(T6)討論StoredProcedures搭配Asp.NetWebForm的SearchBar  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T6)討論StoredProcedures搭配Asp.NetWebForm的SearchBar  
=======================================================================  
0. Introduction

0.1. What to learn

0.2. In Summary

-----------

1. Create Sample Data

2. Store Procedure

3. Create Stored Procedure with parameters

4. Alter the stored procedure, WITH ENCRYPTION

5. Delete stored procedure

-----------

6. Stored Procedure output parameter V.s. Stored Procedure Return Value

6.1. Stored Procedure output parameter 1

6.2. Stored Procedure output parameter 2

6.3. Stored Procedure Return Value 1

6.4. Stored Procedure output parameter 3

6.5. Stored Procedure Return Value 2

-----------

7. sp\_help, sp\_helptext, sp\_depends

8. Create Sample Data

-----------

9. Web Application - Stored Procedure Optional Parameters

9.1. Set up SQL Authentication

9.2. Create Web Application

9.3. Code

9.3.1. Web.config

9.3.2. WebForm1.aspx

9.3.3. Default.aspx.cs

9.3.4. Run it  
=======================================================================

0. Introduction

0.1. What to learn

- Stored Procedure is a group of TSQL

- Create/Alter/Drop Procedure

- Stored Pocedure Output parameters VS Return Value

0.2. In Summary

In Summary :

1.

Reference:

<http://searchsqlserver.techtarget.com/definition/T-SQL>

A stored procedure is group of T-SQL (Transact SQL) statements.

T-SQL (Transact-SQL) is a set of programming extensions from

Sybase and Microsoft that add several features

to the Structured Query Language (SQL),

including transaction control, exception and error handling, row processing and declared variables.

---------------------------------------------------------------

2.

Create/Alter PROCEDURE

--CREATE PROCEDURE spGetAllEmployees

----ALTER PROCEDURE spGetAllEmployees

--(

--    @parameterA INT,

--    @parameterB INT OUTPUT

--    --@parameterB INT OUT

--) --WITH ENCRYPTION

--AS

--    BEGIN

--        ...

--    END;

--GO

2.1.

--WITH ENCRYPTION

Once encrypted, you can not read or modify the procedure text.

2.2.

All parameter and variable names in SQL server, need to have the @symbol.

2.3.

use "sp" prefix means stored procedure.

Don't use "sp\_" prefix because "sp\_" prefix is for system stored procedure

---------------------------------------------------------------

3.

Delete PROCEDURE

--DROP PROCEDURE spGetAllEmployees

--GO

---------------------------------------------------------------

4.

Stored Pocedure Output parameters VS Return Value

-----------------------------

4.1.

Stored Pocedure Return Value

--CREATE PROCEDURE spGetCountAllEmployees2

--AS

--    BEGIN

--        RETURN

--        ( SELECT    COUNT(e.EmployeeID)

--          FROM      dbo.Employee e

--        );

--    END;

--DECLARE @TotalEmployees2 INT;

--EXECUTE @TotalEmployees2 = spGetCountAllEmployees2;

--PRINT @TotalEmployees2;

4.1.1.

Stored Pocedure Return Value can ONLY return ONE INTEGER value.

-----------------------------

4.2.

Stored Pocedure Output parameters

--CREATE PROCEDURE spGetCountAllEmployees1

--    (

--      @TotalCount int OUTPUT

--    )

--AS

--    BEGIN

--        SELECT  @TotalCount = COUNT(e.EmployeeID)

--        FROM    dbo.Employee e;

--    END;

--DECLARE @TotalEmployees INT;

--DECLARE @@Status\_spGetCountAllEmployees1 INT;

--EXECUTE @@Status\_spGetCountAllEmployees1

--    = spGetCountAllEmployees1 @TotalEmployees OUTPUT;

--PRINT @TotalEmployees;

--PRINT @@Status\_spGetCountAllEmployees1;

4.2.1.

Stored Pocedure Output parameters can output more than one value and any Data type.

E.g. Output string, Data time, int, ....etc.

-----------------------------

4.3.

When you execute a stored procedure, it will always return an integer value.

5.3.1.

If you use "Stored Pocedure Output parameters",

then you will also get a "return integer status value".

zero means success, and non-zero means failure.

4.3.2.

If you use "Stored Pocedure Return Value",

then you will still get "return integer value".

But this "return integer value" is not "status value" any more.

It is whatever value which was returned by stored prcedure.

4.3.3.

In SSMS,

Database Name --> Programmability --> Stored Procedures -->

Stored Procedure Name --> Right click --> Execute Stored Procedures

--> then you will see ONE RETURN INTEGER VALUE.

If you use output parameters,

then you will also see the value of output parameters

and ONE RETURN INTEGER VALUE indicates the status,

which 0 means successful.

---------------------------------------------------------------

5.

system stored procedures for help

5.1.

--sp\_help databaseObjectName

Same as you highlight all kind of database object

such as stored procedure name, table name, view name, trigger name ...etc.

and then press Ctrl + F1

Then you will see all the information regarding the database object.

------------------------------

5.2.

--sp\_helptext spName

See the stored procedure text.

Only for stored procedure.

------------------------------

5.3.

--sp\_depends databaseObjectName

See the dependencies of database object.

E.g.

--sp\_depends spName

you will see what table and what column were used in this stored procedure.

Thus, before you delete or edit these columns' name,

you have to edit this stored procedure first.

E.g.

--sp\_depends tableName

Show you all the stored procedure or

any other database object

which were created by this table columns.

Thus, before you delete or edit this table columns' name,

you have to double check if it will affect these database objects first.

---------------------------------------------------------------

6.

6.1.

Execution plan of Stored Procedure is reusable.

-->

--Select \* from Employee WHERE EmployeeID=1

When you execute the queryA at first time.

It will create Execution plan.

Thus, it will be quicker when execute this query in second time.

