Build a SQL Server 2008 Stored Procedure dbo.FacultyInfo

A key point to build this SQL stored procedure is that our sample database **CSE_DEPT.mdf** should have been built and located at the default location, which is **C:\Program Files\Microsoft SQL Server\MSSQL10.SQL2008EXPRESS\MSSQL\DATA**. Refer to Chapter 2 to build this sample database if it has not been built.

To build a SQL Server stored procedure, many different methods can be adopted. Six possible ways can be used to create a stored procedure are:

1. Using SQL Server Enterprise Manager
2. Using Query Analyzer
3. Using ASP Code
4. Using Visual Studio.NET – Real Time Coding Method
5. Using Visual Studio.NET – Server Explorer
6. Using Enterprise Manager Wizard

In this section, we try to build this stored procedure with two of them, using Visual Studio.NET 2010 Server Explorer and SQL Server Management Studio. First let's discuss how to build this stored procedure with Visual Studio.NET 2010 Server Explorer.

H.1 Build the SQL Server 2008 Stored Procedure Using Server Explorer

Open the Microsoft Visual Studio.NET 2010 and open the Server Explorer by going to the **View | Server Explorer** menu item. Make sure that our sample database **CSE_DEPT.mdf** has been connected to the Visual Studio.NET 2010. If not, you need to use the Data Source window to first connect it by adding a new data source. The point to be noted is that you need to check the `Data Source` you are connecting is **SQL2008EXPRESS**. To do this checking, click on the **Advanced** button in the `Add Connection` wizard and then the `Data Source` property.

Expand our sample database **CSE_DEPT.mdf** from the Server Explorer window, and right click on the **Stored Procedures** folder, select `Add New Stored Procedure` item from the popup menu to open the `New Stored Procedure` window, which is shown in Figure H.1.

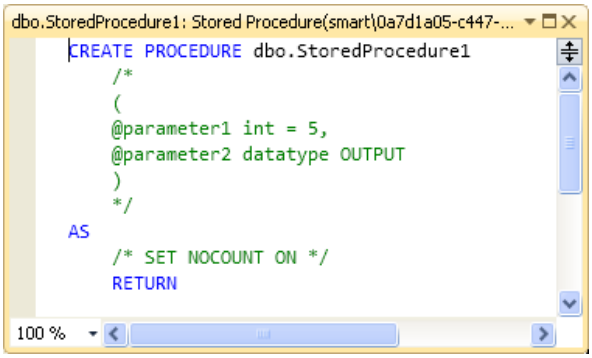

Figure H.1 The opened new stored procedure wizard.

Remove all comment out marks and replace the name of this stored procedure with the **dbo.FacultyInfo**. Add the codes that are shown in Figure H.2 into this stored procedure to make it as our target stored procedure. Go to **File | Save StoredProcedure1** menu item to save this stored procedure.

```
CREATE PROCEDURE dbo.FacultyInfo
(
A     @FacultyName VARCHAR(50),
      @Result VARCHAR(800) OUTPUT
)
AS
B     DECLARE @facultyID VARCHAR(50)
      DECLARE @fName VARCHAR(100)
      DECLARE @title VARCHAR(100)
      DECLARE @office VARCHAR(100)
      DECLARE @phone VARCHAR(100)
      DECLARE @college VARCHAR(100)
      DECLARE @email VARCHAR(100)
      DECLARE @message VARCHAR(800)

C     SET @facultyID=(SELECT faculty_id FROM Faculty WHERE faculty_name LIKE @FacultyName)
      SET @fName = (SELECT faculty_name FROM Faculty WHERE faculty_name LIKE @FacultyName)
      SET @title = (SELECT title FROM Faculty WHERE faculty_name LIKE @FacultyName)
      SET @office = (SELECT office FROM Faculty WHERE faculty_name LIKE @FacultyName)
      SET @phone = (SELECT phone FROM Faculty WHERE faculty_name LIKE @FacultyName)
      SET @college = (SELECT college FROM Faculty WHERE faculty_name LIKE @FacultyName)
      SET @email = (SELECT email FROM Faculty WHERE faculty_name LIKE @FacultyName)

D     SET @Result = @facultyID + ',' + @fName + ',' + @title + ',' + @office + ',' +
                    @phone + ',' + @college + ',' + @email

      PRINT ' '
E     SELECT @message = '----- ResultSet =: ' + @Result

      PRINT @message
      RETURN
```

Figure H.2 The codes for the stored procedure dbo.FacultyInfo.

