**Lab Session 4: Data Access with C#**

Accessing Data from a database is one of the important aspects of any programming language. It is an absolute necessity for any programming language to have the ability to work with databases. C# is no different.

It can work with different types of databases. It can work with the most common databases such as Oracle and Microsoft SQL SERVER.

**Fundamentals of Database connectivity**

C# and .Net can work with a majority of databases, the most common being Oracle and Microsoft SQL Server. But with every database, the logic behind working with all of them is mostly the same. Working with databases, the following are the concepts which are common to all databases.

1. **Connection** – To work with the data in a database, the first obvious step is the connection. The connection to a database normally consists of the below-mentioned parameters.
   1. **Database name or Data Source** – The first important parameter is the database name to which the connection needs to be established. Each connection can only work with one database at a time.
   2. **Credentials** – The next important aspect is the username and password which needs to be used to establish a connection to the database. It ensures that the username and password have the necessary privileges to connect to the database.
   3. **Optional parameters** – For each database type, you can specify optional parameters to provide more information on how .net should handle the connection to the database. For example, one can specify a parameter for how long the connection should stay active. If no operation is performed for a specific period of time, then the parameter would determine if the connection has to be closed.
2. **Selecting data from the database** – Once the connection has been established, the next important aspect is to fetch the data from the database. C# can execute ‘SQL’ select command against the database. The ‘SQL’ statement can be used to fetch data from a specific table in the database.
3. **Inserting data into the database** – C# can also be used to insert records into the database. Values can be specified in C# for each row that needs to be inserted into the database.
4. **Updating data into the database** – C# can also be used to update existing records into the database. New values can be specified in C# for each row that needs to be updated into the database.
5. **Deleting data from a database** – C# can also be used to delete records into the database. Select commands to specify which rows need to be deleted can be specified in C#.

Ok, now that we have seen the theory of each operation, let’s jump into the further sections to look at how we can perform database operations in C#.

**SQL Command in c#**

**SqlCommand in C#** allow the user to query and send the commands to the database. SQL command is specified by the SQL connection object. Two methods are used, ExecuteReader method for results of query and ExecuteNonQuery for insert, Update, and delete commands. It is the method that is best for the different commands.

**How to connect C# to Database**

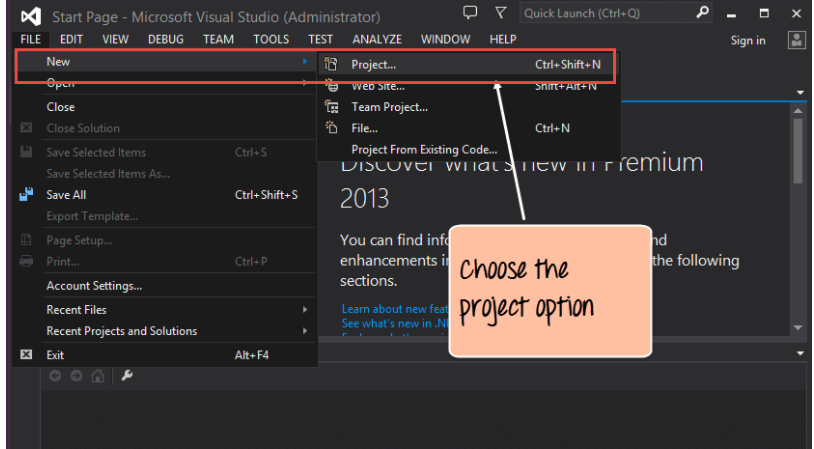
Let’s now look at the code, which needs to be kept in place to create a connection to a database. In our example, we will connect to a database which has the name of Demodb. The credentials used to connect to the database are given below

* + **Username – sa**
  + **Password – demo123**

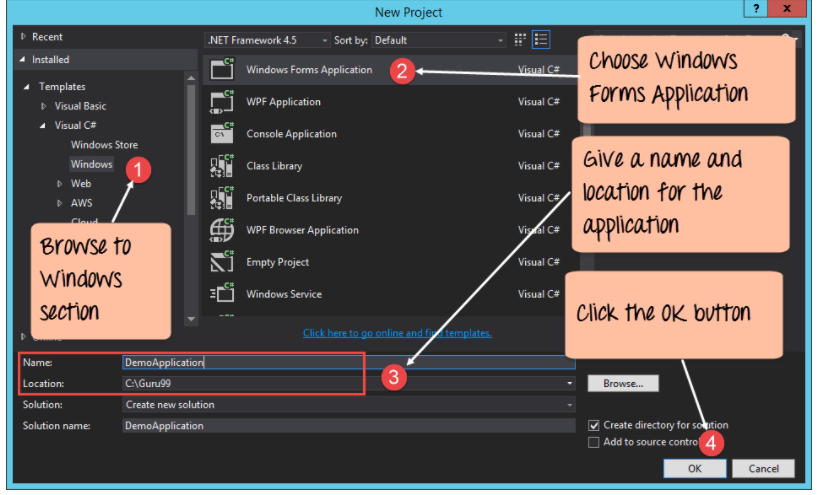
We will see a simple Windows forms application to work with databases. We will have a simple button called “Connect” which will be used to connect to the database.

