

ASP.NET MVC 5 Recipes: A Practical Solution

This free book is provided by courtesy of [C# Corner](#) and Mindcracker Network and its authors. Feel free to share this book with your friends and co-workers. Please do not reproduce, republish, edit or copy this book.

This book is the second edition of my previous [Programming ASP.NET MVC 5](#) ebook. It includes basic and advance Receipie of ASP.NET MVC 5. This book is basically for programmers and developers who want to learn complete tutorial of new contents with example of ASP.NET MVC 5.

Nimit Joshi

Microsoft MVP & C# Corner MVP

Sno	INDEX	Page
1	Start with the New Taste of MVC 5	1-9
2	Social Provider Access in Web Application with MVC 5	10-20
3	Working with MVC 5 in Visual Studio 2012	21-38
4	Working with External Providers in MVC 5	39-44
5	Getting Started with Areas in MVC 5	45- 53
6	Perform CRUD Functionality in ASP.Net MVC 5	54-62
7	Paging, Searching and Sorting in ASP.Net MVC 5	63-73
8	Perform Code First Migration in ASP.Net MVC 5	74-80
9	Performing Data Annotation in ASP.Net MVC 5	81-86
10	Promoting MVC 4 and Web API Project to MVC 5 and Web API 2	87-95
11	Getting Started with Enum Support in MVC 5 View	96-102
12	Working with DropDownList in MVC 5 Using jQuery	103-112
13	Getting Started with Wizard in ASP.Net MVC 5: Part 1	113-132
14	Introducing Mobile Site in MVC 5 and jQuery Mobile	133-137
15	Working with Yahoo External Provider in MVC 5	138-142
16	Working with SSL Certificate Warning in MVC 5 Application	143-150
17	CRUD Operations Using Asynchronous Programming in MVC 5	151-156
18	Stored Procedures in Entity Framework 6 in MVC 5	157-175
19	ASP.Net MVC 5 Using Visual Basic: Getting Started	176-182
20	ASP.Net MVC 5 Using Visual Basic: Adding Controller	183-189
21	New Release Candidate Update for ASP.Net MVC 5, Web API 2	190-193
22	Implement ASP.Net Web API 2 in ASP.Net MVC 5	194-204
23	Model First Approach in ASP.Net MVC 5	205-225
24	Cascading DropDownList in MVC 5 Using Web API 2	226-245
25	Introducing Google Chart in ASP.NET MVC 5	246-267
26	Binding View with Model Using Enterprise Library in ASP.NET MVC 5	268-292
27	Paging, Sorting, And Filtering with Partial View in ASP.NET MVC 5	293-320
28	Benefits of Partial View in MVC 5	321-355

About The Author

Nimit Joshi is a .NET techie, author and software developer. Nimit's background includes Master's in Computer Application and Bachelor's in Computer Application. Nimit is two times Microsoft MVP and C# Corner MVP. Nimit also wrote a book on ASP.NET MVC 5. Currently he is working with ASP.NET MVC 5, Web API 2 technologies and write articles in C# Corner (C-Sharp Corner). Nimit loves to sing, listen classic and old music, play cricket and watch movies.



NIMIT JOSHI
(Software Developer)

Chapter 1: Start with the New Taste of MVC 5

1.1 Introduction

Microsoft released the latest version of MVC, MVC 5. MVC 5 was released with Visual Studio 2013 Preview. In this chapter I am introducing you to creating an ASP.NET Web Application with the latest MVC Project Template with which you will learn the basics of MVC 5. In that context, here I am creating a MVC 5 Web Application in which I introduce you to all the aspects of MVC 5. As you know, MVC stands for Model View Controller and here we also create a new controller for the web application.

Prerequisites

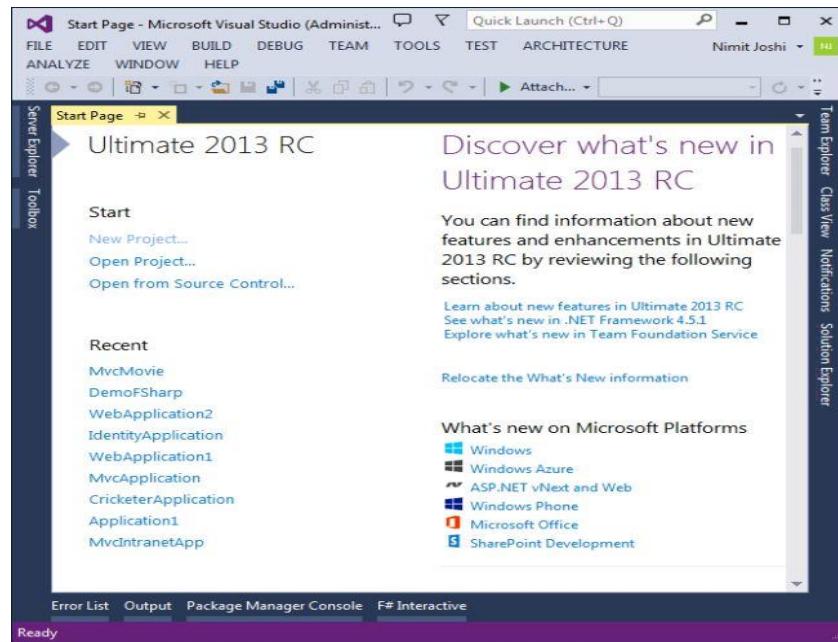
The following are the prerequisites to start the MVC 5 application:

- Visual Studio 2013 Preview or later
- ASP.NET Web Tools for Visual Studio 2013
- NuGet Package Manager for Visual Studio 2013

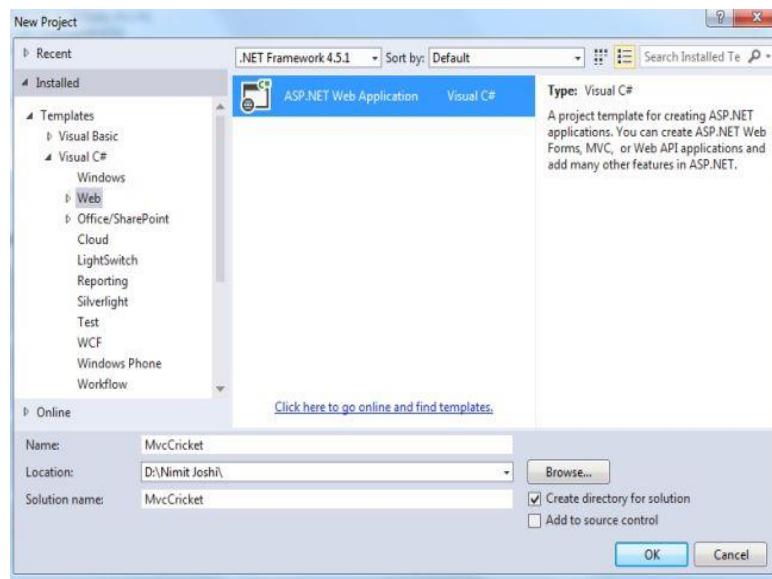
1.2 Create Application in MVC 5

So let's start to create an ASP.NET Web Application with the MVC 5 Project template. Follow the procedure given below.

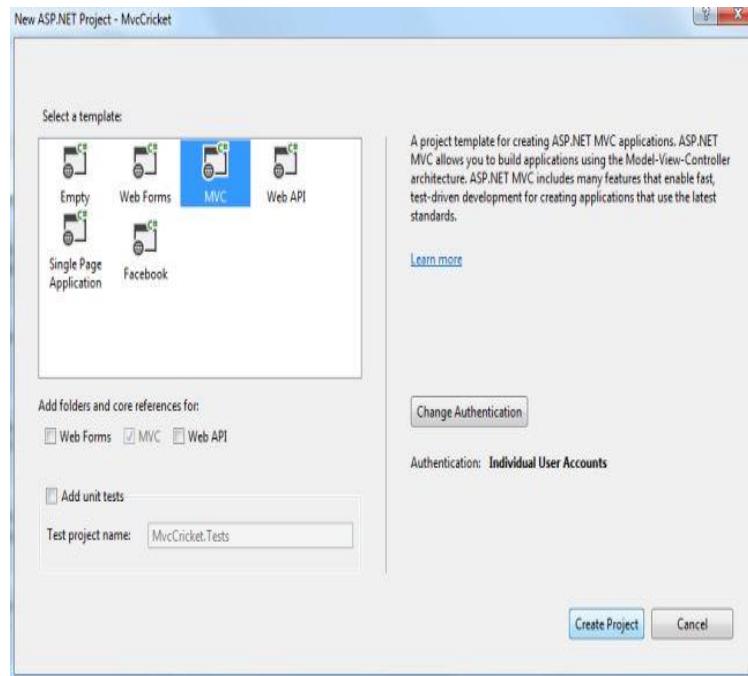
Step 1: Open Visual Studio 2013. (I am using Visual Studio 2013 RC).



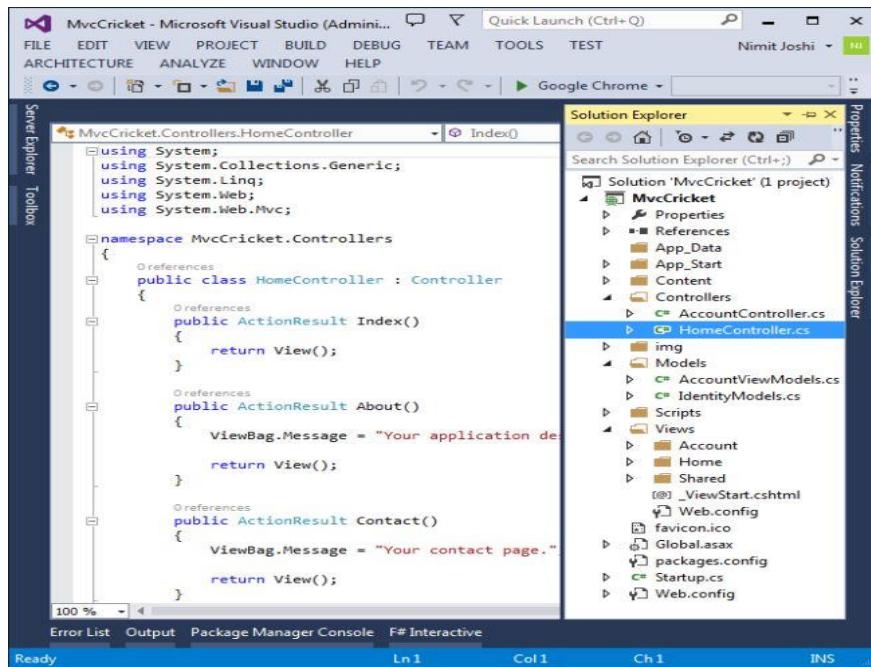
Step 2: Click on "New Project" to create a web application.



Step 3: Select the MVC template.

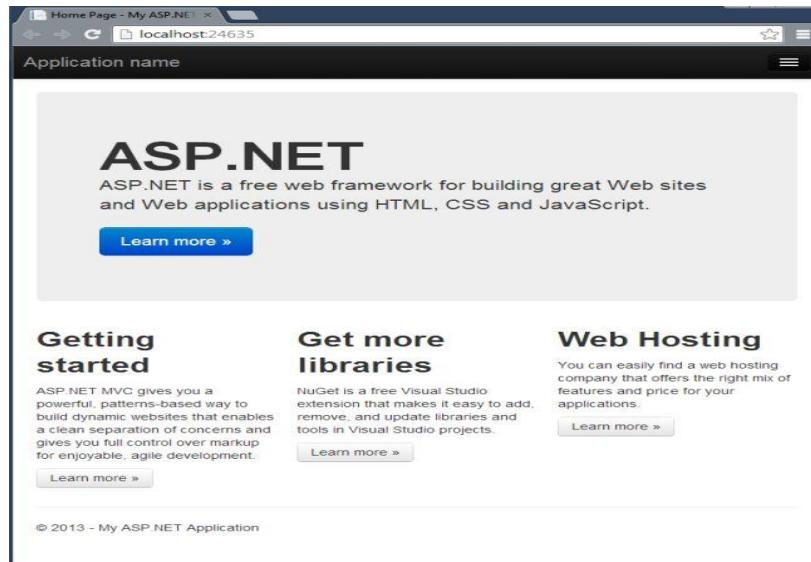
**Step 4:** Click on "Create Project".

When you click on Create Project, Visual Studio automatically creates a MVC web application in which you can see the folders like Models, Controllers and Views. There are two files named Account and Home present in the Controller and View folders. As you can see in the following figure the *HomeController.cs* file has the corresponding view in the Views folder.



1.3 Debug Application

Now, you have created an application, so its time to debug the application. Press F5 to debug your application. Visual Studio allows IIS Express to debug the application. The browser will open and show the application's home page.

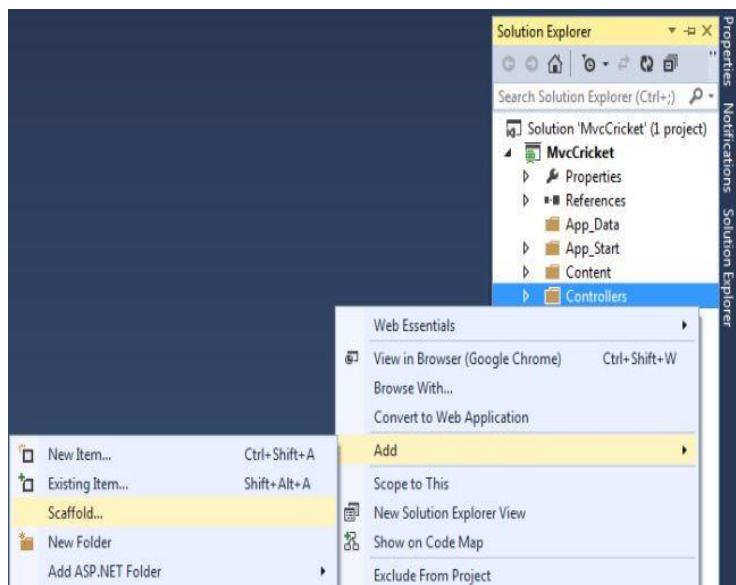


In the browser's URL, you will see "localhost" and a port number. Localhost represents that you are running it on local computer and port number generated by Visual Studio that is used for the web server.

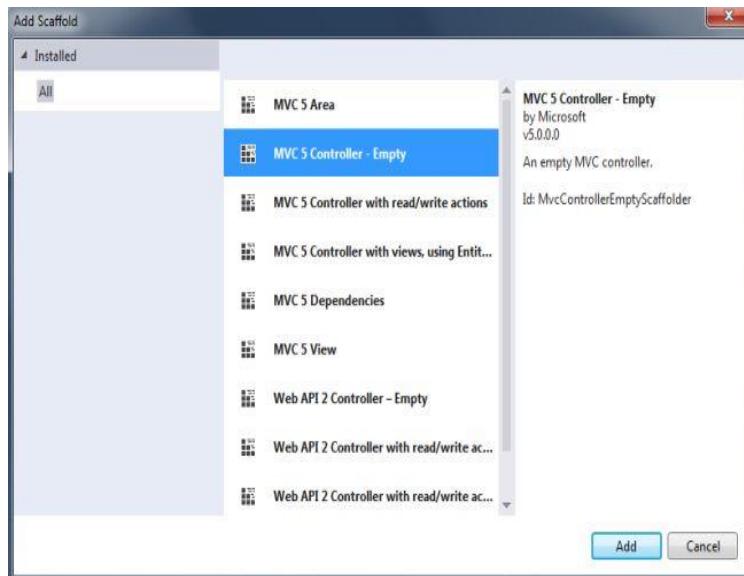
Working with Controller

As you know, there are various controllers already present in the application. So, let's start to create a new controller with the following procedure.

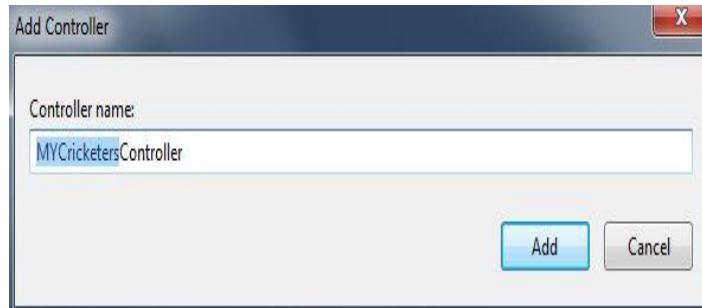
Step 1: Right-click on your Controller folder.



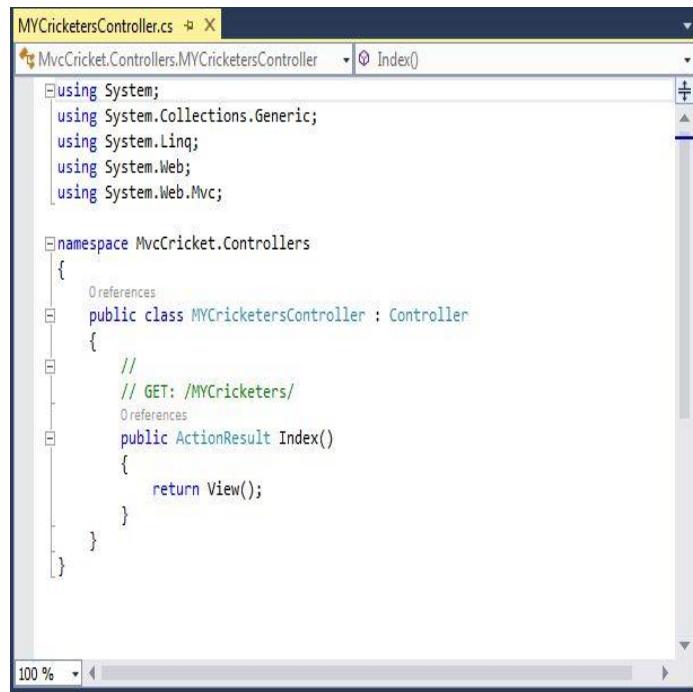
Step2: Click on "Add Scaffold".



Step 3: In the next wizard, enter your Controller name as "MYCrikceters".



Visual Studio will create the *MYCricketersController.cs* file and display your new controller.



```

MYCricketersController.cs  X
MvcCricket.Controllers.MYCricketersController  Index()
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcCricket.Controllers
{
    public class MYCricketersController : Controller
    {
        // GET: /MYCricketers/
        public ActionResult Index()
        {
            return View();
        }
    }
}

```

Step 4: Modify your code with the following code:

```

using System.Web.Mvc;

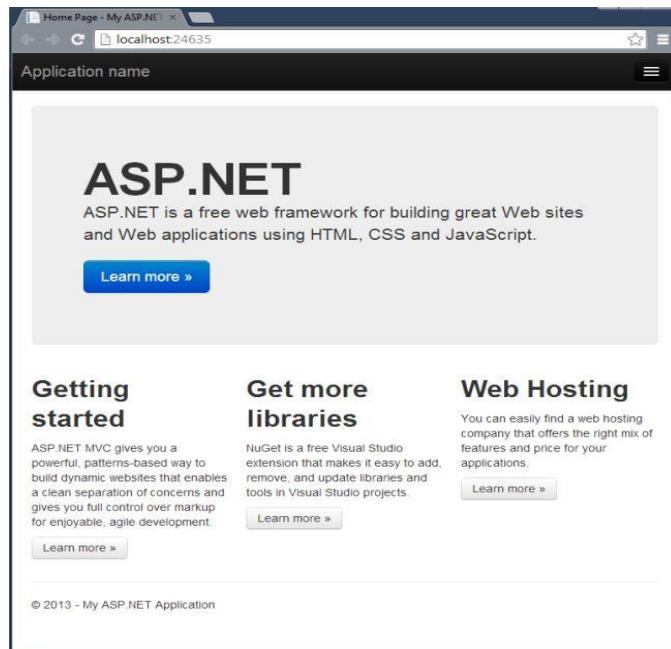
namespace MvcCricket.Controllers
{
    public class MYCricketersController : Controller
    {
        // GET: /MYCricketers/
        public string Index()
        {
            return "Hello this is My First Controller";
        }

        public string Welcome()
        {
            return "Welcome to My Cricketer Controller";
        }
    }
}

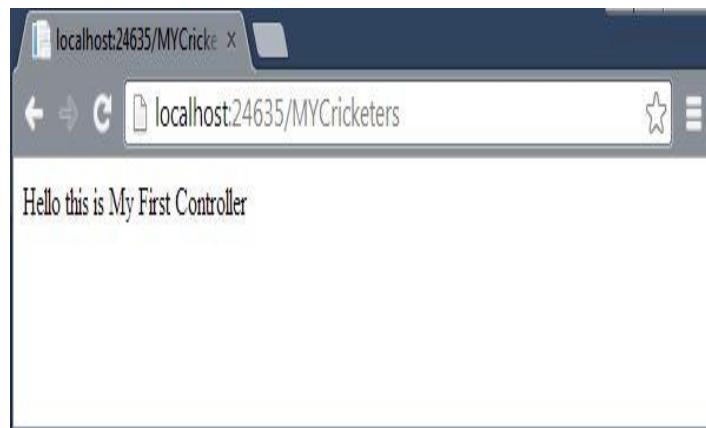
```

There are two controller methods used in here. These methods will return a string.

