

SQL Stored Procedures, INSERT Template script

Posted At : 4 March 2009 11:08 | Posted By : Shaun McCran
Related Categories: Software Architecture, Best practices, SQL

This article deals with creating a SQL stored procedure for inserting a record.

In each of these stored procedure templates I am declaring a variety of documentation parameters in the header.

I've found these handy in the past when you are working in a team environment, or when you go back to a procedure at a later date. Its much easier to read a simple description in the header, than trawl through the SQL code looking for what it is doing.

So, this declares the procedure name, any parameters and return codes, and also details what it does, and who made it.

In a modified version of this I also hold the SVN revision number here.

```
/* ***** */
/*          Company Name          ***** */
/* Procedure Name   : dbo.ssp_stored_procname   */
/* Parameters       :                           */
/* Return Codes     :                           */
/* Description      : Description of what it does, params etc */
/*                                                         */
/*                                                         */
/* Author           : Authorname                 */
/* Date written     : Date                       */
/* History          : version number             */
/* ***** */
```

The next block of code performs a select on the sysObjects table (part of the Master database). It is checking for the existence of itself. If it finds itself, it will drop the procedure. Note that throughout all of these scripts we are telling the user at each stage what is going on, by printing useful english output back to the screen.

```

SELECT 1 FROM sysobjects where id = object_id('dbo.ssp_stored_procname') and syssta
        BEGIN
        PRINT 'Dropping old version of dbo.ssp_stored_procname'
        DROP PROCEDURE dbo.ssp_stored_procname
        END
        GO
    
```

By now we have identified whether or not the procedure previously existed, and if it did, we have dropped it, so we know that we are all good to go. So to create our Insert procedure, we print out a message to the user, then using the "CREATE PROCEDURE" command we create our procedure.

At this point you substitute the "@field" value with your field name, and the [datatype] and (datasize) with the correct values. As an example of this the first line in this proc will create a field named 'varName' with a data type of varchar with a character limit of 100. Just list your fields one after another, separating with a comma.

```

        PRINT 'Creating procedure dbo.ssp_stored_procname - START'
        GO

        CREATE PROCEDURE dbo.ssp_stored_procname
        (varName [varchar](100),
        @field [int],
        @field [datatype](datasize),
        @field [datatype],
        @field [datatype],
        @field [datatype])
    
```

After that we create the SQL code, as per usual. We have an Insert statement, listing the fields, and the SQL variable declaration (@var). Then we check for any errors, and return a success message if it all worked ok!

```
AS INSERT INTO table
    ([field],
     [field],
     [field],
     [field],
     [field],
     [field])
VALUES
    (@field,
     @field,
     @field,
     @field,
     @field,
     @field)

RETURN @@ERROR
GO

PRINT 'Creating procedure dbo.ssp_stored_procname - END'
GO
```

By using a script like this I've found that its really simple to have a repeatable standard process that is easy to implement across a team of developers, ensuring you get the same results, no matter who writes the query. It is also very useful if you have a seperate implementation team, as these scripts are re-runnable, they clear up after themselves.

Download the full template [here](#) .