# Handling Variable Form Data in a Stored Procedure

Posted At : August 13, 2007 9:49 PM | Posted By : Mark Kruger
Related Categories: MS SQL Server, Coldfusion & Databases

Lots of projects have a requirement that interaction with a database must be done using stored procedures only. Stored procedures are generally quite easy to write, but there are some things that are slightly more difficult that using a straight CFQUERY. For example, perhaps you have seen code that handles a search form. You might see a query that looks something like this:

```
<cfquery name="get" datasource="#dsn#">
    SELECT * FROM USERS
    WHERE    userID IS NOT NULL

    <cfif NOT isEmpty(form.username)>
        AND UserName = '#form.username#'
    </cfif>
    <cfif NOT isEmpty(form.address)>
        AND Address = '#form.address#'
    </cfif>
</cfquery>
```

Please note, I'm use a UDF called "isEmpty" that simply trims the string and checks the length. Also keep in mind that I'm not adding the required Cfqueryparam to save room. How would you duplicate this code in a T-SQL stored procedure?

### Sample Stored Procedur

Ok, let's have a go at that stored procedure (I'm putting inside of a cfquery to get the color coding).

```
<Cfquery>
CREATE PROCEDURE dbo.usp_SearchUsers
     (
        @username varchar(50),
        @Address varchar(50)
         )

WITH RECOMPILE
AS
SET NOCOUNT ON

SELECT * FROM USERS
WHERE     username = @username
AND        Address = @address
</cfquery>
```

Please note that I am omitting the nocount and ansi_nulls stuff to save space. If you look at this stored procedure you can see that it compares the username and address with the variables passed in. This is not exactly what we were looking for. In the code above I check the form variables and I only compare them if they are not blank. In other words, the form is smart enough to only check variables the user is filling in. How do I do that in a stored proc?

If you are like me you might jump to the conclusion that you are going to need a temp table or a cursor - or perhaps a block of code for each case. But thankfully none of

those are necessary. You can solve this problem by remembering that it is possible to compare a variable with something inside of a WHERE clause. So you can do the following:

```
<cfquery>
CREATE PROCEDURE dbo.usp_SearchUsers
     (
        @username varchar(50) = null,
        @Address varchar(50) = null
         )

WITH RECOMPILE
AS
SET NOCOUNT ON

SELECT * FROM USERS
WHERE     (@username IS NULL OR username = @username)
AND        (@address IS NULL OR Address = @address)
</cfquery>
```

Ahah! Or as my daughters homey former sixth grade teacher would say, "well bless my thumpin' gizzard!" It turns out I can check for a null within each of the where clauses that I care about. The key features that have changed are the assignment of a null during the input variable declaration (**varchar(50) = null**) and the addition of check within the where clause while examining the variable (@username IS NULL OR username = @username). Of course you could also pass in an empty string and do something like "@address = '' OR Address = @address", and that would work in this case. But it would fail in the case of a date or numeric value because an empty strings is of course *not* a date or a number. Nulls are explicitly "nothing" and better represent (in this case) what we are describing.

There is just one thing left to do and that is to alter your stored proc call to account for nulls. The Procedure above could be called with something like this.

```
<cfstoredproc datasource="#dsn#" procedure="usp_SearchUsers">

    <cfprocparam cfsqltype="CF_SQL_VARCHAR"
            value="#form.username#"
            null="#YesNoFormat(Not Len(form.username))#" />,

    <cfprocparam cfsqltype="CF_SQL_VARCHAR"
            value="#form.address#"
            null="#YesNoFormat(Not Len(form.address))#" />

        <cfprocresult name="get" resultset="1"/>
</cfstoredproc>
```

The important item in this code is the attribute for null (**null="#YesNoFormat(Not Len(form.address))#"**). This inline code returns a YES or a NO. When form.username is an empty string it will return a YES - resulting in a NULL being sent to the stored procedure. When that happens the SP line that reads "@username IS NULL OR username = @username" evaluates as true and the username column is not examined.