Unit 3 Database Access

3.1. Introduction about ADO.NET

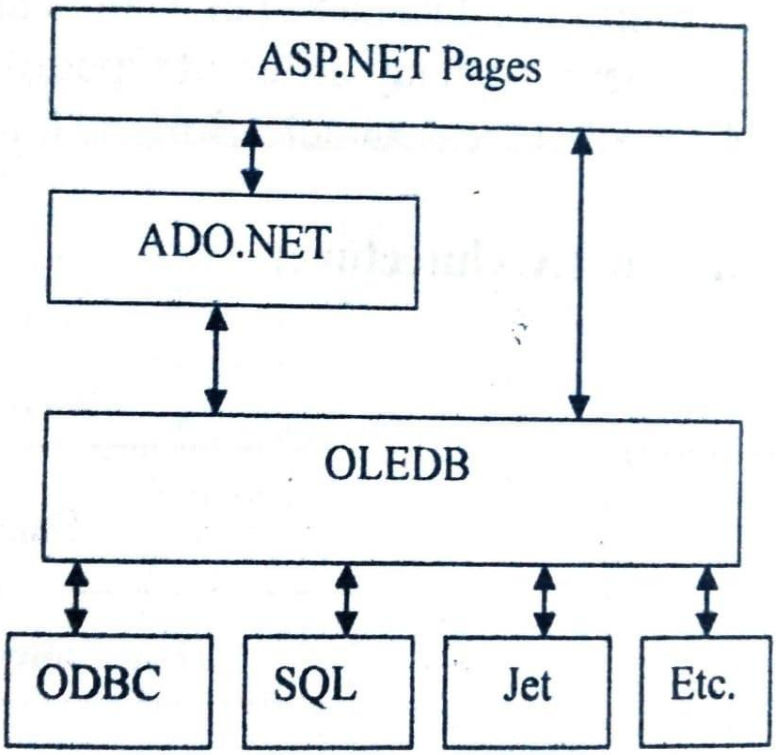
ADO.NET is the major version of ADO (ActiveX Data Objects) that enables ASP.NET pages to present data in much more efficient and different ways.

For example, it fully makes use of XML and easily able to communicate with any XML – complaint application.

In terms of ASP.NET development, ADO.NET provides the framework for accessing any type of data that can be used with ASP.NET pages.

This allows users to view or change information stored in any kind of database, text files, and XML data dynamic application development.

It shares Common Type System, design pattern and naming converstions.



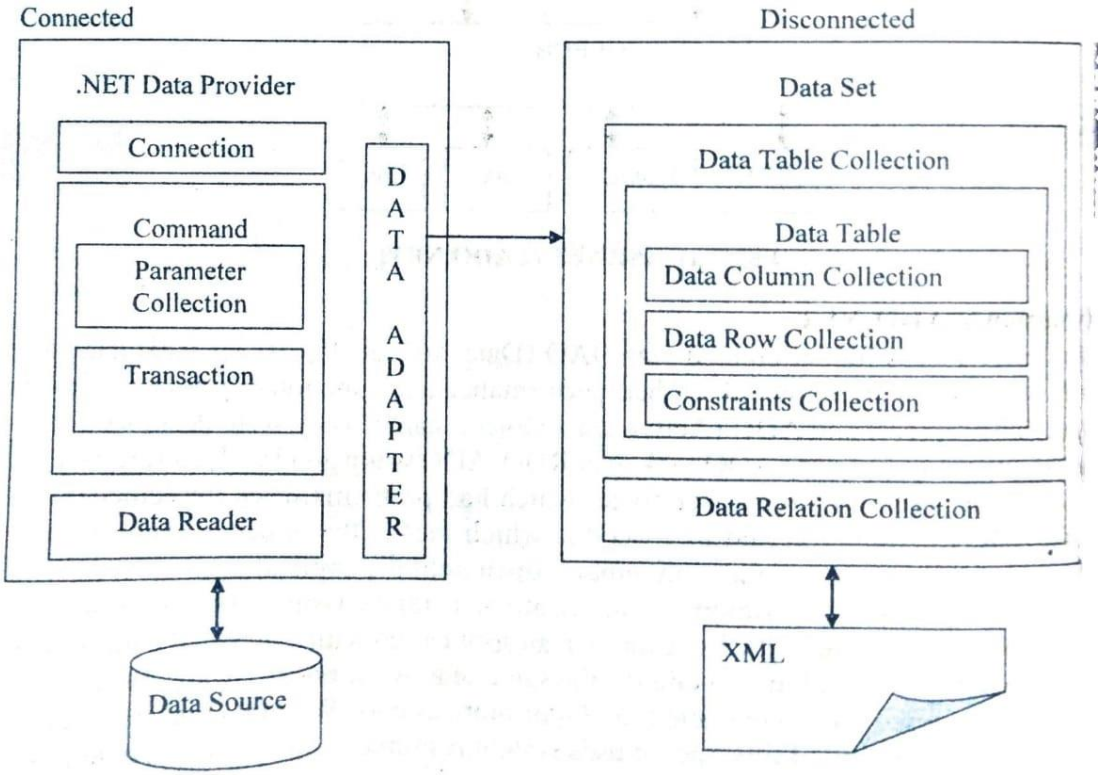
3.2. Introduction about Provider, Adapter, Reader, CommandBuilder