Let's have a closer look at this new added piece of codes to see how it works.

A. Both IN and OUT parameters are first declared in the parameter section. The **@FacultyName** is an input parameter with a data type of VARCHAR(50) and the **@Result** is an output parameter with a data type of VARCHAR(800). The keyword OUTPUT must be attached after the OUT parameter to indicate that this is an output parameter in this stored procedure.

B. Eight local variables are created and the top seven are used to save the queried seven pieces of faculty information, and the last variable `@message` is used to display the running result of this stored procedure when it is tested in the Server Explorer environment.

C. The **SET** instruction is used to fetch seven pieces of faculty information from the Faculty table based on the input argument @FacultyName. The fetched columns are assigned to the associated seven local variables defined in step **B**.

D. The **SET** instruction is used to combine all seven pieces of fetched faculty information together into the OUT parameter @Result, and separate them with a comma mark.

E. To test this stored procedure, we use a SELECT statement to collect the values of the OUT parameter and use the PRINT command to display it.

Now let's test this stored procedure by right click on any place in this stored procedure, and select the **Execute** item from the popup menu to open the **Run Stored Procedure** dialog, which is shown in Figure H.3.

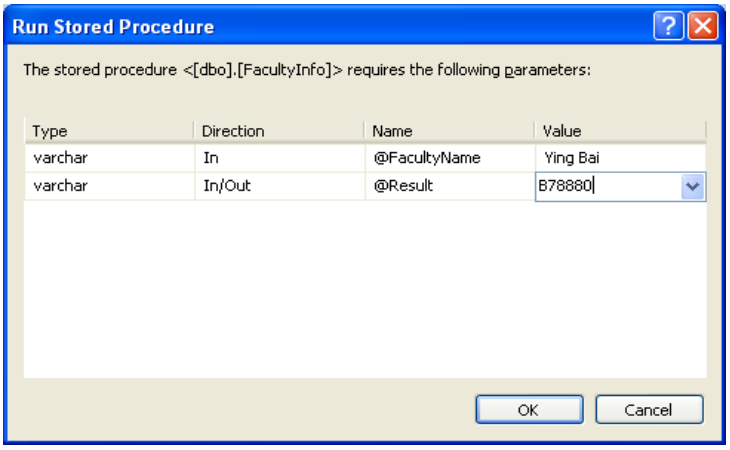

Figure H.3 The running status of the stored procedure.

Enter **Ying Bai** and **B78880** into the Value box as the input and the output parameters, as shown in Figure H.3. Click on the **OK** button to run this stored procedure. The running result is shown in the Output window, which is shown in Figure H.4.

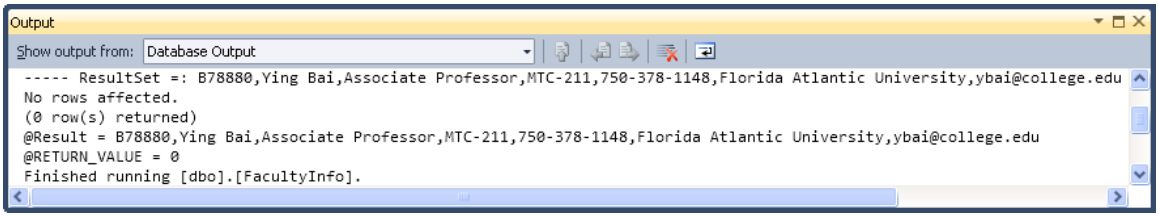

Figure H.4 The running result of the stored procedure.

Our SQL stored procedure **dbo.FacultyInfo** is successful!

Before we can call this stored procedure from our FacultyFrame Form to test the CallableStatement interface, make sure that you have closed the connection between our sample database **CSE_DEPT.mdf** and the Visual Studio.NET 2010. To do that, right click on our sample database **CSE_DEPT.mdf** from the Server Explorer window, and select the **Close Connection** item from the popup menu. Otherwise, you may encounter a connection exception when you run our Java application project.

Next let's discuss how to build this stored procedure using the SQL Server Management Studio.

