Number Formatting in Coldfusion - a look "Under the Hood"

Posted At : February 18, 2006 3:24 PM | Posted By : Mark Kruger
Related Categories: Coldfusion MX 7, Coldfusion Tips and Techniques

Formatting numbers is a pain. It would be great if our little human pea brains could read a number without commas in it to group the hundreds (or periods for my European readers). Sadly, we find ourselves unable to cope without the commas, so a good deal of display code regarding numbers is written to simply output them in the correct format. Most CF programmers use *numberFormat( )* or *decimalFormat( )* to control the output. Decimal format seems handy because it doesn't require a mask and produces the typical format you would expect. The 2 functions are quite different however and it may cause some perplexity when you are working with large numbers. Let me explain.

Note: Examples provided by Russ "Snake" Michaels from the CFGURU list :)

The decimalFormat() Function

The decimalFormat() seems to convert the argument into an actual Java data type like double or float, then it applies a specific mask to it and returns the result. That works fine, except that there is a maximum size *for decimal notation* of 18 with a scale of 2 for this function. That means if you get *beyond* 18 characters the number object (for the function) will not contain the expected data. Instead, it will contain scientific notation. You can use the javaCast() function to demonstrate what is happening behind the scenes. Check out this example:

```
<cfset num = 123456789123456789.12>

<cfset javaNum = Javacast("float",num)>
<cfoutput>
Java casted Number: #javanum#
</cfoutput>
Result is: 1.23456791E17

<cfset num = 123456789123456789.12>

<cfset javaNum = Javacast("float",num)>
<cfoutput>
Java casted Number: #javanum#
</cfoutput>
Result is: 1.23456789123E+017
```

The decimalFormat() function isn't programmed to convert this number back into the necessary *character array* for the mask. So the results are wrong with any number that has more than 16 digits. Why 16 and not 18? Because decimalFormat() tacks on the 2 decimal places if they are not passed in - so a 16 digit number with no decimal places looks like an 18 digit number to the function. The example above has 20 digits. When the mask is applied the result is never exactly what is excepted.

```
<cfset num = 123456789123456789.12>
<cfoutput>
decimal format: #decimalformat(num)#
</cfoutput>
Result is: 92,233,720,368,547,760.00
```

Obviously the result of trying to mask the scientific notation results in some sort of

calculation being run or perhaps a reference problem. In any case, the result is wildly inaccurate. The lesson, don't use decimalFormat() when dealing with very large numbers.

The numberFormat() Function

The numberFormat() function isn't exempt from this problem either. Internally it is doing something very similar, but it has a different approach. numberFormat() simply takes the first 18 digits starting at the left and formats them with the mask. So this code:

```
<cfset num = 123456789123456789.12>
<cfoutput>
numberformat: #numberformat(num,'999,999,999,999,999,999,999.99')#
</cfoutput>
```

Returns a *nearly correct* result of 123,456,789,123,456,784.00. What's wrong with this result? It has 2 zeros (.00) trailing it. It should have .12 at the end? So the function returns the first 18 digits from the left and the mask inserts the zeros. This can lead to odd (meaning strange) numbers being returned. for example, let's add a few digits to the front of our number:

```
Adding a 9 to the left side:
<cfset num = 9123456789123456789.123>
<cfoutput>
numberformat: #numberformat(num,'999,999,999,999,999,999,999.99')#
</cfoutput>
Result is: 9,123,456,789,123,457,000.00

Adding two 9s to the left side:
<cfset num = 99123456789123456789.123>
<cfoutput>
numberformat: #numberformat(num,'999,999,999,999,999,999,999.99')#
</cfoutput>
Result is: 99,123,456,789,123,450,000.00
```

The function maintains the correct number of decimal places but only populates the first 16 places. That's strange because it seemed to handle 18 places until the 19th digit was added. In either case (decimalFormat() or numberFormat()) you are going to get innacurate results when dealing with very large numbers.

Obviously if you are working with such large numbers it's time to consider a different approach anyway - scientific notation is a good place to start. Still, it's worth remembering that outputting numbers using these 2 functions *does* have it's limitations. I will leave you with a neat trick I just learned from Jacob Munson on CF Talk.

numberFormat() Mask Tip

If the reason you choose decimalFormat( ) over numberFormat() is because you don't like typing out long masks, this tip is worth it's weight in Gold. Instead of typing out a long mask, try a comma, a period and two 9s - as in ",.99". This code:

```
<cfset num = 99123456789123456789.123>
<cfoutput>
number format ',.99': #numberformat(num,',.99')#
</cfoutput>
```

Produces the exact same result as the long mask of 9's above. Thanks Jacob for a

Produces the exact same result as the long mask of 9s above. Thanks Jacob for a *great* time saving tip.