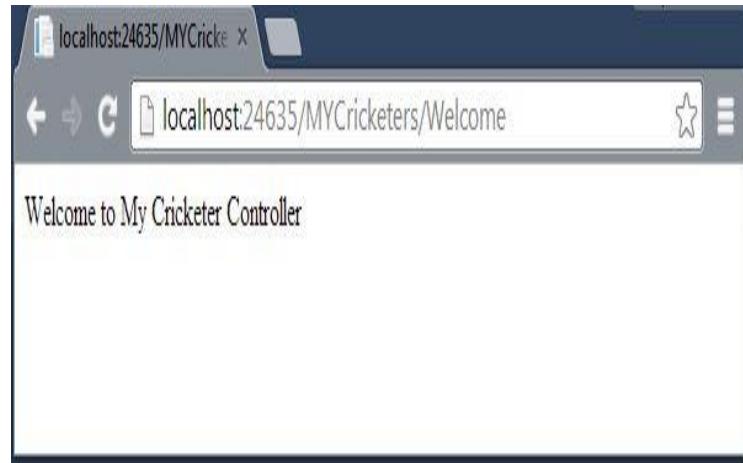
Step 5: Debug your application,



Visual Studio by default opens HomeController. You need to provide your controller name with the localhost like localhost:1234/MYCricketers. Enter your controller name with the localhost. In here, *MYCricketersController* is the first method named as Index.



Similarly open the Welcome method that will open by providing it with the URL as shown below:



Chapter 2: Social Provider Access in Web Application With MVC 5

2.1 Introduction

This chapter explains how to access the social providers in a MVC based ASP.NET Web Application. Facebook is used for the example of a social provider. There are various types of social providers available to be accessed, such as Google, Twitter and Microsoft. You can access your information in your social account such as you can access your friend's photos in your Facebook account.

In that context, I am creating a MVC based ASP.NET Web Application and I am also assuming that you have the knowledge necessary to configure the social login provider in your Web Application. You can visit [Configure the External Login Options in MVC](#) as a reference to access the external authentication for your ASP.NET Web Application.

Prerequisites

The following are the prerequisites:

- Create an ASP.NET MVC Web Application in Visual Studio 2013.
- Enable the Facebook Provider

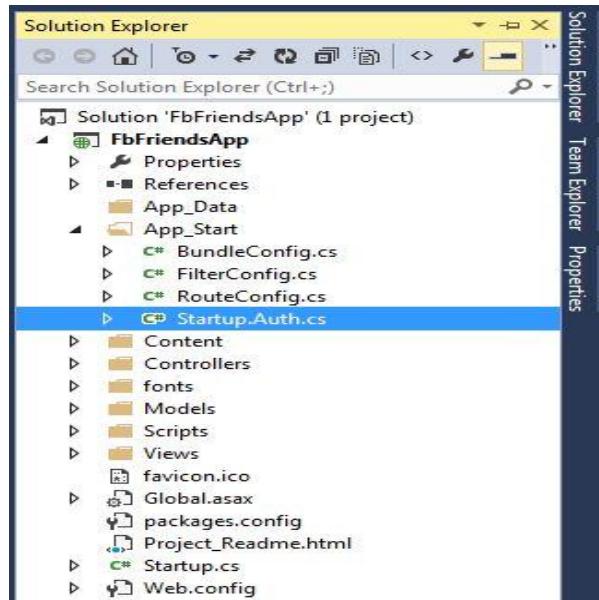
Now let's follow the sections mentioned below to proceed:

- Facebook Authentication
- Controller and Model Editing
- Adding View
- Access Application

2.2 Facebook Authentication

To authenticate the Facebook provider, use the following procedure.

Step 1: In your Solution Explorer, open the Startup authentication file as shown below:



Step 2: Find the *ConfigureAuth()* method and modify your code with the following code:

```

1. //app.UseTwitterAuthentication(
2. //  consumerKey: "",
3. //  consumerSecret: "");
4.
5. List<string> Social = new List<string>() { "email", "friends_about_me", "friends_photos" };
6.
7. var FbFriends = new FacebookAuthenticationOptions();
8.
9. FbFriends.Scope.Add("email");
10. FbFriends.Scope.Add("friends_about_me");
11. FbFriends.Scope.Add("friends_photos");
12. FbFriends.AppId = "App ID Value";
13. FbFriends.AppSecret="App Secret Value";
14.
15. FbFriends.Provider=new FacebookAuthenticationProvider()
16. {
17.     OnAuthenticated = async FbContext =>
18.     {
19.
20.         FbContext.Identity.AddClaim(
21.             new System.Security.Claims.Claim("FacebookAccessToken", FbContext.AccessToken));
22.     }
23. };
24.
25. FbFriends.SignInAsAuthenticationType=DefaultAuthenticationTypes.ExternalCookie;
26. app.UseFacebookAuthentication(FbFriends);

```

```

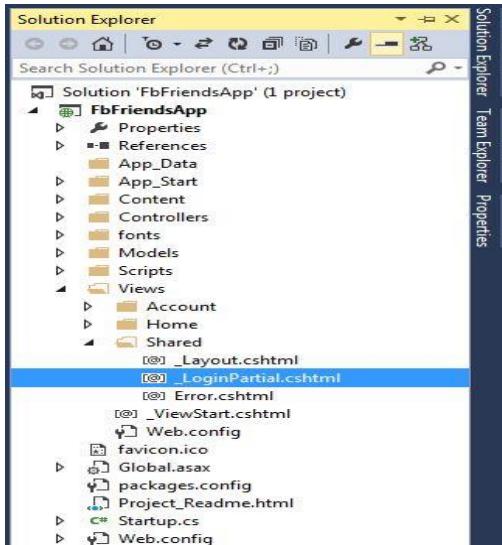
27.
28.
29. //app.UseGoogleAuthentication();
  
```

In the code above, *FacebookAuthenticationProvider()* is called each time the user authenticates with Facebook and the data is to be stored in the *FbContext.FacebookAccessToken* to access the friends of the user.

Controller and Model Editing

Now we need to edit our Controller and Model to access the information of User Facebook Account. Use the following procedure to do that.

Step 1: At first open your Login file to create an Action.



Step 2: Modify your code with the following code:

```

1. <ul class="nav navbar-nav navbar-right">
2.   <li>
3.     @Html.ActionLink("Hello " + User.Identity.GetUserName() + "!", "Manage", "Account", routeValues: null, htmlAttributes: new { title = "Manage" })
4.   </li>
5.
6.   <li>
  
```

```

7.      @Html.ActionLink("Facebook Friends", "FacebookFriends", "Account")
8.    </li>
9.
10.   <li><a href="javascript:document.getElementById('logoutForm').submit()">Log off</a></li>
11. </ul>

```

Step 3: Now open the *AccountController.cs* file from the Solution Explorer and proceed with the following sections to edit the controller:

- **Adding Task**

Add the following Task in your Account Controller:

```

1.  privateasync Task FbAuthenticationToken(ApplicationUser User)
2.  {
3.    var claims = await AuthenticationManager.GetExternalIdentityAsync(DefaultAuthenticationTypes.External
Cookie);
4.    if (claims != null)
5.    {
6.      var getclaim = await UserManager.GetClaimsAsync(User.Id);
7.      var FbToken = claims.FindAll("FacebookAccessToken").First();
8.      if (getclaim.Count() <= 0)
9.      {
10.        await UserManager.AddClaimAsync(User.Id, FbToken);
11.      }
12.    }
13.  }

```

- **Modify the ExternalLoginCallback()**

Modify the external login provider in this method with the following code:

```

1. // Sign in the user with this external login provider if the user already has a login
2.
3. var user = await UserManager.FindAsync(loginInfo.Login);
4. if (user != null)
5. {
6.   await FbAuthenticationToken(user);
7.   await SignInAsync(user, isPersistent: false);
8.   return RedirectToAction(returnUrl);
9. }

```

- **Modify the LinkLoginCallback()**

Modify this method with this code:

```

1. // GET: /Account/LinkLoginCallback

```

```

2.    public async Task<ActionResult> LinkLoginCallback()
3.    {
4.        var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey, User.Identity.GetUserId());
5.        if (loginInfo == null)
6.        {
7.            return RedirectToAction("Manage", new { Message = ManageMessageId.Error });
8.        }
9.        var result = await UserManager.AddLoginAsync(User.Identity.GetUserId(), loginInfo.Login);
10.       if (result.Succeeded)
11.       {
12.           var FbUser = await UserManager.FindByIdAsync(User.Identity.GetUserId());
13.           await FbAuthenticationToken(FbUser);
14.           return RedirectToAction("Manage");
15.       }
16.       return RedirectToAction("Manage", new { Message = ManageMessageId.Error });
17.   }

```

- **Modify the ExternalLoginConfirmation()**

Modify the external login provider in this method with the following code:

```

1. // Get the information about the user from the external login provider
2.
3. var info = await AuthenticationManager.GetExternalLoginInfoAsync();
4. if (info == null)
5. {
6.     return View("ExternalLoginFailure");
7. }
8. var user = new ApplicationUser() { UserName = model.UserName };
9. var result = await UserManager.CreateAsync(user);
10. if (result.Succeeded)
11. {
12.     result = await UserManager.AddLoginAsync(user.Id, info.Login);
13.     if (result.Succeeded)
14.     {
15.         await FbAuthenticationToken(user);
16.         await SignInAsync(user, isPersistent: false);
17.         return RedirectToAction(returnUrl);
18.     }
19. }
20. AddErrors(result);

```

- **Facebook Package**

Install the [Facebook NuGet Package](#) in your application.

- **Modify the AccountViewModel**

Add the following code in your *AccountViewModel.cs* file in the Account Folder:

```

1. public class FacebookViewModel
2.
3. {
4.     [Required]
5.     [Display(Name = "Friend's name")]
6.
7.     public string Name { get; set; }
8.
9.     public string Image { get; set; }
10. }
```

- **Modify Controller**

Add the following code in the Account Controller to access the Facebook Friends Information:

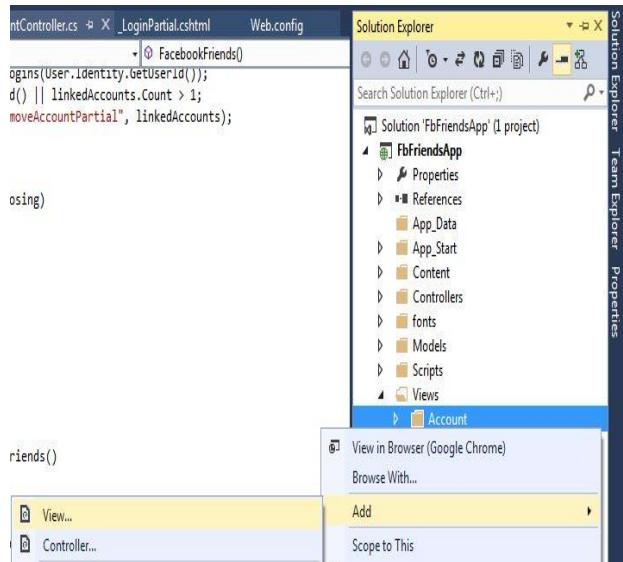
```

1. public async Task<ActionResult> FacebookFriends()
2. {
3.     var UserClaim = await
4.
5.     UserManager.GetClaimsAsync(User.Identity.GetUserId());
6.
7.     var token = UserClaim.FirstOrDefault(fb => fb.Type == "FacebookAccessToken").Value;
8.
9.     var FbClient = new FacebookClient(token);
10.
11.     dynamic FacebookFriends = FbClient.Get("/me/friends");
12.     var FbFriends = newList<FacebookViewModel>();
13.     foreach (dynamic MyFriend in FacebookFriends.data)
14.     {
15.         FbFriends.Add(new FacebookViewModel()
16.         {
17.             Name = MyFriend.name,
18.             Image = @"https://graph.facebook.com/" + MyFriend.id + "/picture?type=large"
19.         });
20.     }
21.
22.     return View(FbFriends);
23. }
```

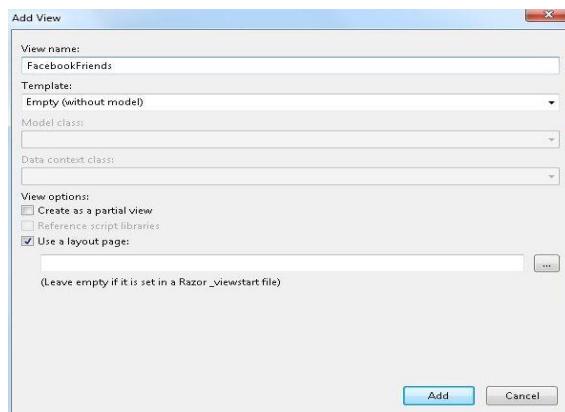
2.3 Adding View

The final step is to add the View to access Facebook. So, proceed with the following procedure.

Step 1: Add a View to your Account Folder:



Step 2: In the wizard, enter the View name:



Step 3: Replace the boilerplate code with the following code:

```
@model IList<FbFriendsApp.Models.FacebookViewModel>

@{
    ViewBag.Title = "Facebook Friends";
}
```

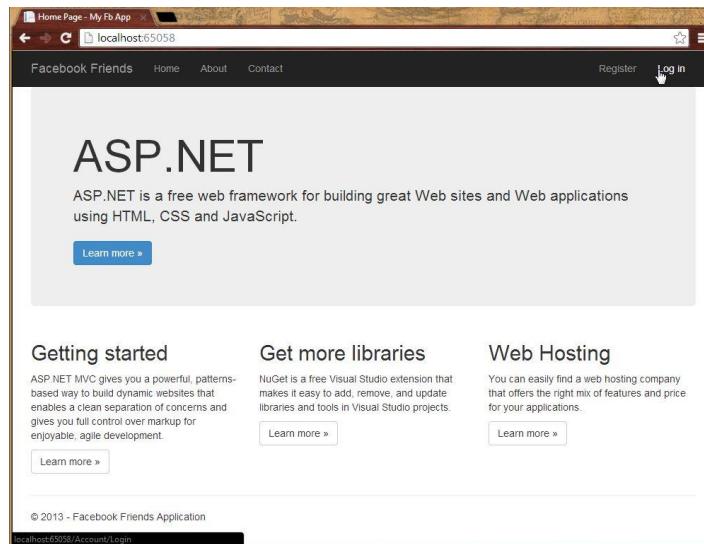
<h2>My Facebook Friends</h2>

```
@if (Model.Count > 0)
{
    <div class="row">
        @foreach (var MyFriend in Model)
        {
            <div class="col-md-3">
                <a href="#" class="thumbnail">
                    <img src=@MyFriend.Image alt=@MyFriend.Name />
                </a>
            </div>
        }
    </div>
}
```

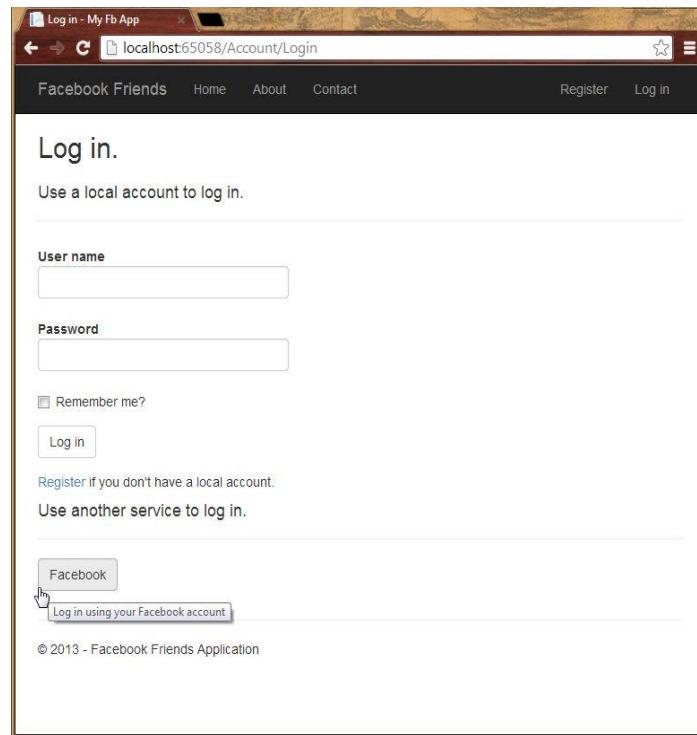
2.4 Access Application

Use the following procedure to run the application.

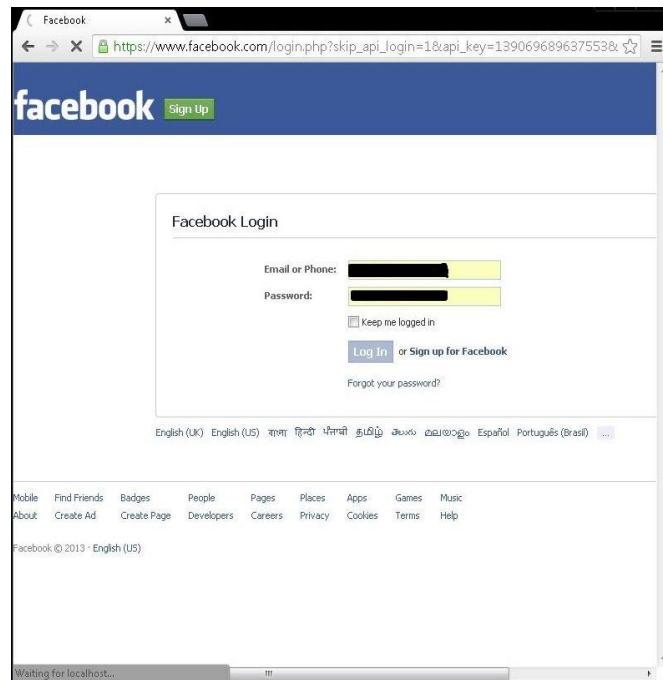
Step 1: Click on the LogIn link on your Home Page:

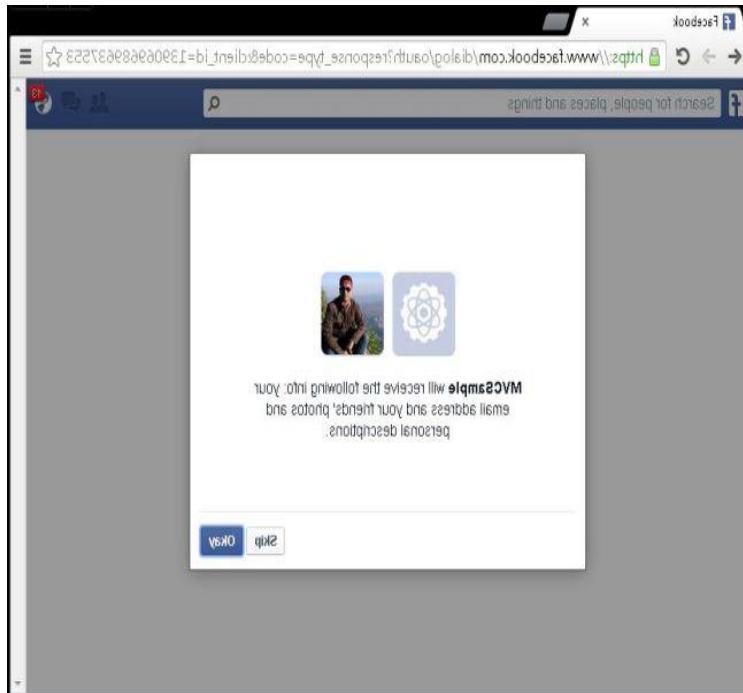
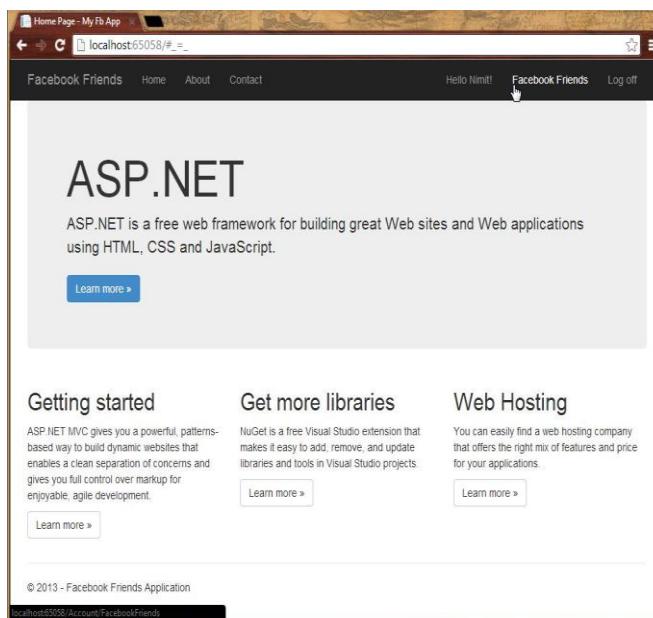


Step 2: Click on Facebook to proceed.



