(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

- 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.

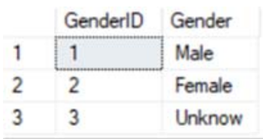
6.5.
Prevent SQL Injection
We can use some error checking to prevent SQL Injection in stored procedure.

1. Create Sample Data

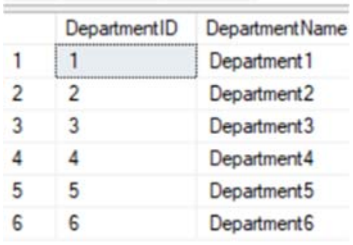
```
--==================================================================================
--T006_01_Create Sample Data
--==================================================================================
--If Table exists then DROP it
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              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  *
```

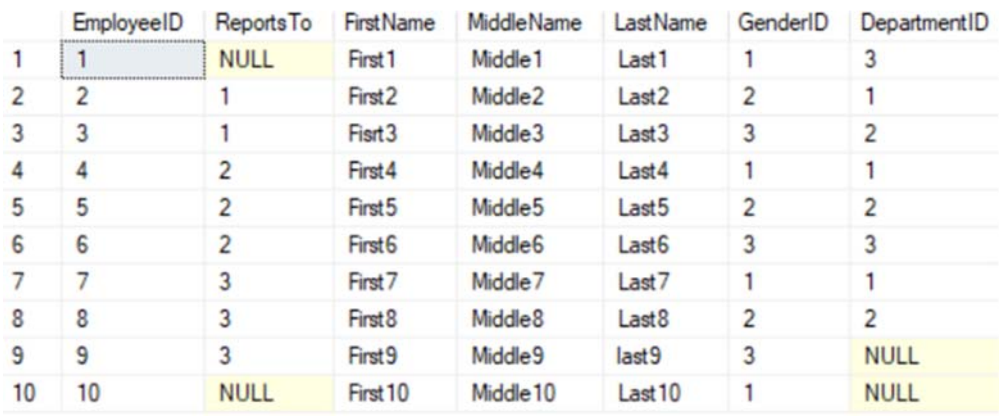
```
FROM    Employee;
GO -- Run the prvious command and begins new batch
```



	GenderID	Gender
1	1	Male
2	2	Female
3	3	Unknow



	DepartmentID	DepartmentName
1	1	Department1
2	2	Department2
3	3	Department3
4	4	Department4
5	5	Department5
6	6	Department6

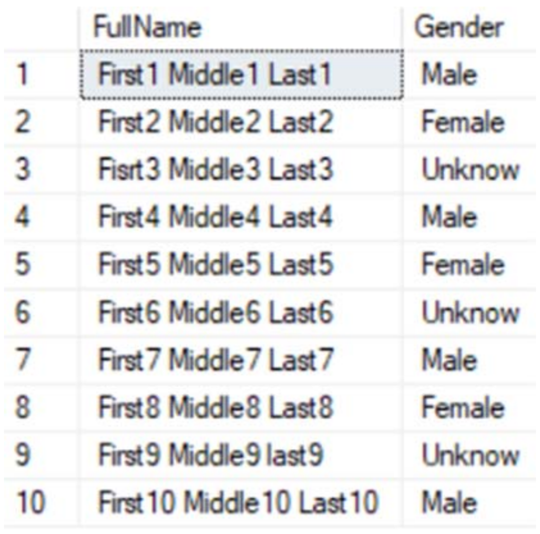


	EmployeeID	ReportsTo	FirstName	MiddleName	LastName	GenderID	DepartmentID
1	1	NULL	First1	Middle1	Last1	1	3
2	2	1	First2	Middle2	Last2	2	1
3	3	1	Fisrt3	Middle3	Last3	3	2
4	4	2	First4	Middle4	Last4	1	1
5	5	2	First5	Middle5	Last5	2	2
6	6	2	First6	Middle6	Last6	3	3
7	7	3	First7	Middle7	Last7	1	1
8	8	3	First8	Middle8	Last8	2	2
9	9	3	First9	Middle9	last9	3	NULL
10	10	NULL	First10	Middle10	Last10	1	NULL

==

2. Store Procedure

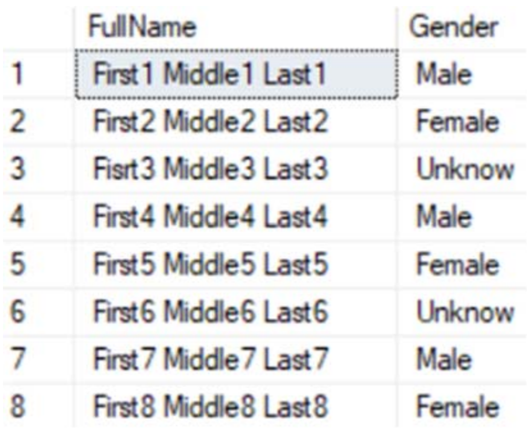
```
--=====================================================================================
--T006_02_Store Procedure
--=====================================================================================
--=====================================================================================
--T006_02_01
--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
*/
```



	FullName	Gender
1	First1 Middle1 Last1	Male
2	First2 Middle2 Last2	Female
3	Fisrt3 Middle3 Last3	Unknow
4	First4 Middle4 Last4	Male
5	First5 Middle5 Last5	Female
6	First6 Middle6 Last6	Unknow
7	First7 Middle7 Last7	Male
8	First8 Middle8 Last8	Female
9	First9 Middle9 last9	Unknow
10	First10 Middle10 Last10	Male

```
--==================================================================================
--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.
*/
```

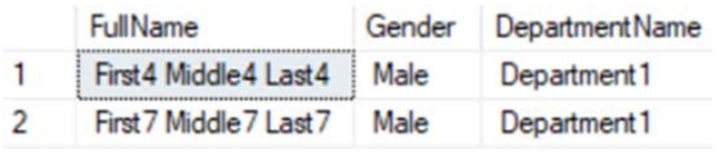


	FullName	Gender
1	First1 Middle1 Last1	Male
2	First2 Middle2 Last2	Female
3	Fisrt3 Middle3 Last3	Unknow
4	First4 Middle4 Last4	Male
5	First5 Middle5 Last5	Female
6	First6 Middle6 Last6	Unknow
7	First7 Middle7 Last7	Male
8	First8 Middle8 Last8	Female

```
=============================================
```

3. Create Stored Procedure with parameters

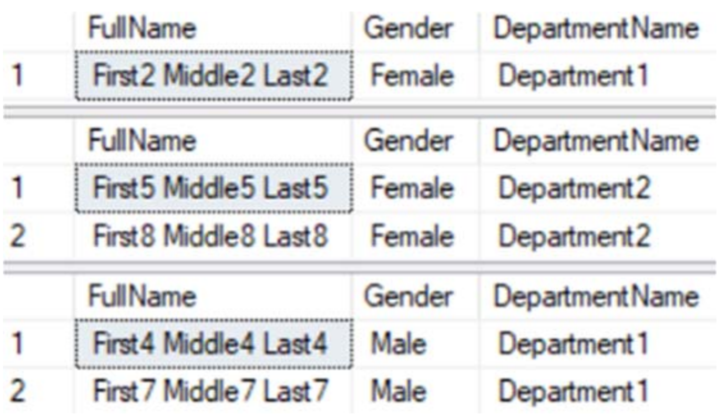
```
--==================================================================================
--T006_03_Create Stored Procedure with parameters
--==================================================================================
--==================================================================================
--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
*/
```



	FullName	Gender	DepartmentName
1	First4 Middle4 Last4	Male	Department1
2	First7 Middle7 Last7	Male	Department1

```
--==================================================================================
--T006_03_02
--Create Stored Procedure with parameters
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.ROUTINES
```

```sql
                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.
*/
```