Because the Execution plan has been already created,

and it will use the same Execution plan

--Select \* from Employee WHERE EmployeeID=2

It only change to EmployeeID=2

but it will create another Execution plan.

Thus, these kind of Execution plan is not usable.

--CREATE PROCEDURE spGetNameById1

--    (

--      @Id int ,

--      @Name nvarchar(50) OUTPUT

--    )

--AS

--    BEGIN

--        SELECT  @Name = e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName

--        FROM    dbo.Employee e

--        WHERE   e.EmployeeID = @Id;

--    END;

--DECLARE @EmployeeName1 NVARCHAR(20);

--DECLARE @EmployeeName2 NVARCHAR(20);

--EXECUTE spGetNameById1 1, @EmployeeName OUT;

--EXECUTE spGetNameById1 2, @EmployeeName OUT;

--PRINT 'Employee1 Name1 : ' + @EmployeeName1;

--PRINT 'Employee1 Name2 : ' + @EmployeeName2;

When create the stored procedure,

and then

--EXECUTE spGetNameById1 1, @EmployeeName OUT;

Then you execute the stored procedure at first time

then it will create an Execution plan of this Stored Procedure.

--EXECUTE spGetNameById1 2, @EmployeeName OUT;

When you execute the stored procedure at second time

then it will re-use this Execution plan of this Stored Procedure.

Thus, it is quicker.

------------------------------

6.2.

Less network traffic

-->

-- SELECT ... FROM ... WHERE ... ORDER BY ...

This is a very large query.

If you create a stored procedure for this query.

I just need to pass the execute stored procedure statement.

-- EXEC PROC spName

This is much shorter than large query.

Thus reduces network traffic.

------------------------------

6.3.

Code reusable and more maintainable

-->

When you change the logic in stored procedure,

it will apply any where you used this stored procedure.

It is more maintainable.

------------------------------

6.4.

Better Security

It is always better we assign the permistion of stored procedure to a database user.

Instead of we assign the dirrect permistion of tables to a database user.

Thus, a database user need to execute the stored procedure to get that table data.

It is easier to control what data a user can access to.

------------------------------

6.5.

Prevent SQL Injection

We can use some error checking to prevent SQL Injection in stored procedure.

1. Create Sample Data

--====================================================================================

--T006\_01\_Create Sample Data

--====================================================================================

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Employee' ) )

    BEGIN

        TRUNCATE TABLE Employee;

        DROP TABLE Employee;

    END;

GO -- Run the previous command and begins new batch

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Department' ) )

    BEGIN

        TRUNCATE TABLE Department;

        DROP TABLE Department;

    END;

GO -- Run the previous command and begins new batch

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gender' ) )

    BEGIN

        TRUNCATE TABLE Gender;

        DROP TABLE Gender;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE Department

(

  DepartmentID INT IDENTITY(1, 1)

                   PRIMARY KEY

                   NOT NULL ,

  DepartmentName NVARCHAR(50) NULL

);

GO -- Run the prvious command and begins new batch

INSERT  Department

VALUES  ( N'Department1' );

INSERT  Department

VALUES  ( N'Department2' );

INSERT  Department

VALUES  ( N'Department3' );

INSERT  Department

VALUES  ( N'Department4' );

INSERT  Department

VALUES  ( N'Department5' );

INSERT  Department

VALUES  ( N'Department6' );

GO -- Run the prvious command and begins new batch

CREATE TABLE Gender

(

  GenderID INT IDENTITY(1, 1)

               PRIMARY KEY

               NOT NULL ,

  Gender NVARCHAR(50) NOT NULL

);

GO -- Run the prvious command and begins new batch

INSERT  Gender

VALUES  ( N'Male' );

INSERT  Gender

VALUES  ( N'Female' );

INSERT  Gender

VALUES  ( N'Unknow' );

GO -- Run the prvious command and begins new batch

CREATE TABLE Employee

(

  EmployeeID INT IDENTITY(1, 1)

                 PRIMARY KEY

                 NOT NULL ,

  [ReportsTo] INT NULL ,

  FirstName NVARCHAR(100) NULL ,

  MiddleName NVARCHAR(100) NULL ,

  LastName NVARCHAR(100) NULL ,

  GenderID INT FOREIGN KEY REFERENCES Gender ( GenderID )

               NOT NULL ,

  DepartmentID INT FOREIGN KEY REFERENCES Department ( DepartmentID )

                   NULL

);

GO -- Run the prvious command and begins new batch

INSERT  Employee

VALUES  ( NULL, N'First1', N'Middle1', N'Last1', 1, 3 );

INSERT  Employee

VALUES  ( 1, N'First2', N'Middle2', N'Last2', 2, 1 );

INSERT  Employee

VALUES  ( 1, N'Fisrt3', N'Middle3', N'Last3', 3, 2 );

INSERT  Employee

VALUES  ( 2, N'First4', N'Middle4', N'Last4', 1, 1 );

INSERT  Employee

VALUES  ( 2, N'First5', N'Middle5', N'Last5', 2, 2 );

INSERT  Employee

VALUES  ( 2, N'First6', N'Middle6', N'Last6', 3, 3 );

INSERT  Employee

VALUES  ( 3, N'First7', N'Middle7', N'Last7', 1, 1 );

INSERT  Employee

VALUES  ( 3, N'First8', N'Middle8', N'Last8', 2, 2 );

INSERT  Employee

VALUES  ( 3, N'First9', N'Middle9', N'last9', 3, NULL );

INSERT  Employee

VALUES  ( NULL, N'First10', N'Middle10', N'Last10', 1, NULL );

GO -- Run the prvious command and begins new batch

SELECT  \*

FROM    Gender;

SELECT  \*

FROM    Department;

SELECT  \*

FROM    Employee;

GO -- Run the prvious command and begins new batch

Graphical user interface, table

Description automatically generated

Table

Description automatically generated with low confidence

==================================================

2. Store Procedure

--====================================================================================

--T006\_02\_Store Procedure

--====================================================================================

--====================================================================================

--T006\_02\_01

--SELECT

SELECT  e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName AS FullName ,

        g.Gender

FROM    dbo.Employee e

        INNER JOIN dbo.Gender g ON e.GenderID = g.GenderID;

GO -- Run the prvious command and begins new batch

/\*

Display name and gender

\*/

Table

Description automatically generated

--====================================================================================

--T006\_02\_02

--CREATE PROCEDURE ... SELECT

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetAllEmployees' ) )

    BEGIN

        DROP PROCEDURE spGetAllEmployees;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spGetAllEmployees

AS

    BEGIN

        SELECT  e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName AS FullName ,

                g.Gender

        FROM    dbo.Employee e

                INNER JOIN dbo.Gender g ON e.GenderID = g.GenderID;

    END;

GO -- Run the prvious command and begins new batch

/\*

1.

