Data Binding Without Using Cfqueryparam

Posted At : July 7, 2005 11:27 AM | Posted By : Mark Kruger
Related Categories: Coldfusion & Databases, Coldfusion Tips and Techniques

I am always rhapsodizing on the benefits of CFQUERYPARAM. But what if you needed to *not* use CFQUERYPARAM? Is it possible to get the benefits of the tag without actually needing to USE the tag? Why yes it is! In order to explain let's look under the covers of how an SQL statement is prepared when you use binding. How about a little lesson from Classis ASP?

A Classic Example

In classic ASP you must build a string containing SQL and pass it into a database connection. Consider this example:

```
<%
set conn =    server.CreateObject("ADODB.Connection")
conn.open    connectionsStr
set rsRelo = conn.Execute("Select name,email from Users where UserId=" & intUser)
%>
```

Note that this code uses the "Execute" method of the ADODB object. The execute method is analogous to using CFQUERY *without* cfqueryparam. It says, to the DB Server, "Here... see what you can make of this and get back to me." To bind parameters in ASP you use the function (oddly enough) "SQLBindParameter", as in this example taken from this **excellent tutorial** on optimizing ODBC (note - this is NOT ASP code - but it could be):

```
{
SQLINTEGER val = 10;
SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_INTEGER, SQL_C_LONG, 0, 0, &val, 0, NULL);
SQLExecDirect(hstmt, "SELECT C1 FROM T1 WHERE C1 = ?", SQL_NTS);
}
```

In this case the database is informed as to the type and can readily access the execution plan if available in the cache (see "**Why you should worry about your execution plan**"). You will notice that the statement itself, "SELECT C1 FROM T1 WHERE C1 = ?", looks familiar. It's exactly like the output of the debug information when you *are* using CFQUERYPARAM. That should tell you how CFQUERYPARAM does it's thing.

Under the Hood

You see when you use CFQUERYPARAM, the driver "prepares" the statement and creates temporary "stored procedure" for it. Let's say we have this code.

```
<cfquery name="getUser" datasource="#dsn#">
      SELECT    name, Email, userId
      FROM   users
      WHERE   username = <cfqueryparam cfsqltype="CF_SQL_CHAR" value="joe">
      AND      pin = <cfqueryparam cfsqltype="CF_SQL_INTEGER" value="5555">
   </cfquery>
```

What actually get's passed to the db server? Something like this:

```
<cfquery name="getUser" datasource="#dsn#">
   DECLARE   @param1 varchar(50)
   DECLARE @param2 int = 5555
```

```
   set @param1 = 'joe'
   set @param2 = 5555

   SELECT    name, Email, userId
   FROM   users
   WHERE   username = @param1
   AND      pin = @param2
</cfquery>
```

The code above is runnable as well - and it has the same benefits as cfqueryparam. In other words, the 2 queries perform the same function *and both bind the data and data types*. You could use either of them and get the benefits of data binding. In the words of Hugh Neutron, "Now you gotta admit that's pretty neat!".

Why would you want to use the latter code and not the former? Well, sometimes it is convenient to build a "query string" and pass it into the cfquery tag. Not often, but there are cases where the construction of the query is super-dynamic, like when the columns or the where clause are pulled from a db - or when the table isn't known in advance. In those special circumstances having a way to do data binding could be an important tool for you.

What do you lose? CFQUERYPARAM does have the benefit of escaping special characters and scrubbing the data for you. I can imagine there is an obscure way to implement an SQL injection attack on the latter code, that would not be effective when using cfqueryparam (though any attempt is far more likely to simply produce errors). Still, I like to think of this technique as one more arrow in my developer's quiver.