So let’s follow the below steps to achieve this

**Step 1)**the first step involves the creation of a new project in Visual Studio. After launching Visual Studio, you need to choose the menu option New->Project.

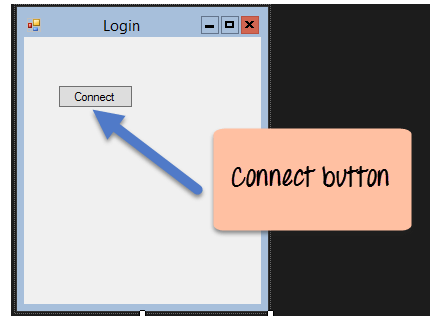


**Step 2)** the next step is to choose the project type as a Windows Forms application. Here, we also need to mention the name and location of our project.

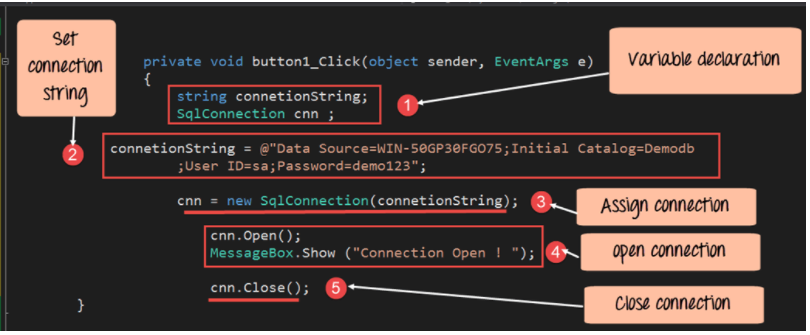


1. In the project dialog box, we can see various options for creating different types of projects in Visual Studio. Click the Windows option on the left-hand side.
2. When we click the Windows options in the previous step, we will be able to see an option for Windows Forms Application. Click this option.
3. We then give a name for the application which in our case is “DemoApplication”. We also need to provide a location to store our application.
4. Finally, we click the ‘OK’ button to let Visual Studio to create our project.

**Step 3)** Now add a button from the toolbox to the Windows form. Put the text property of the Button as Connect. This is how it will look like



**Step 4)** now double click the form so that an event handler is added to the code for the button click event. In the event handler, add the below code.



**CODE:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string connetionString;

SqlConnection cnn;

connetionString = @"Data Source=WIN-50GP30FGO75;Initial Catalog=Demodb;User ID=sa;Password=demol23";

cnn = new SqlConnection(connetionString);

cnn.Open();

if (cnn != null)

{

Label1.Text = "OPEN";

Label1.Visible = true;

}

else

{

Label1.Text = "Error in Connection";

Label1.Visible = true;

} cnn.Close();

}

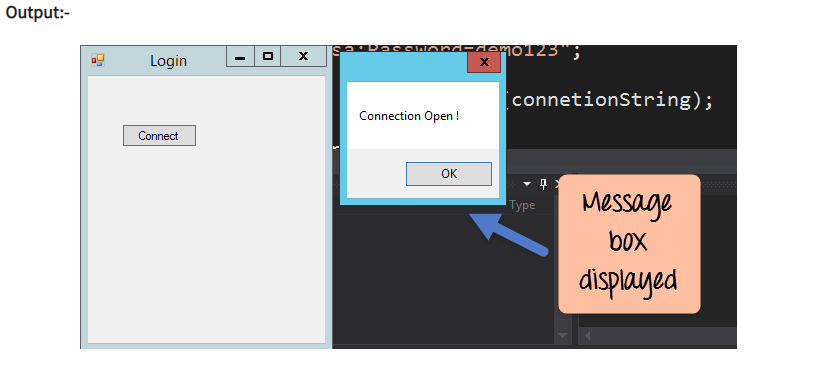
}

}

**Code Explanation:-**

* 1. The first step is to create variables, which will be used to create the connection string and the connection to the SQL Server database.
  2. The next step is to create the connection string. The connecting string needs to be specified correctly for C# to understand the connection string. The connection string consists of the following parts
     1. Data Source – This is the name of the server on which the database resides. In our case, it resides on a machine called WIN- 50GP30FGO75.
     2. The Initial Catalog is used to specify the name of the database
     3. The UserID and Password are the credentials required to connect to the database.
  3. Next, we assign the connecting string to the variable cnn. The variable cnn, which is of type SqlConnection is used to establish the connection to the database.
  4. Next, we use the Open method of the cnn variable to open a connection to the database. We then just display a message to the user that the connection is established.
  5. Once the operation is completed successfully, we then close the connection to the database. It is always a good practice to close the connection to the database if nothing else is required to be done on the database.

When the above code is set, and the project is executed using Visual Studio, you will get the below output. Once the form is displayed, click the Connect button.