	FullName	Gender	DepartmentName
1	First2 Middle2 Last2	Female	Department1

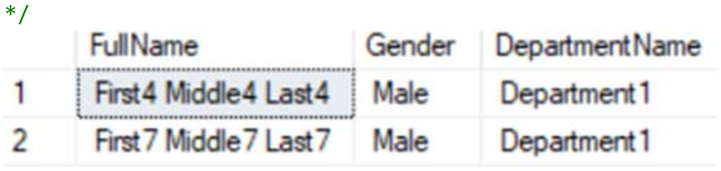
	FullName	Gender	DepartmentName
1	First5 Middle5 Last5	Female	Department2
2	First8 Middle8 Last8	Female	Department2

	FullName	Gender	DepartmentName
1	First4 Middle4 Last4	Male	Department1
2	First7 Middle7 Last7	Male	Department1

==

4. Alter the stored procedure, WITH ENCRYPTION

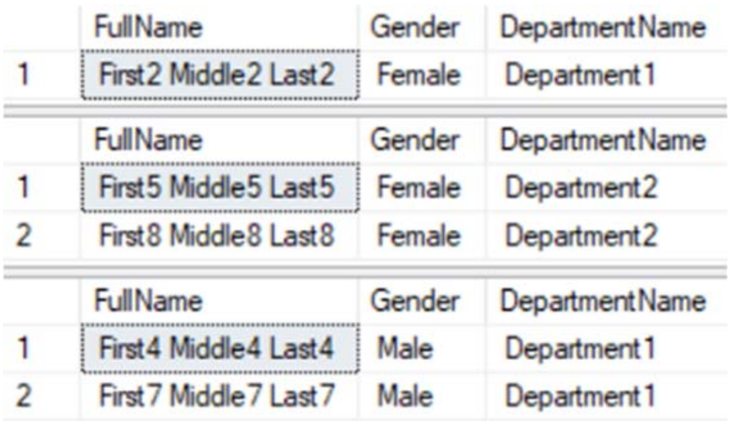
```
--=====================================================================================
--T006_04_Alter the stored procedure, WITH ENCRYPTION
--=====================================================================================
--=====================================================================================
--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.
*/
```



	FullName	Gender	DepartmentName
1	First4 Middle4 Last4	Male	Department1
2	First7 Middle7 Last7	Male	Department1

```
--=====================================================================================
--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.
*/
```



	FullName	Gender	DepartmentName
1	First2 Middle2 Last2	Female	Department1

	FullName	Gender	DepartmentName
1	First5 Middle5 Last5	Female	Department2
2	First8 Middle8 Last8	Female	Department2

	FullName	Gender	DepartmentName
1	First4 Middle4 Last4	Male	Department1
2	First7 Middle7 Last7	Male	Department1

```
--==================================================================================
--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
*/
```

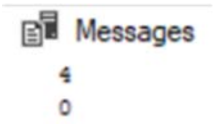
6. Stored Procedure output parameter   V.s.   Stored Procedure Return Value

```
--==================================================================================
--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.
*/
```

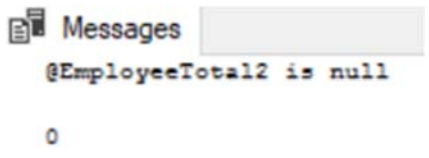
6.1. Stored Procedure output parameter 1

```
--==============================================================================
--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.
*/
```



```
Messages
4
0
```

```
-----------------------------------------------------------
--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.
*/
```

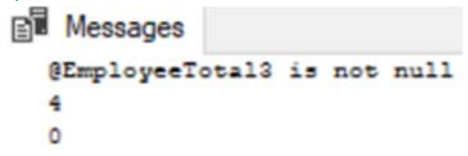


```
Messages
@EmployeeTotal2 is null

0
```

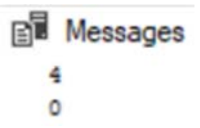
```
-----------------------------------------------------------
--T006_06_01_03
DECLARE @EmployeeTotal3 INT;
DECLARE @Status_spGetEmployeeCountByGenderID3 INT;
EXECUTE @Status_spGetEmployeeCountByGenderID3 = spGetEmployeeCountByGenderID 1,
    @EmployeeTotal3 OUTPUT;
-- EXEC @Status_spGetEmployeeCountByGenderID3 = 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.
*/
```



```
Messages
@EmployeeTotal3 is not null
4
0
```

```
------------------------------------------------------------
--T006_06_01_04
DECLARE @EmployeeTotal4 INT;
DECLARE @Status_spGetEmployeeCountByGenderID4 INT;
EXECUTE @Status_spGetEmployeeCountByGenderID4 = spGetEmployeeCountByGenderID @EmployeeCount = @EmployeeTotal
4 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.
*/
```

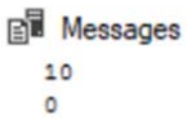


```
Messages
4
0
```

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' ) )
```

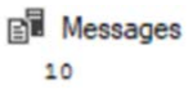
```
    BEGIN
        DROP PROCEDURE spGetCountAllEmployees1;
              --DROP PROC spGetEmployeesByGenderIDAndDepartmentID;
    END;
GO -- Run the previous command and begins new batch
CREATE PROCEDURE spGetCountAllEmployees1
(
  @TotalCount int OUTPUT
 )
AS
    BEGIN
        SELECT  @TotalCount = 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.
*/
```


```
Messages
   10
   0
```

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;
```

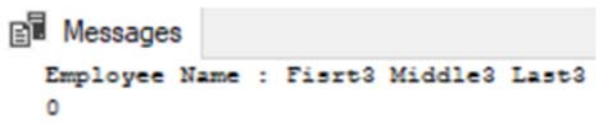
```
GO -- Run the previous command and begins new batch
CREATE PROCEDURE spGetCountAllEmployees2
AS
    BEGIN
        RETURN
        ( SELECT    COUNT(e.EmployeeID)
          FROM      dbo.Employee e
        );
    END;
GO -- Run the prvious command and begins new batch
-------------------------------------------------------------
--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.
*/
```


Messages
10

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.
*/
```

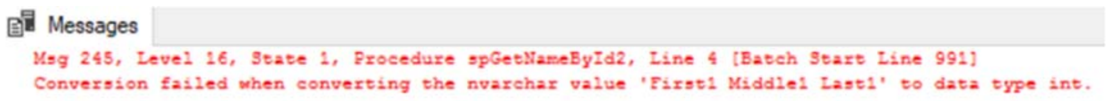


```
Employee Name : Fisrt3 Middle3 Last3
0
```

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
*/
```

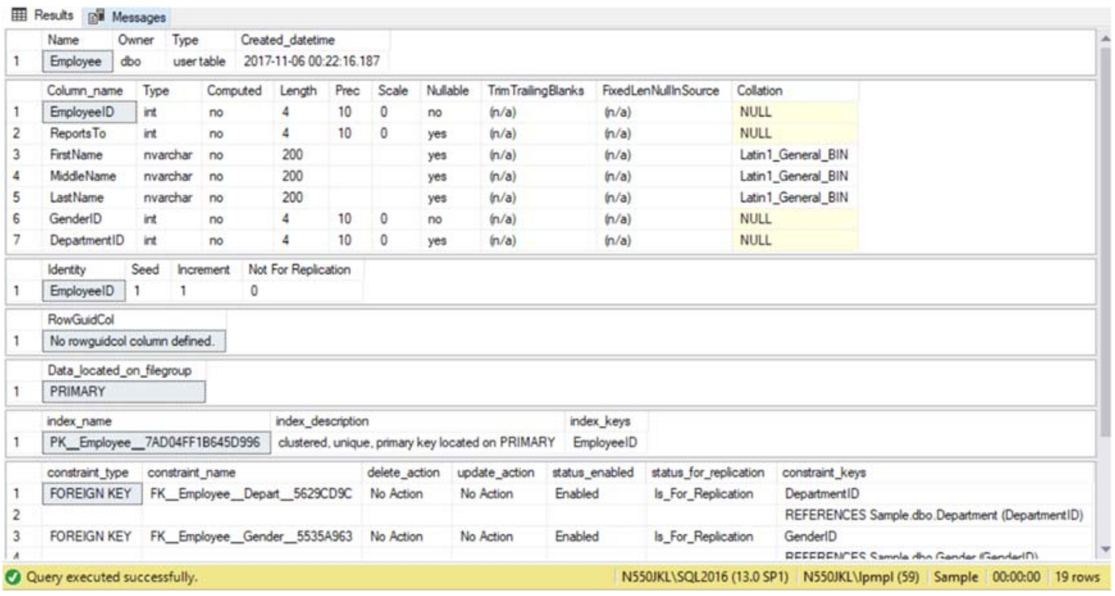


```
Msg 245, Level 16, State 1, Procedure spGetNameById2, Line 4 [Batch Start Line 991]
Conversion failed when converting the nvarchar value 'First1 Middle1 Last1' to data type int.
```