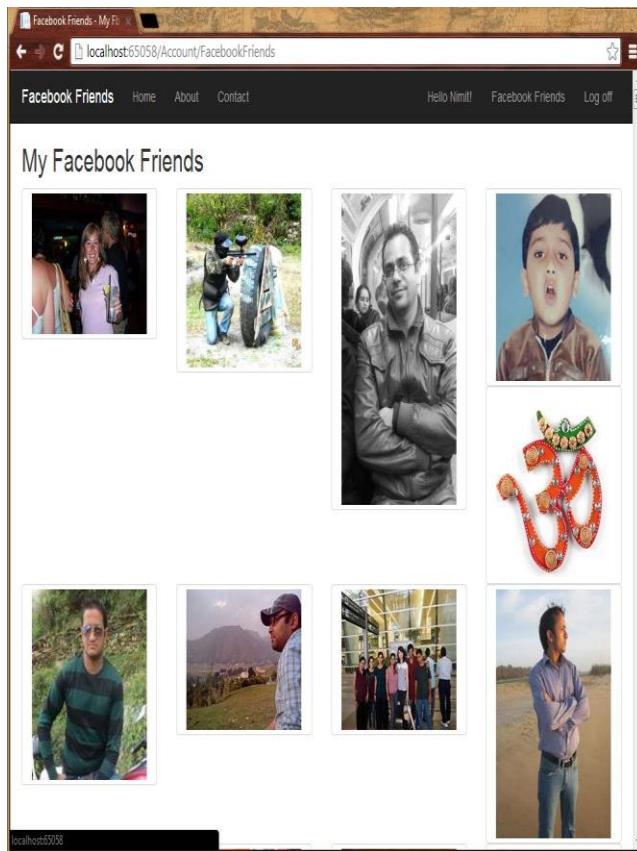
Step 3: Enter the Facebook Credentials.



Step 4: Authenticate the Facebook App.**Step 5:** Register yourself and click on the Facebook Friends to proceed.**Step 6:** Now you can view your Facebook Friends in your Web Application.

© 2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

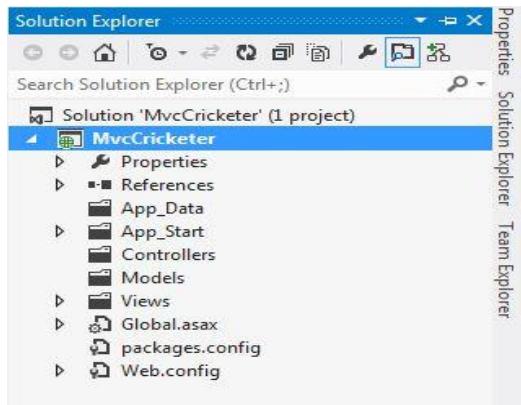


Step 7: Please log off.

Chapter 3: Working With MVC 5 in Visual Studio 2012

3.1 Introduction

Today, I am creating an ASP.NET Web Application using the MVC 5 Project Template in Visual Studio 2012. Surprised? Well, I think you don't know that Microsoft revealed the ASP.NET and Web Tools 2013.1 version for the Visual Studio 2012. I have also described this update in [New Web Tools for Visual Studio](#). As I said, we will describe the MVC 5 template later, so here I am creating the MVC 5 application in Visual Studio 2012. The selection of a MVC 5 project creates an Empty Project in Visual Studio 2012. We need to create an entire application. We can view some folders in the application solution like Models, Controllers, Views, App_Start and so on as shown in the following screenshot:



Prerequisites

- Visual Studio 2012 (updated with Web Tools 2013.1)

Note: Please note that you must have updated your Visual Studio 2012, otherwise you cannot work on a MVC 5 project in Visual Studio 2012. Read the [Full Information](#). We will now create our Home Page to display the Application Home Page and then we'll perform the CRUD Operations. So, proceed with the following sections:

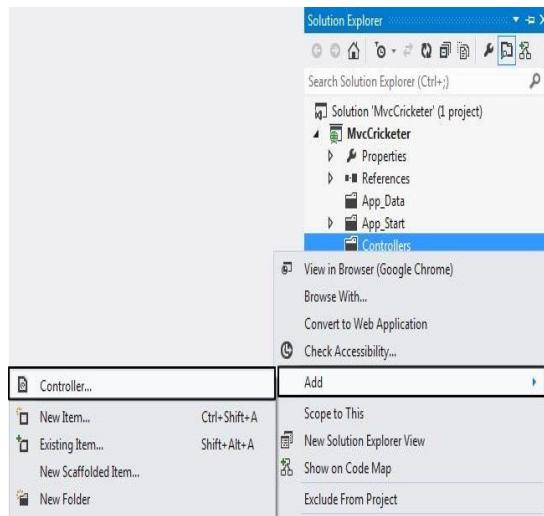
- Getting Started with Empty Application
- Perform CRUD Operations
- Database Connection

Getting Started with Empty Application

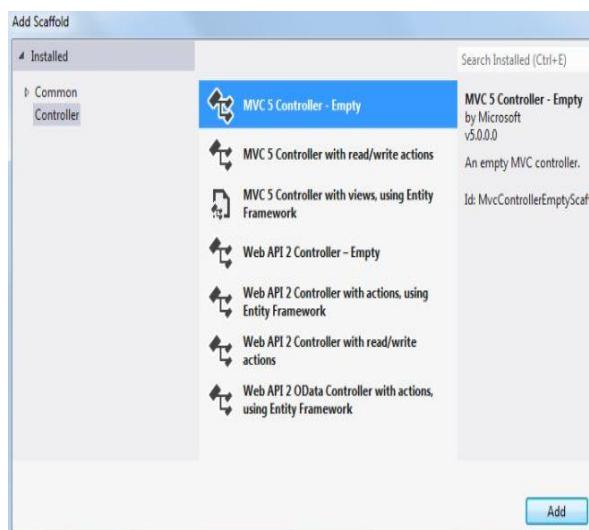
Use the following procedure to create a sample of your empty application.

3.2 Working with Controller

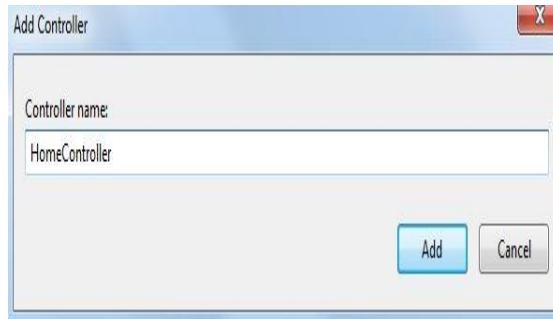
Step 1: At first we need to add a Controller. Just right-click on your Controllers folder to add one.



Step 2: Select the MVC 5 Empty Controller in the next wizard.



Step 3: Enter the Controller name as HomeController.



Step 4: Modify your code with the following code:

```
using System.Web.Mvc;

namespace MvcCricketer.Controllers
{
    public class HomeController : Controller
    {
        //
        // GET: /Home/
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult Welcome()
        {
            ViewBag.Message = "Your Welcome Page";

            return View();
        }
    }
}
```

Step 5: Build your solution.

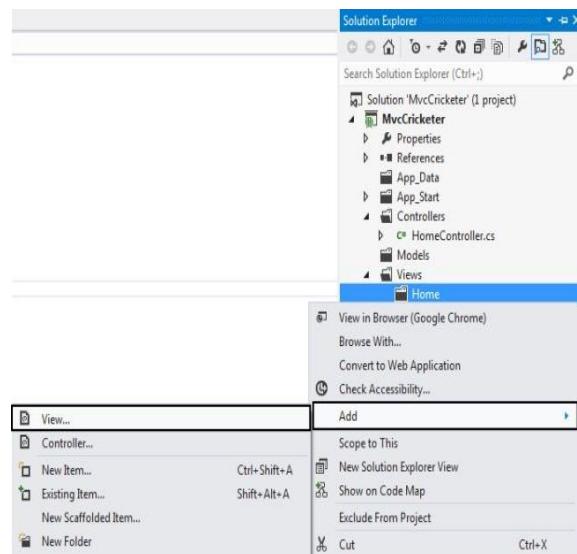
After scaffolding the HomeController, by default Index() is created. I have added one more method, in other words Welcome() to this controller.

3.3 Working with View

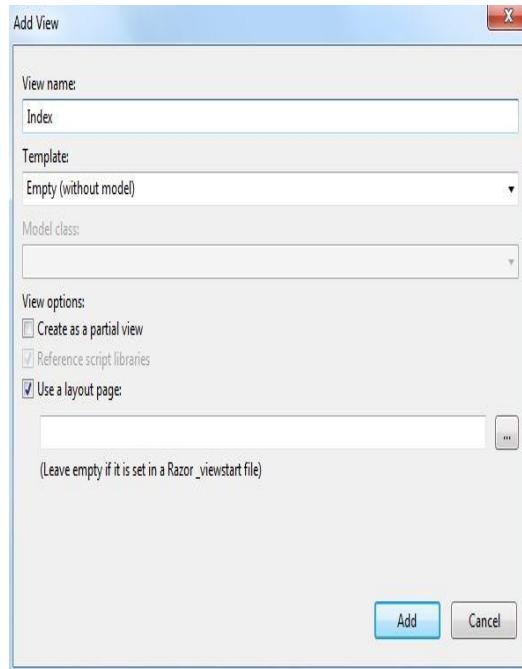
As you saw, we have created the Controllers, so now we'll create a View for our Controller using the following procedure.

Step 1: Create a new Home folder in the Views folder.

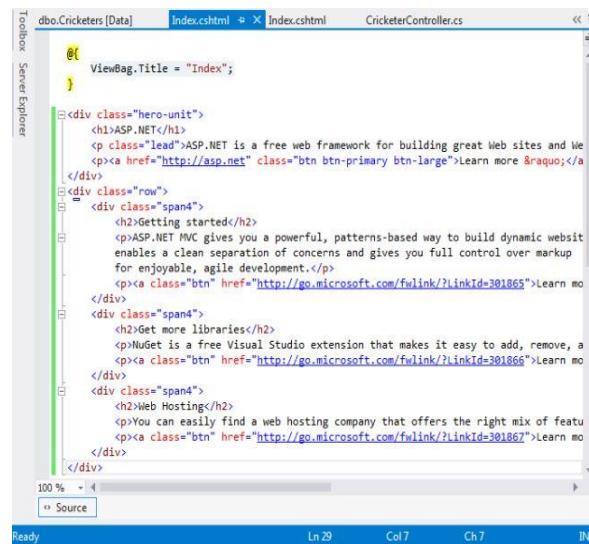
Step 2: Right-click on the "Home" folder to add a View.



Step 3: Enter the View name as "Index".



Step 4: Create some content to be shown in your home page. You can see my screenshot of "Index.cshtml".



```

@{
    ViewBag.Title = "Index";
}

<div class="hero-unit">
    <h1>ASP.NET</h1>
    <p class="lead">ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript. ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that are enjoyable, agile development.</p>
    <p><a href="http://asp.net" class="btn btn-primary btn-large">Learn more &rarr;</a></p>
</div>
<div class="row">
    <div class="span4">
        <h2>Getting started</h2>
        <p>ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that are enjoyable, agile development.</p>
        <p><a class="btn" href="http://go.microsoft.com/fwlink/?LinkId=301865">Learn more &rarr;</a></p>
    </div>
    <div class="span4">
        <h2>Get more libraries</h2>
        <p>Get a free Visual Studio extension that makes it easy to add, remove, and manage libraries for your ASP.NET MVC projects.</p>
        <p><a class="btn" href="http://go.microsoft.com/fwlink/?LinkId=301866">Learn more &rarr;</a></p>
    </div>
    <div class="span4">
        <h2>Web Hosting</h2>
        <p>You can easily find a web hosting company that offers the right mix of features for your application.</p>
        <p><a class="btn" href="http://go.microsoft.com/fwlink/?LinkId=301867">Learn more &rarr;</a></p>
    </div>
</div>

```

Step 5: Create a "Welcome.cshtml" page in the "Home" folder and you can replace your code with the following code:

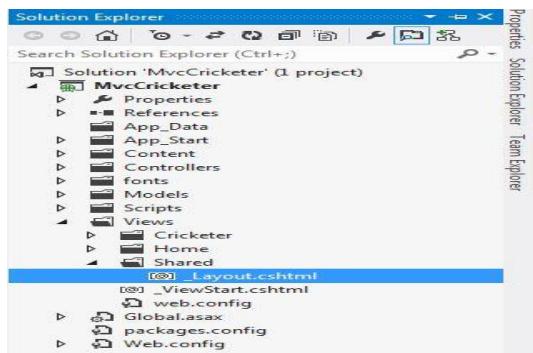
```
@{
```

```
ViewBag.Title = "Welcome";  
}  
  
<h2>Welcome</h2>  
<p>Welcome to Visual Studio 2012 that is upgraded by ASP.NET Web Tools 2013.1</p>
```

3.4 Working with View Layout

Use the following procedure to create your home page layout:

Step 1: Open the _Layout.cshtml file.



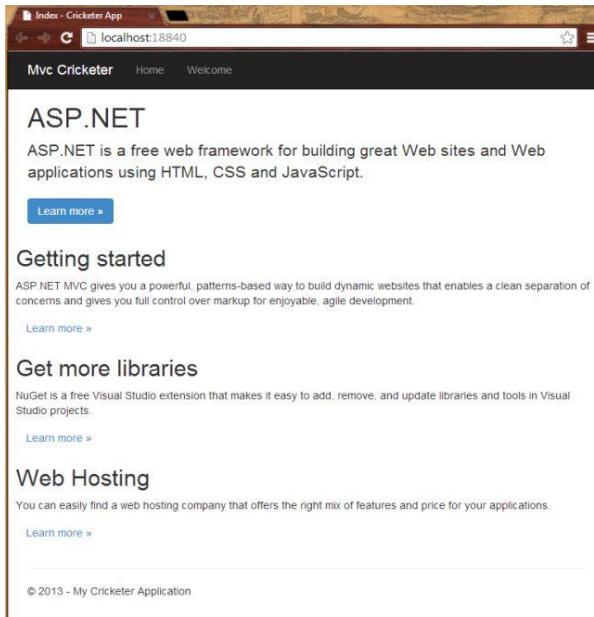
Step 2: Modify your code with the code below in the screenshot:

```

Layout.cshtml  X ViewStart.cshtml  dbo.Cricketers [Data]  Index.cshtml  Index.cshtml  CricketerController.cs
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>ViewBag.Title - Cricketer App</title>
    <link href="/Content/Site.css" rel="stylesheet" type="text/css" />
    <link href="/Content/bootstrap.min.css" rel="stylesheet" type="text/css" />
    <script src="/Scripts/modernizr-2.6.2.js"></script>
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <@Html.ActionLink("Mvc Cricketer", "Index", "Home", null, new { @class = "navbar-brand" })>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><@Html.ActionLink("Home", "Index", "Home")></li>
                    <li><@Html.ActionLink("Welcome", "Welcome", "Home")></li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        <@RenderBody()>
        <hr />
        <footer>
            <p>© &copy; @DateTime.Now.Year - My Cricketer Application</p>
        </footer>
    </div>
</body>
</html>

```

Step 3: Debug your application from HomeController.



The screenshot shows a browser window titled 'Index - Cricketer App' with the URL 'localhost:18840'. The page content is as follows:

ASP.NET
 ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.
[Learn more »](#)

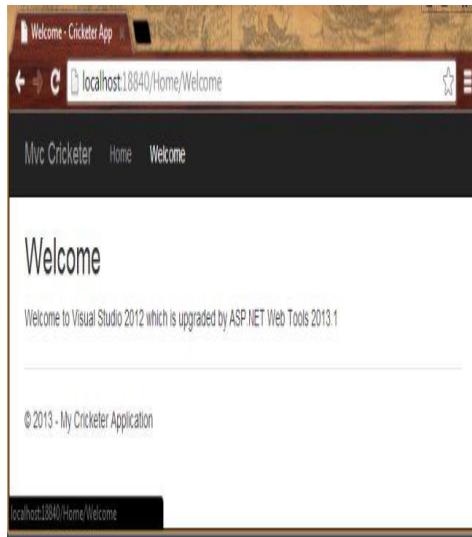
Getting started
 ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.
[Learn more »](#)

Get more libraries
 NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.
[Learn more »](#)

Web Hosting
 You can easily find a web hosting company that offers the right mix of features and price for your applications.
[Learn more »](#)

© 2013 - My Cricketer Application

Step 4: Click on "Welcome" to open the Welcome Page.



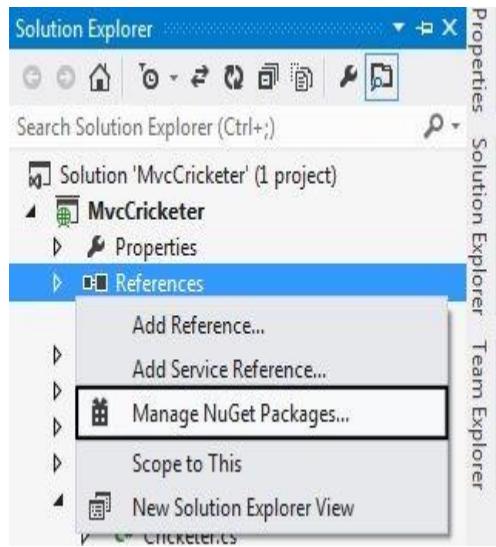
3.5 Perform CRUD Operations

Now, as we saw that our empty project has a new controller and view to display the page using a page layout. In this section we will perform Create, Read, Update and Delete (CRUD) operations to the records we saved in our application. So, proceed with the following sections.

