**Business Objects as Model in ASP.NET MVC**

The entities are mapped to the database tables and object-relational mapping (ORM) frameworks like Entity Framework, NHibernate, etc. are used to retrieve and save the data into a database. The business object contains both state (data) and behavior that is logic specific to the business.  Let us understand how to use Business Objects as Model in ASP.NET MVC Application.

##### **Create an Empty ASP.NET MVC Application:**

First, create an Empty MVC Application with the name **MVC\_DEMO.**Once you create the application, then create a controller with the name as **HomeController** within the Controllers Folder and then copy and paste the below code in it.

**namespace** *MVC\_DEMO.Controllers*

**{**

**public** **class** HomeController : Controller

**{**

**public** ViewResult Index**()**

**{**

ViewData**[**"Countries"**]** = new List**<**string**>()**

**{**

"India",

"US",

"Canada",

"Brazil"

**}**;

**return** View**()**;

**}**

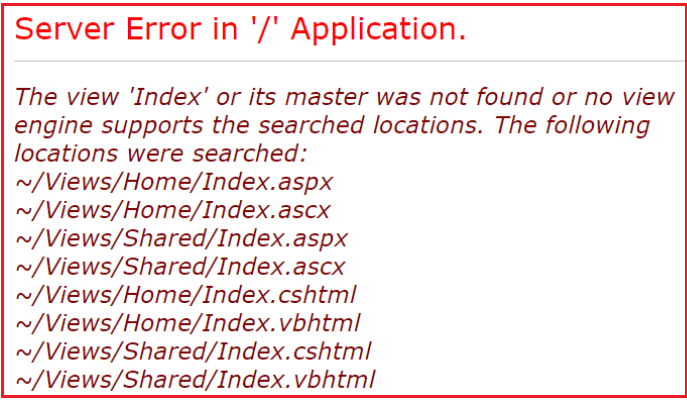
**}**

**}**

The following URL will invoke the **Index()** action method of the **HomeController**. Notice that, the HomeController class is inherited from the base **Controller** class which in turn inherits from **ControllerBase** class. ControllerBase, in turn, inherits from the **IController** interface.

**http://localhost:53657/Home/Index**

The**return View()** statement within the index action method by default looks for a view with the name “**Index**” in “**/Views/Home/**” and “**/Views/Shared/**” folders. If a view with the name “**Index**” is not found then we will get the following error.



So, In the ASP.NET MVC application, there are several conventions that we need to follow while working. For example, controllers need to have the word controller in them and should implement the IController interface either directly or indirectly. Views should be placed in a specific location that MVC can find them.

But with models, there are no strict rules. In fact, the “**Models**” folder is optional and they can place anywhere within the application. They can even be present in a separate project. Let’s now turn our attention to using business objects as the model. We will be using the table “**Employee**” for this demo.

##### **Step1: Create the Required Database**

Please use the below SQL script to create and populate the Employee table with some test data. Also, we are creating one stored procedure to retrieve the employee data.

-- Create Employee Table

**Create** **table** Employee

(

Id int Primary Key Identity(1,1),

Name nvarchar(50),

Gender nvarchar(10),

City nvarchar(50),

Salary decimal(18,2),

DateOfBirth DateTime

)

**GO**

-- Insert some test data into Employee table

**Insert** **into** Employee values('Pranaya','Male','Mumbai',4000,'02/03/1977')

**Insert** **into** Employee values('Anurag','Male','Hyderabad',5000,'04/06/1979')

**Insert** **into** Employee values('Priyanka','Female','Bangalore',1000,'01/05/1979')

**Insert** **into** Employee values('Subrat','Male','Hyderabad',2000,'03/07/1981')

**Insert** **into** Employee values('Sudhanshu','Male','Mumbai',3000,'02/04/1978')

**Insert** **into** Employee values('Preety','Female','Bangalore',4000,'02/03/1974')

**Insert** **into** Employee values('Sandeep','Male','Hyderabad',5000,'04/06/1972')

**Insert** **into** Employee values('Sambit','Male','Bangalore',6000,'07/05/1975')

**Insert** **into** Employee values('Hina','Female','Mumbai',3000,'09/08/1976')

**GO**

--Stored procedure to retrieve data

**Create** procedure spGetAllEmployees

**as**

Begin

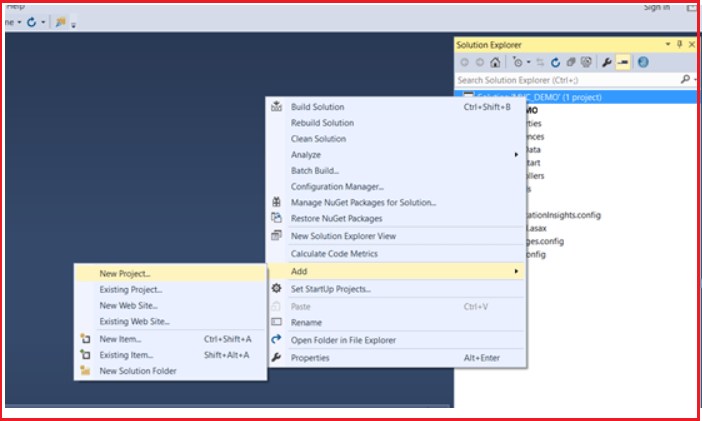
**Select** Id, Name, Gender, City, Salary, DateOfBirth