Display name and gender

2.

CREATE a stored procedure "spGetAllEmployees"

use "sp" prefix means stored procedure.

don't use "sp\_" prefix

because "sp\_" prefix is for system stored procedure

\*/

--====================================================================================

--T006\_02\_03

--EXECUTE stored procedure

spGetAllEmployees;

GO -- Run the prvious command and begins new batch

EXEC spGetAllEmployees;

GO -- Run the prvious command and begins new batch

EXECUTE spGetAllEmployees;

GO -- Run the prvious command and begins new batchs

/\*

--spGetAllEmployees;

--EXEC spGetAllEmployees;

--EXECUTE spGetAllEmployees;

3 Query ways to execute the Stored Procedure.

or

Database Name --> Programmability --> Stored Procedures -->

Stored Procedure Name --> right click --> Execute Stored Procedure.

\*/

Graphical user interface, text, table

Description automatically generated

==================================================

3. Create Stored Procedure with parameters

--====================================================================================

--T006\_03\_Create Stored Procedure with parameters

--====================================================================================

--====================================================================================

--T006\_03\_01

--SELECT

SELECT  e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName AS FullName ,

        g.Gender ,

        d.DepartmentName

FROM    dbo.Employee e

        INNER JOIN dbo.Gender g ON e.GenderID = g.GenderID

        INNER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID

WHERE   e.GenderID = 1

        AND e.DepartmentID = 1;

GO -- Run the prvious command and begins new batch

/\*

Display name and gender and DepartmentName

with the condition, e.GenderID = 1 AND e.DepartmentID = 1

\*/

Graphical user interface, table

Description automatically generated

--====================================================================================

--T006\_03\_02

--Create Stored Procedure with parameters

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetEmployeesByGenderIDAndDepartmentID' ) )

    BEGIN

        DROP PROCEDURE spGetEmployeesByGenderIDAndDepartmentID;

    END;

GO -- Run the previous command and begins new batch

CREATE PROC spGetEmployeesByGenderIDAndDepartmentID

(

  @GenderID INT ,

  @DepartmentID INT

 )

AS

    BEGIN

        SELECT  e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName AS FullName ,

                g.Gender ,

                d.DepartmentName

        FROM    dbo.Employee e

                INNER JOIN dbo.Gender g ON e.GenderID = g.GenderID

                INNER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID

        WHERE   e.GenderID = @GenderID

                AND e.DepartmentID = @DepartmentID;

    END;

GO -- Run the prvious command and begins new batch

/\*

1.

Display name and gender and DepartmentName by the GenderID and DepartmentID

2.

--CREATE PROC spGetEmployeesByGenderIDAndDepartmentID

--    (

--      @GenderID INT ,

--      @DepartmentID INT

--    )

--AS

--    BEGIN

--        ...

--    END;

Create a stored procedure "spGetEmployeesByGenderIDAndDepartmentID"

with 2 parameters @GenderID INT  and   @DepartmentID INT

\*/

--====================================================================================

--T006\_03\_03

--EXECUTE stored procedure

spGetEmployeesByGenderIDAndDepartmentID 2, 1;

GO -- Run the prvious command and begins new batch

EXEC spGetEmployeesByGenderIDAndDepartmentID 2, 2;

GO -- Run the prvious command and begins new batch

EXECUTE spGetEmployeesByGenderIDAndDepartmentID 1, 1;

GO -- Run the prvious command and begins new batchs

/\*

--spGetEmployeesByGenderIDAndDepartmentID 2, 1;

--EXEC spGetEmployeesByGenderIDAndDepartmentID 2, 2;

--EXECUTE spGetEmployeesByGenderIDAndDepartmentID 1, 1;

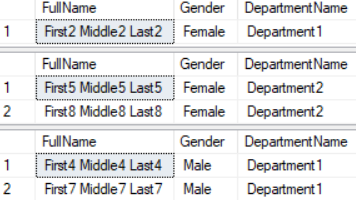
3 Query ways to execute the Stored Procedure.

or

Database Name --> Programmability --> Stored Procedures -->

Stored Procedure Name --> right click --> Execute Stored Procedure.

\*/



==================================================

4. Alter the stored procedure, WITH ENCRYPTION

--====================================================================================

--T006\_04\_Alter the stored procedure, WITH ENCRYPTION

--====================================================================================

--====================================================================================

--T006\_04\_01

--SELECT

SELECT  e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName AS FullName ,

        g.Gender ,

        d.DepartmentName

FROM    dbo.Employee e

        INNER JOIN dbo.Gender g ON e.GenderID = g.GenderID

        INNER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID

WHERE   e.GenderID = 1

        AND e.DepartmentID = 1

ORDER BY e.FirstName;

GO -- Run the prvious command and begins new batch

/\*

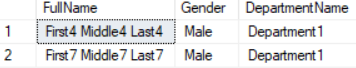
1.

Display name and gender and DepartmentName

with the condition, e.GenderID = 1 AND e.DepartmentID = 1

and order by the FirstName.