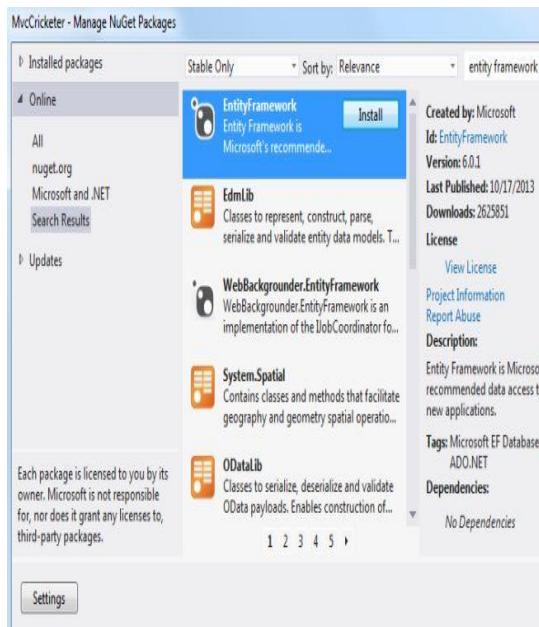
Creating Model

At first we need to create a Model in our application using following steps:

Step 1: Add an Entity Framework reference to your project.



Step 2: Install an Entity Framework.

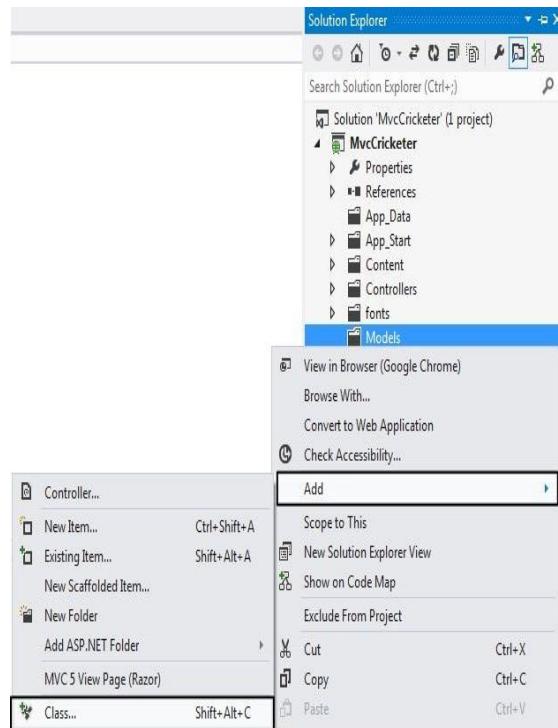


The screenshot shows the 'MvcCricketer - Manage NuGet Packages' window. On the left, there's a sidebar with 'Installed packages', 'Online' (selected), and 'Updates'. Under 'Online', there are links for 'All', 'nuget.org', 'Microsoft and .NET', and 'Search Results'. Below this is a note about package licensing. The main area lists several packages:

- EntityFramework**: Microsoft's recommended package. It has an 'Install' button highlighted in blue. Details: Created by Microsoft, Id: EntityFramework, Version: 6.0.1, Last Published: 10/17/2013, Downloads: 2625851, License: View License.
- EdmLib**: Classes to represent, construct, parse, serialize and validate entity data models. T...
- WebBackground.EntityFramework**: WebBackground.EntityFramework is an implementation of the IblobCoordinator fo...
- System.Spatial**: Contains classes and methods that facilitate geography and geometry spatial operatio...
- ODatalib**: Classes to serialize, deserialize and validate OData payloads. Enables construction of...

At the bottom, there are navigation links (1, 2, 3, 4, 5) and a 'Settings' button.

Step 3: Add a class as Cricketer by right-clicking on the Models folder.



Step 4: Replace the boilerplate code with the following code:

```
using System.Data.Entity;

namespace MvcCricketer.Models
{
    public class Cricketer
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public int ODI { get; set; }
        public int Test { get; set; }
        public string Grade { get; set; }
    }

    public class CricketerDbContext : DbContext
    {
        public DbSet<Cricketer> Cricketers { get; set; }
    }
}
```

Step 5: Build the solution.

Database Connectivity

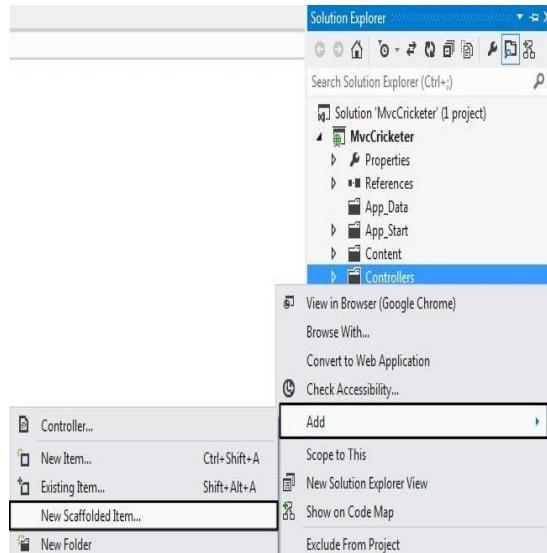
We need to set our context named "CricketerDbContext" with the database. So, open the Web.Config file and paste in the following code:

```
<connectionStrings>
  <add name="CricketerDbContext"
    connectionString="Data Source=(LocalDb)\v11.0;
    AttachDbFilename=|DataDirectory|\MyCricketers.mdf;
    Integrated Security=true"
    providerName="System.Data.SqlClient"/>
</connectionStrings>
```

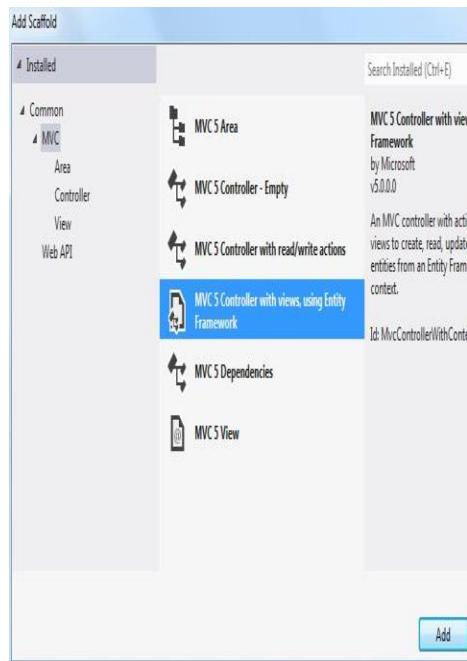
5.6 Connectivity with Controller

You have successfully created your model, now we need to connect the model with the controller using the following procedure.

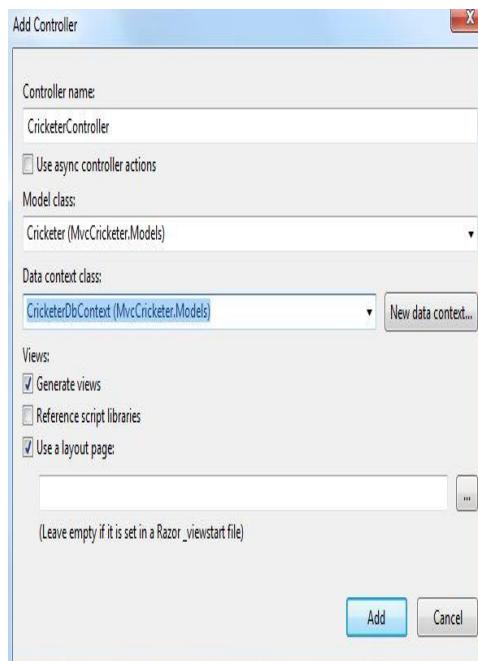
Step 1: Right-click on "Controllers" and add a new Scaffolded item.



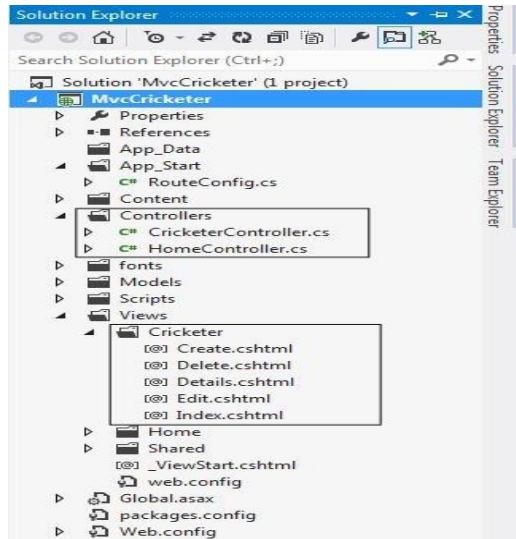
Step 2: Select the MVC 5 Controller with the Views option.



Step 3: Enter the controller name as "CricketController" then select the model and data context class,



Visual Studio automatically scaffolds your CricketerController and creates the view also. Check out the Solution Explorer for the Cricketer folder newly created in the Views folder.



Step 3: Modify the page links in your home page from the "Page_Layout" file with the following code:

```
<li>@Html.ActionLink("Home", "Index", "Home")</li>
<li>@Html.ActionLink("Welcome", "Welcome", "Home")</li>
<li>@Html.ActionLink("Cricketer", "Index", "Cricketer")</li>
```

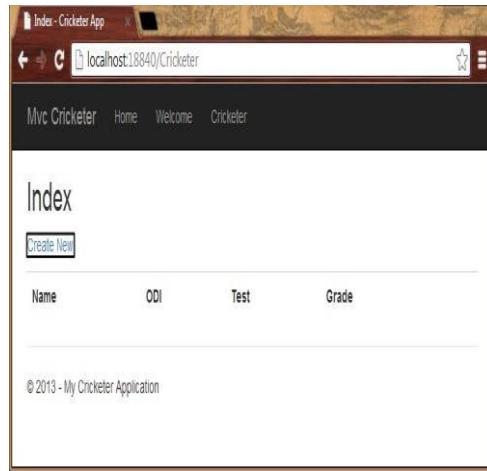
I added a Cricketer link in my home page.

5.7 Working with CRUD Operation on the application

We now have the Model, View and Controller. We are prepared to run our application.

- **Create**

Step 1: Debug the application and click on the "Cricketers" link. It will open the page and you can create new records. Click on the "Create New" link as shown below:



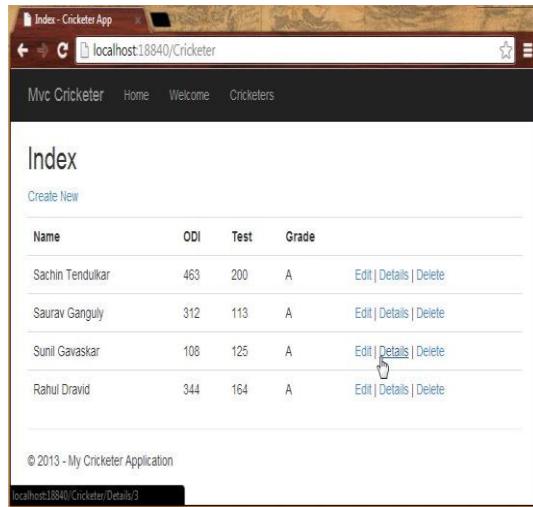
Step 2: Enter the records

Enter more records in your application.

- **Read**

In this section you will read the record.

Step 1: Click on the "Details" link on the record.



Index

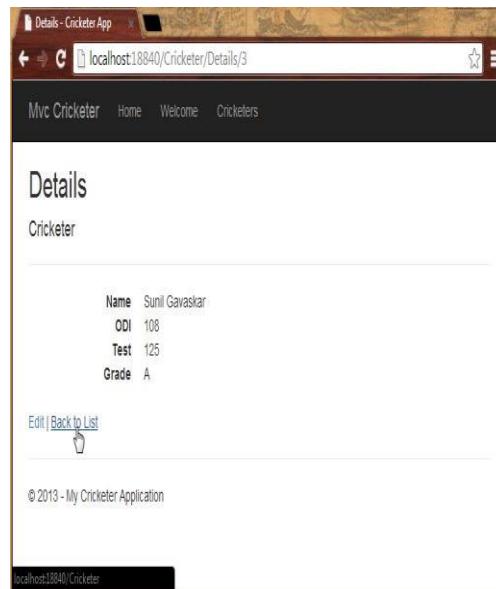
Create New

Name	ODI	Test	Grade	
Sachin Tendulkar	463	200	A	Edit Details Delete
Saurav Ganguly	312	113	A	Edit Details Delete
Sunil Gavaskar	108	125	A	Edit Details Delete
Rahul Dravid	344	164	A	Edit Details Delete

© 2013 - My Cricketer Application

<localhost:18840/Cricketer/Details/3>

Step 2: Show the details and click on "Back to the List".



Details

Cricketer

Name	Sunil Gavaskar
ODI	108
Test	125
Grade	A

[Edit](#) | [Back to List](#)

© 2013 - My Cricketer Application

<localhost:18840/Cricketer>

- **Update**

In this section you will update the record.

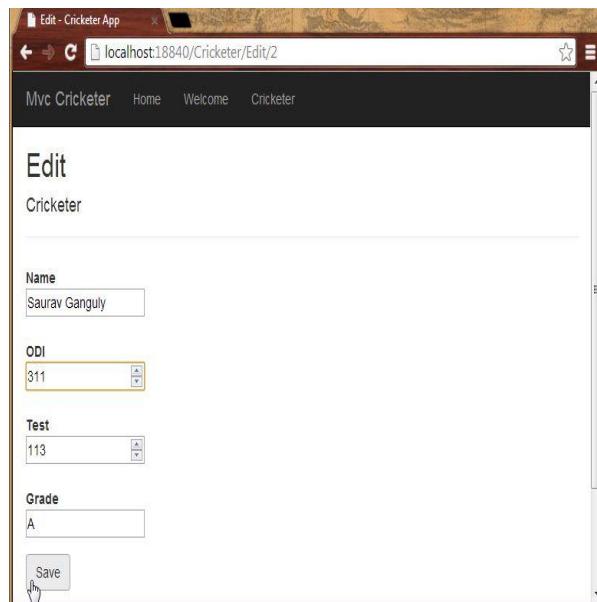
Step 1: Click on the "Edit" link on the record that is to be updated.

Name	ODI	Test	Grade	
Sachin Tendulkar	463	200	A	Edit Details Delete
Saurav Ganguly	312	113	A	Edit Details Delete
Sunil Gavaskar	108	125	A	Edit Details Delete
Rahul Dravid	344	164	A	Edit Details Delete
Javagal Srinath	229	67	B	Edit Details Delete

© 2013 - My Cricketer Application

localhost:18840/Cricketer/Edit/2

Step 2: Update the record and save it.



Edit

Cricketer

Name
Saurav Ganguly

ODI
311

Test
113

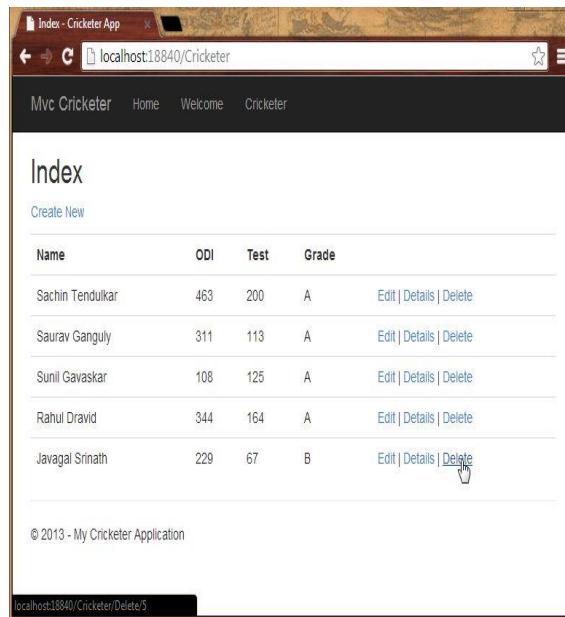
Grade
A

Save

- **Delete**

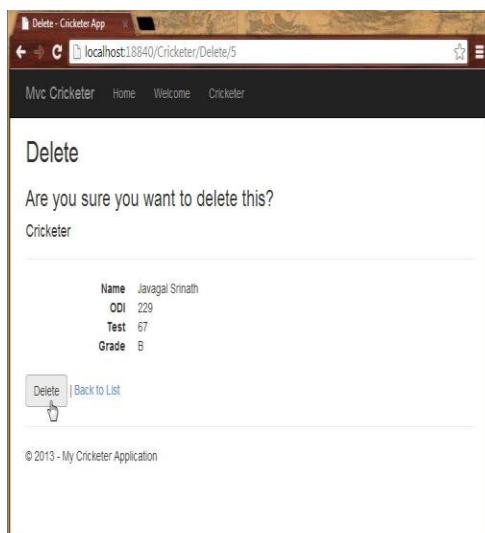
In this section we will delete the records.

Step 1: Click on the "Delete" link to delete the record.



The screenshot shows the 'Index' view of a web application. The page title is 'Index - Cricketer App'. The URL in the browser is 'localhost:18840/Cricketer'. The navigation bar includes 'Mvc Cricketer', 'Home', 'Welcome', and 'Cricketer'. Below the navigation is a heading 'Index' and a 'Create New' button. A table lists five cricketers with columns: Name, ODI, Test, and Grade. Each row contains 'Edit | Details | Delete' links. The last row, for Javagal Srinath, has the 'Delete' link highlighted with a mouse cursor. At the bottom of the page is a copyright notice: '© 2013 - My Cricketer Application'.

Step 2: The final step to delete any record.



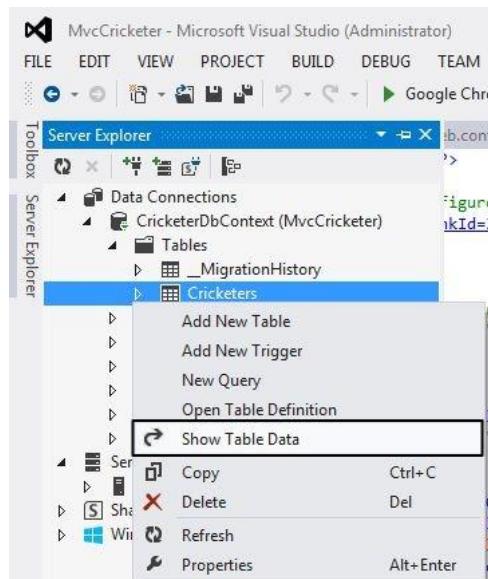
The screenshot shows the 'Delete' view of the application. The page title is 'Delete - Cricketer App'. The URL is 'localhost:18840/Cricketer/Delete/5'. The navigation bar is identical to the Index view. The main content asks 'Are you sure you want to delete this?'. Below this is a 'Cricketer' table with one row for Javagal Srinath, showing his details: Name (Javagal Srinath), ODI (229), Test (67), and Grade (B). At the bottom are 'Delete' and 'Back to List' buttons, with the 'Delete' button being clicked by a mouse cursor. The footer copyright notice is present.

3.7 Database Connection

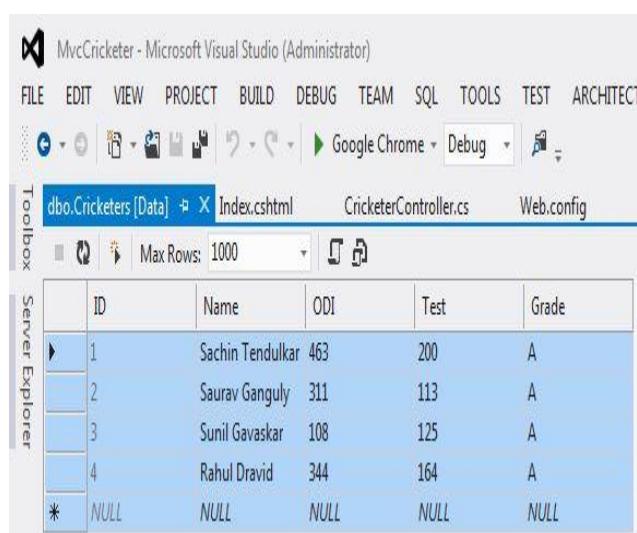
We can also check out the records in our application using the Server Explorer.