ADO.NET is a set of libraries that allows you to interact with data source.

Commonly, the data source is a data base, but it could also be a text file, an Excel spread sheet, or an XML file.

There are many different types of databases available such as Microsoft SQL Server, Microsoft access, Oracle, IBM DB2 etc.

Connected & Disconnected Data (Architecture)



[Fig (7.b): Connected Disconnected Architecture of ADO.NET]

The data access with ADO.NET consists of two parts:

1. Data Provider
2. DataSet

ADO.NET allows us to interact with different types of data sources and different types of data bases.

There are many set of classes that allows you to access different types of databases.

Since different data sources are interpreted with different protocols, we need a way to communicate with the right data source using the right protocol.

Some older data sources use the ODBC (Open DataBase Connectivity) protocol, many new data sources use the OleDb (Object Linking and Embedding DataBase) protocol, and there are many other data sources that allow you to communicate with them directly through .NET ADO.NET class library.

1. Data Provider

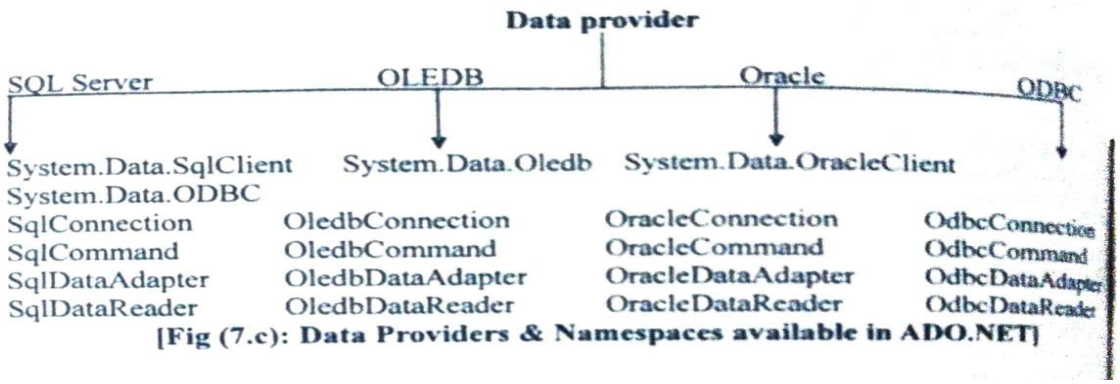
The Data Provider is used for providing and maintaining the connection to the database. It is a set of related components that work together to provide data in an efficient manner.

Provider Name	API prefix	Data Source Description
ODBC Data Provider	Odbc	Data Sources with an ODBC interface. Normally older data bases.
OleDb Data Provider	OleDb	Data Sources that expose an OleDb interface, i.e. Access or Excel.
Oracle Data Provider	Oracle	For Oracle Data Bases.
SQL Data Provider	Sql	For interacting with Microsoft SQL Server.
Borland Data Provider	Bdp	Generic access to many data bases such as Interbase, SQL Server, IBM DB2, and Oracle.

[Table:1 List of Various Data Providers]

Each DataProviders a consists of the following classes:

The connection object which provides a connection to the database. The Command object which is used to execute a command. The DataReader object which provides a forward-only, read only, connected recordset . The DataAdapter object which populates a disconnected DataSet with data and performs update.