**from** Employee

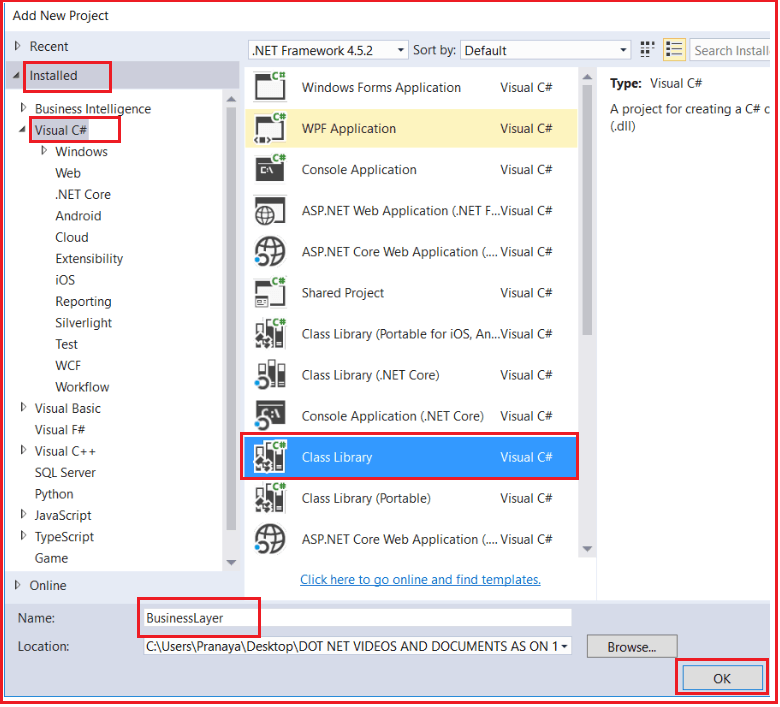
End

**GO**

**Step2: Add a Class Library project with Name=”BusinessLayer” to the Solution**  
Right-click on the Solution Folder => Add => New Project as shown in the below image.



From the new project window, select **Visual C#** from Installed Template from the left pane and then select **Class Library** Template from the middle pane. Provide the name as **BusinessLayer** and click on the **OK** as shown in the below image.



Now it will add the BusinessLayer class library project to our existing solution.

##### **Step3: Adding Models to the Class Library Project**

Right-click on the business layer class library project and add a class file with the name **Employee.cs**. Once you created the **Employee** class then copy and paste the following code into it. The following class is very straightforward. We simply created the class with 6 properties.

**namespace** *BusinessLayer*

**{**

**public** **class** Employee

**{**

**public** **int** ID **{** **get**; **set**; **}**

**public** string Name **{** **get**; **set**; **}**

**public** string Gender **{** **get**; **set**; **}**

**public** string City **{** **get**; **set**; **}**

**public** **decimal** Salary **{** **get**; **set**; **}**

**public** DateTime DateOfBirth **{** **get**; **set**; **}**

**}**

**}**

##### **Step4: Adding Required References**

Right-click on the “**References**” folder of the business layer class library project and add a reference to the “**System.Configuration**” assembly. This is required as we want to read the connection string from the web config file using the ConfigurationManager class and this class belongs to **System.Configuration namespace.**

##### **Step5: Adding EmployeeBusinessLayer class**

Right-click on the business layer class library project and add a class file with the name **EmployeeBusinessLayer.cs**. Once you created the **EmployeeBusinessLayer** class then copy and paste the following code into it. In the following class, we define one method i..e. GetAllEmployess(). This method is used to get the employee details from the database. The following code is self-explained, so please go through the comment lines.

**using** *System;*

**using** *System.Collections.Generic;*

**using** *System.Configuration;*

**using** *System.Data;*

**using** *System.Data.SqlClient;*

**namespace** *BusinessLayer*

**{**

**public** **class** EmployeeBusinessLayer

**{**

**public** List**<**Employee**>** GetAllEmployess**()**

**{**

//Reads the connection string from web.config file. The connection string name is DBCS

string connectionString = ConfigurationManager.ConnectionStrings**[**"DBCS"**]**.ConnectionString;

//Create List of employees collection object which can store list of employees

List**<**Employee**>** employees = new List**<**Employee**>()**;

//Establish the Connection to the database

**using** **(**SqlConnection con = new SqlConnection**(**connectionString**))**

**{**

//Creating the command object by passing the stored procedure that is used to

//retrieve all the employess from the tblEmployee table and the connection object

//on which the stored procedure is going to execute

SqlCommand cmd = new SqlCommand**(**"spGetAllEmployees", con**)**;

//Specify the command type as stored procedure

cmd.CommandType = CommandType.StoredProcedure;

//Open the connection

con.Open**()**;

//Execute the command and stored the result in Data Reader as the method ExecuteReader