==

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.
*/
```



Results | Messages

	Name	Owner	Type	Created_datetime
1	Employee	dbo	user table	2017-11-06 00:22:16.187

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	EmployeeID	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	ReportsTo	int	no	4	10	0	yes	(n/a)	(n/a)	NULL
3	FirstName	nvarchar	no	200			yes	(n/a)	(n/a)	Latin1_General_BIN
4	MiddleName	nvarchar	no	200			yes	(n/a)	(n/a)	Latin1_General_BIN
5	LastName	nvarchar	no	200			yes	(n/a)	(n/a)	Latin1_General_BIN
6	GenderID	int	no	4	10	0	no	(n/a)	(n/a)	NULL
7	DepartmentID	int	no	4	10	0	yes	(n/a)	(n/a)	NULL

	Identity	Seed	Increment	Not For Replication
1	EmployeeID	1	1	0

	RowGuidCol
1	No rowguidcol column defined.

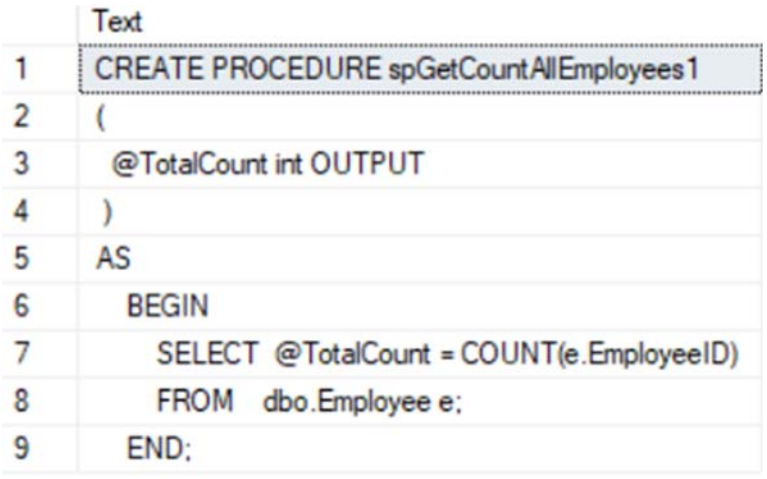
	Data_located_on_filegroup
1	PRIMARY

	index_name	index_description	index_keys
1	PK__Employee__7AD04FF1B645D996	clustered, unique, primary key located on PRIMARY	EmployeeID

	constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
1	FOREIGN KEY	FK__Employee__Depart__5629CD9C	No Action	No Action	Enabled	Is_For_Replication	DepartmentID
2							REFERENCES Sample.dbo.Department (DepartmentID)
3	FOREIGN KEY	FK__Employee__Gender__5535A963	No Action	No Action	Enabled	Is_For_Replication	GenderID
4							REFERENCES Sample.dbo.Gender (GenderID)

Query executed successfully. | N550JKL\SQL2016 (13.0 SP1) | N550JKL\lpmpl (59) | Sample | 00:00:00 | 19 rows

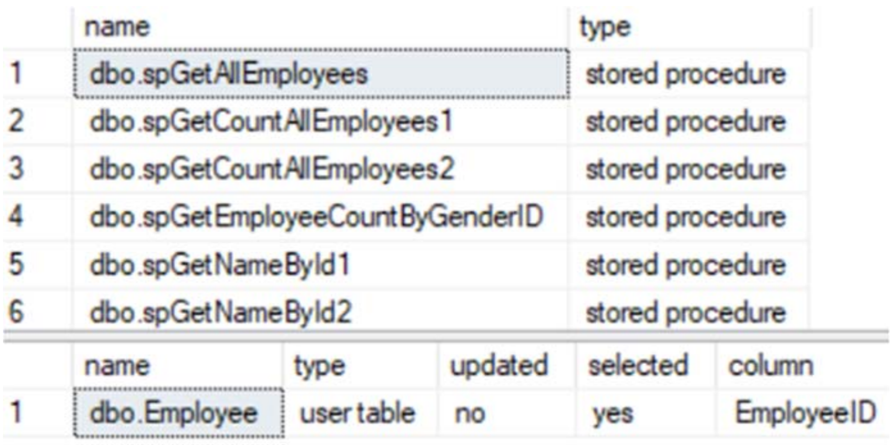
```
--====================================================================================
--T006_07_02
--sp_helptext spName
sp_helptext spGetCountAllEmployees1;
GO -- Run the prvious command and begins new batch
/*
--sp_helptext spName
See the stored procedure text.
Only for stored procedure.
*/
```



```
Text
CREATE PROCEDURE spGetCountAllEmployees1
(
 @TotalCount int OUTPUT
)
AS
  BEGIN
    SELECT  @TotalCount = COUNT(e.EmployeeID)
    FROM    dbo.Employee e;
  END;
```

