Cairo University
Faculty of Engineering
Computer Engineering Dept.
CMP N202

Fall 2015

LAB #4
Accessing SQL Server database from C# application

Objectives

After this lab, the student should be able to use Visual Studio to:

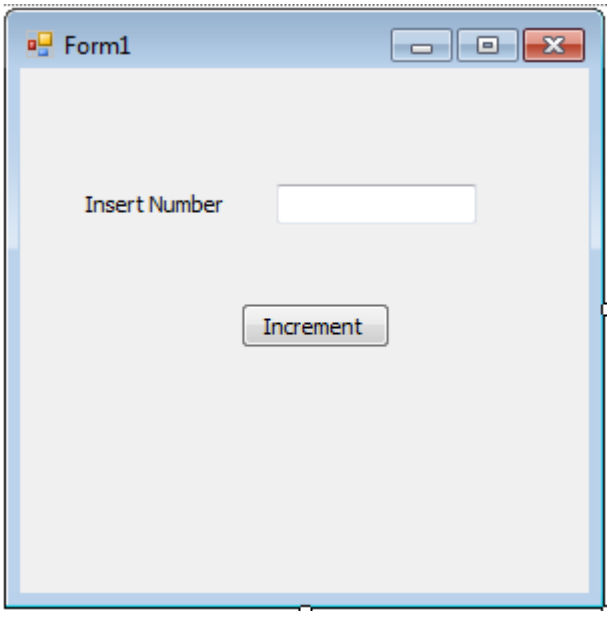
- Create Simple windows Application using C#.
- Write scripts to create database, create tables, or fill tables data
- Connect C# application to a database.
- Execute simple SQL queries on the database by calling direct SQL statements in C# application.

Lab Outline

I. Create a windows application

Creating Simple Windows Application

1. Open **Microsoft Visual Studio 2010** , From **File** select **New** then **Project**.
2. Select from **Visual C#** , **Windows Forms Application.**
3. The application creates a default form for you. And You can add more forms if you want by right-clicking on your project in the Solution Explorer and selecting **Add Windows Form** or **Add New Item Windows Form.** (But we will use the default form created "Form1")
4. Add GUI elements Textbox, button to the form.
   Open **Toolbox**(If it doesn't appear, choose it from view)
   Choose textbox, button and Label and drag them into your created form.
   Let the event handler of the button to add 1 into the value entered in textbox and show the result into the same text box.
   You can **right click** any GUI element and modify its name, color, etc from **Properties**.
   Your form will look like this



5. Add **event Handler** (function that will be executed when the button is clicked)for the button by double clicking it.

```
            int value=0;
            //TryParse in order not to prevent exception if
non-integer value was entered
            if (int.TryParse(textbox.Text, out value))
            {
                value = value + 1;
                //set the result to the text box
                textbox.Text = value.ToString();
            }
```

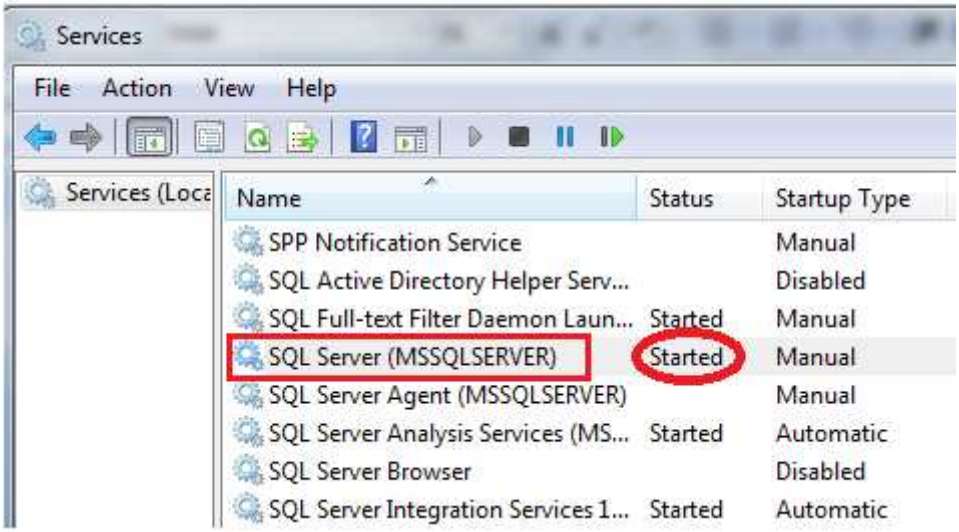
6. Run the application **(CTRL+F5**).

II. Create a data connection to a SQL Server database

Before you create a connect to a database, you should

Make sure SQL server is running

1. Click Windows Start Button and type "local services". Select "view local service"
2. Make sure that "**SQL Server**" is "**Started**". If not, right-click and start it. You can also make it start automatically with Windows through its **Properties.**
   Note: Here server name is **MSSQLSERVER**, you may have a different server name.