//is going to return a Data Reader result set

SqlDataReader rdr = cmd.ExecuteReader**()**;

//Read each employee from the SQL Data Reader and stored in employee object

**while** **(**rdr.Read**())**

**{**

//Creating the employee object to store employee information

Employee employee = new Employee**()**;

employee.ID = Convert.ToInt32**(**rdr**[**"Id"**])**;

employee.Name = rdr**[**"Name"**]**.ToString**()**;

employee.Gender = rdr**[**"Gender"**]**.ToString**()**;

employee.City = rdr**[**"City"**]**.ToString**()**;

employee.Salary = Convert.ToDecimal**(**rdr**[**"Salary"**])**;

employee.DateOfBirth = Convert.ToDateTime**(**rdr**[**"DateOfBirth"**])**;

//Adding that employee into List of employees collection object

employees.Add**(**employee**)**;

**}**

**}**

//Return the list of employees that is stored in the list collection of employees

**return** employees;

**}**

**}**

**}**

##### **Step6: Adding a Reference to class Library Project in ASP.NET MVC Application:**

Right-click on the “**References**” folder of the “**MVC\_DEMO**” project and add a reference to the “**BusinessLayer**” class library project. Then Include a connection string with name = “**DBCS**” in **Web.Config** file as shown below.

**<connectionStrings>**

**<add** name="DBCS"

connectionString="Data Source=LAPTOP-2HN3PT8T\SQLEXPRESS;Initial Catalog=MVC\_DB;Integrated Security=True"

providerName="System.Data.SqlClient"**/>**

**</connectionStrings>**

##### **Step8: Creating Controller**

Right-click on the “**Controllers**” folder and add a Controller with the name “**EmployeeController**” and then copy and paste the following code into it. In the below controller we have only one action method i.e. Index. This method creates an instance of **EmoloyeeBusinessLayer** class and then calls the **GetAllEmployees** method which will return the list of employees. The list of employees is then handed over to the **Index** view.

**using** *BusinessLayer;*

**using** *System.Collections.Generic;*

**using** *System.Web.Mvc;*

**namespace** *MVC\_DEMO.Controllers*

**{**

**public** **class** EmployeeController : Controller

**{**

**public** ActionResult Index**()**

**{**

EmployeeBusinessLayer employeeBusinessLayer = new EmployeeBusinessLayer**()**;

List**<**Employee**>** employees = employeeBusinessLayer.GetAllEmployess**()**;

**return** View**(**employees**)**;

**}**

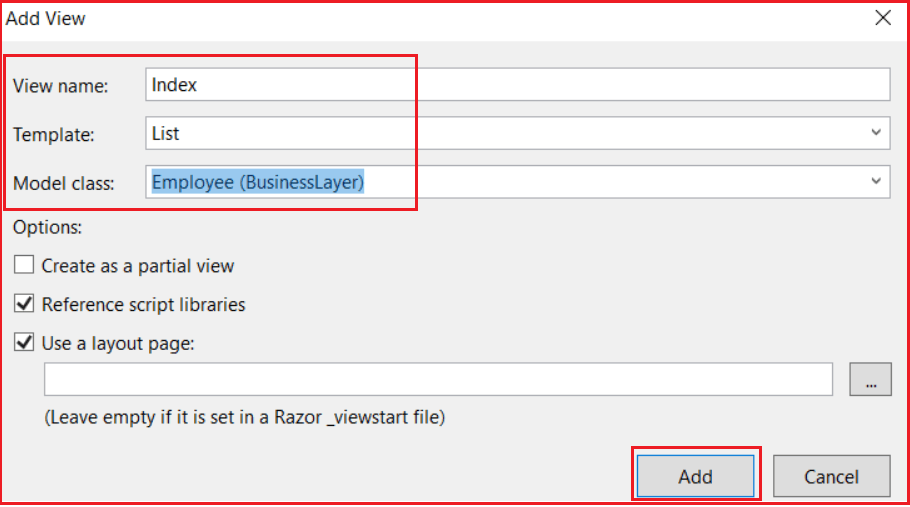
**}**

**}**

##### **Step9: Adding Index View**

Right-click on the **Index()** action method in the “**EmployeeController**” class and then select “**Add View**” from the context menu. Set

**View name = Index**  
**Model class = Employee (BusinessLayer)**  
**Template = List**  
**Click on the “Add” button as shown below**



###### **Change the RouteConfig.cs as shown below**

We are setting the controller as Employee and the default action method as Index.

**namespace** *MVC\_DEMO*

**{**

**public** **class** RouteConfig

**{**

**public** **static** **void** RegisterRoutes**(**RouteCollection routes**)**

**{**

routes.IgnoreRoute**(**"{resource}.axd/{\*pathInfo}"**)**;

routes.MapRoute**(**

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new **{** controller = "Employee", action = "Index", id = UrlParameter.Optional **}**

**)**;

**}**

**}**

**}**

Set MVC\_DEMO as your startup project and run the application, then navigate to **http://localhost:54094/Employee/Index**. It should display the output as expected as shown in the below image.

