Cfdocument and Performance

Posted At : February 16, 2006 5:16 PM | Posted By : Mark Kruger
Related Categories: Coldfusion MX 7, Coldfusion Tips and Techniques

> CF Muse Reader Asks:
> I am using CFMX7 on Windows 2003. We are seeing some serious processor peaks when the Cfdocument tag takes off to write out a report. We have the latest hot fixes in place...any suggestions?

Cfdocument is a new weapon in the Coldfusion arsenal. It's not without it's detractors, but personally I think it is splendid for what it can do. When it comes to performance, however, there are a number of things to keep in mind. If you think about what Cfdocument is being tasked to do *under the hood* and I'm sure you could come up with a few as well. Since it is my blog, we might as well work from my list...

Images

This issue initially caused quite a few headaches with developers. The first release of CF 7 (pre-Merrimack) had a significant bug when trying to render an image. The image would be resampled much larger on the page. Some developers discovered that setting the image to a percentage (from 65% to 75%) of the original size caused it to render *nearly* correct. Others came up with clever ways of dictating sizing to the image like embedding it in a table cell as a background image so that the cell properties would dicate size, as in this example:

```
<table>
<tr>
<td width="490" height="118" background="Images/my_image.gif">
</td>
</tr>
</table>
```

This problem has improved a great deal with CF 7.01, but it is still a challenge to get images to render correctly. Why? Because under the hood CF uses iText classes to do it's formatting. These classes often must resample the image using methods that maintain perspective, but *change the dpi*. In other words, they are mangling the file a bit to make it work in the new format. This is, naturally, a processor intensive operation. The larger or higher resolution the image file, the more processor intensive it becomes. So my first bit of advice is to try your best to embed images with a width and height parameter and "pre-process" the images so they can be used "as is" in the PDF file.

Think Print!

HTML allows you to do cool things that you simply *can't do* in a printable document. A print document can't "expand or contract" based on the user environment. It has a finite set of boundaries. Therefore, when you think of the HTML you are going to be rendering you should take as much of the "unknown" out of the equation as possible. Try using absolute values for width and height attributes. Don't nest tables. Set margins that make sense. Use values that make sense for printing. For example, this snippet (which I confess I have not used) comes from a sample posted on livedocs.

```
<style type="text/css">
```

```
   body {
    width: 8in;
    height: 10.5in;
    margin: .25in;
   }
</style>
```

Maybe you forgot, or never knew, that you could use "inches" as values in CSS. While it doesn't make sense to do it in HTML, it makes perfect sense in a printable document. Why does this impact performance? Because the rendering engine doesn't have to run calculations to determine the width of a table or div tag. It "knows" the width in advance. You see, when you set a width of an item to 80% on an HTML page you don't really think about what that means. The browser must first determine the width of the container object, which might be a document, table, div tag etc. Then it determines the width of the item in question relative to that original value.

As a developer you are not privy to this *behind the scenes* calculation. It happens on the client - and you are using the power of the desktop to do some of your work for you (good for you!). But Cfdocument has to play both client *and* server. If you use a value like 80% the rendering engine will have to work out the math. Things get more complicated when the item is *nested*. For example, if you produce a table that is 90% wide with a nested cell that is 50% wide and a nested table that is 90% with 2 cells that are 40% and 60% (whew) think of the process for rendering the inner most cell at 40%. If you could be a bug in your computer... er... I mean a fly on the wall in your computer.. you might hear something like this:

- Cfdocument: What is the width of this item?
- Document Fairy: 40% of the containing object
- Cfdocument: Cool.. what's the width of the containing object then?
- Document Fairy: 90% of *it's* containing object
- Cfdocument: O...k... then what is the width of that containing object?
- Document Fairy: I'm sorry.. which object?
- Cfdocument: The object containing the 90% object containing the 40% object..?
- Document Fairy: Right, it's 50%.
- Cfdocument: Excellent excellent... [drumming it's little cf fingers]...50% of what?
- Document Fairy: [triumphantly as only a fairy can be] ... it's containing object.
- Cfdocument: Yes... right... ok... what's the width of that containing object?
- Document Fairy: What containing obj.....
- Cfdocument: [interrupting] Look, I don't have all day! It's been like... 20 milliseconds already... What is the width of the object containing the 50% object containing the 90% object that contains the 40% object?
- Document Fairy: [A little fairy sob in her voice] ... there's no need to get angry. I didn't write this code you know. I'm only a figment of CF Muse's imagination anyway...
- Cfdocument: [soothing] ... yes... all right, sorry. The width?
- Document Fairy: [clearly back in happy land] It's 8 and 1/2 inches - the width of the page!
- Cfdocument: Right! excellent... let's see, 90% of 8.5 is 7.65. 50% of that is 3.825. 90% of that is 3.4425 (dang! I wish they would let us round). And 40% of that is 1.377 inches. Excellent! What's the width of the next cell?
- Document Fairy: 60% of the containing obj...
- Cfdocument: Arggh!!

You can see how taking the process of determining widths out of the equation would

benefit the processing speed.

External resources

The final thing that comes to mind is one you may not have thought about. An HTML file is not really a *self contained* entity. When the page is received by the browser it does a "catalog" of all the resources needed to render that page. Then it goes out and *gets* all those external files - images, css, flash, Javascript etc. What you see on the page is generally a result of *many separate http requests*, not just the one that called the page.

A PDF file on the other hand really is a *document* in the traditional sense. All the resources it needs to be viewed must be embedded in the file. The only exceptions are fonts. With fonts you can choose to embed them or not embed them. If you *don't* embed them, then the PDF will display with whatever fonts the user has installed. If they match, the PDF will display correctly.

This "encapsulation" principle of a PDF file has implications for Cfdocument. Take the example of an image embedded in a document. Let's say I have the following:

```
<cfdocument ...>
      <img src="/images/mygif.gif"/>
   </cfdocument>
```

If this image tag was embedded on an HTML page, the *browser* would be tasked with going out and grabbing the image and rendering it on the page. This task would occur *subsequent to the inital request*. But Cfdocument doesn't have that option. Not only that, Cfdocument must embed the *contents* of the file into the document - not merely a pointer to be handled by the browser. How is this accomplished? Presumably, Cfdocument opens the file, serializes it with the correct "PDFish" formatting for image and sticks it in the right spot in the document.

Now consider, in the example above, how does CF knows where the file is? Answer: CF will use regular file mappings starting with the location of template containing the Cfdocument call. In other words, it treats it like a Cfinclude. Of course this causes problems at times because you don't really think of images relative to templates. You are more likely to think of them relative to the web root. So perhaps you decided on this elegant solution. You might simply convert this to an absolute path as in:

```
<cfdocument ...>
      <img src="http://www.mydomain.com/images/mygif.gif"/>
   </cfdocument>
```

This works well. The image is rendered correctly. But consider what is going on behind the scenes. The server is:

- Resolving www.mydomain.com using DNS or a HOST file.
- Sending an HTTP request via Cfhttp - even if the file is on your own server, CF will still be using CFHTTP to get it.
- Serializing the contents of the file and placing it into the PDF file - I don't know, but the file system could be involved here as well.

The same scenario may be drawn for CSS files, movies, charts, fonts etc. These resources have to be *gathered* by Cfdocument and *embeded* into the document.

Personally, I consider this latter issue to be the most likely cause of performance

problems. A slow server at the other end of a cfhttp request (even your own server) could cause a hanging thread while Cfdocument waits for the contents of that file. A PDF with a lot of external resources (like a catalog for example) would really hammer the resources of your server. DNS issues are *always* problematic on web servers and *rarely understood*.

I'm afraid I didn't answer your question in a way that points you in one direction. Still, I hope there are some clues here for you. If anyone has any other Cfdocument tips feel free to comment. Keep those questions coming.