Make sure your database is created

You can create and fill a database either through graphical interface (Management Studio) or by running a script.

In this lab, you are given an SQL script that creates and fills the database

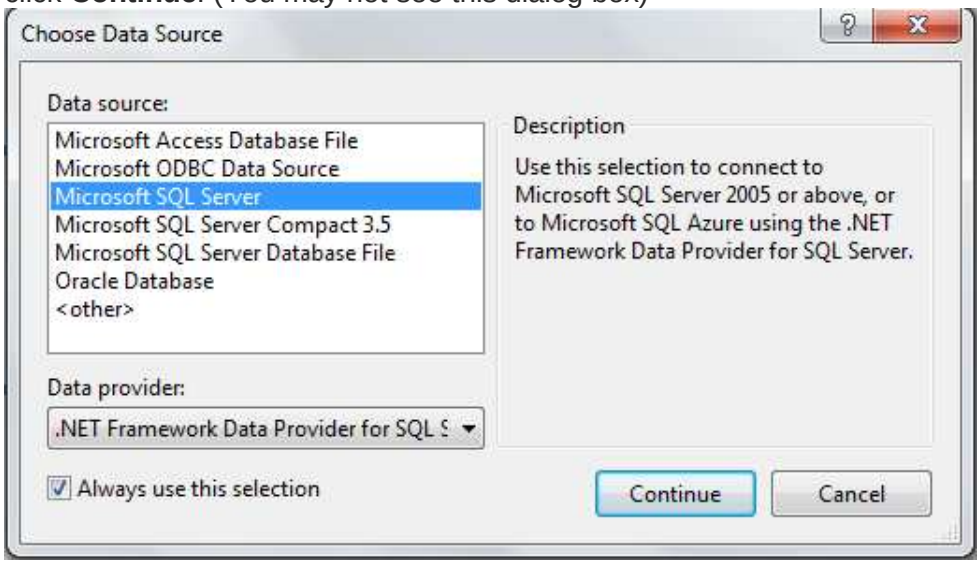
1. Open SQL Server Management Studio and connect to the server (as in previous lab)
2. From File → Open → File, browse to open the SQL script SPJ_DB_Lab4. The script is provided with the lab material under folder
3. Execute the script (click Execute)
4. This script creates a database SPJ_DB_Lab4, creates database tables, and fills tables with some sample data.

Create a connection to get connections string

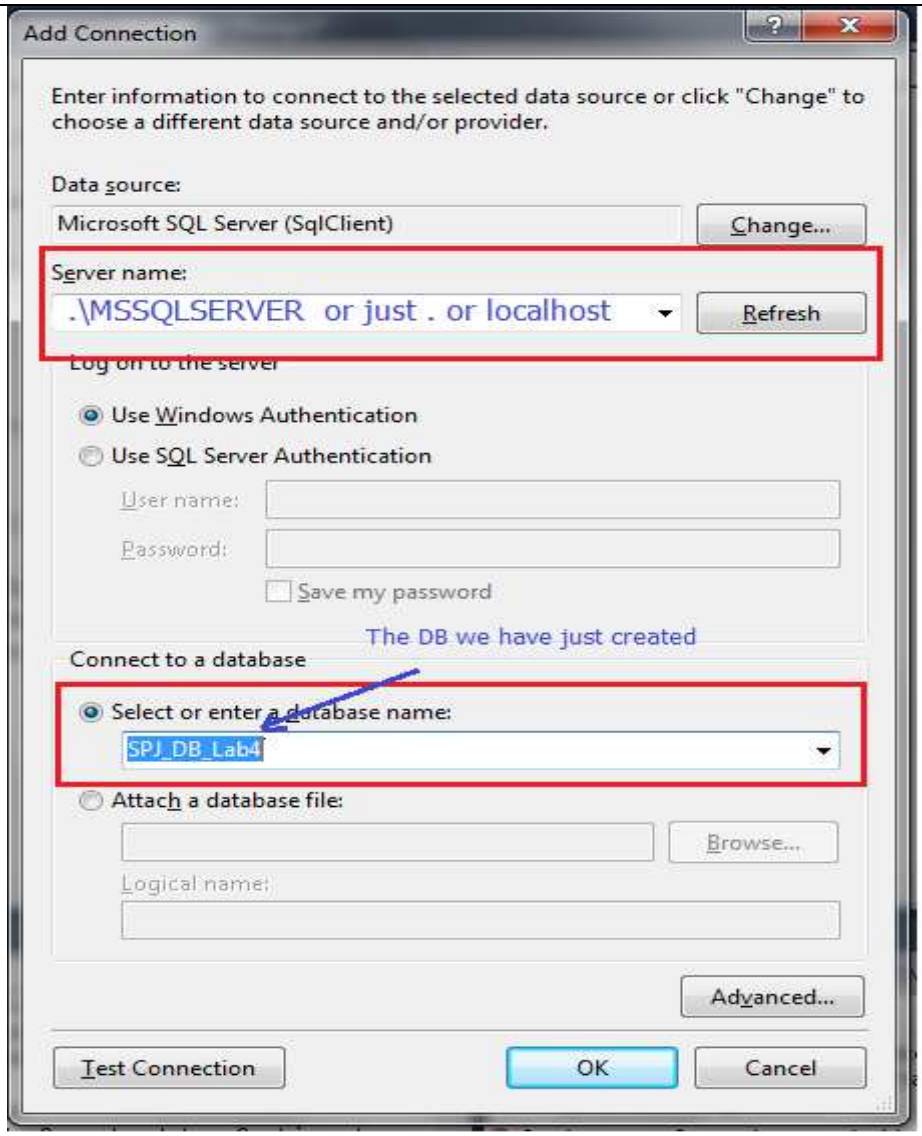
Go back to Visual Studio and create a new Windows Application then:

1. In **Server Explorer/Database Explorer** right-click **Data Connections, select Add Connection**

2. In the **Choose Data Source** dialog box, select **Microsoft SQL Server**, and then click **Continue**. (You may not see this dialog box)



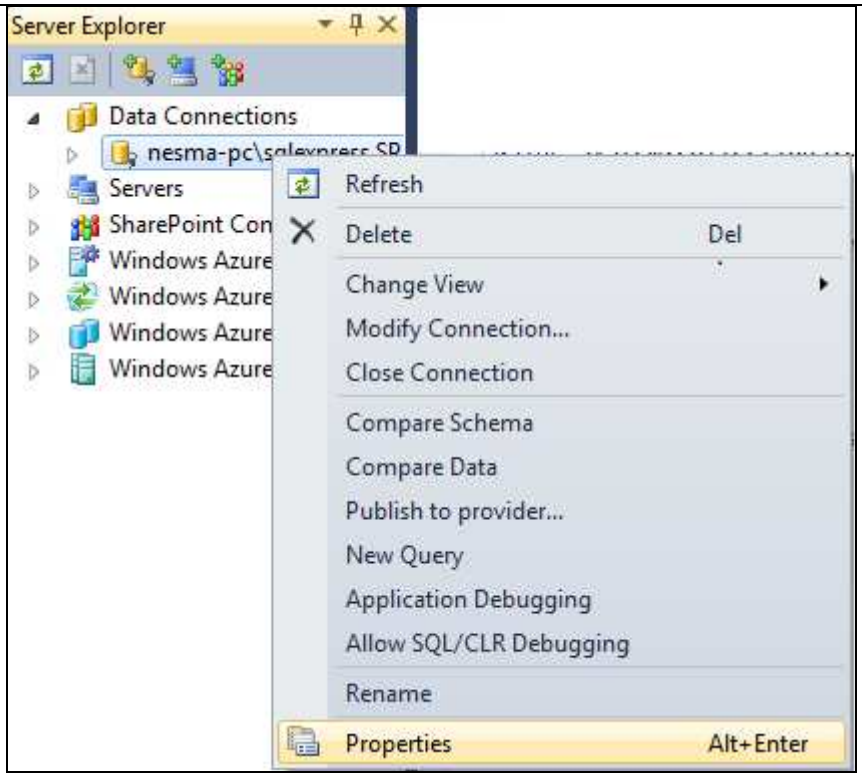
3. Select a server name from the drop-down list, or type the name of the server where the database you want to access is located. (see next figure)
4. Based on the requirements of your database or application, select either **Windows Authentication** or use a specific user name and password to log on to the SQL Server (**SQL Server Authentication**).
5. Select the database you want to connect to from the drop-down list. (select SPJ_DB_Lab4)
6. Click **OK**.



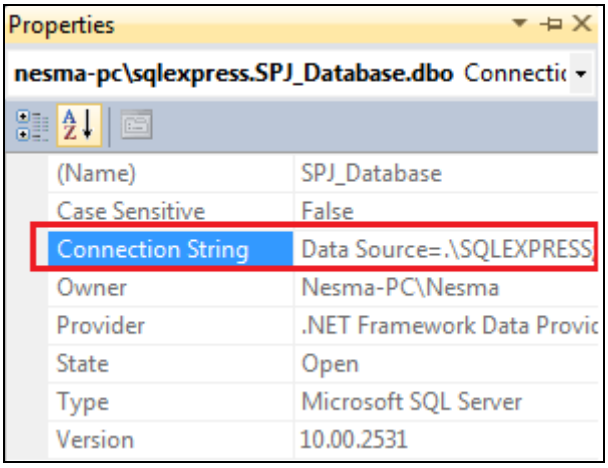
III. Get the connection string

Steps

1. In **Server Explorer/Database Explorer**, right click on the database connection you created and choose **Properties**



2. In the Properties list you will find a property called **"Connection String"**, copy its value to use it in the c# code



The **Connection string** specifies information about a data source, the way of connecting to it and other attributes for connection such as security.

