Debugging and a Return to Dodge City

Posted At : November 1, 2012 10:49 AM | Posted By : Mark Kruger
Related Categories: Coldfusion Troubleshooting

One of the things the Muse likes best about ColdFusion is the excellent debug information provided during development. Of course you should never ever leave debugging enabled on a production server. Not only are you generating a great deal of additional data with each request (adding overhead), you are potentially exposing a mother lode of technical information that a nefarious hacker would salivate to see. But during development, the debug information is where you ought to live. Indeed, if you are not constantly checking the debug information start doing it now - make a habit of it! You will learn things about performance, iterations, database interactions, cookies, paths, and all sorts of goodies that will make you a better programmer.

I've had my head buried in the debug information since I started with ColdFusion. Back then (in the Wild West days of CF 4.01) we never heard of newfangled ideas like "cfqueryparam". We just stuffed our variables into queries willy nilly and trusted the good Lord to protect us. It feels like I have spent the last 7 or 8 years cleaning up after code written like that. But writing queries in the raw (unprotected I mean... I don't generally code naked, although I did experiment in college) had one main advantage. As you probably know a lot of debugging goes back to the database. The debug output pre-cfqueryparam was "well formed" query code that could be copied and pasted directly into a query tool like MSSQL studio or Navicat. This made debugging pretty easy. You could swipe a problem query out of the debug, run it and tweak it unit it gave you what you needed, then past it back into CF. But that changed when we all started using CFQUERYPARAM.

Now that we are all using CFQUERYPARAM (and if you are not you *should* be) to protect against SQL Injection (SQLi), the classic debug template outputs the query like this:

```
qry_***query name*** (Datasource=**dsn***, Time=1ms, Records=1) ...
 SELECT  someID, name
 FROM infoSummary
 WHERE itemID = ?
 AND  isDefault = ?
 AND  userID = ?

Query Parameter Value(s) -
Parameter #1(CF_SQL_INTEGER) = 1
Parameter #2(CF_SQL_INTEGER) = 1
Parameter #3(CF_SQL_INTEGER) = 1
```

In order to use such output you have to paste it into your query analyzer and then replace all the question marks with the values from the parameter array below the query.

Yesterday @mrbusche tweeted that in Railo, when you use cfqueryparam, the debug output is still fully pastable. That got me to thinking. Why do I put up with debug output that is *not* pastable. If I want it to be pastable it should be pastable right? After all, the debug template is open and editable by little old me. Why not edit it to reinsert the SQL values back into the query. So that's what the Muse did.

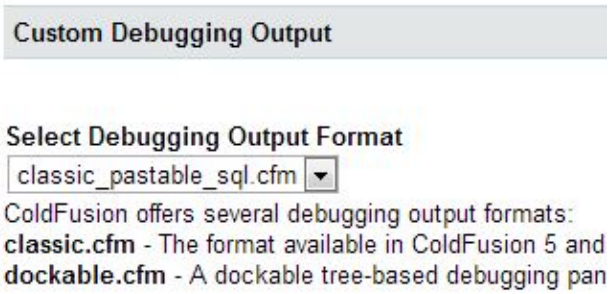
Step 1 - Clone the Debug Templates

First, I navigated to my CF installation (in this case I was on a CF 9 server) and located the

"classic.cfm" template. In a multi-server install it is under:

```
JRun4\servers\**instance name**\cfusion.ear\cfusion.war\WEB-INF\debug
```

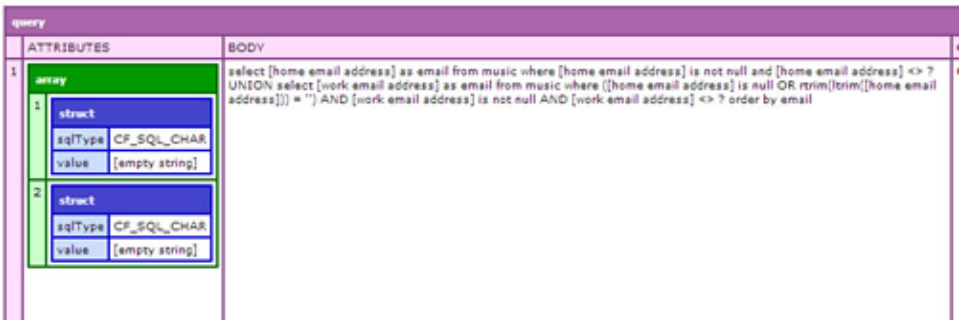
In a standard install look under wwwroot/WEB-INF for the debug folder. I made a copy of this file in the same directory and called it classic_pastable_sql.cfm. Then I logged into the CF Admin and went to the "debug output settings". I selected my new template listed under "select debugging output format" like so:



Now that it was selected I could make changes to it as needed without altering the original "classic" template. So far so good.