H.2 Build the SQL Server Stored Procedure Using SQL Server Management Studio

Launch the Microsoft SQL Server Management Studio by going to **Start > All**

Programs > Microsoft SQL Server 2008 > SQL Server Management Studio. Click the `Connect` button to open this studio server. On the opened studio, expand the **Databases** and our sample database **CSE_DEPT** nodes. Then expand the **Programmability** node and right click on the **Stored Procedures** node, select the **New Stored Procedure** to open a new stored procedure template, as shown in Figure H.5.

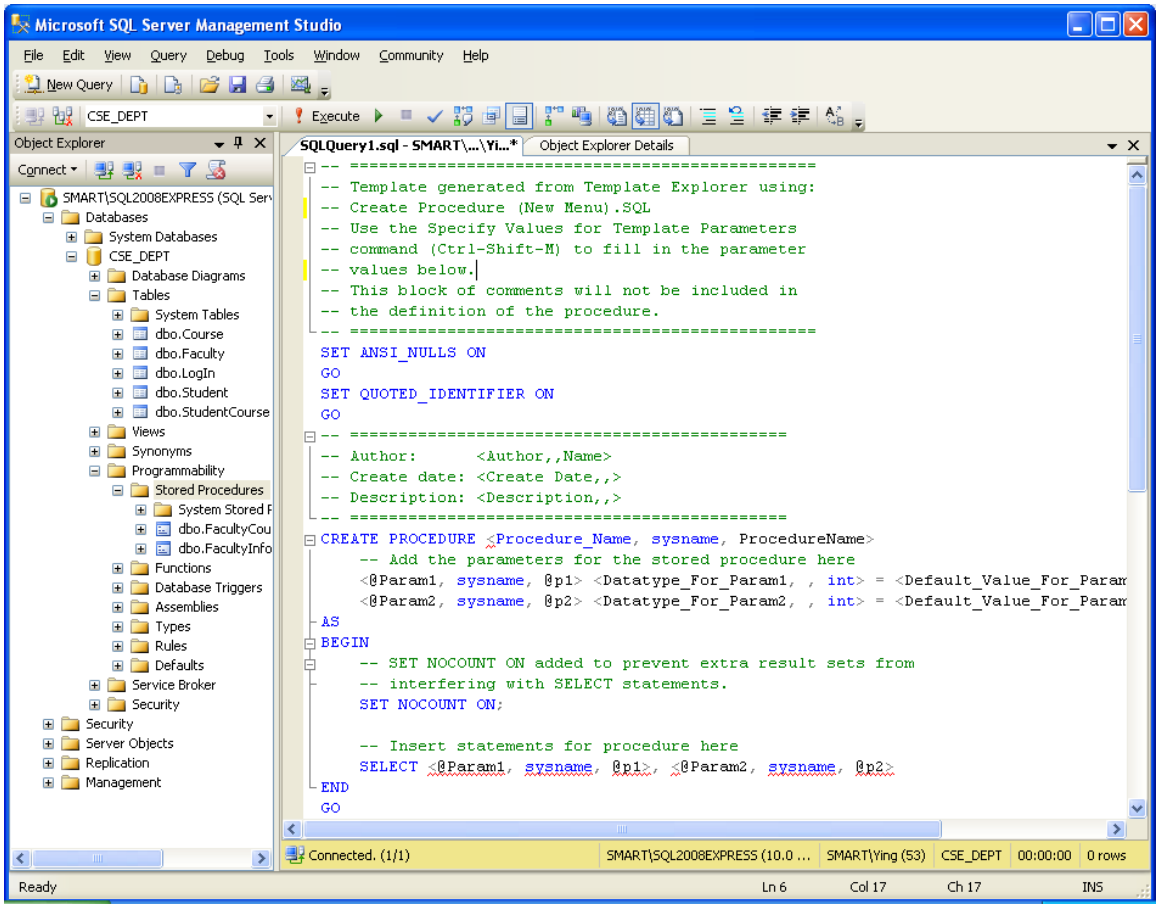

Figure H.5 The new stored procedure template.

You can use the `Ctrl-Shift-M` combination keys to enter all parameters for this stored procedure. However, an easy way to do that is to directly enter all parameters manually. On the opened new stored procedure template, enter the following codes that are shown in Figure H.6 into this stored procedure template as the body of our new stored procedure.