Step 1: Check out the data context in the Server Explorer.

Step 2: In your table, right-click and click "Show table data".



Step 3: You can see the table records.



	ID	Name	ODI	Test	Grade
1	Sachin Tendulkar	463	200	A	
2	Saurav Ganguly	311	113	A	
3	Sunil Gavaskar	108	125	A	
4	Rahul Dravid	344	164	A	
*	NULL	NULL	NULL	NULL	NULL

Chapter 4: Working with External Providers in MVC 5

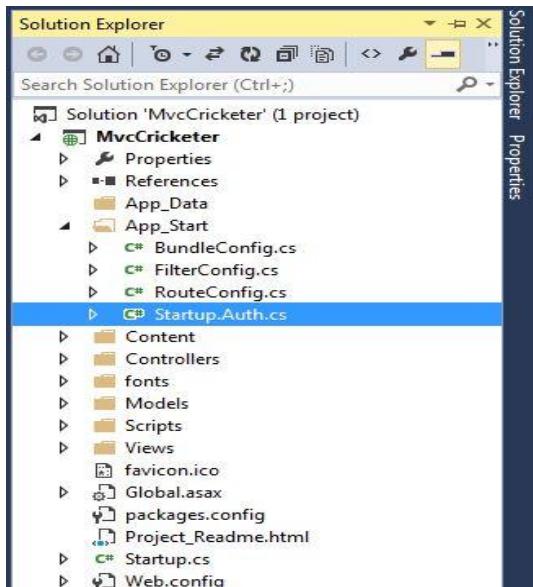
4.1 Introduction

This chapter explains customization of a social providers button in the ASP.NET MVC 5 using Visual Studio Studio 2013. As you all know, we can create the MVC 5 Project based ASP.NET Web Application in Visual Studio 2013, so here I am customizing the login provider's buttons.

4.2 Registration of OAuth Providers

We need to first enable the authentication providers for the project. Use the following procedure to create a sample of doing that.

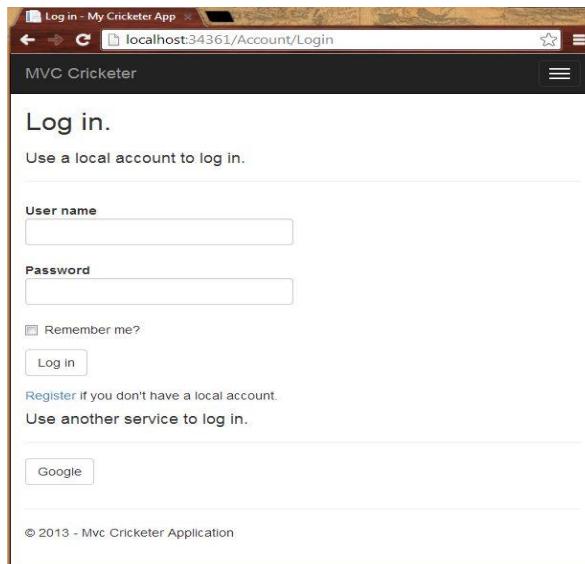
Step 1: Open the Startup.Auth.cs file.



Step 2: Enable the Google provider as shown below:

```
app.UseGoogleAuthentication();
```

Step 3: Run your application and open the Login page.

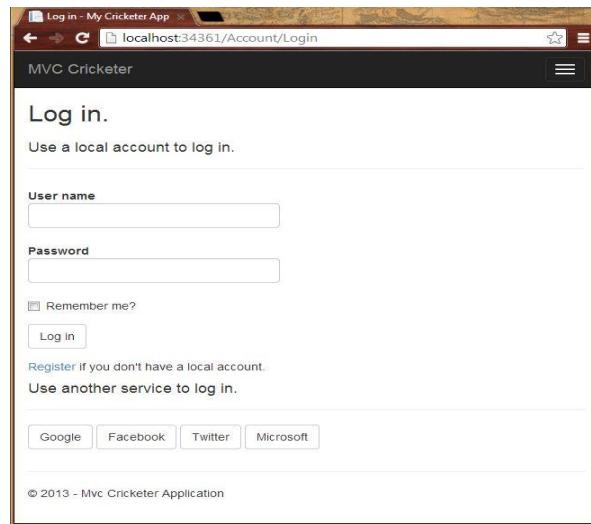


Step 4: Use the same and open other providers and enter the app_id and app_secret value as "x" as shown below:

```
app.UseTwitterAuthentication(
    consumerKey: "x",
    consumerSecret: "x");
```

```
app.UseFacebookAuthentication(
    appId: "x",
    appSecret: "x");
```

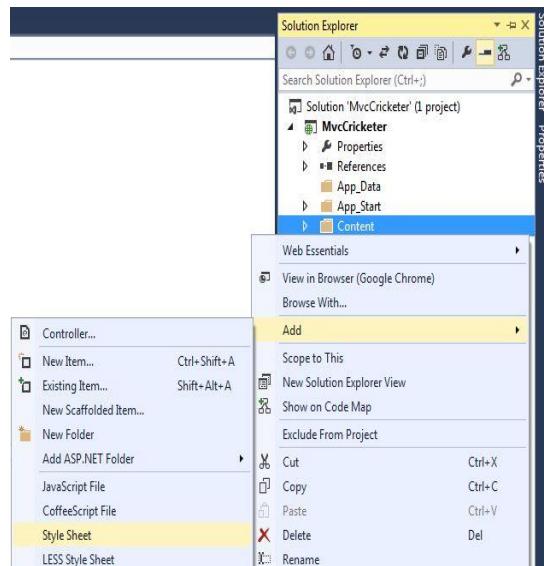
These are the simple social provider buttons to be displayed on your login page.



4.3 Customization of OAuth Providers

I am using a CSS file that is expressed on GitHub. Use the following procedure to do that.

Step 1: Create a CSS file in Content.

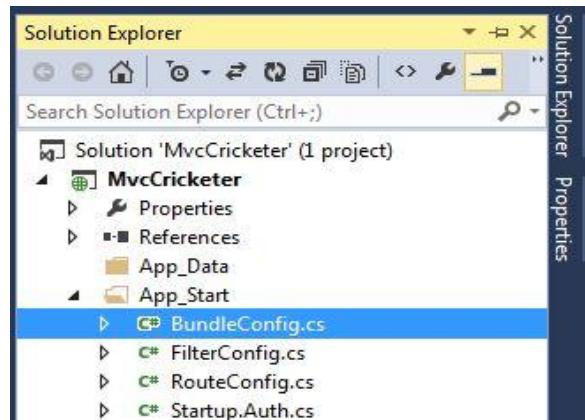


Step 2: Name it "Zocial.css".



Step 3: Copy the content in it from the CSS file stored in [GitHub](#).

Step 4: Open the Bundle Configuration file.



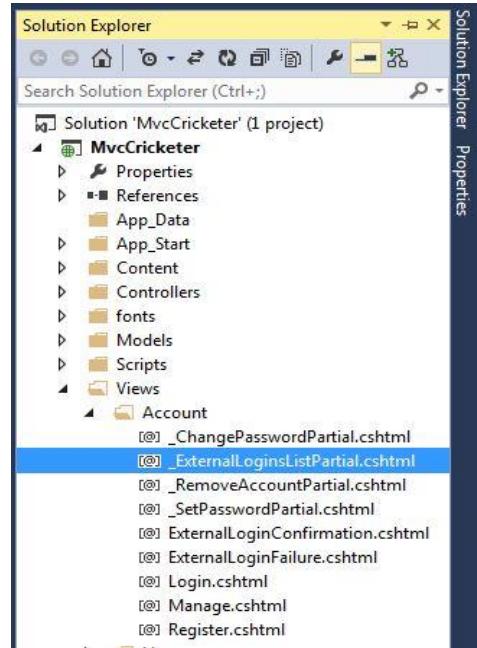
Step 5: Modify your code with the following code:

```
bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
    "~/Scripts/bootstrap.js",
    "~/Scripts/respond.js"));

bundles.Add(new StyleBundle("~/Content/css").Include(
    "~/Content/bootstrap.css",
    "~/Content/site.css",
    "~/Content/zocial.css"));
```

In the code above I added the reference of my CSS file created in the Content folder.

Step 6: Now, open the external login file as in the following:



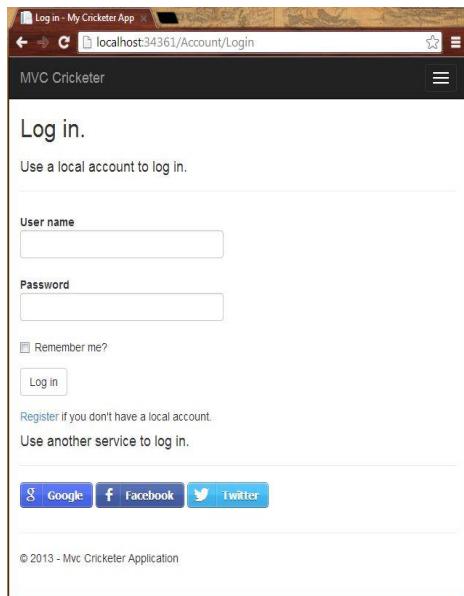
Step 7: Modify the code with the following code:

```

using (Html.BeginForm(action, "Account", new { ReturnUrl = returnUrl }))
{
    @Html.AntiForgeryToken()
    <div id="socialLoginList">
        <p>
            @foreach (AuthenticationDescription p in loginProviders)
            {
                <button type="submit"
                    class="zocial @p.AuthenticationType.ToLower()"
                    id="@p.AuthenticationType"
                    name="provider" value="@p.AuthenticationType"
                    title="Log in using your @p.Caption account">
                    @p.AuthenticationType</button>
            }
        </p>
    </div>
}

```

Step 8: Run your application and open the Login page.



5 Getting Started with Areas in MVC 5

5.1 Introduction

You all know that MVC (Model, View, Controller) is a design pattern to separate the data logic from the business and presentation logic. We can also design the structure physically, where we can keep the logic in the controllers and views to exemplify the relationships.

It is also possible that we can have large projects that use MVC, and then we need to split the application into smaller units called *areas* that isolate the larger MVC application into smaller functional groupings. A MVC application could contain several MVC structures (areas).

In this chapter I am creating a simple application for defining the area in MVC 5. MVC 5 is the latest version of MVC used in Visual Studio 2013.

Prerequisites

You need to have the following to complete this chapter:

- MVC 5
- Visual Studio 2013

In that context, we'll follow the sections given below:

- MVC 5 application
- Adding Controller for Area
- Adding Views for Area
- Area Registration
- Application Execution

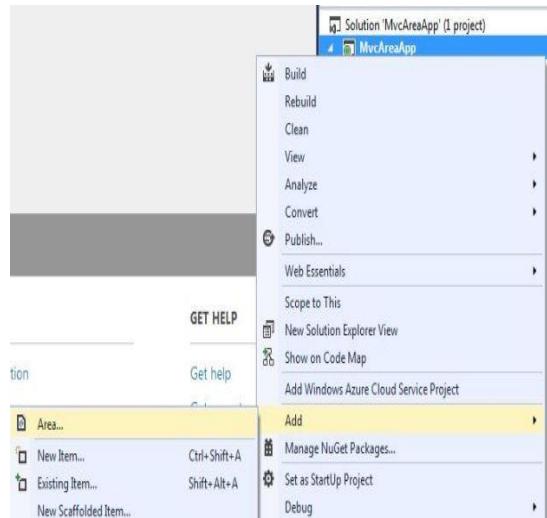
5.2 MVC 5 Application

Use the following procedure to create a Web application based on a MVC 5 template.

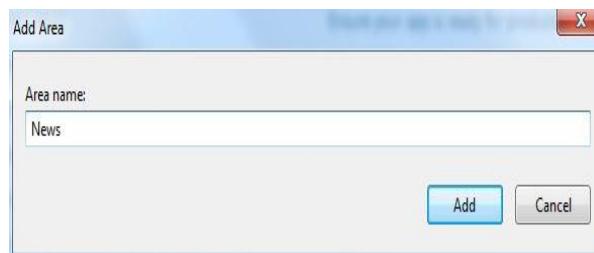
Step 1: Open Visual Studio 2013.

Step 2: Create an ASP.NET Web Application with MVC 5 project template.

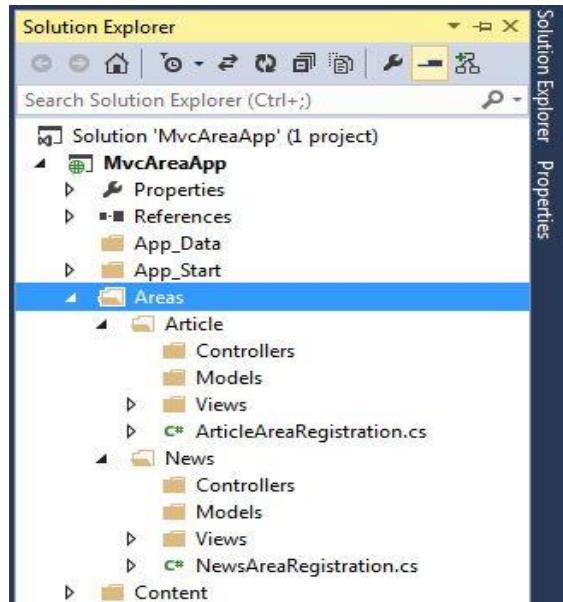
Step 3: In Solution Explorer, right-click on the project and click "Add" to add an area as shown below:



Step 4: Enter the name for the area, such as "News".



Step 5: Similarly add an other area named "Article".

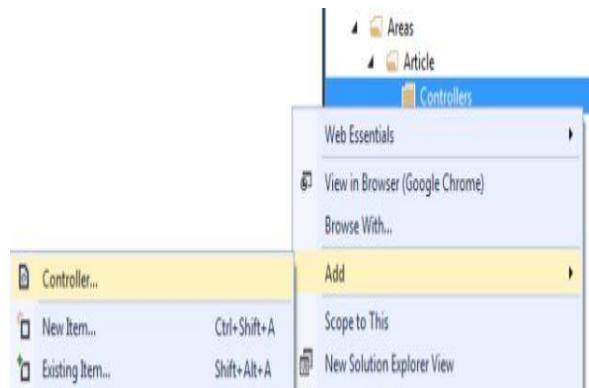


Now from the steps above you have added two areas for your application named News and Article.

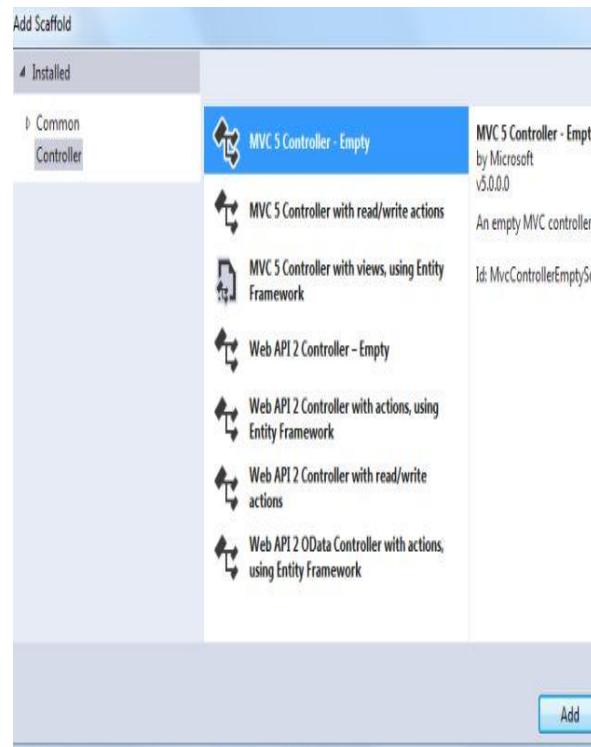
5.3 Adding Controller for Area

We have successfully added an area, now we'll add controllers for each of our areas using the following procedure.

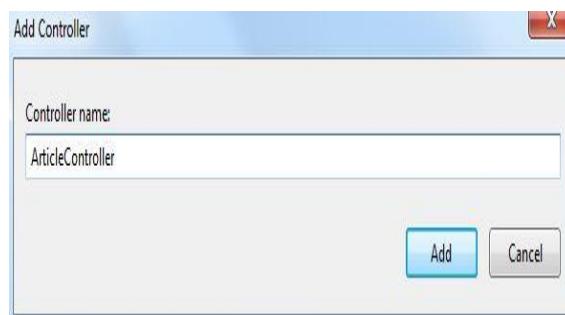
Step 1: Right-click on the Controller in your Article area to add a controller.



Step 2: Select "MVC 5 Empty Controller".

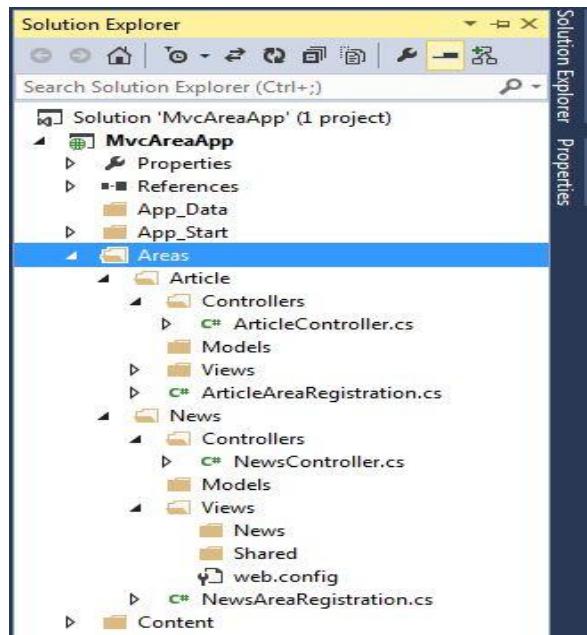


Step 3: Enter the name as "ArticleController".



Step 4: Similarly add the controller for "News".

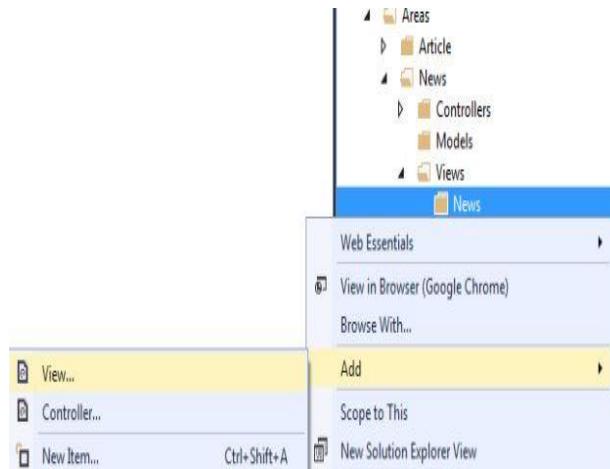
Now your Area folder should be as in the following screenshot:



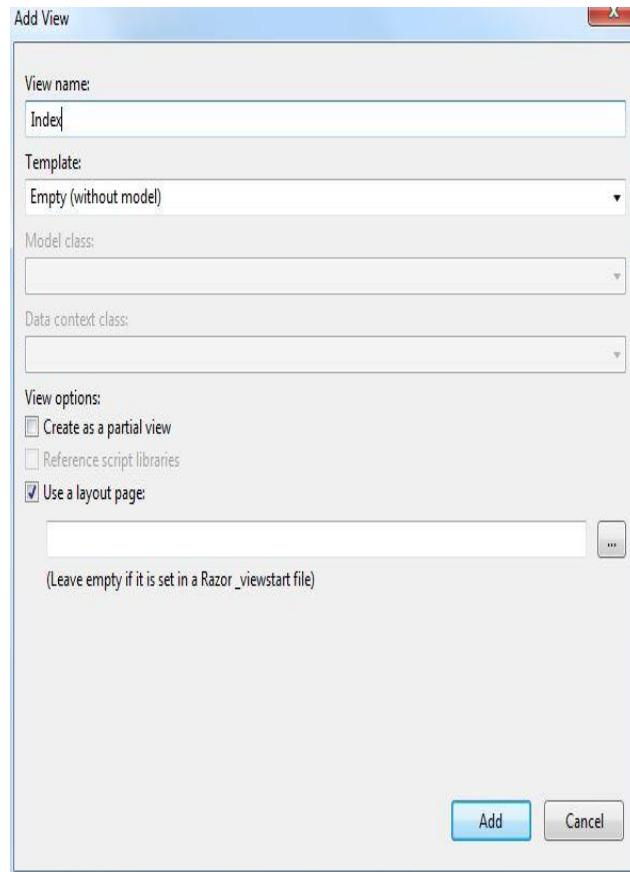
5.4 Adding Views for Area

We have successfully added a controller for our area, now to add a view for the area using the following procedure.