\*/



--====================================================================================

--T006\_04\_02

--Alter the stored procedure

ALTER PROC spGetEmployeesByGenderIDAndDepartmentID

(

  @GenderID INT ,

  @DepartmentID INT

 )

AS

    BEGIN

        SELECT  e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName AS FullName ,

                g.Gender ,

                d.DepartmentName

        FROM    dbo.Employee e

                INNER JOIN dbo.Gender g ON e.GenderID = g.GenderID

                INNER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID

        WHERE   e.GenderID = @GenderID

                AND e.DepartmentID = @DepartmentID

             --The change is here

        ORDER BY e.FirstName;

    END;

GO -- Run the prvious command and begins new batch

/\*

1.

Display name and gender and DepartmentName

by the GenderID and DepartmentID

and order by the FirstName

2.

--ALTER PROC spGetEmployeesByGenderIDAndDepartmentID

--    (

--      @GenderID INT ,

--      @DepartmentID INT

--    )

--AS

--    BEGIN

--        ...

--    END;

Alter the stored procedure "spGetEmployeesByGenderIDAndDepartmentID"

with 2 parameters @GenderID INT  and   @DepartmentID INT

3.

In SSMS,

Database Name --> Programmability --> Stored Procedures -->

Stored Procedure Name --> Right Click --> Modify

\*/

--====================================================================================

--T006\_04\_03

--EXECUTE stored procedure

spGetEmployeesByGenderIDAndDepartmentID 2, 1;

GO -- Run the prvious command and begins new batch

EXEC spGetEmployeesByGenderIDAndDepartmentID 2, 2;

GO -- Run the prvious command and begins new batch

EXECUTE spGetEmployeesByGenderIDAndDepartmentID 1, 1;

GO -- Run the prvious command and begins new batchs

/\*

--spGetEmployeesByGenderIDAndDepartmentID 2, 1;

--EXEC spGetEmployeesByGenderIDAndDepartmentID 2, 2;

--EXECUTE spGetEmployeesByGenderIDAndDepartmentID 1, 1;

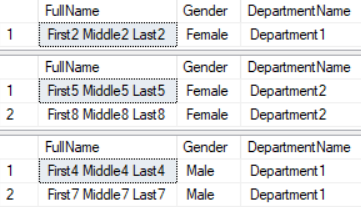
3 Query ways to execute the Stored Procedure.

or

Database Name --> Programmability --> Stored Procedures -->

Stored Procedure Name --> right click --> Execute Stored Procedure.

\*/



--====================================================================================

--T006\_04\_04

--WITH ENCRYPTION

ALTER PROC spGetEmployeesByGenderIDAndDepartmentID

(

  @GenderID INT ,

  @DepartmentID INT

 )

    WITH ENCRYPTION

AS

    BEGIN

        SELECT  e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName AS FullName ,

                g.Gender ,

                d.DepartmentName

        FROM    dbo.Employee e

                INNER JOIN dbo.Gender g ON e.GenderID = g.GenderID

                INNER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID

        WHERE   e.GenderID = @GenderID

                AND e.DepartmentID = @DepartmentID

        ORDER BY e.FirstName;

    END;

GO -- Run the prvious command and begins new batch

/\*

1.

Display name and gender and DepartmentName

by the GenderID and DepartmentID

and order by the FirstName

2.

--ALTER PROC spGetEmployeesByGenderIDAndDepartmentID

--    (

--      @GenderID INT ,

--      @DepartmentID INT

--    )

--    WITH ENCRYPTION

--AS

--    BEGIN

--           ...

--    END;

Alter the stored procedure "spGetEmployeesByGenderIDAndDepartmentID"

with 2 parameters @GenderID INT  and   @DepartmentID INT

--    WITH ENCRYPTION

Once encrypted, you can not read or modify the procedure text again.

You can only

-- DROP PROCEDURE 'SPName'

to delete the stored procedure

3.

In SSMS, to delete stored procedure.

Database Name --> Programmability --> Stored Procedures -->

Stored Procedure Name --> Right Click --> Delete

\*/

==================================================

5. Delete stored procedure

--====================================================================================

--T006\_05\_Delete stored procedure

--====================================================================================

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetEmployeesByGenderIDAndDepartmentID' ) )

    BEGIN

        DROP PROCEDURE spGetEmployeesByGenderIDAndDepartmentID;

             --DROP PROC spGetEmployeesByGenderIDAndDepartmentID;

    END;

GO -- Run the previous command and begins new batch

/\*

In SSMS, to delete stored procedure.

Database Name --> Programmability --> Stored Procedures -->

Stored Procedure Name --> Right Click --> Delete

\*/

6. Stored Procedure output parameter   V.s.   Stored Procedure Return Value

--====================================================================================

--T006\_06\_Stored Procedure output parameter   V.s.   Stored Procedure Return Value

--====================================================================================

/\*

1.

Store Pocedure Output parameters can output more than one value and any Data type.

E.g. Output string, Data time, int, ....etc.

2.

When you execute a stored procedure, it will always return an integer value.

2.1.

If you use "Store Pocedure Output parameters",

then you will also get a "return integer status value".

zero means success, and non-zero means failure.

2.2.

If you use "Store Pocedure Return Value",

then you will still get "return integer value".

But this "return integer value" is not "status value" any more.

It is whatever value which was returned by store prcedure.

2.3.

In SSMS,

Database Name --> Programmability --> Stored Procedures -->

Stored Procedure Name --> Right click --> Execute Stored Procedures

--> then you will see ONE RETURN INTEGER VALUE.

If you use output parameters,

then you will also see the value of output parameters

and ONE RETURN INTEGER VALUE indicates the status,

which 0 means successful.