```
--====================================================================================
--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.
*/
```

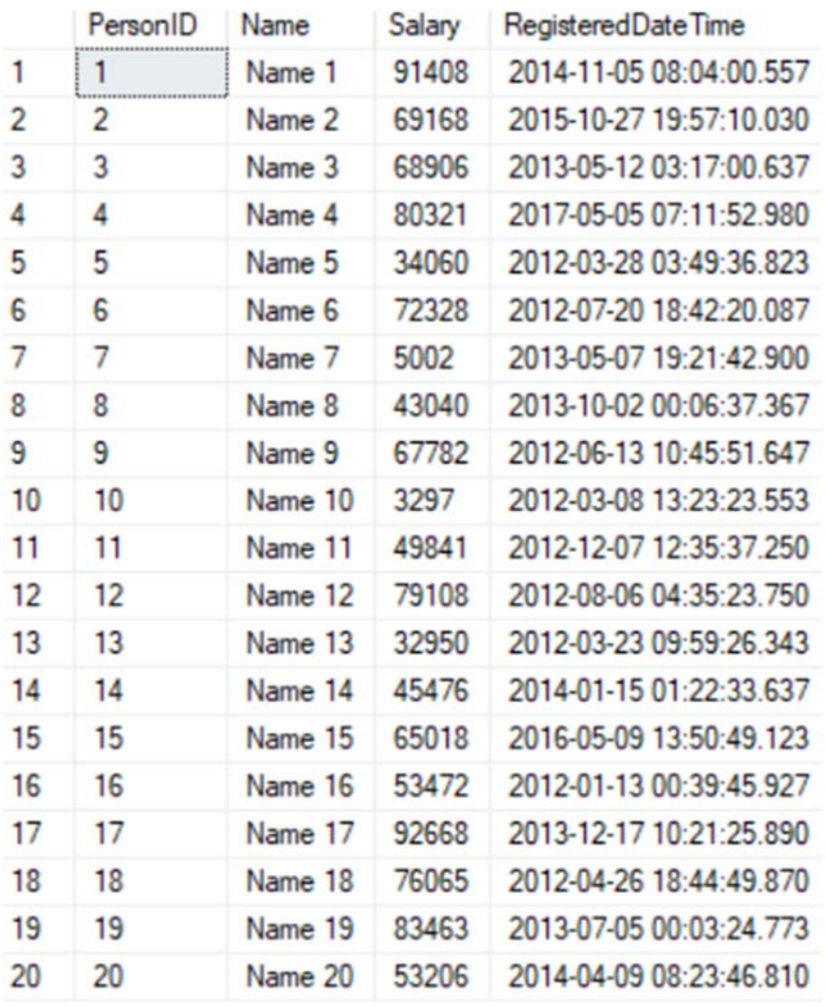


	name	type
1	dbo.spGetAllEmployees	stored procedure
2	dbo.spGetCountAllEmployees1	stored procedure
3	dbo.spGetCountAllEmployees2	stored procedure
4	dbo.spGetEmployeeCountByGenderID	stored procedure
5	dbo.spGetNameById1	stored procedure
6	dbo.spGetNameById2	stored procedure

	name	type	updated	selected	column
1	dbo.Employee	user table	no	yes	EmployeeID

8. Create Sample Data

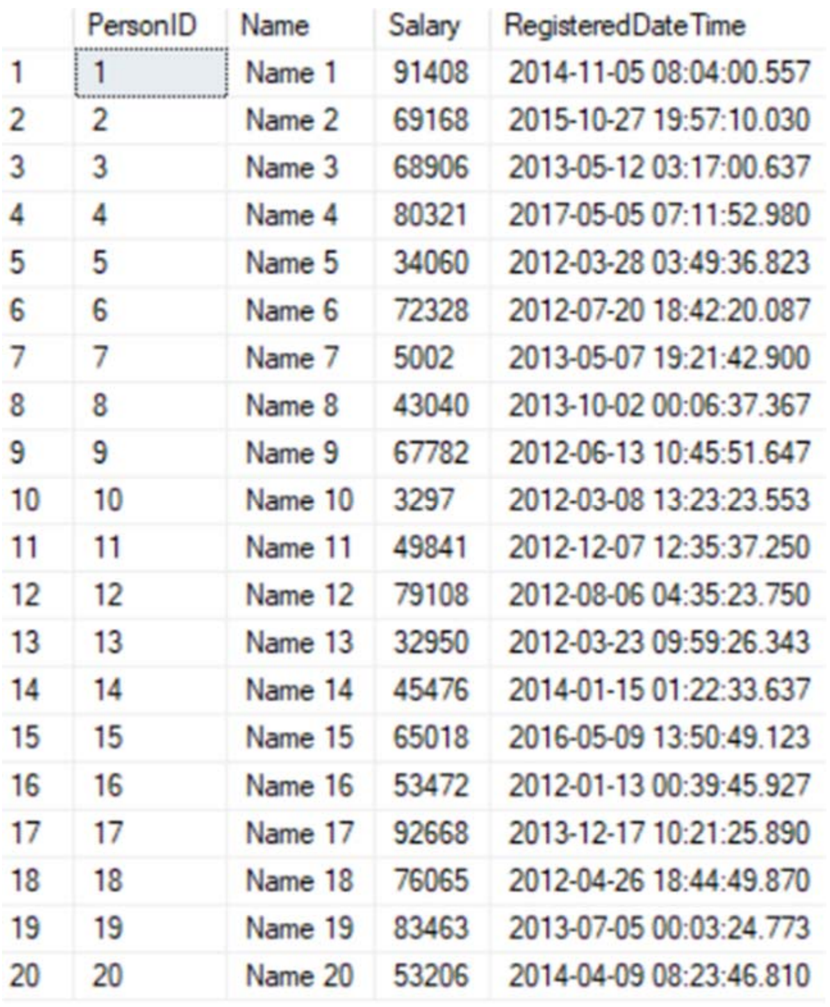
```
--==================================================================================
--T006_08_01
--Create or Recreate Table
IF ( EXISTS ( SELECT    *
             FROM      INFORMATION_SCHEMA.TABLES
             WHERE     TABLE_NAME = 'Person3' ) )
   BEGIN
       TRUNCATE TABLE Person3;
       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
```



	PersonID	Name	Salary	RegisteredDateTime
1	1	Name 1	91408	2014-11-05 08:04:00.557
2	2	Name 2	69168	2015-10-27 19:57:10.030
3	3	Name 3	68906	2013-05-12 03:17:00.637
4	4	Name 4	80321	2017-05-05 07:11:52.980
5	5	Name 5	34060	2012-03-28 03:49:36.823
6	6	Name 6	72328	2012-07-20 18:42:20.087
7	7	Name 7	5002	2013-05-07 19:21:42.900
8	8	Name 8	43040	2013-10-02 00:06:37.367
9	9	Name 9	67782	2012-06-13 10:45:51.647
10	10	Name 10	3297	2012-03-08 13:23:23.553
11	11	Name 11	49841	2012-12-07 12:35:37.250
12	12	Name 12	79108	2012-08-06 04:35:23.750
13	13	Name 13	32950	2012-03-23 09:59:26.343
14	14	Name 14	45476	2014-01-15 01:22:33.637
15	15	Name 15	65018	2016-05-09 13:50:49.123
16	16	Name 16	53472	2012-01-13 00:39:45.927
17	17	Name 17	92668	2013-12-17 10:21:25.890
18	18	Name 18	76065	2012-04-26 18:44:49.870
19	19	Name 19	83463	2013-07-05 00:03:24.773
20	20	Name 20	53206	2014-04-09 08:23:46.810

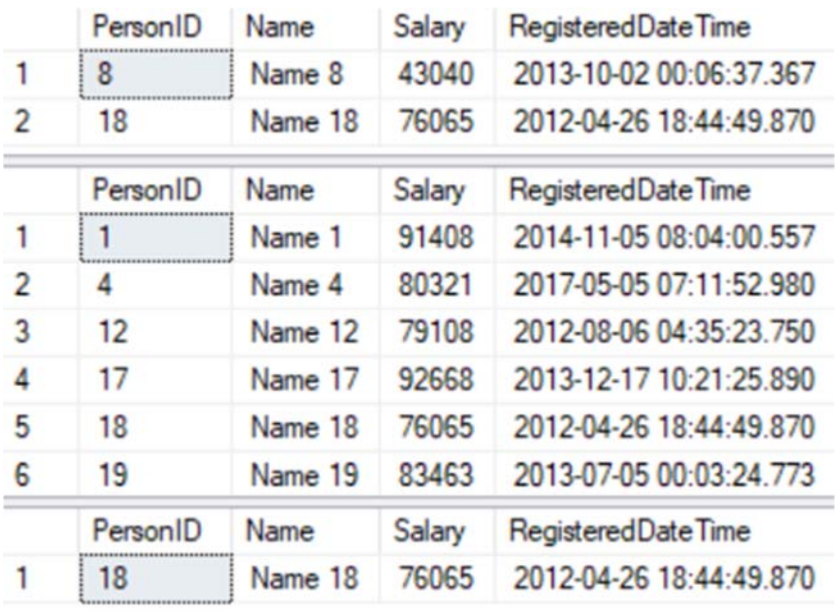
```
--===========================================================================================
--T006_08_03
--Create or Recreate store procedure
IF ( EXISTS ( SELECT   *
              FROM      INFORMATION_SCHEMA.ROUTINES
              WHERE     ROUTINE_TYPE = 'PROCEDURE'
                        AND 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.
```



	PersonID	Name	Salary	RegisteredDateTime
1	1	Name 1	91408	2014-11-05 08:04:00.557
2	2	Name 2	69168	2015-10-27 19:57:10.030
3	3	Name 3	68906	2013-05-12 03:17:00.637
4	4	Name 4	80321	2017-05-05 07:11:52.980
5	5	Name 5	34060	2012-03-28 03:49:36.823
6	6	Name 6	72328	2012-07-20 18:42:20.087
7	7	Name 7	5002	2013-05-07 19:21:42.900
8	8	Name 8	43040	2013-10-02 00:06:37.367
9	9	Name 9	67782	2012-06-13 10:45:51.647
10	10	Name 10	3297	2012-03-08 13:23:23.553
11	11	Name 11	49841	2012-12-07 12:35:37.250
12	12	Name 12	79108	2012-08-06 04:35:23.750
13	13	Name 13	32950	2012-03-23 09:59:26.343
14	14	Name 14	45476	2014-01-15 01:22:33.637
15	15	Name 15	65018	2016-05-09 13:50:49.123
16	16	Name 16	53472	2012-01-13 00:39:45.927
17	17	Name 17	92668	2013-12-17 10:21:25.890
18	18	Name 18	76065	2012-04-26 18:44:49.870
19	19	Name 19	83463	2013-07-05 00:03:24.773
20	20	Name 20	53206	2014-04-09 08:23:46.810

