #foo# variable, I used a backslash immediately prior to an open curly brace ({), which is another regEx special character. Because of that positioning, the reUnescape() method sees that as being an escaped curly brace, as opposed to a backslash and a curly brace.

However, if you move the backslash to the end of the string, e.g.

```
1. <cfset foo = "+$[(.){}" />
```

the reUnescape() method now handles it just fine. While I would totally love to figure out how to write the reUnescape() regEx so that this doesn't happen, I feel that it is very unlikely in a real-world scenario that I'm going to encounter a string with those characters back-to-back. Therefore, for the time being, I am willing to live with it.

Thanks again to Sean for offering his eyes to my dilemma. It's amazing how time and time again you can stare at something for hours on end and not "see" it, while a fresh pair of eyes (especially when we're talking one of the best pair of code eyes in the world) can see the issue within a matter of minutes.

</update>

<update date="12/03/07">

Okay, it seems that I was not very clear in communicating the code that I am actually using, so I am going to attempt to clear that up right here with this update. :-)

reEscape2() was purely for demonstration purposes, so that the different regEx's employed could be seen. In my real code, there are only two methods: reEscape() and reUnescape().

The only method I ever actually call from my various controllers, etc, is reEscape(). It makes an internal call to reUnescape(). While I could eliminate reUnescape() altogether and just add that regEx to reEscape(), I like having them separated so that I can easily alter it later should I need to.

So, in reality, what is reEscape2() in this post is actually my reEscape() method.

Hope all of that makes sense.

</update>

Functions

```
<!--- BEGIN reEscape() method --->
<cffunction
name= "reEscape"
hint= "Returns the provided string after escaping regular expression special characters"
returntype= "string"
output= "no"
access= "public" >
<!--- set the method arguments --->
<cfargument name= "strIn"
hint= "The string to process"
required= "yes"
type= "string" />
<!--- per Adobe LiveDocs, the following characters must be escaped:
+ * ? . [ ^ $ ( ) { |
--->
<cfreturn reReplace ( reReplace (arguments.strIn, "\" , "\" , "all" ), "(+|*|?|.|[|^|$|(|)|{||)" , "\1" ,
"all" ) />
</cffunction>
<!--- END reEscape() method --->
```

```
<!--- BEGIN reEscape2() method --->
```

```
<cffunction
name= "reEscape2"
hint= "Returns the provided string after escaping regular expression special characters"
returntype= "string"
output= "no"
access= "public" >
<!--- set the method arguments --->
<cfargument name= "strIn"
hint= "The string to process"
required= "yes"
type= "string" />
<cfreturn reReplace (
reReplace (
reUnescape(arguments.strIn),
"\" ,
"\" ,
"all" ),
"(+|*|?|.|[|^|$|(|)|{||)" ,
"\1" ,
"all" ) />
</cfscript>
</cffunction>
<!--- END reEscape2() method --->
```

```
<!--- BEGIN reUnescape() method --->
<cffunction
name= "reUnescape"
hint= "Returns the provided string with any regular expression escape characters removed"
returntype= "string"
output= "no"
access= "public" >
<!--- set the method arguments --->
<cfargument name= "strIn"
hint= "The string to process"
required= "yes"
type= "string" />
<cfscript>
var rtn = arguments.strIn;
rtn = reReplace (rtn, "(\)(+|*|?|.|[|^|$|(|)|{|||\)" , "2" , "all" );
return rtn;
</cfscript>
</cffunction>
<!--- END reUnescape() method --->
```