Question : Is it professional and secure to embed it in the code?

Store the connection string as we will be using it in next steps

Now, Open the C# application provided with the lab. You will find it under Code\DBapplication

To deal with the DB, the application code makes use of three classes; **SqlConnection, SQLCommand** and **SqlDataReader**

To use them you should write next line outside the project namespace (see DBManager.cs code file)

```
using System.Data.SqlClient;
```

IV.Open the connection to the database

Code (see file DBManager.cs)

```
// SqlConnection class is used to open a connection to a database.
string  DB_Connection_String = @"Data Source=.;Initial Catalog=SPJ_DB_Lab4;Integrated
Security=True"
SqlConnection myConnection; myConnection = new SqlConnection(DB_Connection_String);
try
{
     myConnection.Open(); //Open a connection with the DB
     Console.WriteLine("The DB connection is opened successfully");
}
catch (Exception e)
{
     Console.WriteLine("The DB connection is failed");
}
```

V. Insert Query Command

To execute Insert/Delete/Update statement, you should use method (function) ***SqlCommand::ExecuteNonQuery( )***

Code (see file DBManager.cs, and see who actually prepares the query)

```
try
{
   //First prepare the statement to be executed
    string query = "INSERT INTO S (S#, Name, City, Status) " +
                   "Values ('S4','Mohamed','Cairo',20);";
 // myConnection is the connection opened in the previous step
 //SqlCommand class is used for executing queries and stored procedures on the    database.
     SqlCommand myCommand = new SqlCommand(query, myConnection);
     // ExecuteNonQuery returns the number of rows affected
     return myCommand.ExecuteNonQuery();
}
catch (Exception ex)
{
     Console.WriteLine(ex.Message);
     return 0;
}
```

VI.Delete Query Command

Code (see file DBManager.cs, and see who actually prepares the query)

```
try
{
     string query = "DELETE FROM S WHERE S#='S4';";
     SqlCommand myCommand = new SqlCommand(query, myConnection);
     return myCommand.ExecuteNonQuery();
}
catch (Exception ex)
{
     Console.WriteLine(ex.Message);
     return 0;
}
```

VII. Update Query Command

Code (see file DBManager.cs, and see who actually prepares the query)

```
try
{
     string query = "UPDATE S SET City='Cairo' WHERE S#='S2';";
     SqlCommand myCommand = new SqlCommand(query, myConnection);
     return myCommand.ExecuteNonQuery();
}
catch (Exception ex)
{
     Console.WriteLine(ex.Message);
     return 0;
}
```

VIII. Select Query Command that returns a Scalar

SqlCommand::ExecuteScalar( ) method

Code (see file DBManager.cs, and see who actually prepares the query)

```
try
{
     string query = "SELECT COUNT(S#) FROM S;";
     SqlCommand myCommand = new SqlCommand(query, myConnection);
     return myCommand.ExecuteScalar();
}
catch (Exception ex)
{
     Console.WriteLine(ex.Message);
     return 0;
}
```

IX.Select Query Command that returns a DataTable

SqlCommand::ExecuteReader( ) method

Code (see file DBManager.cs, and see who actually prepares the query)

```
try
{      SqlCommand myCommand = new SqlCommand(query, myConnection);
       SqlDataReader reader = myCommand.ExecuteReader();
       if (reader.HasRows)
       {      DataTable dt = new DataTable();
              dt.Load(reader);
              reader.Close();
              return dt;
       }
       else
       {  return null;  }
}
catch (Exception ex)
{
       Console.WriteLine(ex.Message);
       return null;
}
```

X. Close the connection to the database

Code

```
try
{
     myConnection.Close();
}
catch (Exception e)
{
     Console.WriteLine(e.Message);
}
```

Requirements

Use file ***CompanyDBLab4.sql*** to create and fill the company database.

Create a simple windows application that is connected to the company database. The application should allow the user to:

1. Insert new employees
2. update the information of a certain employee
3. Get all employees who work in a certain department.
4. Get the number of employees who work on a certain project
5. Delete a project

Hint:
Try to organize your code by separating it into classes for example one class dealing with database and another for application logic.

Evaluation Criteria:
1 mark for each operation
2 marks for modularity
1 mark for validation
2 mark for GUI

Bonus:
Reading the connection string from a file rather than embedding its value in the code.