[Fig (7.c): Data Providers & Namespaces available in ADO.NET]

In fig (7.c), System.Data.SqlClient, System.Data.Oledb, System.Data.OracleClient and System.Data.ODBC are the namespces.

ADO.NET Objects
ADO.NET consists of many objects that we use to work with data.

a) Connection Object
b) Command Object
c) Parameter Object
d) DataReader Object
e) DataAdapter Object

Connection Object
To establish connection with a database, you must have an object. The Connection helps to identify the database server, the databse name, user name, password and other parameters that are required for connecting to the database.

Connection String – A string that specifies information about a data source and the means of connecting to it is called Connection String.

To create SqlConnection Object:

```
Dim conn As New SqlConnection(Data Source =DatabaseSerever;Intial Catalog=DBName;User ID;Password=”Your Password”)
```

Command Object
It is used to retrieve a subset of data. Also invoking SQL insert, update, delete statement directly require you to set certain parameters on the command before executing the statement.

Properties

Properties	Description
Connection	To set a connection object.
CommandText	It specifies the SQL string or stored procedure to be executed.
CommandType	It is used to determine how to interpret command text. i.e. CommandText is StoredProcedure or Text or DirectTable.
Transaction	It is used to set the Transaction Object that is to be used for this command.
Paremeters collection	It is used to specify placeholders instead of direct values.
CommandTimeOut	Gets the time to wait while trying to execute the command before terminating the attempt and generating an error.

[Table:4 Properties of Command Object]

Methods

Methods	Description
ExecuteNonQuery	It will execute the SQL statement and returns the number of rows affected by the query.
ExecuteScalar	It will execute the SQL statement which return the singleton value.
ExecuteReader	It will execute the SQL statement and returns the records in the form of DataReader. i.e. it is used to create the Object of DataReader.
CreateParameter	It creates and returns a SqlParameter object associated with the SqlCommand.
Cancel	It is used to cancel the command given for execution.
Prepare	It is used to compile the statement before the execution.
ResetCommandTimeOut	It is used to reset Command time out property to its default value

[Table:5 Methods of Command Object]

Parameter Object

When we are working with data, you may also need to filter results based on some criteria. Typically, this is done by accepting input from a user and using that input in SQL query.

For example, an Admin want to see the data of employees working in specific department. Another query might be to filter Employees by City. So if you want to filter records, you need to build the string dynamically. But this is not the good programming approach.

Dim cmd As New SqlCommand("Select * from tblEmp where City = '"+varCity+"'",conn)

Here the input variable, varCity is retrived from a TextBox control on the Web Page. Anything placed into that TextBox control will be put imto varCity & added to your SQL string. This situation invites a hacker to replace that string with something malicious.
Instead of dynamically building a string, use parameter, which makes your application much more secure.

You have to use @ symbol as a prefix for the parameter name as shown below:

Dim cmd As New SqlCommand("Select * from tblEmp where City=@City",conn)

DataReader

A SqlDataReader is used to read data in the most efficient manner. You cannot use it for writing data. You can forward – only and in sequential manner from SqlDataReader.

Once you have read some data, you must saveit because you will not be able to go back and read it again. A single row of data is in the memory at a time. When the SqlDataReader object is created the pointer is placed before the first row.

Properties

Properties	Description
FieldCount	It stores number of fields in a row.
HasRows	It specifies that the rows are selected or not for reading.
IsClosed	It specifies that DataReader is closed or not.
RecordsAffected	It returns -1 as DataReader is created on server.
Item	It gets the value of the specified column name.

[Table:6 Properties of DataReader Object]

DataAdapter

It acts as a bridge between data source and in-memory data objects such as the DataSet.

Properties

Properties	Description
SelectCommand	It is used to hold a Command that retrieves data from the data source.
UpdateCommand	It is used to hold a Command that updates data to the data source.
DeleteCommand	It is used to hold a Command that delete data from the data source.
InsertCommand	It is used to hold a Command that insert data into the data source.
CommandType	It indicates CommandText property contains SQL statement or stored procedure. If CommandText property contains stored procedure than set the value to CommandType.StoredProcedure. Default value is CommandType.Text for SQL statement.

[Table:8 Properties of DataAdapter Object]

Methods

Methods	Description
Fill	It is used to populate a dataset object with the data that the DataAdapter object retrieve from the data store using its SelectCommand. But before that we must initialize a Dataset object.
Update	It is used to update the database according to the changes that are made in the DataSet.