Step 1: Right-click on the "News" folder in the View to add a View for the News Area.



Step 2: Enter the view name as defined in the NewsController.



Step 3: Generate some content in the View of News as in the following screenshot:

Index.cshtml

```

@{
    ViewBag.Title = "Today News";
}



## Latest News



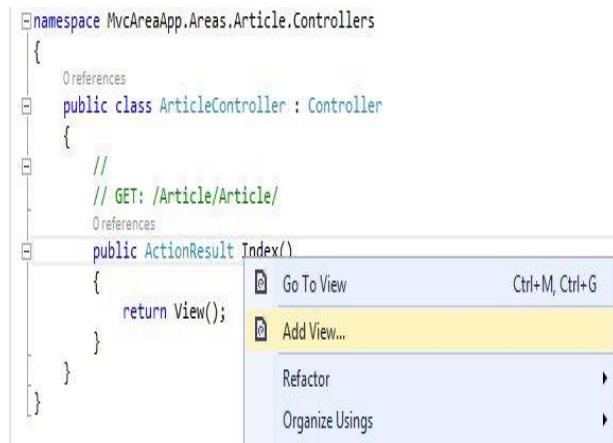
This time Delhi Election makes new record in the Election history of Delhi.



Read More


```

Step 4: You can also add a view as shown in the following screenshot:



Step 5: Generate some content for the Article view.

5.5 Area Registration

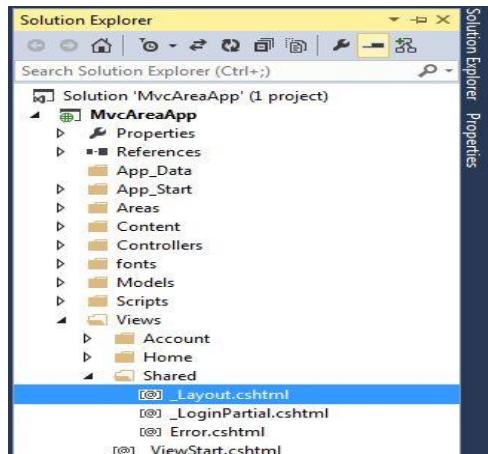
Step 1: Open the "Global.asax" file.

Step 2: Add the following code in your Application_Start() method:

```
AreaRegistration.RegisterAllAreas();
```

Application Execution

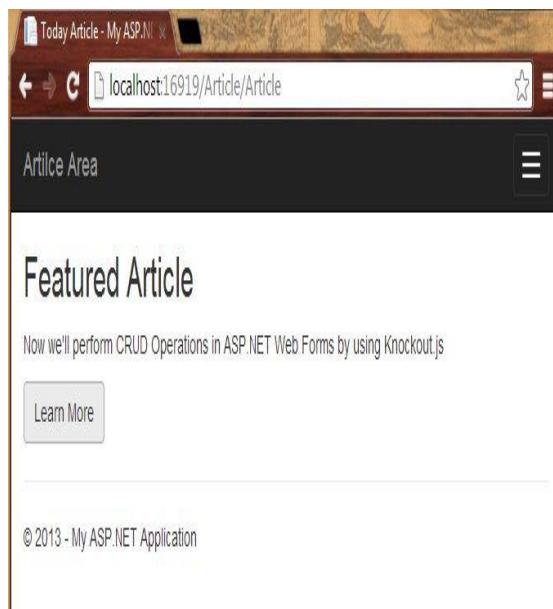
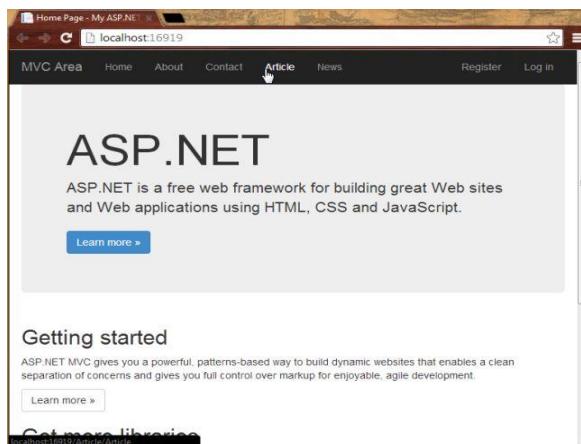
Step 1: Open the project view Layout file.



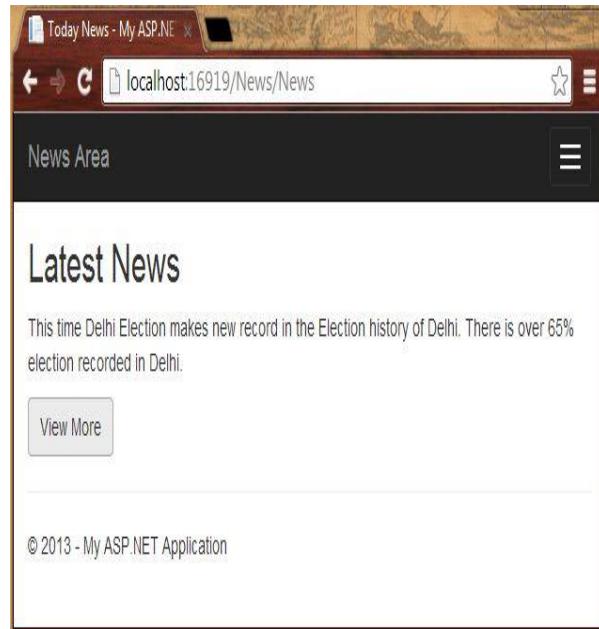
Step 2: Modify the in the layout file as shown in the following code:

```
<ul class="nav navbar-nav">
    <li>@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("About", "About", "Home")</li>
    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
    <li>@Html.ActionLink("Article", "Index", "Article", new {area= "Article" }, null)</li>
    <li>@Html.ActionLink("News", "Index", "News", new { area = "News" }, null)</li>
</ul>
```

Step 3: Debug the application and open the Article link as shown below:



Step 4: Similarly open the News link:



6 Perform CRUD Functionality in ASP.Net MVC 5

6.1 Introduction

In this chapter we'll learn how to do Create, Read, Update and Delete (CRUD) operations with the Entity Framework Data Model in the ASP.NET Web Application based on the MVC Project Template in Visual Studio 2013. As I described in the MVC with Entity Framework Data Model, we can create a [MVC Application with Entity Framework 6](#), therefore here we perform the CRUD functionality.

So, let's proceed with the following section:

- Updating the application and Perform CRUD Operations

Updating the application and doing CRUD Operations

At first we update our application to designate more briefly. We can also modify our pages to define more details of students about their enrollments and courses in their details. Use the following procedure to update the details page.

6.2 Editing Details Page

As we define the enrollments property in the student that hold the collection, so here we define it in the details page so that we can show which courses the student enrolls in. **Step 1:** You can the *Details Action* in the *Controllers\StudentController.cs* page that it used in the *Find()* to get the student entity.

```
public ActionResult Details(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Student student = db.Students.Find(id);
    if (student == null)
    {
```

```

        return HttpNotFound();
    }
    return View(student);
}

```

Step 2: Open the *Views\Student\Details.cshtml* page and add the highlighted code below:

```

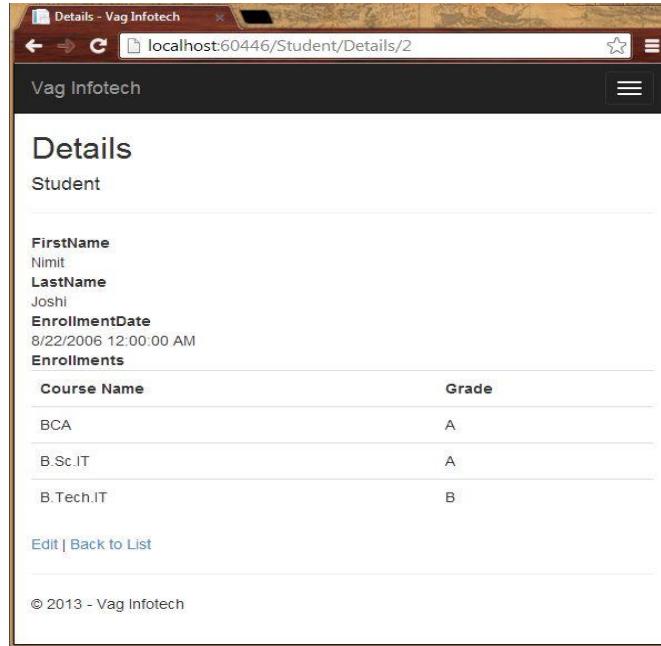
<dd>
    @Html.DisplayFor(model => model.EnrollmentDate)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.Enrollments)
</dt>
<dd>
    <table class="table">
        <tr>
            <th>Course Name</th>
            <th>Grade</th>
        </tr>
        @foreach (var items in Model.Enrollments)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelitems => items.Course.Name)
                </td>
                <td>
                    @Html.DisplayFor(mdoelitems => items.Grade)
                </td>
            </tr>
        }
    </table>
</dd>

```

You can press "Ctrl+K+D" to apply the indentation in your code. In the code above the Enrollments navigation property is used to display the Course name and Grade of the student, if the student enrolls in one or more courses then it is displayed again by the foreach loop. All of this data is retrieved from the database automatically when it is needed.

Step 3: Press "Ctrl+F5" to run the application.

Step 4: Click on the "Students" link and then we want the details of Nimit. So, click on the details link:



In the preceding screenshot, you can see that the student enrollments in various courses and grade. It shows the list of enrollments for the selected student.

6.3 Editing Create Page

Step 1: Open the *Controllers\StudentController.cs* page and find the Create Action and modify it with the highlighted code below:

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include="FirstName,LastName,EnrollmentDate")] Student student)
{
    try
    {
        if (ModelState.IsValid)
        {

```

```

        db.Students.Add(student);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
}

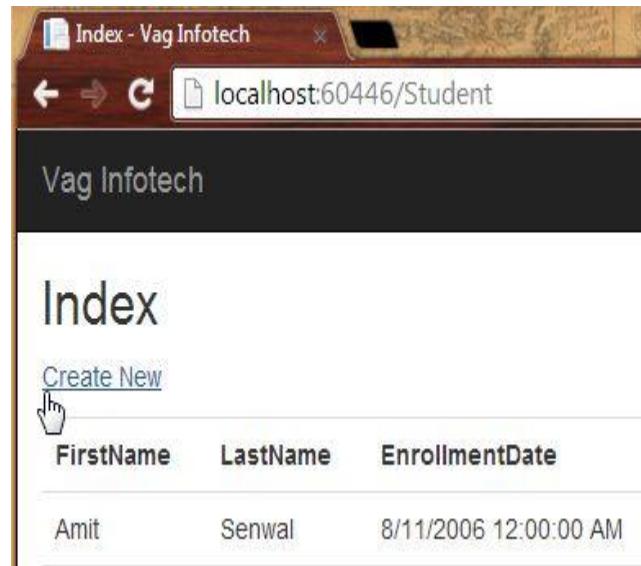
catch (DataException /*dataex*/)
{
    ModelState.AddModelError("", "Unable to save changes. Try again or Contact to the Administrator");
}

return View(student);
}

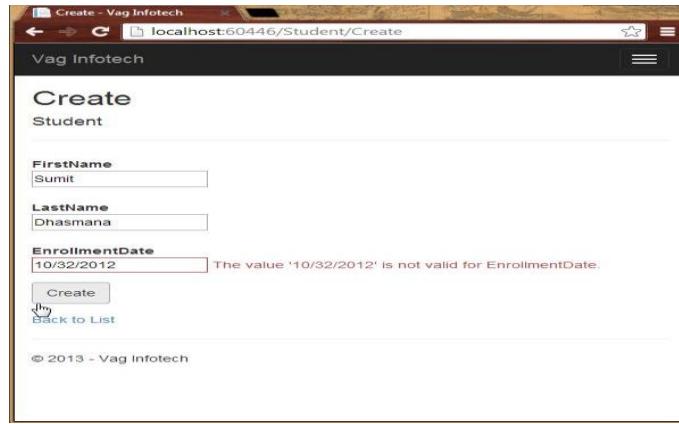
```

The code above will add the Student entity in the Student entity set that is created by the MVC model binder and save the changes. We remove the ID because it is the primary key column and when the new row is inserted, SQL Server will set this automatically. The user cannot set the ID value from Input.

Step 2: Run the application and click on "Create new link".



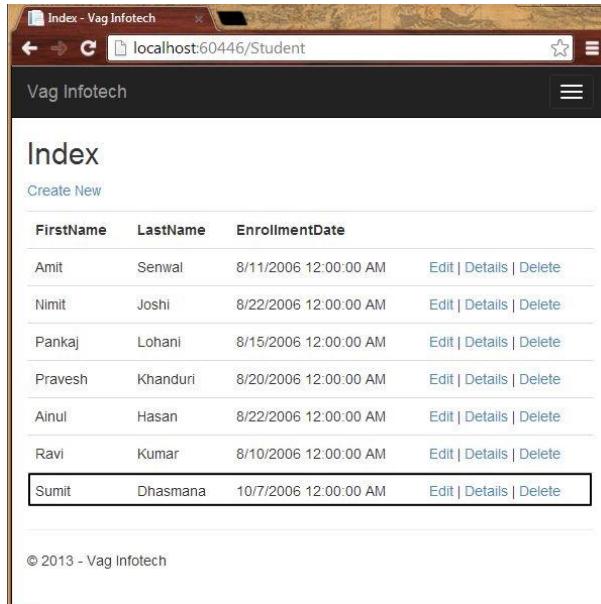
Step 3: In Create New page we can show the server-side validation by entering the wrong date as shown below:



You can see the Server-Side validation (by-default) in the code below:

```
if (ModelState.IsValid)
{
    db.Students.Add(student);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Step 4: Enter the valid date and you can see the added record as in the following:



FirstName	LastName	EnrollmentDate	
Amit	Senwal	8/11/2006 12:00:00 AM	Edit Details Delete
Nimit	Joshi	8/22/2006 12:00:00 AM	Edit Details Delete
Pankaj	Lohani	8/15/2006 12:00:00 AM	Edit Details Delete
Pravesh	Khanduri	8/20/2006 12:00:00 AM	Edit Details Delete
Ainul	Hasan	8/22/2006 12:00:00 AM	Edit Details Delete
Ravi	Kumar	8/10/2006 12:00:00 AM	Edit Details Delete
Sumit	Dhasmana	10/7/2006 12:00:00 AM	Edit Details Delete

6.4 Editing Edit Page

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

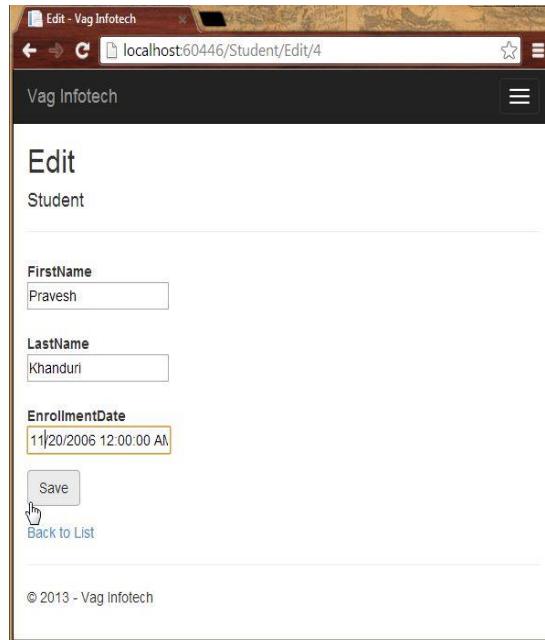
We add the "try-catch" block here to update the Edit page. It is optional. It uses the Find() to get the Student entity.

Step 1: Modify your code with the highlighted code below:

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include="ID,FirstName,LastName,EnrollmentDate")] Student student)
{
    try
    {
        if (ModelState.IsValid)
        {
            db.Entry(student).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("Index");
        }
    }
    catch(DataException /*DataEx*/)
    {
        ModelState.AddModelError("", "Unable to save changes. Try again or Contact to the Administrator");
    }
    return View(student);
}
```

The code above sets a flag on the entity and indicates that this has been changed. When the flag has been changed the SaveChanges() runs and the modified flag is responsible for informing the Entity Framework to run the update query in the SQL statement and update the row.

Step 2: Press "Ctrl+F5" to run the application and click on the "Edit" link to edit the student.



Edit

Student

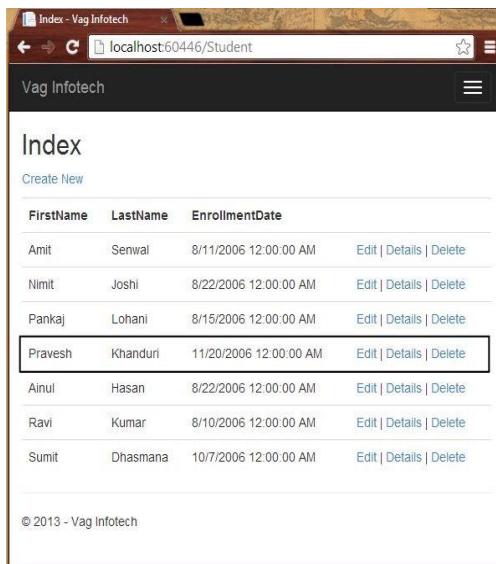
FirstName
Pravesh

LastName
Khanduri

EnrollmentDate
11/20/2006 12:00:00 AM

© 2013 - Vag Infotech

Step 3: You can see the Edited record in the following:



Index

Create New

FirstName	LastName	EnrollmentDate	Action
Amit	Senwal	8/11/2006 12:00:00 AM	Edit Details Delete
Nimit	Joshi	8/22/2006 12:00:00 AM	Edit Details Delete
Pankaj	Lohani	8/15/2006 12:00:00 AM	Edit Details Delete
Pravesh	Khanduri	11/20/2006 12:00:00 AM	Edit Details Delete
Ainul	Hasan	8/22/2006 12:00:00 AM	Edit Details Delete
Ravi	Kumar	8/10/2006 12:00:00 AM	Edit Details Delete
Sumit	Dhasmana	10/7/2006 12:00:00 AM	Edit Details Delete

© 2013 - Vag Infotech

6.5 Editing Delete Page

The Delete method in the *StudentController.cs* page uses the *Find()* to delete the row as we saw above in the "Details" and "Create" pages. We can add some custom error message if the *SaveChanges()* failed to execute.

There are two types of action methods called in here. When the method calls and the response is GET, it shows the confirmation page to delete the selected data and when the user accepts it, a POST request is created. The *HttpPost Delete* method is called and then delete operation is actually performed.

We'll add a "try-catch" block here to show the custom message if any error occur during the update of the database.

Step 1: Find and replace the *Delete* action method, with the highlighted code below:

```
public ActionResult Delete(int? id, bool? ErrorinSaveChanges=false)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    if (ErrorinSaveChanges.GetValueOrDefault())
    {
        ViewBag.ErrorMessage = "Failed to Delete. Try again or Contact to System Administrator";
    }
    Student student = db.Students.Find(id);
    if (student == null)
    {
        return HttpNotFound();
    }
    return View(student);
}
```

Step 2: Modify the *HttpPost DeleteConfirmed* action method with the code below:

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Delete(int id)
{
    try
    {
        Student student = db.Students.Find(id);
        db.Students.Remove(student);
```

```

        db.SaveChanges();
    }
    catch (DataException /*DataEx*/)
    {
        return RedirectToAction("Delete", new { id = id, errorinsavechanges = true });
    }
    return RedirectToAction("Index");
}

```

The code above gets the selected entity and calls Remove() to change the status to "Delete". Now the SQL DELETE command executes, when the *SaveChanges()* method runs.