```
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
```



	PersonID	Name	Salary	RegisteredDateTime
1	8	Name 8	43040	2013-10-02 00:06:37.367
2	18	Name 18	76065	2012-04-26 18:44:49.870

	PersonID	Name	Salary	RegisteredDateTime
1	1	Name 1	91408	2014-11-05 08:04:00.557
2	4	Name 4	80321	2017-05-05 07:11:52.980
3	12	Name 12	79108	2012-08-06 04:35:23.750
4	17	Name 17	92668	2013-12-17 10:21:25.890
5	18	Name 18	76065	2012-04-26 18:44:49.870
6	19	Name 19	83463	2013-07-05 00:03:24.773

	PersonID	Name	Salary	RegisteredDateTime
1	18	Name 18	76065	2012-04-26 18:44:49.870

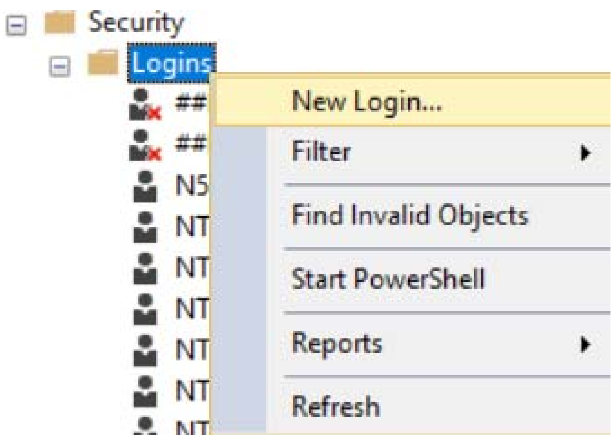
==

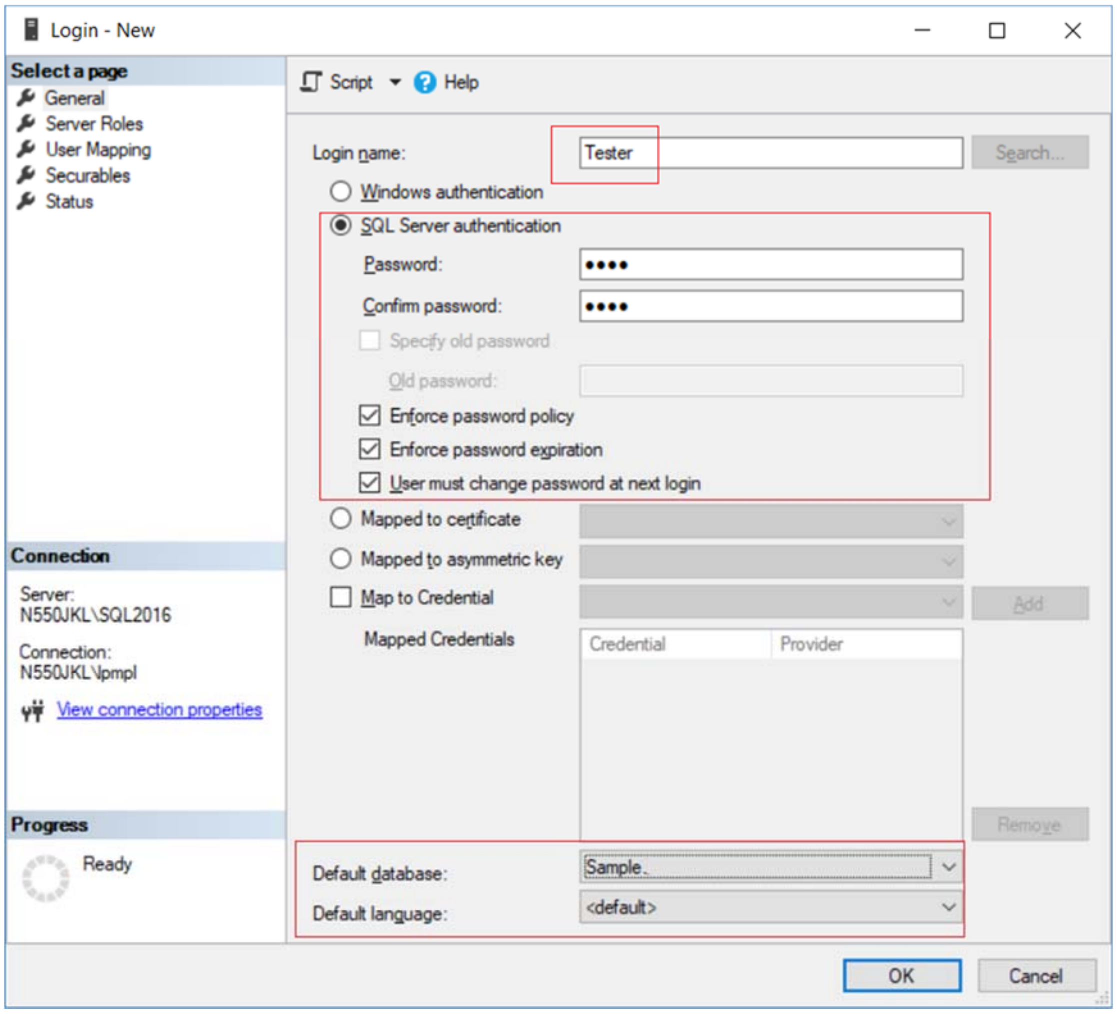
9. Web Application - Stored Procedure Optional Parameters

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.




Login - New

Select a page
- General
- Server Roles
- User Mapping
- Securables
- Status

Script ▾ Help

Login name: Tester [Search...]

Windows authentication
[● SQL Server authentication]

Password: ••••

Confirm password: ••••

☐ Specify old password

Old password:

☑ Enforce password policy

☑ Enforce password expiration

☑ User must change password at next login

Mapped to certificate

Mapped to asymmetric key

☐ Map to Credential [Add]

Mapped Credentials

Credential	Provider

[Remove]

Connection

Server:
N550JKL\SQL2016

Connection:
N550JKL\lpmpl

View connection properties

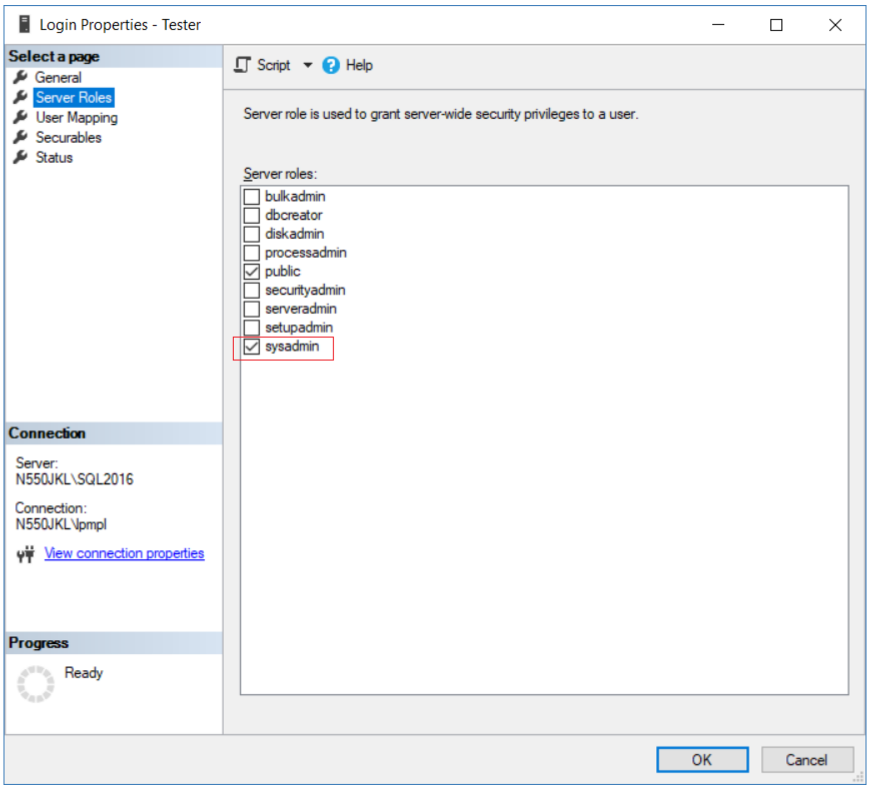
Progress

Ready

Default database: Sample

Default language: <default>

[OK] [Cancel]


Login Properties - Tester

Select a page
- General
- Server Roles
- User Mapping
- Securables
- Status

Script ▾ Help

Server role is used to grant server-wide security privileges to a user.

Server roles:
- [ ] bulkadmin
- [ ] dbcreator
- [ ] diskadmin
- [ ] processadmin
- [x] public
- [ ] securityadmin
- [ ] serveradmin
- [ ] setupadmin
- [x] sysadmin

Connection

Server:
N550JKL\SQL2016

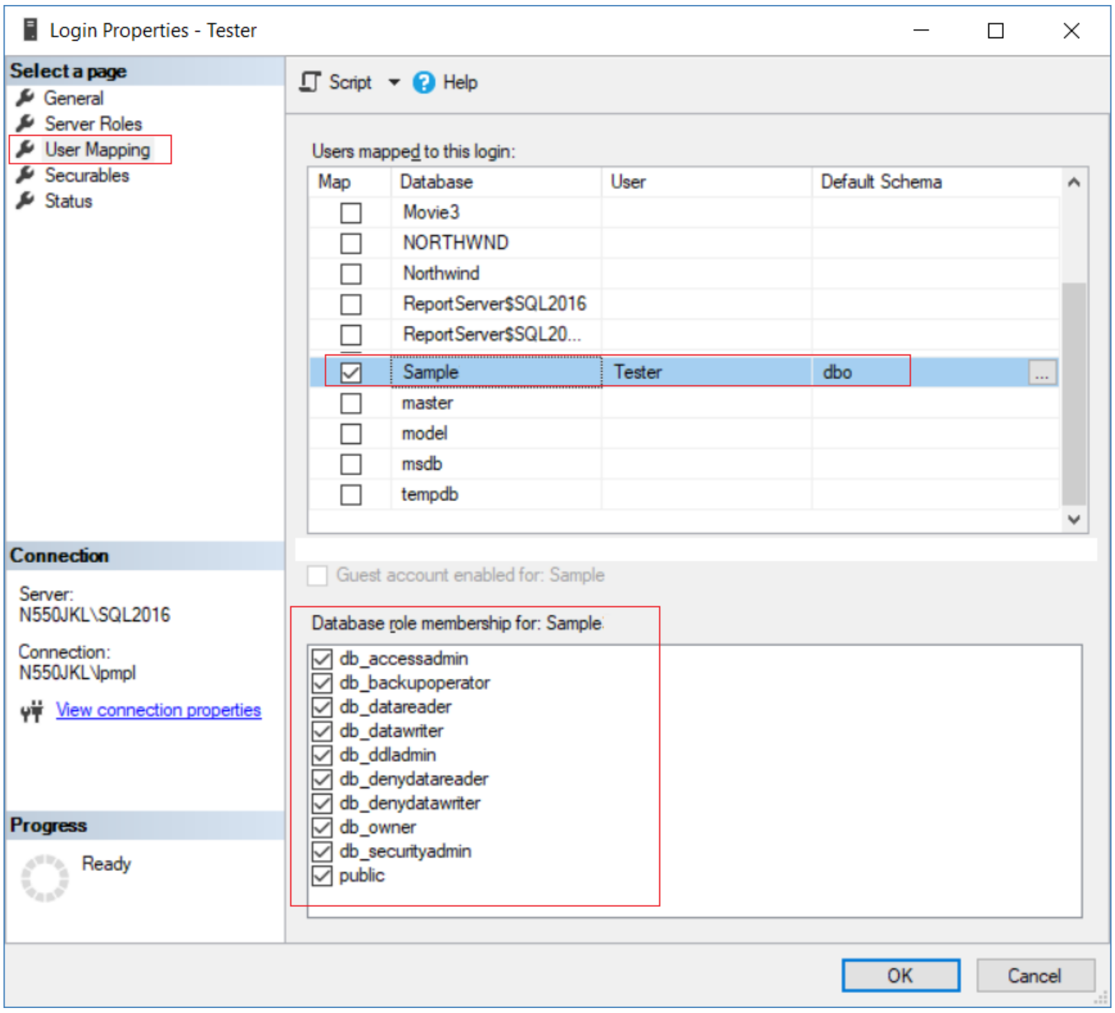
Connection:
N550JKL\lpmpl

View connection properties

Progress

Ready

OK Cancel