[Table:9 Methods of DataAdapter Object]

- ✓ **CommandBuilder**

When we use Update() method to update the database, the changes remains only in the DataSet, it is not reflected in the Database. To implement changes in the database, you need to use CommandBuilder.

SqlCommandbuilder class of System.Data.SqlClient namespace automatically generates single – table commnds that are used to update the changes made to the DataSet with the associated SQL server database.

You need to add following code:

```
Dim cb As New SqlCommandBuilder(da)
Da.Update(ds,”Emp”)
```

2. DataSet

The dataset is a disconnected, in-memory representation of data. It can be considered s a local copy of the portions of the database. The DataSet is persisted in memory and it can be manipulated and updated independent of the databse.

The DataSet class resides in the System.Data namespace.

The DataSet object contains a collection of tables, relationships and constraints that are consistent with the data read from the data store.

Example:

```
Dim conn As New SqlConnection(Data
Source=.\SQLEXPRESS;AttachDbFilename="C:\Users\Hp\Documents\Visual
Studio 2010\WebSites\dbtest\App_Data\Database.mdf";Integrated
Security=True;User Instance=True)

Dim da As New SqlDataAdpater
da.SelectCommand.Connection=conn
da.SelectCommand.CommandText=”Select * from tblEmp”
Dim ds As New DataSet
da.Fill(ds,”Emp”)
```

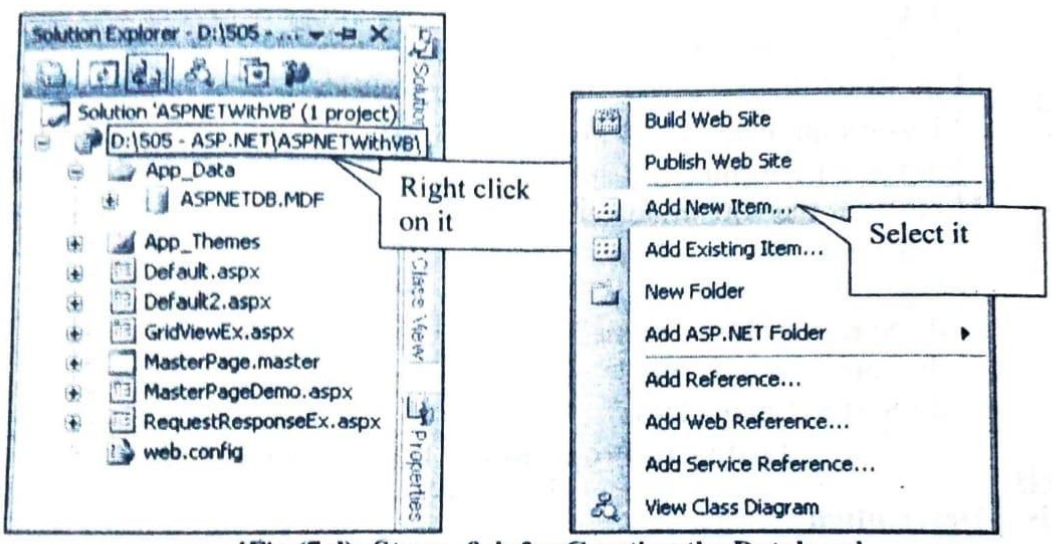
DataRelation

It is the object used to maintain the relationship within the DataSet.