Step 2 - Unpack the Query Output Code

Now I needed to know how the debug output was gathered. It turns out that the debug template creates ColdFusion queries of the query code and then loops through these queries to output the data onto the screen. The query name is *cfdebug_queries*. A column called "body" contains the SQL statement and a column called "attributes" contains an *array* of structures - each having *sqltype* and *value* as members. SqlType Contained the original "cf_sql_**TYPE**" constant passed to the cfqueryparam and value contained whatever data was passed. Here's a dump of those 2 columns:



Step 3 - Code to Reinsert the Params back into the query.

So the trick would be to replace the question mark placeholders with the values from the array while inserting single quote marks where appropriate. Now I did not need this to be perfect - just acceptable for the lion's share of queries I might tease out of the debug. So I created a function like so:

```
<cfscript>
function debugResetSQL (body, attr) {

        var i = 1;
        var testlist = "CHAR,DATE,IDSTAMP,LONGVARCHAR,VARCHAR,TIME,TIMESTAMP";
        for(i = 1; i LTE arrayLen(attr); i++) {
            IF(listfind(testlist,listlast(attr[i].sqlType,'_'))){
                body = replace(body,"?","'" & attr[i].value & "'");
```

```
            } ELSE {
                body = replace(body,"?",attr[i].value);
            }
        }

        return body;
    }
</cfscript>
```

This function simply loops through the array of cfqueryparam values and matches question marks it finds in the body of SQL with the current value from the array - adding or not adding single quotes along the way. I put single quotes around character values, hashes, and time and date - trusting implicit conversion on the database server to do some work on my behalf. I put this function into my new debug script (after testing).

Next I had to figure out where the script output the query information. I found what I wanted on or around line 570. It looked like this:

```
#htmleditformat(cfdebug_queries.body)#

<cfif arrayLen(cfdebug_queries.attributes) GT 0>
Query Parameter Value(s) -<br />

<cfloop index="x" from=1 to="#arrayLen(cfdebug_queries.attributes)#">
<cfset thisParam = #cfdebug_queries.attributes[cfdebug_queries.currentRow][x]#>
Parameter ###x#<cfif StructKeyExists(thisParam, "sqlType")>(#thisParam.sqlType#)</cfif> =
<cfif StructKeyExists(thisParam, "value")>#htmleditformat(thisParam.value)#</cfif><br />

</cfloop>
</cfif>
```

I left the array loop in place as additional info on the theory that more is better. I changed the output of the "body" variable to look like this:

```
<cfif arraylen(cfdebug_queries.attributes)>
    <pre>#htmleditformat(debugResetSQL(cfdebug_queries.body,cfdebug_queries.attributes))#</pre>
<cfelse>
    <pre>#htmleditformat(cfdebug_queries.body)#</pre>
</cfif>
```

The result is that my queries are output cleanly and in a pastable format - with the array and types below like before. The output looks like this:

```
getemail (Datasource=***, Time=4ms, Records=112)....
select [home email address] as email
from music where [home email address] is not null
and [home email address]   ''
UNION
select [work email address] as email
from music where
 ([home email address] is null  OR rtrim(ltrim([home email address])) = '') AND
[work email address] is not null
AND [work email address]   ''
order by email

Query Parameter Value(s) -
Parameter #1(CF_SQL_CHAR) =
Parameter #2(CF_SQL_CHAR) =
Parameter #3(CF_SQL_CHAR) =
```

A few Caveats

Of course with different database servers you might get different results and need to further tweak this code. It also doesn't really differentiate between a question mark added as a placeholder and one that is part of a query in some fashion. I could foresee some queries being mangled because of that. But all things being equal I suspect it will work for 99.9% of the queries on my dev installation.

So now I have the best of both worlds. Here's a link to the modified debug template. Perhaps Muse readers will find it useful. A great deal more can be done with custom debug templates to make life easier for the developer. I have an example I will share next time - as soon as I am confident it is working as expected. Happy debugging!