Login Properties - Tester

Select a page
- General
- Server Roles
- User Mapping
- Securables
- Status

Script ▾ Help

Users mapped to this login:

Map	Database	User	Default Schema
☐	Movie3		
☐	NORTHWND		
☐	Northwind		
☐	ReportServer$SQL2016		
☐	ReportServer$SQL20...		
✓	Sample	Tester	dbo
☐	master		
☐	model		
☐	msdb		
☐	tempdb		

- [ ] Guest account enabled for: Sample

Database role membership for: Sample
- [x] db_accessadmin
- [x] db_backupoperator
- [x] db_datareader
- [x] db_datawriter
- [x] db_ddladmin
- [x] db_denydatareader
- [x] db_denydatawriter
- [x] db_owner
- [x] db_securityadmin
- [x] public

Connection

Server:
N550JKL\SQL2016

Connection:
N550JKL\lpmpl

View connection properties

Progress

Ready

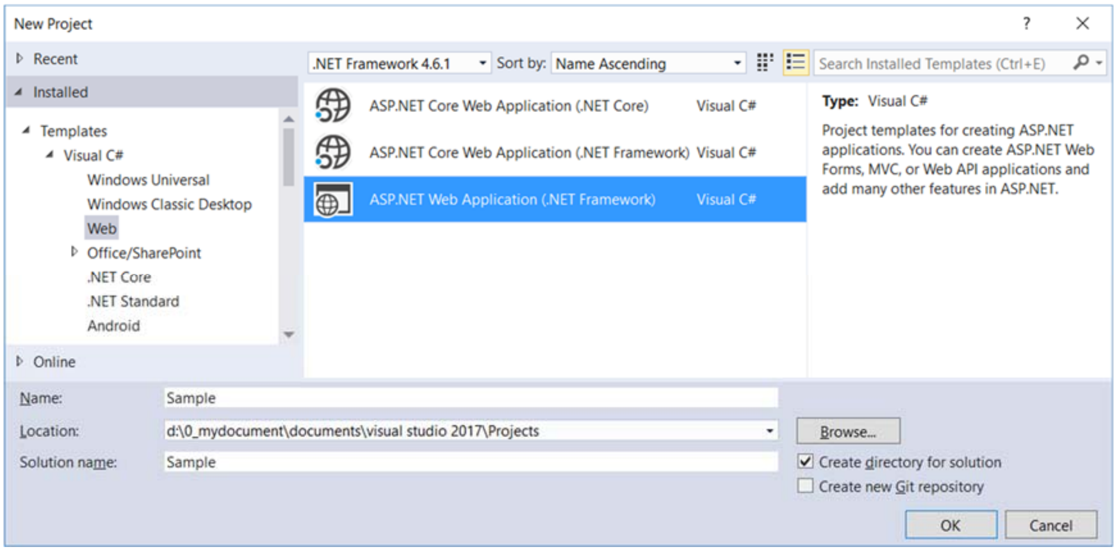
OK Cancel

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


New Project

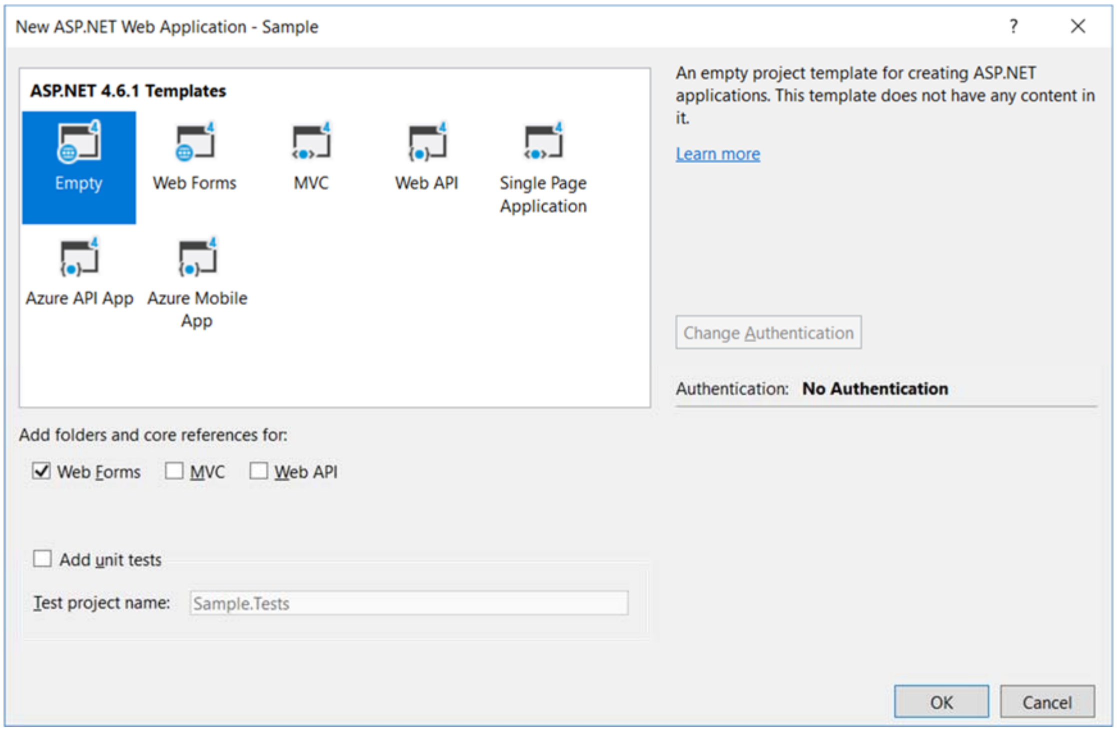
Recent
Installed
Templates
Visual C#
Windows Universal
Windows Classic Desktop
Web
Office/SharePoint
.NET Core
.NET Standard
Android
Online

.NET Framework 4.6.1 | Sort by: Name Ascending | Search Installed Templates (Ctrl+E)

ASP.NET Core Web Application (.NET Core)	Visual C#
ASP.NET Core Web Application (.NET Framework)	Visual C#
ASP.NET Web Application (.NET Framework)	Visual C#

Type: Visual C#
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

Name: Sample
Location: d:\0_mydocument\documents\visual studio 2017\Projects
Solution name: Sample
[✓] Create directory for solution
[ ] Create new Git repository


New ASP.NET Web Application - Sample

ASP.NET 4.6.1 Templates
Empty, Web Forms, MVC, Web API, Single Page Application, Azure API App, Azure Mobile App

An empty project template for creating ASP.NET applications. This template does not have any content in it.
Learn more

Authentication: **No Authentication**

Add folders and core references for:
[✓] Web Forms [ ] MVC [ ] Web API

[ ] Add unit tests
Test project name: Sample.Tests

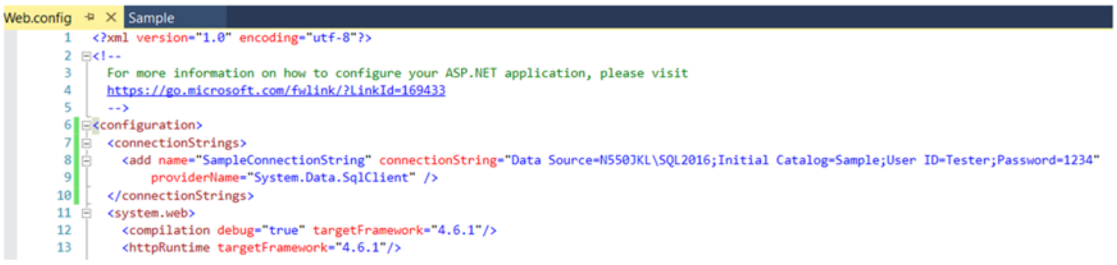
9.3. Code

9.3.1. Web.config

Add connection String