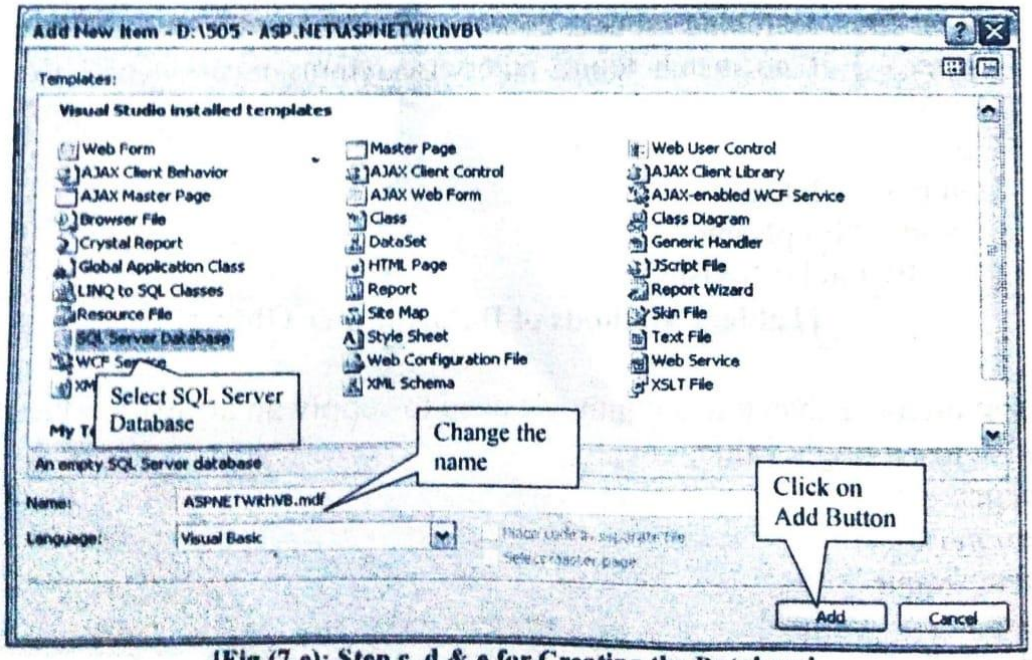
3.3. Database Access using ADO.NET

Creating Database in Visual Studio .NET

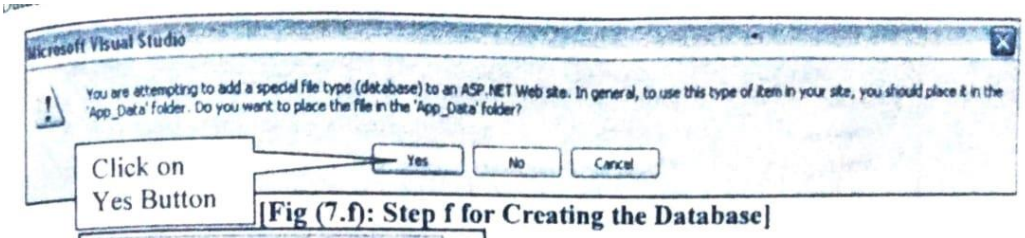
a) Right click on the WebSite in Soluation Explorer
b) Select Add New Item from the List
c) Select SQL Server Database from the Templates
d) Change the Name of the Database as you need
e) Click on Add Button
f) It will ask to add the Database in App_Data Folder click on YES Button.



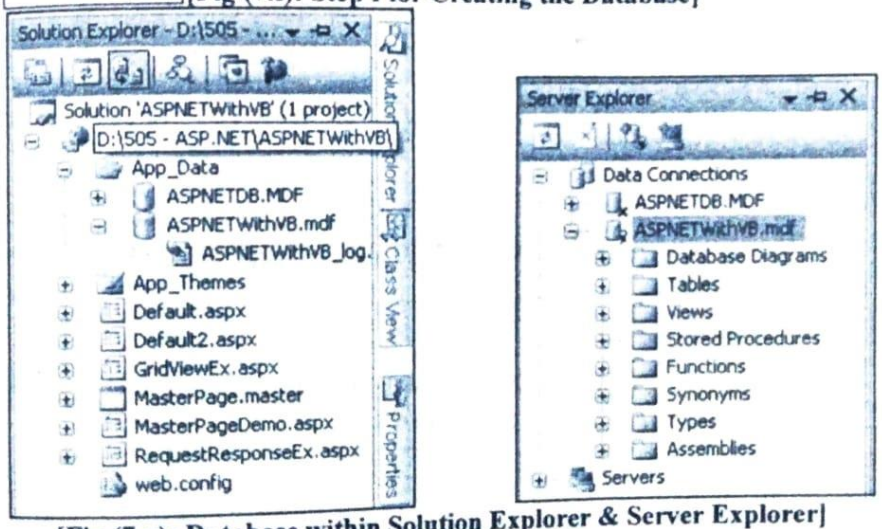
[Fig (7.d): Step a & b for Creating the Database]



[Fig (7.e): Step c, d & e for Creating the Database]



[Fig (7.f): Step f for Creating the Database]



[Fig (7.g): Database within Solution Explorer & Server Explorer]

Creating tables within the Database

a) Right click on the Tables
b) Select Add New Table, it will open the Table Creation Wizard
c) Add Columns and Set the constraints for it.
d) Save the Table with the appropriate name.