Go to **File > Save SQLQuery1.sql** to save this stored procedure.

Right click on any location inside our new stored procedure and select the **Execute** item to try to run it. Then right click on the Stored Procedures node from the Object Explorer window and select the **Refresh** item to refresh it to get our new created stored procedure **dbo.FacultyInfo**. Right click on our new stored procedure and select the **Execute Stored Procedure** to open the `Execute Procedure` wizard, which is shown in Figure H.7.

Enter a set of parameters shown in Figure H.7 into the associated **Value** columns as a new course record, and click on the `OK` button to run this stored procedure to test its functionality.

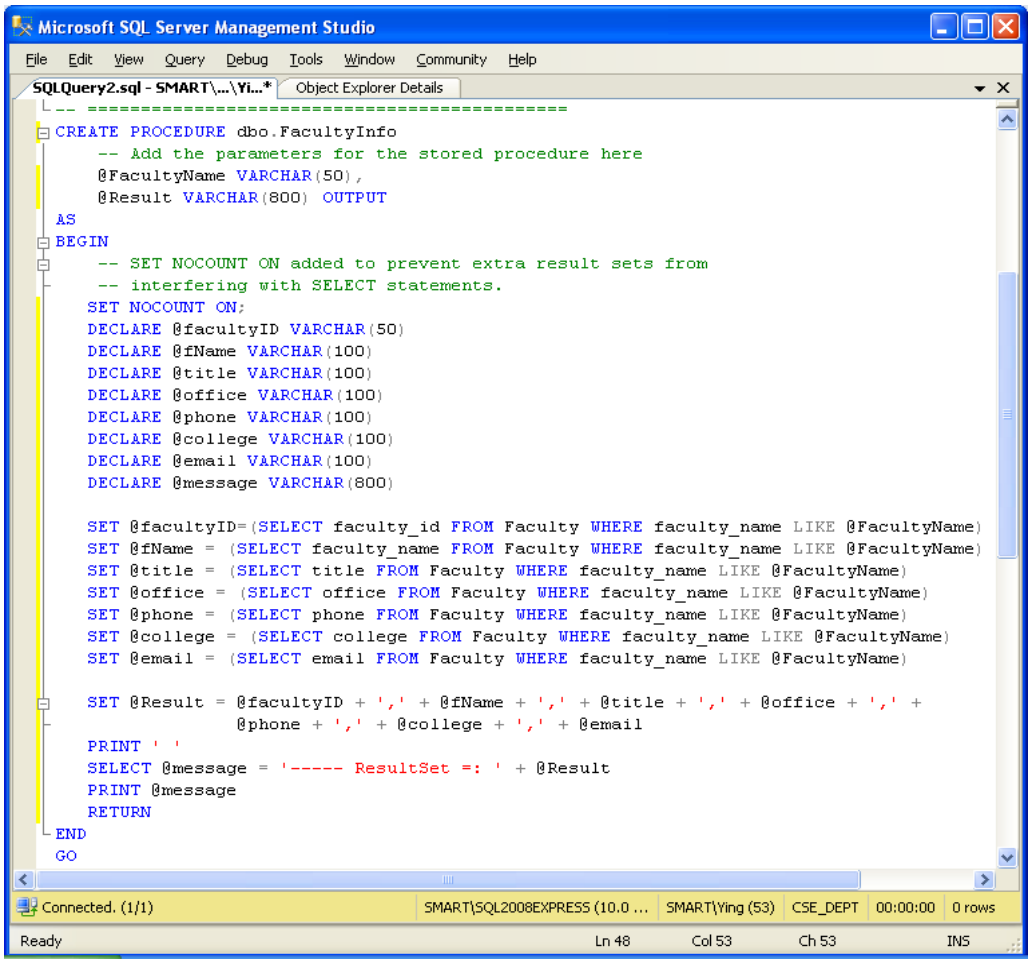

Figure H.6 The codes for our new stored procedure.