```
<configuration>
  <connectionStrings>
```

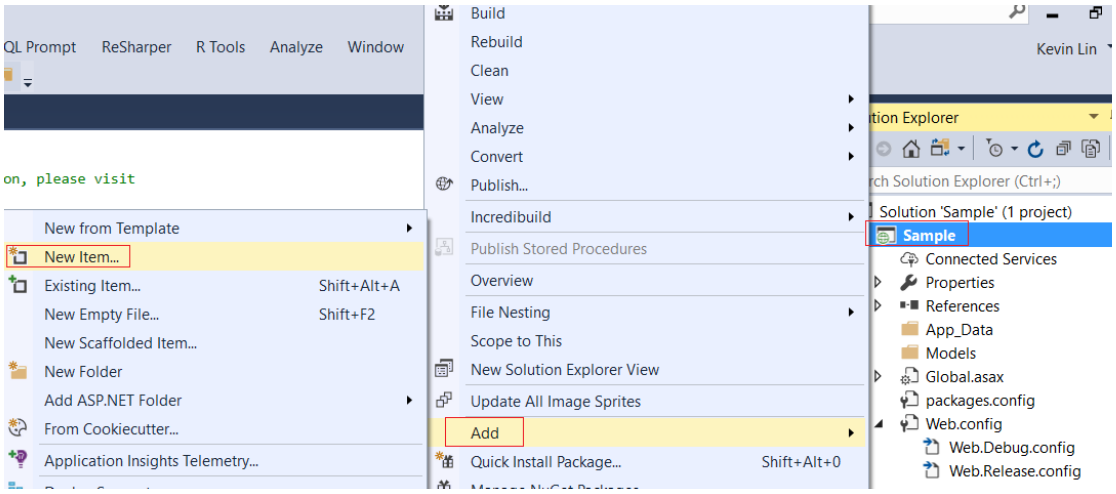
```
    <add name="SampleConnectionString" connectionString="Data Source=N550JKL\SQL2016;Initial
Catalog=Sample;User ID=Tester;Password=1234"
        providerName="System.Data.SqlClient" />
  </connectionStrings>
```