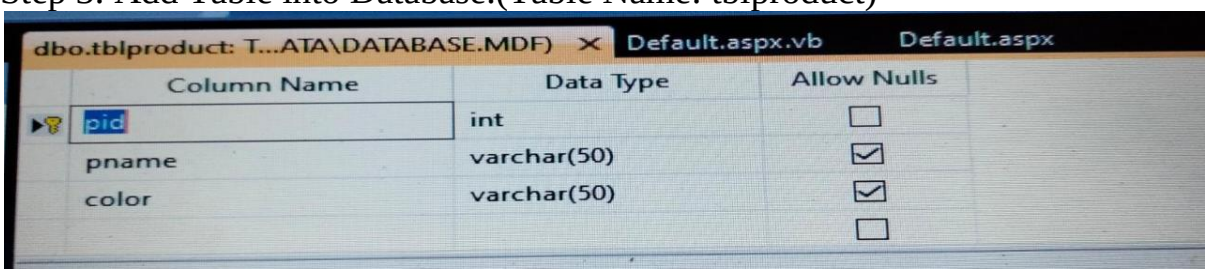
Example: Perform insert, update, delete and select operation.

Step 1: Create One Website

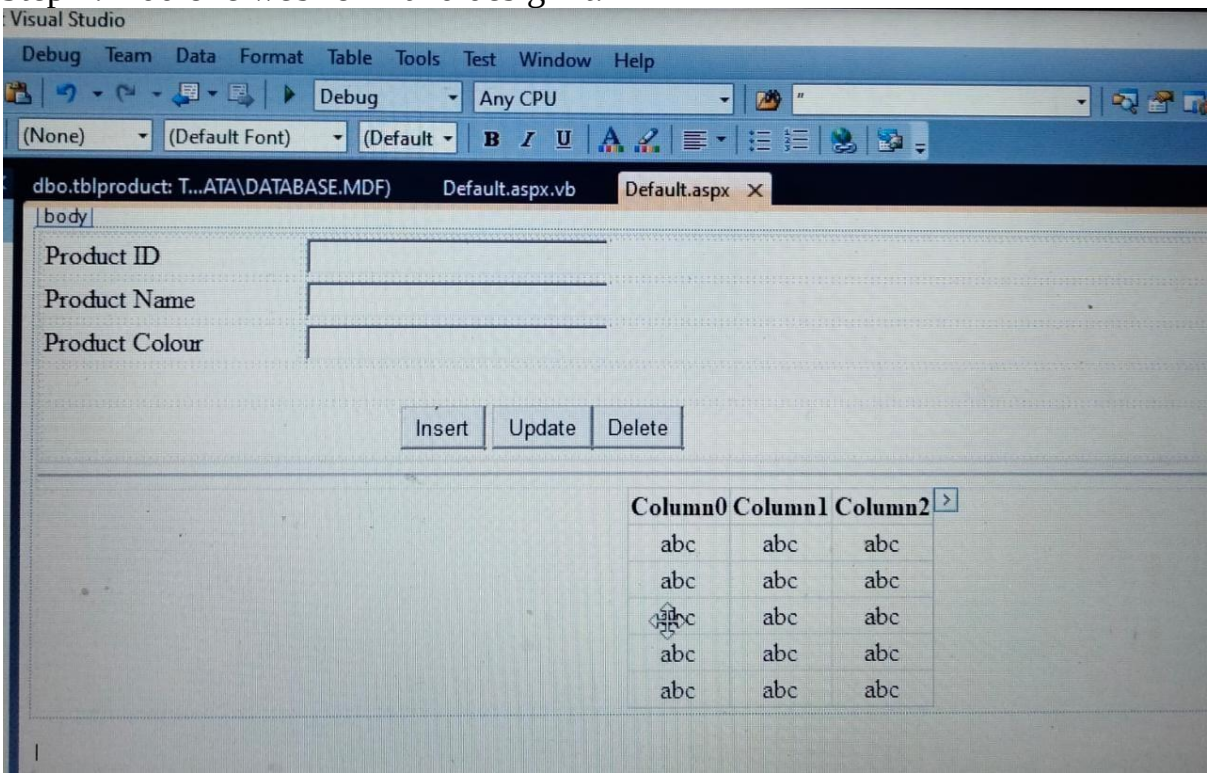
Step 2: Add Database in your Website.

a) Right click on the WebSite in Soluation Explorer
b) Select Add New Item from the List
c) Select SQL Server Database from the Templates
d) Change the Name of the Database as you need
e) Click on Add Button
f) It will ask to add the Database in App_Data Folder click on YES Button.