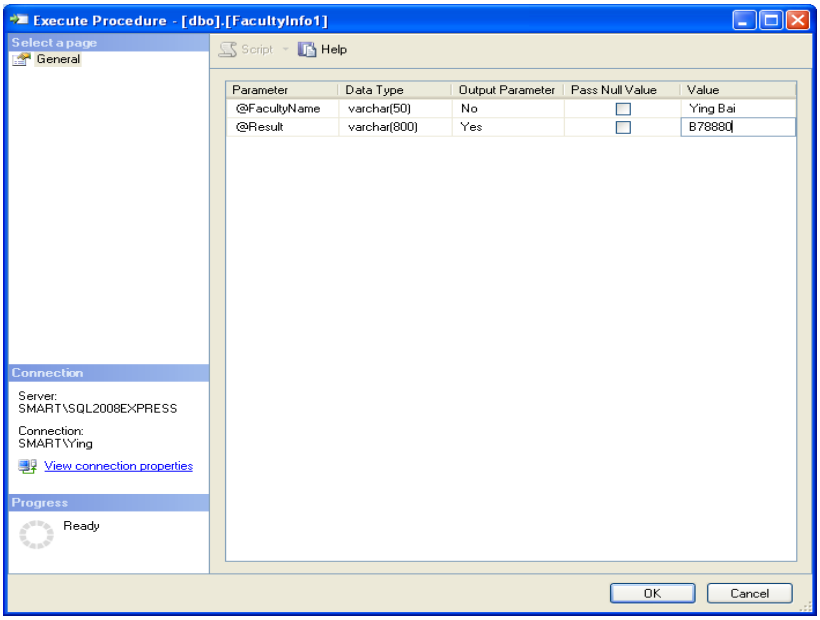

Figure H.7 The opened Execute Procedure wizard.

The test result is shown in Figure H.8. It can be found that a successful message, **1 row(s) affected**, is displayed in the **Output** window.

Now close the Microsoft SQL Server Management Studio Express and we can continue to develop the codes for the CallableStatement method to call this stored procedure to perform a new course insertion action against our sample database.

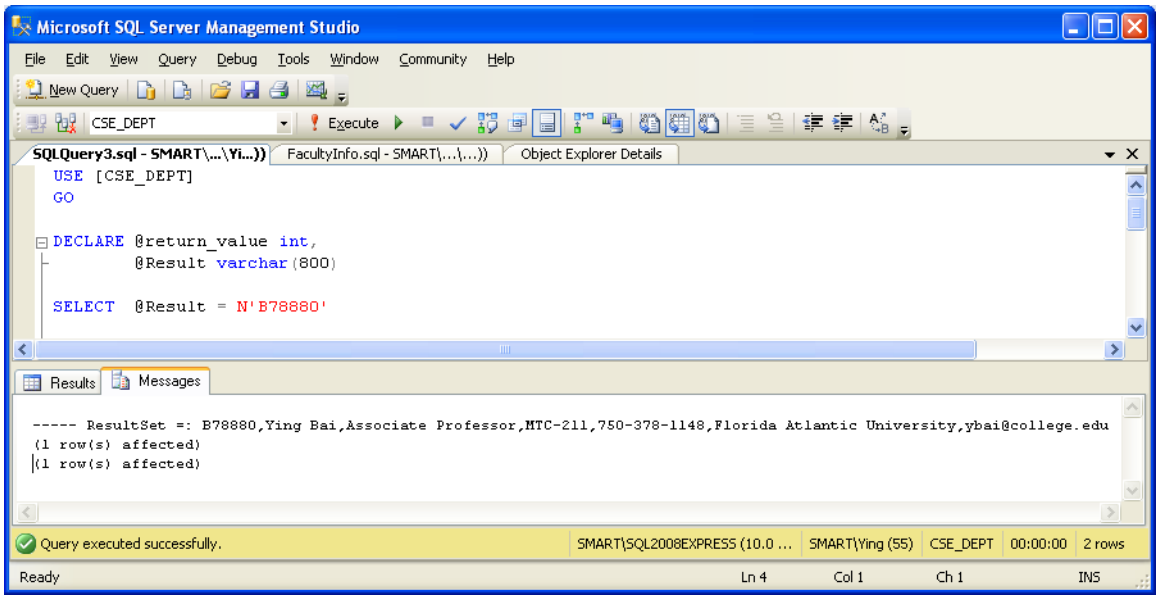

Figure H.8 The running result of the stored procedure.