```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <connectionStrings>
    <add name="SampleConnectionString" connectionString="Data Source=N550JKL\SQL2016;Initial Catalog=Sample;User ID=Tester;Password=1234"
        providerName="System.Data.SqlClient" />
  </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.6.1"/>
    <httpRuntime targetFramework="4.6.1"/>
```

9.3.2. WebForm1.aspx

ProjectName --> Right Click --> Add --> New Item...
-->
WebForm
Name :
WebForm1.aspx



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Sample.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">;
<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>
```

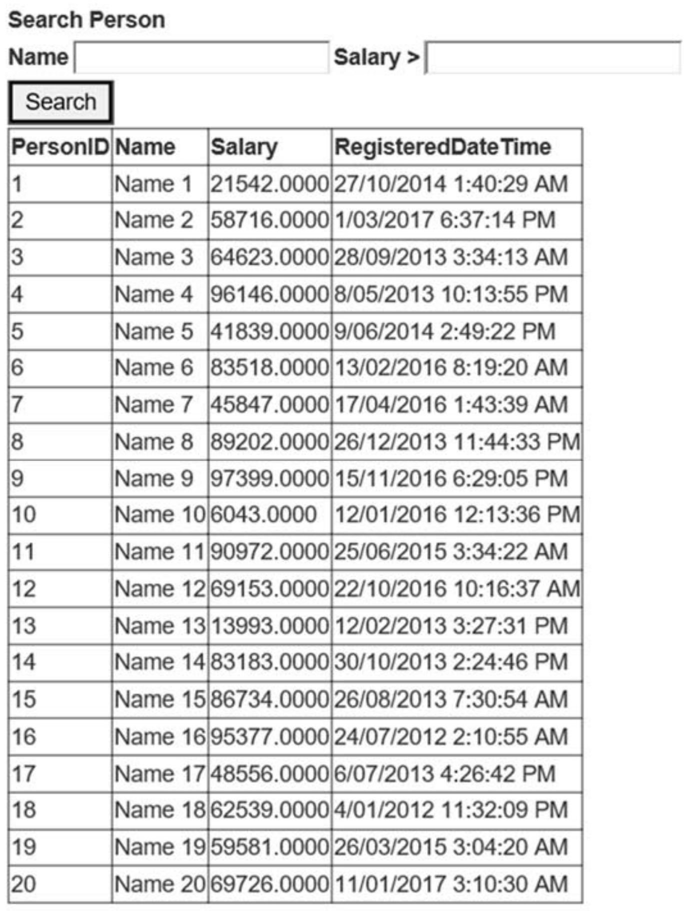
```
                    <td>
                        <b>Name</b>
                    </td>
                    <td>
                        <asp:TextBox ID="txtNameLike" runat="server"></asp:TextBox>
                    </td>
                    <td>
                        <b>Salary > </b>
                    </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 = CommandType.StoredProcedure;
                    AttachParameter(cmd, "@NameLike", txtNameLike);
                    AttachParameter(cmd, "@SalaryGreaterThan", txtSalaryGreaterThan);
                    con.Open();
                    gvEmployees.DataSource = cmd.ExecuteReader();
                    gvEmployees.DataBind();
                }
            }
        }
    }
```

9.3.4. Run it

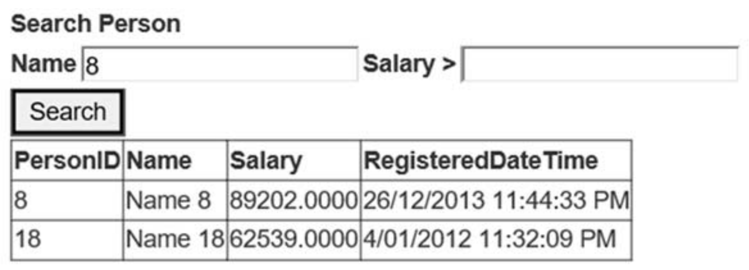


Search Person

Name [ ] **Salary >** [ ]

Search

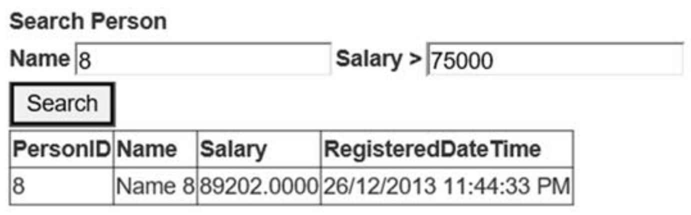
PersonID	Name	Salary	RegisteredDateTime
1	Name 1	21542.0000	27/10/2014 1:40:29 AM
2	Name 2	58716.0000	1/03/2017 6:37:14 PM
3	Name 3	64623.0000	28/09/2013 3:34:13 AM
4	Name 4	96146.0000	8/05/2013 10:13:55 PM
5	Name 5	41839.0000	9/06/2014 2:49:22 PM
6	Name 6	83518.0000	13/02/2016 8:19:20 AM
7	Name 7	45847.0000	17/04/2016 1:43:39 AM
8	Name 8	89202.0000	26/12/2013 11:44:33 PM
9	Name 9	97399.0000	15/11/2016 6:29:05 PM
10	Name 10	6043.0000	12/01/2016 12:13:36 PM
11	Name 11	90972.0000	25/06/2015 3:34:22 AM
12	Name 12	69153.0000	22/10/2016 10:16:37 AM
13	Name 13	13993.0000	12/02/2013 3:27:31 PM
14	Name 14	83183.0000	30/10/2013 2:24:46 PM
15	Name 15	86734.0000	26/08/2013 7:30:54 AM
16	Name 16	95377.0000	24/07/2012 2:10:55 AM
17	Name 17	48556.0000	6/07/2013 4:26:42 PM
18	Name 18	62539.0000	4/01/2012 11:32:09 PM
19	Name 19	59581.0000	26/03/2015 3:04:20 AM
20	Name 20	69726.0000	11/01/2017 3:10:30 AM



Search Person

Name: 8 Salary >

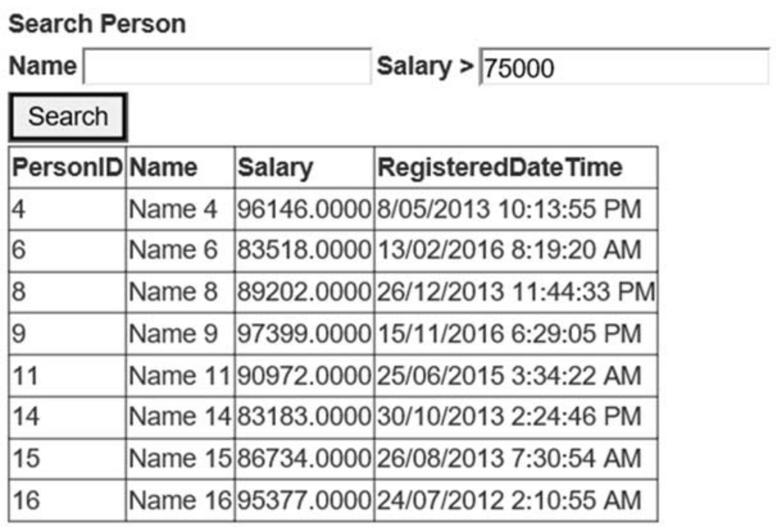
PersonID	Name	Salary	RegisteredDateTime
8	Name 8	89202.0000	26/12/2013 11:44:33 PM
18	Name 18	62539.0000	4/01/2012 11:32:09 PM



Search Person

Name: 8 Salary >: 75000

PersonID	Name	Salary	RegisteredDateTime
8	Name 8	89202.0000	26/12/2013 11:44:33 PM



Search Person

Name: Salary >: 75000

PersonID	Name	Salary	RegisteredDateTime
4	Name 4	96146.0000	8/05/2013 10:13:55 PM
6	Name 6	83518.0000	13/02/2016 8:19:20 AM
8	Name 8	89202.0000	26/12/2013 11:44:33 PM
9	Name 9	97399.0000	15/11/2016 6:29:05 PM
11	Name 11	90972.0000	25/06/2015 3:34:22 AM
14	Name 14	83183.0000	30/10/2013 2:24:46 PM
15	Name 15	86734.0000	26/08/2013 7:30:54 AM
16	Name 16	95377.0000	24/07/2012 2:10:55 AM