\*/

----------------------------------------------------------------------------------------------------------------------------

6.1. Stored Procedure output parameter 1

--====================================================================================

--T006\_06\_01

--Stored Procedure output parameter 1

------------------------------------------------------------

--T006\_06\_01\_00

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetEmployeeCountByGenderID' ) )

    BEGIN

        DROP PROCEDURE spGetEmployeeCountByGenderID;

             --DROP PROC spGetEmployeesByGenderIDAndDepartmentID;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spGetEmployeeCountByGenderID

(

  @GenderID INT ,

  @EmployeeCount int OUTPUT

 )

AS

    BEGIN

        SELECT  @EmployeeCount = COUNT(e.EmployeeID)

        FROM    dbo.Employee e

                INNER JOIN dbo.Gender g ON e.GenderID = g.GenderID

        WHERE   e.GenderID = @GenderID;

    END;

GO -- Run the prvious command and begins new batch

------------------------------------------------------------

--T006\_06\_01\_01

DECLARE @EmployeeTotal INT;

DECLARE @Status\_spGetEmployeeCountByGenderID INT;

EXECUTE @Status\_spGetEmployeeCountByGenderID = spGetEmployeeCountByGenderID 1,

    @EmployeeTotal OUTPUT;

PRINT @EmployeeTotal;

PRINT @Status\_spGetEmployeeCountByGenderID;

GO -- Run the prvious command and begins new batch

/\*

--PRINT @EmployeeTotal;

--PRINT @Status\_spGetEmployeeCountByGenderID;

Output will be

--4

--0

The returned integer value is status value, 0 means success.

\*/

A picture containing graphical user interface

Description automatically generated

------------------------------------------------------------

--T006\_06\_01\_02

DECLARE @EmployeeTotal2 INT;

DECLARE @Status\_spGetEmployeeCountByGenderID2 INT;

EXECUTE @Status\_spGetEmployeeCountByGenderID2 = spGetEmployeeCountByGenderID 1,

    @EmployeeTotal2;

IF ( @EmployeeTotal2 IS NULL )

    BEGIN

        PRINT '@EmployeeTotal2 is null';

    END;

ELSE

    BEGIN

        PRINT '@EmployeeTotal2 is not null';

    END;

PRINT @EmployeeTotal2;

PRINT @Status\_spGetEmployeeCountByGenderID2;

GO -- Run the prvious command and begins new batch

/\*

--PRINT '@EmployeeTotal2 is null';

--PRINT @EmployeeTotal2;

--PRINT @Status\_spGetEmployeeCountByGenderID2;

Output will be

--@EmployeeTotal2 is null

--

--0

Because

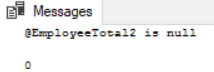
--EXECUTE @Status\_spGetEmployeeCountByGenderID2 = spGetEmployeeCountByGenderID 1, @EmployeeTotal2;

It does not has "Output" keyword after "@EmployeeTotal2",

Thus, @EmployeeTotal2 is NULL

Then the returned integer value is status value, 0 means success.

\*/



------------------------------------------------------------

--T006\_06\_01\_03

DECLARE @EmployeeTotal3 INT;

DECLARE @Status\_spGetEmployeeCountByGenderID3 INT;

EXECUTE @Status\_spGetEmployeeCountByGenderID3 = spGetEmployeeCountByGenderID 1,

    @EmployeeTotal3 OUTPUT;

-- EXEC @Status\_spGetEmployeeCountByGenderID3 = spGetEmployeeCountByGenderID 1, @EmployeeTotal3 OUT

IF ( @EmployeeTotal3 IS NULL )

    BEGIN

        PRINT '@EmployeeTotal3 is null';

    END;

ELSE

    BEGIN

        PRINT '@EmployeeTotal3 is not null';

    END;

PRINT @EmployeeTotal3;

PRINT @Status\_spGetEmployeeCountByGenderID3;

GO -- Run the prvious command and begins new batch

/\*

--PRINT '@EmployeeTotal3 is not null';

--PRINT @EmployeeTotal3;

--PRINT @Status\_spGetEmployeeCountByGenderID3;

Output will be

--@EmployeeTotal3 is not null

--4

--0

Because

--EXECUTE @Status\_spGetEmployeeCountByGenderID2 = spGetEmployeeCountByGenderID 1, @EmployeeTotal3 Output;

It has "Output" keyword after "@EmployeeTotal3",

Thus, @EmployeeTotal3 is not NULL

Then the returned integer value is status value, 0 means success.

\*/

Graphical user interface, application

Description automatically generated

------------------------------------------------------------

--T006\_06\_01\_04

DECLARE @EmployeeTotal4 INT;

DECLARE @Status\_spGetEmployeeCountByGenderID4 INT;

EXECUTE @Status\_spGetEmployeeCountByGenderID4 = spGetEmployeeCountByGenderID @EmployeeCount = @EmployeeTotal4 OUT,

    @GenderID = 1;

PRINT @EmployeeTotal4;

PRINT @Status\_spGetEmployeeCountByGenderID4;

GO -- Run the prvious command and begins new batch

/\*

--PRINT @EmployeeTotal4;

--PRINT @Status\_spGetEmployeeCountByGenderID4;

Output will be

--4

--0

The returned integer value is status value, 0 means success.

\*/



----------------------------------------------------------------------------------------------------------------------------

6.2. Stored Procedure output parameter 2

--====================================================================================

--T006\_06\_02

--Stored Procedure output parameter 2

------------------------------------------------------------

--T006\_06\_02\_00

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetCountAllEmployees1' ) )

    BEGIN

        DROP PROCEDURE spGetCountAllEmployees1;

             --DROP PROC spGetEmployeesByGenderIDAndDepartmentID;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spGetCountAllEmployees1

(

  @TotalCount int OUTPUT

 )

AS

    BEGIN

        SELECT  @TotalCount = COUNT(e.EmployeeID)

        FROM    dbo.Employee e;

    END;

GO -- Run the prvious command and begins new batch

------------------------------------------------------------

--T006\_06\_02\_01

DECLARE @TotalEmployees INT;

DECLARE @Status\_spGetCountAllEmployees1 INT;

EXECUTE @Status\_spGetCountAllEmployees1 = spGetCountAllEmployees1 @TotalEmployees OUTPUT;

PRINT @TotalEmployees;

PRINT @Status\_spGetCountAllEmployees1;

GO -- Run the prvious command and begins new batch

/\*

1.

