**In this video, we will discuss using business objects as model.** Until now, we have been using entity framework and entities. Entities are mapped to database tables, and object relational mapping tools like Entity Framework, nHibernate, etc are used to retrieve and save data. Business objects contain both state(data) and behaviour, that is logic specific to the business.

**In MVC there are several conventions that needs to be followed.** For example, controllers need to have the word controller in them and should implement IController interface either directly or indirectly. Views should be placed in a specific location that MVC can find them.   
  
public class HomeController : Controller  
{  
    public ViewResult Index()  
    {  
        ViewData["Countries"] = new List<string>()  
        {  
            "India",  
            "US",  
            "UK",  
            "Canada"  
        };  
  
        return View();  
    }  
}  
  
**The following URL will invoke Index() action method with in the HomeController.**Notice that our HomeController inherits from base Controller class which inturn inherits from ControllerBase class. ControllerBase inturn inherits from IController class.  
**http://localhost/MVCDemo/Home/Index**  
  
**return View() statement** with in the **HomeController**by default looks for a view with name = "Index" in "/Views/Home/" and "/Views/Shared/" folders. If a view with name = "Index" is not found, then, we get an error stating  
The view 'Index' or its master was not found or no view engine supports the searched locations. The following locations were searched:  
~/Views/Home/Index.aspx  
~/Views/Home/Index.ascx  
~/Views/Shared/Index.aspx  
~/Views/Shared/Index.ascx  
~/Views/Home/Index.cshtml  
~/Views/Home/Index.vbhtml  
~/Views/Shared/Index.cshtml  
~/Views/Shared/Index.vbhtml  
  
**But with models, there are no strict rules.** Infact **"Models"** folder is optional and they can live anywhere. They can even be present in a separate project.  
  
Let's now turn our attention to using business objects as model. We will be using table **"tblEmployee"** for this demo. Use the sql script to create and populate this table.  
  
Create table tblEmployee  
(  
 Id int Primary Key Identity(1,1),  
 Name nvarchar(50),  
 Gender nvarchar(10),  
 City nvarchar(50),  
 DateOfBirth DateTime  
)  
  
Insert into tblEmployee values('Mark','Male','London','01/05/1979')  
Insert into tblEmployee values('John','Male','Chennai','03/07/1981')

Insert into tblEmployee values('Mary','Female','New York','02/04/1978')  
Insert into tblEmployee values('Mike','Male','Sydeny','02/03/1974')  
Insert into tblEmployee values('Scott','Male','London','04/06/1972')  
  
**Stored procedure to retrieve data**  
Create procedure spGetAllEmployees  
as  
Begin  
 Select Id, Name, Gender, City, DateOfBirth  
 from tblEmployee  
End  
  
**Step 1:** Create an ASP.NET MVC 4 Web application with name = **MVCDemo**  
  
**Step 2:** Add a Class Library project with Name=**"BusinessLayer"**  
  
**Step 3:** Right click on the **BusinessLayer**class library project, and add a class file with name = **Employee.cs.**  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
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 DateTime DateOfBirth { get; set; }  
    }  
}  
  
**Step 4:** Right click on the **"References"** folder of the **"BusinessLayer"** class library project, and add a reference to **"System.Configuration"** assembly.  
  
**Step 5:** Right click on the **BusinessLayer** class library project, and add a class file with name = **EmployeeBusinessLayer.cs**.  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Data;  
using System.Data.SqlClient;  
using System.Configuration;  
  
namespace BusinessLayer  
{  
    public class EmployeeBusinessLayer  
    {  
        public IEnumerable<Employee> Employees  
        {  
            get  
            {  
                string connectionString =  
                    ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;  
  
                List<Employee> employees = new List<Employee>();  
  
                using (SqlConnection con = new SqlConnection(connectionString))  
                {  
                    SqlCommand cmd = new SqlCommand("spGetAllEmployees", con);  
                    cmd.CommandType = CommandType.StoredProcedure;  
                    con.Open();  
                    SqlDataReader rdr = cmd.ExecuteReader();  
                    while (rdr.Read())  
                    {  
                        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.DateOfBirth = Convert.ToDateTime(rdr["DateOfBirth"]);  
  
                        employees.Add(employee);  
                    }  
                }  
  
                return employees;  
            }  
        }  
    }  
}  
  
**Step 6:** Right click on the **"References"** folder of the **"MVCDemo"** project, and add a reference to **"BusinessLayer"** project.  
  
**Step 7:** Include a connection string with **name = "DBCS"** in Web.Config file  
  <add name="DBCS"   
        connectionString="server=.; database=Sample; integrated security=SSPI"  
        providerName="System.Data.SqlClient"/>  
  
**Step 8:** Right click on the **"Controllers"** folder and add Controller with name = **"EmployeeController.cs".**  
public class EmployeeController : Controller  
{  
    public ActionResult Index()  
    {  
        EmployeeBusinessLayer employeeBusinessLayer =  
            new EmployeeBusinessLayer();  
  
        List<Employee> employees = employeeBusinessLayer.Employees.ToList();  
        return View(employees);  
    }  
}  
  
**Step 9:** Right click on the **Index**() action method in the **"EmployeeController"** class and select **"Add View"** from the context menu. Set  
View name = **Index**  
View engine = **Razor**  
Select **"Create a strongly-typed view"** checkbox  
Scaffold Template = **List**  
Click **"Add"** button  
  
Run the application and navigate to http://localhost/MVCDemo/Employee/Index. The output should be as shown below.   