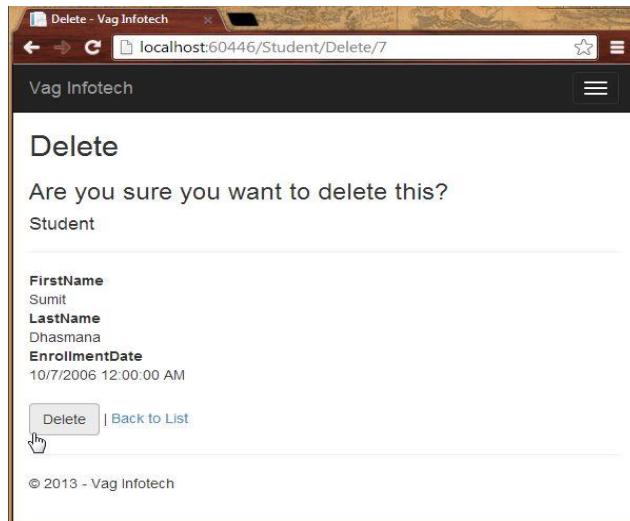
Step 3: Open the *Views\Student\Delete.cshtml* page and modify it with the highlighted code below:

```

<h2>Delete</h2>
<p class="error">@ViewBag.ErrorMessage</p>
<h3>Are you sure you want to delete this?</h3>

```

Step 4: Run the application and open "Students" link. Click on the "Delete" link to delete the data.



Step 5: When in the conformation page and you click on Delete, the data is deleted and the Index page is displayed.

Chapter 7: Paging, Searching and Sorting in ASP.Net MVC 5

7.1 Introduction

From my explanation in my [CRUD in ASP.NET MVC 5](#), you are now able to do basic CRUD operations MVC applications. This section explains how to do sorting, searching and paging in a MVC 5 application with Entity Framework 6 in Visual Studio 2013.

In that context we'll perform the paging and sorting for the Student entity and it'll be displayed in the Student's Index page. In the following chapter you will see how sorting works by clicking the headings. The headings are the links to show the sorted data.

So, let's proceed with the following sections:

- Perform Sorting
- Perform Searching
- Perform Paging

Perform Sorting

Now in this section we will do the sorting of the Student entity. Please use the following procedure to do that.

7.2 Adding Sorting Functionality in Controller

Step 1: Open the *StudentController.cs* file and replace the *Index()* method with the code below:

```
public ActionResult Index(string Sorting_Order)
{
    ViewBag.SortingName = String.IsNullOrEmpty(Sorting_Order) ? "Name_Description" : "";
    ViewBag.SortingDate = Sorting_Order == "Date_Enroll" ? "Date_Description" : "Date";

    var students = from stu in db.Students select stu;

    switch (Sorting_Order)
{
```

```

        case "Name_Description":
            students = students.OrderByDescending(stu=> stu.FirstName);
            break;
        case "Date_Enroll":
            students = students.OrderBy(stu => stu.EnrollmentDate);
            break;
        case "Date_Description":
            students = students.OrderByDescending(stu => stu.EnrollmentDate);
            break;
        default:
            students = students.OrderBy(stu => stu.FirstName);
            break;
    }
    return View(students.ToList());
}

```

In the code above, the *Sorting_Order* parameter is responsible for getting the value from the query string in the URL. The parameter is a string and it is either a "Name" or a "Date". By default the sorting order is ascending.

The students are displayed as an ascending order the first time by their First Name. There are two variables of ViewBag used here for configuring the column heading hyperlinks with the appropriate query string values.

7.3 Adding Heading Hyperlinks in View

Step 2: Open the *Views\Student\Index.cshtml* page and modify it with the highlighted code below:

```

<p>
    @Html.ActionLink("Create New", "Create")</p>

<table class="table">
    <tr>
        <th>
            @Html.ActionLink("First Name", "Index", new { Sorting_Order = ViewBag.SortingName })
        </th>
        <th>

```

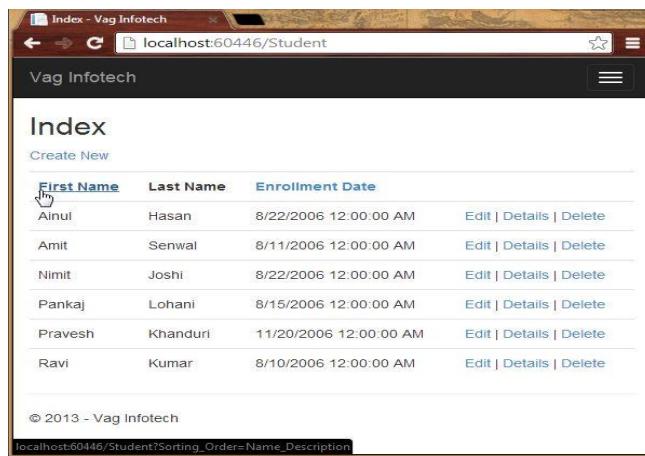
```

        Last Name
    </th>
<th>
    @Html.ActionLink("Enrollment Date", "Index", new { Sorting_Order = ViewBag.SortingDate })
</th>
<th></th>
</tr>

@foreach (var item in Model) {

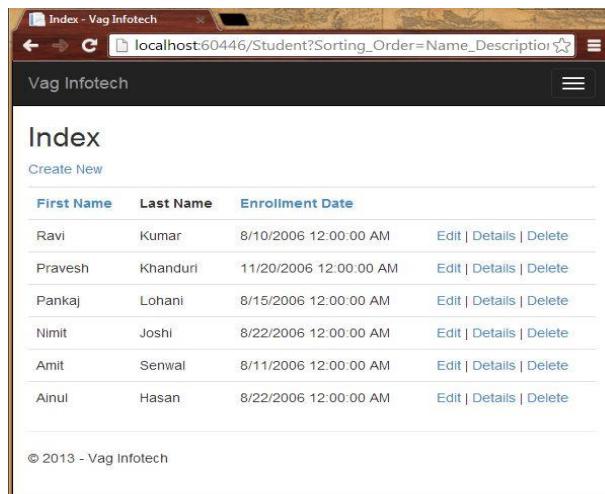
```

Step 3: Run the app and open Students.



First Name	Last Name	Enrollment Date	
Ainul	Hasan	8/22/2006 12:00:00 AM	Edit Details Delete
Amit	Senwal	8/11/2006 12:00:00 AM	Edit Details Delete
Nimit	Joshi	8/22/2006 12:00:00 AM	Edit Details Delete
Pankaj	Lohani	8/15/2006 12:00:00 AM	Edit Details Delete
Pravesh	Khanduri	11/20/2006 12:00:00 AM	Edit Details Delete
Ravi	Kumar	8/10/2006 12:00:00 AM	Edit Details Delete

Step 4: Now click on the First Name (heading) and you'll see the descending order of data.



First Name	Last Name	Enrollment Date	
Ravi	Kumar	8/10/2006 12:00:00 AM	Edit Details Delete
Pravesh	Khanduri	11/20/2006 12:00:00 AM	Edit Details Delete
Pankaj	Lohani	8/15/2006 12:00:00 AM	Edit Details Delete
Nimit	Joshi	8/22/2006 12:00:00 AM	Edit Details Delete
Amit	Senwal	8/11/2006 12:00:00 AM	Edit Details Delete
Ainul	Hasan	8/22/2006 12:00:00 AM	Edit Details Delete

Searching

To do the searching in the application we need to add a TextBox to enter the searching credentials and using button, we can show the corresponding record. So, let's proceed with the steps below.

7.4 Adding Searching Functionality in Controller

Step 1: Open the *StudentController.cs* file and modify the *Index()* method with the highlighted code below:

```
public ActionResult Index(string Sorting_Order, string Search_Data)
{
    ViewBag.SortingName = String.IsNullOrEmpty(Sorting_Order) ? "Name_Description" : "";
    ViewBag.SortingDate = Sorting_Order == "Date_Enroll" ? "Date_Description" : "Date";

    var students = from stu in db.Students select stu;

    {
        students = students.Where(stu => stu.FirstName.ToUpper().Contains(Search_Data.ToUpper()))
                           || stu.LastName.ToUpper().Contains(Search_Data.ToUpper()));
    }

    switch (Sorting_Order)
    {
        case "Name_Description":
            students = students.OrderByDescending(stu=> stu.FirstName);
            break;
        case "Date_Enroll":
            students = students.OrderBy(stu => stu.EnrollmentDate);
            break;
        case "Date_Description":
            students = students.OrderByDescending(stu => stu.EnrollmentDate);
            break;
        default:
            students = students.OrderBy(stu => stu.FirstName);
            break;
    }
}
```

```

    return View(students.ToList());
}

```

In the code above, we've added the *Search_Data* parameter and the LINQ statements. The where clause finds and selects only those student with a first name or last name containing the *Search_Data* value and the record is displayed in the Index view.

7.5 Adding Searching Button in View

Step 2: Open the *Views\Student\Index.cshtml* page and modify it with the highlighted code below:

```

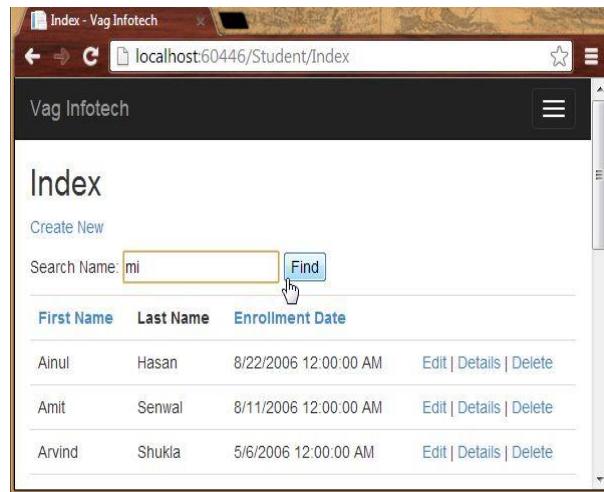
<p>
    @Html.ActionLink("Create New", "Create")
</p>

@using (Html.BeginForm())
{
    <p>
        Search Name: @Html.TextBox("Search_Data", ViewBag.FilterValue as string)
        <input type="submit" value="Find" />
    </p>
}

<table class="table">

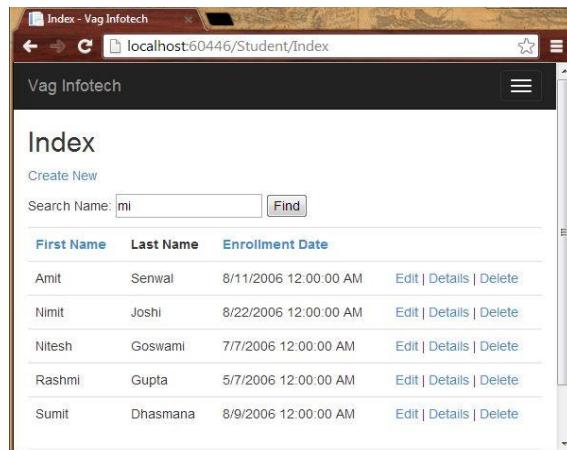
```

Step 3: Run the app and open Students and enter the value to search for.



First Name	Last Name	Enrollment Date
Ainul	Hasan	8/22/2006 12:00:00 AM
Amit	Senwal	8/11/2006 12:00:00 AM
Arvind	Shukla	5/6/2006 12:00:00 AM

The searched record:



First Name	Last Name	Enrollment Date
Amit	Senwal	8/11/2006 12:00:00 AM
Nimit	Joshi	8/22/2006 12:00:00 AM
Nitesh	Goswami	7/7/2006 12:00:00 AM
Rashmi	Gupta	5/7/2006 12:00:00 AM
Sumit	Dhasmana	8/9/2006 12:00:00 AM

You can also notice that the URL doesn't contain the searching value, in other words you cannot bookmark this page.

Perform Paging

We perform paging here by adding the NuGet Package named *PagedList.Mvc*. We add the links for paging in our *Student\Index.cshtml*. This NuGet Package is one of many good paging and sorting packages for ASP.NET MVC programming.

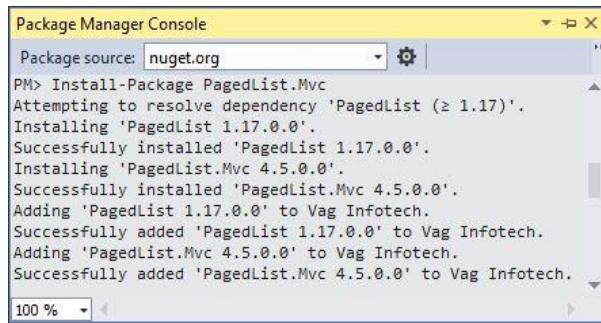
7.6 Adding NuGet Package

We install the `PagedList.Mvc` NuGet Package in the application that will automatically add a `PagedList` package. It has the `PagedList` collection type and extension methods for the `IQueryable` and `IEnumerable` collections to provide the paging. This NuGet Package is used to show the paging buttons.

Step 1: Open the Package Manager Console from the "Tools" -> "Library Package Manager".

Step 2: Enter the following command:

```
Install-Package PagedList.Mvc
```



7.7 Adding Paging Functionality

Step 3: Open the `StudentController.cs` file and the following namespace:

using `PagedList`;

Step 4: Modify the `Index()` with the highlighted code below:

```
public ActionResult Index(string Sorting_Order, string Search_Data, string Filter_Value, int? Page_No)
{
    ViewBag.CurrentSortOrder = Sorting_Order;
    ViewBag.SortingName = String.IsNullOrEmpty(Sorting_Order) ? "Name_Description" : "";
    ViewBag.SortingDate = Sorting_Order == "Date_Enroll" ? "Date_Description" : "Date";

    if (Search_Data != null)
    {
        // Search logic here
    }
}
```

```

        Page_No = 1;
    }
    else
    {
        Search_Data = Filter_Value;
    }

    ViewBag.FilterValue = Search_Data;

var students = from stu in db.Students select stu;

if (!String.IsNullOrEmpty(Search_Data))
{
    students = students.Where(stu => stu.FirstName.ToUpper().Contains(Search_Data.ToUpper())
        || stu.LastName.ToUpper().Contains(Search_Data.ToUpper()));
}

switch (Sorting_Order)
{
    case "Name_Description":
        students = students.OrderByDescending(stu=> stu.FirstName);
        break;
    case "Date_Enroll":
        students = students.OrderBy(stu => stu.EnrollmentDate);
        break;
    case "Date_Description":
        students = students.OrderByDescending(stu => stu.EnrollmentDate);
        break;
    default:
        students = students.OrderBy(stu => stu.FirstName);
        break;
}

int Size_Of_Page = 4;
int No_Of_Page = (Page_No ?? 1);
return View(students.ToPagedList(No_Of_Page, Size_Of_Page));
}

```

If you do not click on any paging or sorting link then the parameters value will be null.

7.8 Adding Paging Links in View

Step 5: Open the *Views\Student\Index.cshtml* page and modify it with the highlighted code below:

```

@model PagedList.IPagedList<Vag_Infotech.Models.Student>
@using PagedList.Mvc;
<link href="~/Content/PagedList.css" rel="stylesheet" />

 @{
    ViewBag.Title = "Students";
}

<h2>Students</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>

@using (Html.BeginForm("Index", "Student", FormMethod.Get))
{
    <p>
        Search Name: @Html.TextBox("Search_Data", ViewBag.FilterValue as string)
        <input type="submit" value="Find" />
    </p>
}

<table class="table">
    <tr>
        <th>
            @Html.ActionLink("First Name", "Index", new { Sorting_Order = ViewBag.SortingName, Filter_Value =
ViewBag.FilterValue })
        </th>
        <th>
            Last Name
        </th>
    
```

```

<th>
    @Html.ActionLink("Enrollment Date", "Index", new { Sorting_Order = ViewBag.SortingDate, Filter_Value =
ViewBag.FilterValue })
</th>
<th></th>
</tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.FirstName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.LastName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.EnrollmentDate)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.ID }) |
            @Html.ActionLink("Details", "Details", new { id=item.ID }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.ID })
        </td>
    </tr>
}

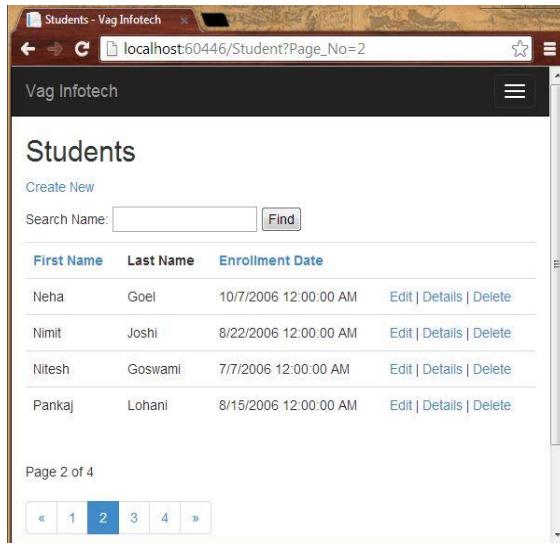
</table>
<br />
Page @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) of @Model.PageCount
@Html.PagedListPager(Model, Page_No =>Url.Action("Index",
    new { Page_No, Sorting_Order= ViewBag.CurrentSortOrder, Filter_Value = ViewBag.FilterValue }))

```

We've added the `@model` statement that specifies that the view now gets a *PagedList* object instead of a *List* object. The `using` statement is used to access the *PagedList.Mvc* that is useful for accessing the paging buttons.

The *PagedListPager* helps to provide a number of options that is helpful for customization including URLs and styling.

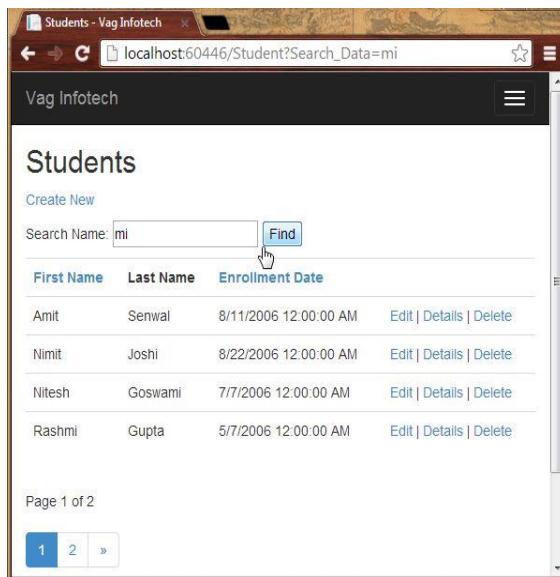
Step 6: Run the application.



First Name	Last Name	Enrollment Date	Action
Neha	Goel	10/7/2006 12:00:00 AM	Edit Details Delete
Nimit	Joshi	8/22/2006 12:00:00 AM	Edit Details Delete
Nitesh	Goswami	7/7/2006 12:00:00 AM	Edit Details Delete
Pankaj	Lohani	8/15/2006 12:00:00 AM	Edit Details Delete

In the preceding screenshot I've clicked on 2.

Step 7: Now search for a string.



First Name	Last Name	Enrollment Date	Action
Amit	Senwal	8/11/2006 12:00:00 AM	Edit Details Delete
Nimit	Joshi	8/22/2006 12:00:00 AM	Edit Details Delete
Nitesh	Goswami	7/7/2006 12:00:00 AM	Edit Details Delete
Rashmi	Gupta	5/7/2006 12:00:00 AM	Edit Details Delete

Chapter 8: Perform Code First Migration in ASP.Net MVC 5

8.1 Introduction

As you know, we can perform CRUD operations in MVC 5 with an Entity Framework Model and we also work in [Paging & Searching in MVC 5](#). Now, suppose we want to update the database with new details in the Students, Courses and Enrollments entities, then we need to migrate our application.