1.1.

----Ch18\_06\_02\_01

--PRINT @TotalEmployees;

--PRINT @Status\_spGetCountAllEmployees1;

Output will be

--10

--0

The returned integer value is status value, 0 means success.

\*/

Text

Description automatically generated with low confidence

----------------------------------------------------------------------------------------------------------------------------

6.3. Stored Procedure Return Value 1

--====================================================================================

--T006\_06\_03

--Stored Procedure Return Value 1

------------------------------------------------------------

--T006\_06\_03\_00

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetCountAllEmployees2' ) )

    BEGIN

        DROP PROCEDURE spGetCountAllEmployees2;

             --DROP PROC spGetEmployeesByGenderIDAndDepartmentID;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spGetCountAllEmployees2

AS

    BEGIN

        RETURN

        ( SELECT    COUNT(e.EmployeeID)

          FROM      dbo.Employee e

        );

    END;

GO -- Run the prvious command and begins new batch

------------------------------------------------------------

--T006\_06\_03\_01

DECLARE @TotalEmployees2 INT;

EXECUTE @TotalEmployees2 = spGetCountAllEmployees2;

PRINT @TotalEmployees2;

GO -- Run the prvious command and begins new batch

/\*

--PRINT @TotalEmployees2;

Output will be

--10

If you use "Store Pocedure Return Value",

then you will still get "return integer value".

But this "return integer value" is not "status value" any more.

It is whatever value which was returned by store prcedure.

\*/



----------------------------------------------------------------------------------------------------------------------------

6.4. Stored Procedure output parameter 3

--====================================================================================

--T006\_06\_04

--Stored Procedure output parameter 3

------------------------------------------------------------

--T006\_06\_04\_00

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetNameById1' ) )

    BEGIN

        DROP PROCEDURE spGetNameById1;

             --DROP PROC spGetEmployeesByGenderIDAndDepartmentID;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spGetNameById1

(

  @Id int ,

  @Name nvarchar(50) OUTPUT

 )

AS

    BEGIN

        SELECT  @Name = e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName

        FROM    dbo.Employee e

        WHERE   e.EmployeeID = @Id;

    END;

GO -- Run the prvious command and begins new batch

------------------------------------------------------------

--T006\_06\_04\_01

DECLARE @EmployeeName NVARCHAR(20);

DECLARE @Status\_spGetNameById1 NVARCHAR(20);

EXECUTE @Status\_spGetNameById1 = spGetNameById1 3, @EmployeeName OUT;

PRINT 'Employee Name : ' + @EmployeeName;

PRINT @Status\_spGetNameById1;

GO -- Run the prvious command and begins new batch

/\*

1.

1.1.

----Ch18\_06\_04\_01

--PRINT 'Employee Name : ' + @EmployeeName;

--PRINT @Status\_spGetNameById1;

Output will be

--Employee Name : Fisrt3 Middle3 Last3

--0

The returned integer value is status value, 0 means success.

If you use "Store Pocedure Output parameters",

then you will also get a "return integer status value".

zero means success, and non-zero means failure.

\*/

Graphical user interface

Description automatically generated

----------------------------------------------------------------------------------------------------------------------------

6.5. Stored Procedure Return Value 2

--====================================================================================

--T006\_06\_05

--Stored Procedure Return Value 2

------------------------------------------------------------

--T006\_06\_05\_00

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetNameById2' ) )

    BEGIN

        DROP PROCEDURE spGetNameById2;

             --DROP PROC spGetEmployeesByGenderIDAndDepartmentID;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spGetNameById2 ( @Id int )

AS

    BEGIN

        RETURN (SELECT

                                 (e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName)

                           FROM dbo.Employee e

                           WHERE e.EmployeeID = @Id);

    END;

GO -- Run the prvious command and begins new batch

------------------------------------------------------------

--Ch18\_06\_05\_01

DECLARE @EmployeeName NVARCHAR(20);

EXECUTE @EmployeeName = spGetNameById2 1;

PRINT 'Employee Name : ' + @EmployeeName;

GO -- Run the prvious command and begins new batch

/\*

1.

1.1.

----Ch18\_06\_05\_01

--PRINT 'Employee Name : ' + @EmployeeName;

Output will be

--Msg 245, Level 16, State 1, Procedure spGetNameById2,

--Line 4 [Batch Start Line 826]

--Conversion failed when converting the nvarchar value

--'First1 Middle1 Last1' to data type int.

When you execute a stored procedure,

it will always return an integer value.

It will fail if you try to return non-int value

\*/



==================================================

7. sp\_help, sp\_helptext, sp\_depends

--====================================================================================

--T006\_07\_01

--sp\_help databaseObjectName

sp\_help Employee;

GO -- Run the prvious command and begins new batch

sp\_help spGetCountAllEmployees1;

GO -- Run the prvious command and begins new batch

/\*

--sp\_help databaseObjectName

Same as you highlight all kind of database object

such as stored procedure name, table name, view name, trigger name ...etc.

and then press Alt + F1

Then you will see all the information regarding the database object.

\*/

Graphical user interface, text, application, email

Description automatically generated

--====================================================================================

--T006\_07\_02

--sp\_helptext spName

sp\_helptext spGetCountAllEmployees1;

GO -- Run the prvious command and begins new batch

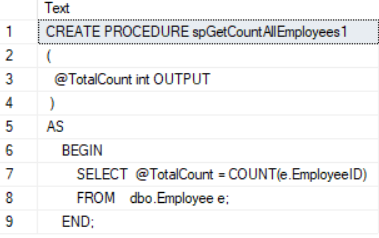
/\*

--sp\_helptext spName

See the stored procedure text.

Only for stored procedure.