When you click on “connect” button, from the output, you can see that the database connection was established. Hence, the message box was displayed.

**Access data with the SqlDataReader**

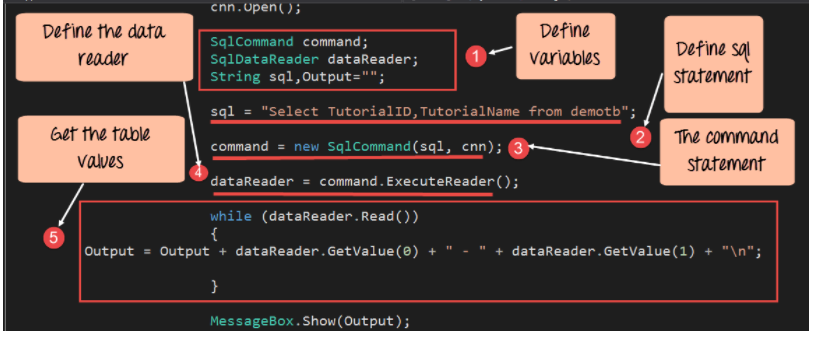
To showcase how data can be accessed using C#, let us assume that we have the following artifacts in our database.

* 1. A table called demotb. This table will be used to store the ID and names of various Tutorials.
  2. The table will have 2 columns, one called “TutorialID” and the other called “TutorialName.”
  3. For the moment, the table will have 2 rows as shown below.

|  |  |
| --- | --- |
| TutorialID | TutorialName |
| 1 | C# |
| 2 | Asp.Net |

Let’s change the code in our form, so that we can query for this data and display the information via a Messagebox. Note that all the code entered below is a continuation of the code written for the data connection in the previous section.

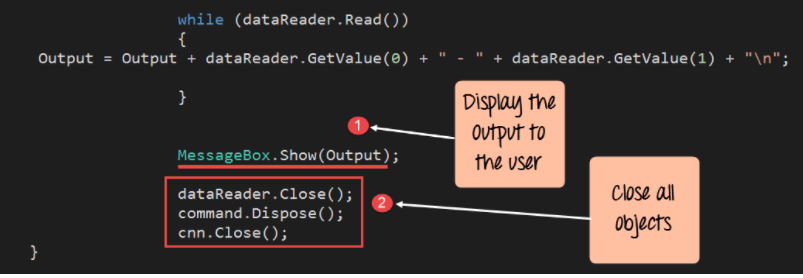
**Step 1)** Let’s split the code into 2 parts so that it will be easy to understand for the user.

* + The first will be to construct our “select” statement, which will be used to read the data from the database.
  + We will then execute the “select” statement against the database and fetch all the table rows accordingly.

**Code Explanation:-**

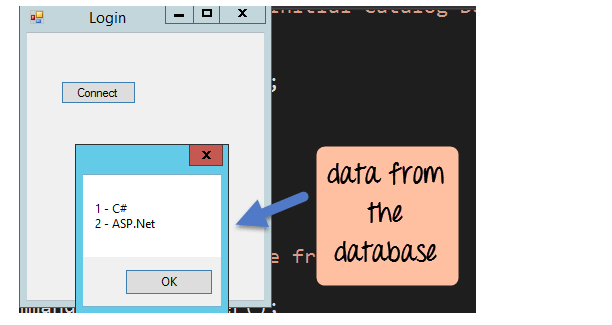
* 1. The first step is to create the following variables
     1. SQLCommand – The ‘SQLCommand’ is a class defined within C#. This class is used to perform operations of reading and writing into the database. Hence, the first step is to make sure that we create a variable type of this class. This variable will then be used in subsequent steps of reading data from our database.
     2. The DataReader object is used to get all the data specified by the SQL query. We can then read all the table rows one by one using the data reader.
     3. We then define 2 string variables, one is “SQL” to hold our SQL command string. The next is the “Output” which will contain all the table values.
  2. The next step is to define the SQL statement, which will be used against our database. In our case, it is “Select TutorialID, TutorialName from demotb”. This will fetch all the rows from the table demotb.
  3. Next, we create the command object which is used to execute the SQL statement against the database. In the SQL command, you have to pass the connection object and the SQL string.
  4. Next, we will execute the data reader command, which will fetch all the rows from the demotb table.
  5. Now that we have all the rows of the table with us, we need a mechanism to access the row one by one. For this, we will use the while statement. The while statement will be used to access the rows from the data reader one at a time. We then use the GetValue method to get the value of TutorialID and TutorialName.

**Step 2)**In the final step, we will just display the output to the user and close all the objects related to the database operation.



**Code Explanation:-**

* 1. We will continue our code by displaying the value of the Output variable using the MessageBox. The Output variable will contain all the values from the demotb table.
  2. We finally close all the objects related to our database operation. Remember this is always a good practice.

When the above code is set, and the project is run using Visual Studio, you will get the below output. Once the form is displayed, click the Connect button.