Sessions and Cookies and Bots (oh my)

Posted At : November 28, 2005 6:33 PM | Posted By : Mark Kruger
Related Categories: Coldfusion Optimization, Podcasts, Coldfusion Tips and Techniques

Would you like to know how to create your own memory leak using the design of the Coldfusion Server to do it? Here's one way. Let's say you have a site that sells products from Narnia. It has a root folder that display your products and prices. You've done a great job of creating friendly links for browsing your Narnia products. You have stuffed Aslan lions both friendly and fearsome, White witch figurines, fauns, nyads, dryads, a toy lamppost and even a wardrobe for sale. Let's say (for the sake of argument) that you have 50 links to Narnia products just on your home page. If a user chooses to buy one of your products he or she clicks on "add to cart". At this point the user is taken into the "/shop/" folder to the page at "www.Nnarniaproducts.com/shop/cart.cfm". So far so good. This is how many online stores are organized and it's just peachy. But let's look under the hood shall we.

Listen Here

Sharing the Application.cfm (or Application.cfc) page

Where you place your cfapplication tag and how you configure it's attributes is very important. For example, it might make sense to put an application.cfm page in root because you have variables that are instantiated and shared between the shopping cart and the main browse pages. So it might make sense to simply include them in one big application. with a single "cfapplication" tag like so...

```
<cfapplication
      name="narnia_products"
         sessionmanagement="Yes"
            setclientcookies="yes"....>
```

You will note that we have enabled session management. Now the truth is, we don't need session management until the user puts something in his or her shopping cart, but since we are sharing the application.cfm page we go ahead and enable it here. When the user adds that big stuffed Aslan to his shopping cart and we need a session variable, we will be able to create one with no problem.

What happens when we do it this way? When a user first arrives on the home page 2 variables - CFID and CFTOKEN - are created. They are placed in memory under the "application" and a cookie is sent to the browser. The browser's responsibility is to return the cookie with each additional request. Subsequent requests "see" the cfid and cftoken, and match them with the data stored in memory. If there are any session variables, they are stored in memory using the CFID and CFTOKEN as a key to figure out which session variables belong to which user. The application knows to set a *new* CFID and CFTOKEN when *it examines the request and finds no existing CFID and CFTOKEN*.

These 2 variables are the basis for the "session". The server hangs on to these variables in memory until the session times out or the server restarts. You can easily test this by outputting the CFID and CFTOKEN on a test page. Output them several times and you will see they remain the same. Then delete all cookies and refresh the page. The values will change. Consider that if you crafted requests devoid of any cookies or url variables (when they are used for sessions) you would, in effect, *be creating a new*

session with each request.

```
<cfoutput>
   #cookie.cfid#
    <br>
    #cookie.cftoken#
</cfoutput>
```

Bots and the Train they Came In On

Actually, *this is exactly how a bot works*. An indexing bot retrieves your home page. The CF server sets a CFID and CFTOKEN in memory and sends back the cookie header with the response. The bot takes stock of your home page and parses out all the links. It then goes out and retrieves each link and indexes the content - exactly what you want it to do. But because a bot, by design, ignores cookie headers, it does not send back the CFID and CFTOKEN with each subsequent request. In effect, every request of the bot is creating a new session. In the case of our Narnia product site - an unneeded session. If you have thousands of products and you want your site to be heavily indexed this can create a resource issue for you. An aggressive bot on a large site can create hundreds and even thousands of *phantom sessions* that take up space in memory until they expire.

Solutions

Be careful how you configure sessions and the application.cfm page. Using a single Application.cfm or Appliciation.cfc page might not be the best solution for you. You might want to configure 2 of them - one with and one without sessions. In most cases, a "session aware" shopping cart can have it's own cfapplication tag while a "public facing" area can have a separate one. You can still use a shared "include" file to populate shared variables and functions - giving you a reasonable level of modularity. Using a robots.txt file to exclude the shopping cart area is also a good idea.

Please note, that while I'm explaining using the CFID and CFTOKEN, there is an alternate variable called "JSESSIONID" that can be used. You can enable JSESSIONID in the Coldfusion admin. jsessionid is a 1 way Hash and it takes the place of both CFID and CFTOKEN variable. In most cases I recommend that you use jsessionid - especially for a new project. However, there are many legacy Coldfusion applications out there that depend on the existence of the CFID and CFTOKEN variable - so you have to take it on a case by case basis.