Step 3: Add Table into Database.(Table Name: tblproduct)



Step 4: Add one web form and design it.



Here, Source code gives proper name.

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="Default.aspx.vb" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        .style1
        {
            width: 100%;
        }
        .style2
        {
            width: 120px;
        }
    </style>
</head>
<body>
```

```
<form id="form1" runat="server">
<div>

    <table class="style1">
        <tr>
            <td class="style2">
                Product ID</td>
            <td>
                <asp:TextBox ID="txtPid" runat="server"
Width="183px"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td class="style2">
                Product Name</td>
            <td>
                <asp:TextBox ID="txtPname" runat="server"
Width="183px"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td class="style2">
                Product Colour</td>
            <td>
                <asp:TextBox ID="txtPColour" runat="server"
Width="183px"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td colspan="2" style="text-align: left">
                 </td>
        </tr>
        <tr>
            <td colspan="2" style="text-align: left">

           
;          &nbs
p;          &nb
sp;          &n
bsp;       
                <asp:Button ID="btnInsert" runat="server" Text="Insert"
Width="51px"
                    style="margin-left: 0px" />
```

```
                 <asp:Button ID="btnUpdate" runat="server" Text="Update"
/>
                <asp:Button ID="btnDelete" runat="server" Text="Delete" />
            </td>
        </tr>
    </table>
  <div align="center">
  <hr />
    <asp:GridView ID="GridView1" runat="server">
    </asp:GridView>
  </div>
  </div>
  </form>
</body>
</html>
```

.

Form Design Complete
Write a connection code and perform insert, update, delete and select using coding

```
Imports System.Data.SqlClient
Imports System.Data

Partial Class _Default
  Inherits System.Web.UI.Page
  Dim conn As New SqlConnection("Data
Source=.\SQLEXPRESS;AttachDbFilename=c:\Users\Hp\documents\visual
studio
2010\WebSites\curd_withcode_demo\App_Data\Database.mdf;Integrated
Security=True;User Instance=True")
  Protected Sub btnInsert_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnInsert.Click



      Dim proid As Integer = txtPid.Text
      Dim proname As String = txtPname.Text
      Dim procolour As String = txtPColour.Text
      conn.Open()
      Dim command As New SqlCommand("insert into tblproduct values ('" &
proid & "','" & proname & "','" & procolour & "')", conn)
      command.ExecuteNonQuery()
```

```
        MsgBox("Sucessfully Inserted", MsgBoxStyle.Information, "Message")
        conn.Close()
        ListProduct()
    End Sub

    Private Sub ListProduct()
        Dim command As New SqlCommand("select * from tblproduct", conn)
        Dim sd As New SqlDataAdapter(command)
        Dim dt As New DataTable
        sd.Fill(dt)
        GridView1.DataSource = dt
        GridView1.DataBind()
    End Sub



    Protected Sub btnUpdate_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnUpdate.Click

        Dim proid As Integer = txtPid.Text
        Dim proname As String = txtPname.Text
        Dim procolour As String = txtPColour.Text
        conn.Open()
        Dim command As New SqlCommand("update tblproduct set pname='" &
proname & "',color='" & procolour & "' where pid= '" & proid & "'", conn)
        command.ExecuteNonQuery()
        MsgBox("Sucessfully Updated", MsgBoxStyle.Information, "Message")
        conn.Close()
        ListProduct()
    End Sub

    Protected Sub btnDelete_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnDelete.Click
        Dim proid As Integer = txtPid.Text
        Dim proname As String = txtPname.Text
        Dim procolour As String = txtPColour.Text
        conn.Open()
        Dim command As New SqlCommand("delete tblproduct where pid= '" &
proid & "'", conn)
        command.ExecuteNonQuery()
        MsgBox("Sucessfully Deleted", MsgBoxStyle.Information, "Message")
        conn.Close()
        ListProduct()
```

```
    End Sub

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        ListProduct()
    End Sub
End Class
```