\*/



--====================================================================================

--T006\_07\_03

--sp\_depends databaseObjectName

sp\_depends Employee;

GO -- Run the prvious command and begins new batch

sp\_depends spGetCountAllEmployees1;

GO -- Run the prvious command and begins new batch

/\*

--sp\_depends databaseObjectName

See the dependencies of database object.

E.g.

--sp\_depends spName

you will see what table and what column were used in this stored procedure.

Thus, before you delete or edit these columns' name,

you have to edit this stored procedure first.

E.g.

--sp\_depends tableName

Show you all the stored procedure or

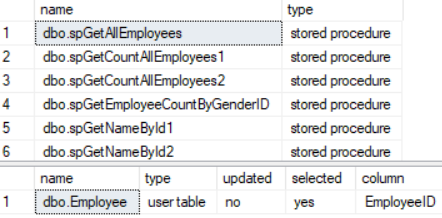
any other database object

which were created by this table columns.

Thus, before you delete or edit this table columns' name,

you have to double check if it will affect these database objects first.

\*/



8. Create Sample Data

--====================================================================================

--T006\_08\_01

--Create or Recreate Table

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Person3' ) )

    BEGIN

        TRUNCATE TABLE Person3;

        DROP TABLE Person3;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE Person3

(

  PersonID INT PRIMARY KEY

                   IDENTITY(1, 1)

                   NOT NULL ,

  [Name] NVARCHAR(100) NULL ,

  Salary NVARCHAR(50) NULL ,

  RegisteredDateTime DATETIME NULL

)

--====================================================================================

--T006\_08\_02

--Insert Data

--Person3 Counter

DECLARE @TotolPerson3Rows INT;

DECLARE @Person3Count INT;

SET @Person3Count = 1;

--\*\*\*\*\* Changeable data rows

SET @TotolPerson3Rows = 20;

-- @RandomSalary

DECLARE @RandomSalary INT;

DECLARE @RandomSalary\_Max INT;

DECLARE @RandomSalary\_Min INT;

SET @RandomSalary\_Min = 1;

SET @RandomSalary\_Max = 100000;

--@RandomRegisteredDateTime

--Reference: <http://crodrigues.com/sql-server-generate-random-datetime-within-a-range/>

DECLARE @RandomRegisteredDateTime DATETIME;

DECLARE @DateFrom DATETIME = '2012-01-01';

DECLARE @DateTo DATETIME = '2017-06-30';

DECLARE @DaysRandom INT= 0;

DECLARE @MillisRandom INT= 0;

WHILE ( @Person3Count <= @TotolPerson3Rows )

    BEGIN

             --1. @RandomSalary

        SELECT  @RandomSalary = FLOOR(RAND() \* ( @RandomSalary\_Max

                                                 - @RandomSalary\_Min )

                                      + @RandomSalary\_Min);

             --2. @RandomRegisteredDateTime

             --get random number of days

        SELECT  @DaysRandom = DATEDIFF(DAY, @DateFrom, @DateTo);

        SELECT  @DaysRandom = ROUND(( ( @DaysRandom - 1 ) \* RAND() ), 0);

             --get random millis

        SELECT  @MillisRandom = ROUND(( ( 99999999 ) \* RAND() ), 0);

        SELECT  @RandomRegisteredDateTime = DATEADD(DAY, @DaysRandom,

                                                    @DateFrom);

        SELECT  @RandomRegisteredDateTime = DATEADD(MILLISECOND, @MillisRandom,

                                                    @RandomRegisteredDateTime);

        INSERT  INTO Person3

        VALUES  ( ( 'Name ' + CONVERT(NVARCHAR, @Person3Count) ),

                  CONVERT(NVARCHAR, @RandomSalary), @RandomRegisteredDateTime );

        PRINT @Person3Count;

        SET @Person3Count += 1;

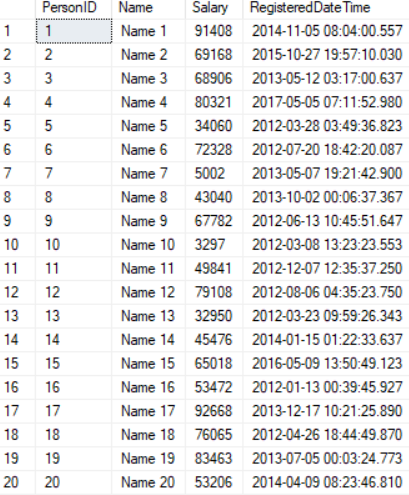
    END;

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    Person3;

GO -- Run the previous command and begins new batch



--====================================================================================

--T006\_08\_03

--Create or Recreate store procedure

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spSearchPerson3' ) )

    BEGIN

        DROP PROCEDURE spSearchPerson3;

             --DROP PROC spSearchPerson3;

    END;

GO -- Run the previous command and begins new batch

CREATE PROC spSearchPerson3

    (

      @NameLike NVARCHAR(100) = NULL ,

      @SalaryGreaterThan MONEY = NULL

       )

AS

    BEGIN

        SELECT  \*

        FROM    Person3 p3

        WHERE   ( p3.[Name] LIKE ('%' +  @NameLike + '%')

                  OR @NameLike IS NULL

                )

                AND ( p3.Salary > @SalaryGreaterThan

                      OR @SalaryGreaterThan IS NULL

                    )

    END;

GO -- Run the previous command and begins new batch

/\*

1.

--CREATE PROC spSearchPerson3

--    (

--      @Name NVARCHAR(100) = NULL ,

--      @Salary MONEY = NULL ,

--      @RegisteredDateTime DATETIME = NULL

--     )

-- ...

 --WHERE   ( p3.[Name] = @Name

 --           OR @Name IS NULL

 --       )

 --       AND ( p3.Salary = @Salary

 --               OR @Salary IS NULL

 --           )

 --       AND ( p3.RegisteredDateTime = @RegisteredDateTime

 --               OR @RegisteredDateTime IS NULL

 --           );

If we set the default value for the parameter,

that will make the parameter become optional.

Without the parameter default value,

the parameter will become compulsory.

Thus, in where clause we need to add the IS NULL for each parameter

\*/

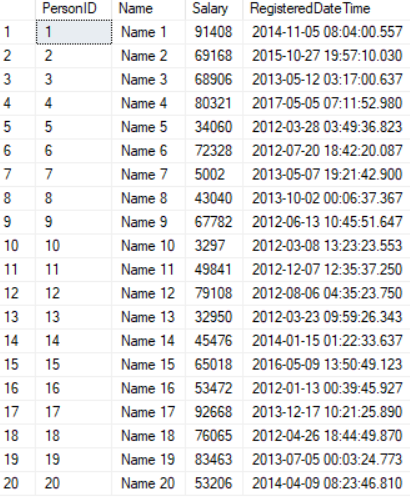
--====================================================================================

--T006\_08\_04

-- Execute Stored Procedure Optional Parameters

EXECUTE spSearchPerson3;

--Return all rows.



EXECUTE spSearchPerson3 @NameLike = '8';

-- Return Name='Name 8'   and   Name='name 18'

EXECUTE spSearchPerson3 @SalaryGreaterThan=75000

-- Retruns all person whoes salary is greater than 75000

EXECUTE spSearchPerson3 @NameLike = '8', @SalaryGreaterThan=75000;

-- Return Name='Name 8' and his/her salary is greater than 75000.

GO -- Run the previous command and begins new batch



=======================================================

9. Web Application - Stored Procedure Optional Parameters

9.1. Set up SQL Authentication

In SQL server

Object Explorer --> Security --> Logins --> New Logins

-->

General Tab

Login Name :

**Tester**

Password:

**1234**

Default Database:

**Sample**

-->

Server Roles Tab

Select

**sysadmin**

-->

User Mapping Tab

Select **Sample**

Select every Roles.









9.2. Create Web Application

Open Visual Studio, I am currently using VS2017

If you don't have it, you may following the instruction here to download.

<http://ithandyguytutorial.blogspot.com/2017/10/ch00install-visual-studio-2017-offline.html>

New Project --> Web --> **ASP.NET** **Web Application (.Net Framework)**

-->

Name:

**Sample**

--> **Empty** --> Select "**Web Forms**"  --> OK

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

9.3. Code

9.3.1. Web.config

Add connection String

<configuration>

  <connectionStrings>

    <add name="SampleConnectionString" connectionString="Data Source=N550JKL\SQL2016;Initial Catalog=Sample;User ID=Tester;Password=1234"

        providerName="System.Data.SqlClient" />

  </connectionStrings>

Graphical user interface, text, application

Description automatically generated

9.3.2. WebForm1.aspx

ProjectName --> Right Click --> Add --> New Item...

-->

**WebForm**

Name :

**WebForm1.aspx**

Graphical user interface, application

Description automatically generated

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Sample.WebForm1" %>

<!DOCTYPE html>

<html xmlns="[http://www.w3.org/1999/xhtml">](http://www.w3.org/1999/xhtml%22%3E);

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

        <div>

            <table>

                <tr>

                    <td colspan="4">

                        <b>Search Person</b>

                    </td>

                </tr>

                <tr>

                    <td>

                        <b>Name</b>

                    </td>

                    <td>

                        <asp:TextBox ID="txtNameLike" runat="server"></asp:TextBox>

                    </td>

                    <td>

                        <b>Salary > </b>

                    </td>

                    <td>

                        <asp:TextBox ID="txtSalaryGreaterThan" runat="server"></asp:TextBox>

                    </td>

                </tr>

                <tr>

                    <td colspan="4">

                        <asp:Button ID="btnSerach" runat="server" Text="Search"

                            OnClick="btnSerach\_Click" />

                    </td>

                </tr>

                <tr>

                    <td colspan="4">

                        <asp:GridView ID="gvEmployees" runat="server">

                        </asp:GridView>

                    </td>

                </tr>

            </table>

        </div>

    </form>

</body>

</html>

9.3.3. Default.aspx.cs

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Sample

{

    public partial class WebForm1 : Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        {

            if (!IsPostBack)

                GetData();

        }

        protected void btnSerach\_Click(object sender, EventArgs e)

        {

            GetData();

        }

        private void AttachParameter(SqlCommand command, string parameterName, Control control)

        {

            if (control is TextBox && ((TextBox) control).Text != string.Empty)

            {

                var parameter = new SqlParameter(parameterName, ((TextBox) control).Text);

                command.Parameters.Add(parameter);

            }

            else if (control is DropDownList && ((DropDownList) control).SelectedValue != "-1")

            {

                var parameter = new SqlParameter(parameterName, ((DropDownList) control).SelectedValue);

                command.Parameters.Add(parameter);

            }

        }

        private void GetData()

        {

            //string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;

            string cs = ConfigurationManager.ConnectionStrings["SampleConnectionString"].ConnectionString;

            using (var con = new SqlConnection(cs))

            {

                var cmd = new SqlCommand("spSearchPerson3", con);

[cmd.CommandType](http://cmd.commandtype/) = CommandType.StoredProcedure;

                AttachParameter(cmd, "@NameLike", txtNameLike);

                AttachParameter(cmd, "@SalaryGreaterThan", txtSalaryGreaterThan);

                con.Open();

                gvEmployees.DataSource = cmd.ExecuteReader();

                gvEmployees.DataBind();

            }

        }

    }

}

9.3.4. Run it

Table

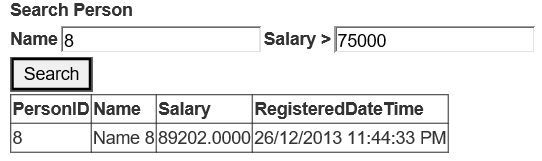
Description automatically generated

-------------------------------------------------------------------------------------

Table

Description automatically generated

-------------------------------------------------------------------------------------



-------------------------------------------------------------------------------------