In that context, you'll learn today to do the Code First Migration in the MVC application. The migration helps you to update and change the data model that you've created for the application and you don't need to re-create or drop your database. So, let's proceed with the following section.

8.2 Apply Code First Migration

The data model in the database usually changes after developing the new application. When the data model changes, it becomes out of sync with the database. We've configured our Entity Framework in the app to drop and recreate the data model automatically. When any type of change is done in the entity classes or in the *DbContext* class, the existing database is deleted and the new database is created that matches the model and sees it with test data.

When the application runs in production, we do not want to lose everything each time we make changes like adding a column. The Code First Migration solved this problem by enabling the code first to update the database instead of dropping or recreating.

Use the following procedure to create a sample of that.

Step 1: Please disable the Database Initializer in the Web.Config file or in the *Global.asax* file like as in the following:

In *Web.Config* File:

```
<!--<contexts>

<context type="Vag_Infotech.DAL.CollegeDbContext, Vag_Infotech">
<databaseInitializer type="Vag_Infotech.DAL.CollegeDatabaseInitializer, Vag_Infotech"></databaseInitializer>
```

```
</context>
</contexts>-->
```

In *Global.asax* File:

```
// Database.SetInitializer(new CollegeDatabaseInitializer());
```

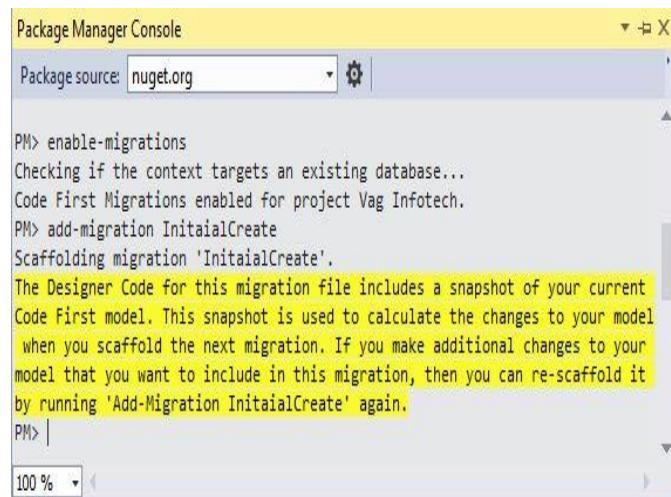
Step 2: Please change the database name in the Web.Config file like as in the following:

```
<connectionStrings>
<add name="CollegeDbContext" connectionString="Data Source=(LocalDb)\v11.0;Initial
Catalog=VInfotechNew;Integrated Security=SSPI" providerName="System.Data.SqlClient" />
</connectionStrings>
```

Step 3: Open the Package Manager Console and enter the following commands:

enable-migration

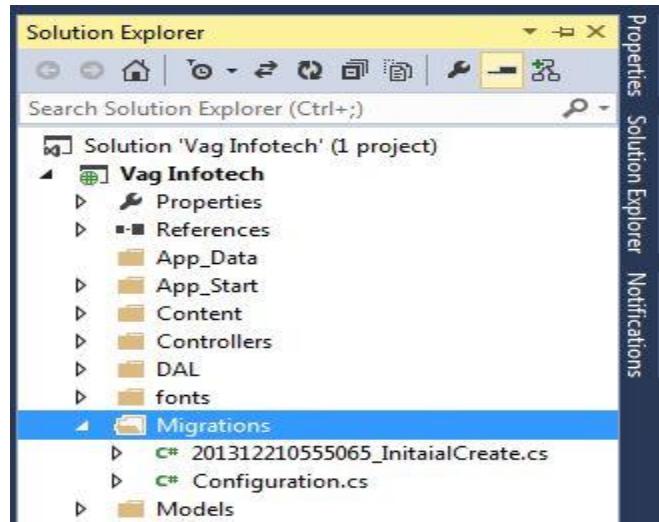
add-migration InitialCreate



The screenshot shows the Package Manager Console window in Visual Studio. The command 'enable-migrations' is run, followed by 'add-migration InitialCreate'. A detailed message about scaffolding is displayed, explaining that it creates a snapshot of the current model for future migrations.

```
Package Manager Console
PM> enable-migrations
Checking if the context targets an existing database...
Code First Migrations enabled for project Vag Infotech.
PM> add-migration InitialCreate
Scaffolding migration 'InitialCreate'.
The Designer Code for this migration file includes a snapshot of your current
Code First model. This snapshot is used to calculate the changes to your model
when you scaffold the next migration. If you make additional changes to your
model that you want to include in this migration, then you can re-scaffold it
by running 'Add-Migration InitialCreate' again.
PM> |
```

It'll create the Migration folder in your application and *Configuration.cs* file in it automatically.



You can see the *Seed()* created in the file and the purpose of this method is to enable the insert or change data after Code First creates or updates the database.

8.3 Editing Seed Method

Now, we add new code for updating our data model in this method. When we do not use the migration, the data model is dropped or re-created during the execution of the application each time, but now the data is retained after database changes.

Step 1: Modify your *Configuration.cs* with the following code:

```
namespace Vag_Infotech.Migrations
{
    using System;
    using System.Collections.Generic;
    using System.Data.Entity;
    using System.Data.Entity.Migrations;
    using System.Linq;
    using Vag_Infotech.Models;

    internal sealed class Configuration : DbMigrationsConfiguration<Vag_Infotech.DAL.CollegeDbContext>
    {
```

```

public Configuration()
{
    AutomaticMigrationsEnabled = false;
}

protected override void Seed(Vag_Infotech.DAL.CollegeDbContext context)
{
    var New_Students = new List<Student>
    {
        new Student{FirstName="Amit",LastName="Senwal",
                    EnrollmentDate=DateTime.Parse("2009-07-01")},
        new Student{FirstName="Nimit",LastName="Joshi",
                    EnrollmentDate=DateTime.Parse("2009-07-01")},
        new Student{FirstName="Pankaj",LastName="Lohani",
                    EnrollmentDate=DateTime.Parse("2009-07-05")},
        new Student{FirstName="Pravesh",LastName="Khanduri",
                    EnrollmentDate=DateTime.Parse("2009-10-01")},
        new Student{FirstName="Ainul",LastName="Hasan",
                    EnrollmentDate=DateTime.Parse("2010-07-01")},
        new Student{FirstName="Ravi",LastName="Kumar",
                    EnrollmentDate=DateTime.Parse("2010-07-10")},
        new Student{FirstName="Arvind",LastName="Shukla",
                    EnrollmentDate=DateTime.Parse("2009-09-01")},
        new Student{FirstName="Sumit",LastName="Dhasmana",
                    EnrollmentDate=DateTime.Parse("2009-08-05")},
    };
    New_Students.ForEach(ns => context.Students.AddOrUpdate(p => p.LastName, ns));
    context.SaveChanges();

    var New_Courses = new List<Course>
    {
        new Course{CourseID=201,Name="MCA",Credit=3},
        new Course{CourseID=202,Name="M.Sc.IT",Credit=2},
        new Course{CourseID=203,Name="M.Tech.CS",Credit=2},
        new Course{CourseID=204,Name="M.Tech.IT",Credit=4}
    };
    New_Courses.ForEach(ns => context.Courses.AddOrUpdate(p => p.Name, ns));
}

```

```

context.SaveChanges();

var New_Enrollments = new List<Enrollment>
{
    new Enrollment{
        StudentID= New_Students.Single(ns=>ns.FirstName=="Amit").ID,
        CourseID=New_Courses.Single(nc=>nc.Name=="MCA").CourseID,
        Grade=StudentGrade.A
    },
    new Enrollment{
        StudentID= New_Students.Single(ns=>ns.FirstName=="Nimit").ID,
        CourseID=New_Courses.Single(nc=>nc.Name=="MCA").CourseID,
        Grade=StudentGrade.A
    },
    new Enrollment{
        StudentID= New_Students.Single(ns=>ns.FirstName=="Nimit").ID,
        CourseID=New_Courses.Single(nc=>nc.Name=="M.Tech.IT").CourseID,
        Grade=StudentGrade.B
    },
    new Enrollment{
        StudentID= New_Students.Single(ns=>ns.FirstName=="Pankaj").ID,
        CourseID=New_Courses.Single(nc=>nc.Name=="M.Sc.IT").CourseID,
        Grade=StudentGrade.A
    },
    new Enrollment{
        StudentID= New_Students.Single(ns=>ns.FirstName=="Pravesh").ID,
        CourseID=New_Courses.Single(nc=>nc.Name=="MCA").CourseID,
        Grade=StudentGrade.A
    },
    new Enrollment{
        StudentID= New_Students.Single(ns=>ns.FirstName=="Pravesh").ID,
        CourseID=New_Courses.Single(nc=>nc.Name=="M.Sc.IT").CourseID,
        Grade=StudentGrade.B
    },
    new Enrollment{
        StudentID= New_Students.Single(ns=>ns.FirstName=="Ainul").ID,
        CourseID=New_Courses.Single(nc=>nc.Name=="M.Tech.IT").CourseID,

```

```

Grade=StudentGrade.A
},
new Enrollment{
    StudentID= New_Students.Single(ns=>ns.FirstName=="Ravi").ID,
    CourseID=New_Courses.Single(nc=>nc.Name=="MCA").CourseID,
    Grade=StudentGrade.B
},
new Enrollment{
    StudentID= New_Students.Single(ns=>ns.FirstName=="Arvind").ID,
    CourseID=New_Courses.Single(nc=>nc.Name=="M.Tech.IT").CourseID,
    Grade=StudentGrade.B
},
new Enrollment{
    StudentID= New_Students.Single(ns=>ns.FirstName=="Sumit").ID,
    CourseID=New_Courses.Single(nc=>nc.Name=="MCA").CourseID,
    Grade=StudentGrade.A
},
new Enrollment{
    StudentID= New_Students.Single(ns=>ns.FirstName=="Sumit").ID,
    CourseID=New_Courses.Single(nc=>nc.Name=="M.Tech.IT").CourseID,
    Grade=StudentGrade.D
},
};

foreach (Enrollment et in New_Enrollments)
{
    var NewEnrollDb = context.Enrollments.Where(
        ns => ns.Student.ID == et.StudentID &&
        ns.Course.CourseID == et.CourseID).SingleOrDefault();
    if (NewEnrollDb == null)
    {
        context.Enrollments.Add(et);
    }
}
context.SaveChanges();
}

```

```
}
```

```
}
```

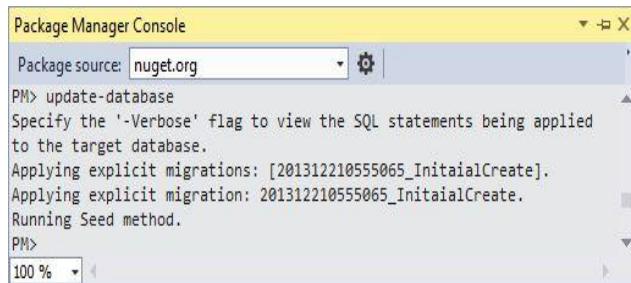
In the code above the *Seed()* takes the context object as an input argument and in the code the object is used to add the entities to the database. We can see how the code creates the collection for the database for each entity type and then saves it by *SaveChanges()* and adds it through the *DbSet* property. It is not necessary to call the *SaveChanges()* after each group of entities.

Step 2: Build the solution.

8.4 Migration Executing

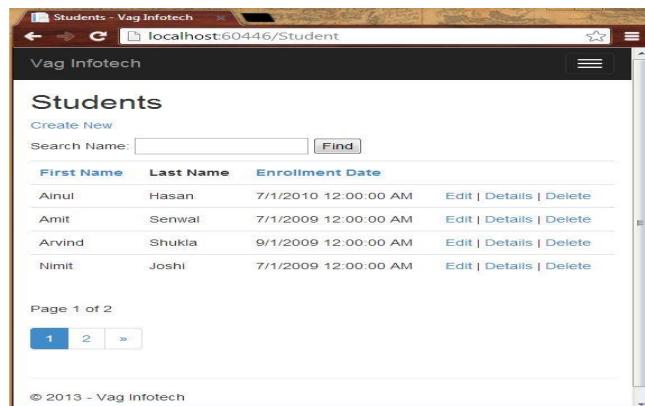
Step 1: Enter the following command in the Package Manager Console:

```
update-database
```



```
Package Manager Console
PM> update-database
Specify the '-Verbose' flag to view the SQL statements being applied
to the target database.
Applying explicit migrations: [201312210555065_InitiaialCreate].
Applying explicit migration: 201312210555065_InitiaialCreate.
Running Seed method.
PM>
100 %
```

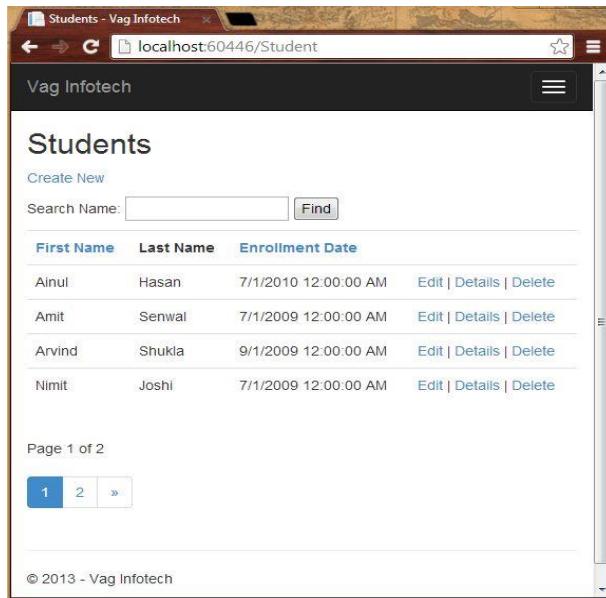
Step 2: Run the application and see the changes in the Students link.



Chapter 9: Performing Data Annotation in ASP.Net MVC 5

9.1. Introduction

This chapter will introduce you to DataAnnotation in MVC 5. In the previous chapter [Perform CRUD Functionality](#) we created some data for the Student entity but in the EnrollmentDate column the date comes with the time. The time is by default 12:00 AM. You can see that in the following screenshot:



First Name	Last Name	Enrollment Date
Ainul	Hasan	7/1/2010 12:00:00 AM
Amit	Senwal	7/1/2009 12:00:00 AM
Arvind	Shukla	9/1/2009 12:00:00 AM
Nimit	Joshi	7/1/2009 12:00:00 AM

Now, in that context we will validate the date time and other data type values for storing the right value and displaying the date without time or with the correct time. We can use the attributes for specifying the formatting, validation and database mapping rules. We can validate the date value in which the time is displayed along with the date and for that we need to make one code change that will fix it. So now let's proceed with the following sections:

- DataAnnotation with DataType Attribute
- DataAnnotation with StringLength Attribute
- DataAnnotation with Column Attribute

9.2 DataAnnotation with DataType Attribute

We can validate the date value in which the time is displayed along with the date and for that we need to make one code change that will fix it. Use the following procedure to do that.

Step 1: Open the *Models\Student.cs* file.

Step 2: Add the following reference:

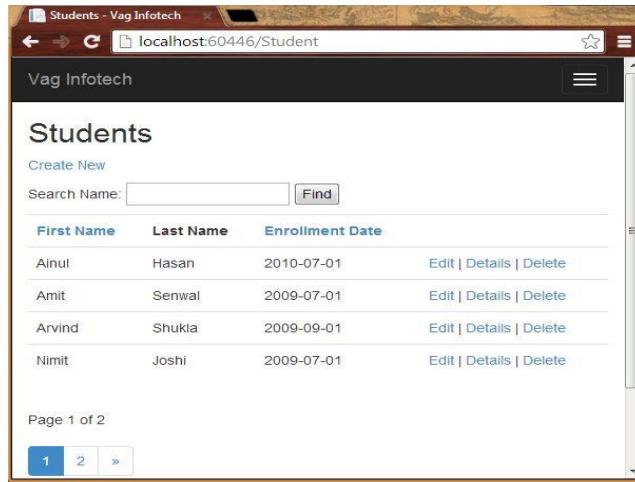
```
using System.ComponentModel.DataAnnotations;
```

Step 3: Modify your code with the highlighted code below:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
namespace Vag_Infotech.Models
{
    public class Student
    {
        public int ID { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
        public DateTime EnrollmentDate { get; set; }
        public virtual ICollection<Enrollment> Enrollments { get; set; }
    }
}
```

In the code above, you can see that we have applied the *DataType* DataAnnotation for the *EnrollmentDate* property of the *Student* entity. In this case you can see that we only want to show the date, not the full date and time. This *DataType* Enumeration provides many types of data types, like Date, Time, Currency, Email and so on. We can also enable the type-specific features that are automatically provided by the application. *ApplyFormatInEditMode* specifies that the specified formatting should also be applied when the value is displayed in a text box for editing.

Step 4: Now run the application and open the Students link



You can see in the preceding screenshot that the time is no longer displayed for the dates. It is applicable for every view that will use the Student entity.

DataAnnotation with StringLength Attribute

The *StringLength* attribute is used to apply the maximum length of the string and used to provide the client-side and server-side validation for the ASP.NET MVC application. The attributes are also used to provide the rules and error messages.

In that context, in your *Student.cs* file apply the following code with the highlighted text:

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

[StringLength(20, ErrorMessage="Your First Name can contain only 20 characters")]
public string FirstName { get; set; }

[StringLength(20)]
public string LastName { get; set; }
```

In the code above, you can see that the *FirstName* property has the *StringLength* attribute and in it we've applied that this can contain only 20 characters.

If you run the application and click on the Students link then you'll get error like:

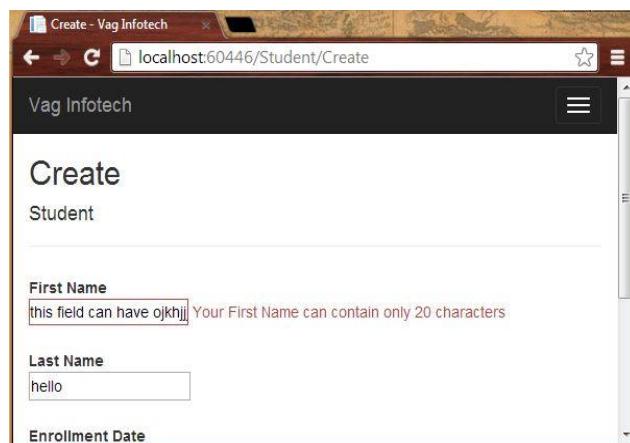
"The model backing the CollegeDbContext context has changed since the database was created. Consider using Code First Migrations to update the database"

So, go to the Package Manager Console and write the following command:

```
add-migration MaxLengthOnNames
```

```
update-database
```

Now, run the application and open the Students link:



9.3 DataAnnotation with Column Attribute

Now in this case we'll see how to use the attribute to change the column name and show the appropriate column name. Suppose we have the property named "FirstName" but you want that the name is to be the First Name, and then this attribute is useful.

This attribute specifies that when the database is created, the column of the Student table that maps to the *FirstName* property will be named *First Name*.

So, let's proceed with the following procedure.

Step 1: Open the *Student.cs* file.

Step 2: Modify your code with the highlighted code below.

At first add the following reference:

```
using System.ComponentModel.DataAnnotations.Schema;
```

The code:

```
[StringLength(20, ErrorMessage="Your First Name can contain only 20 characters")]
[Column("First Name")]
public string FirstName { get; set; }

[StringLength(20)]
[Column("Last Name")]
public string LastName { get; set; }
```

Final Code for Student Entity

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Vag_Infotech.Models
{
    public class Student
    {
        public int ID { get; set; }

        [Required]
        [StringLength(20, ErrorMessage="Your First Name can contain only 20 characters")]
        [Display (Name="First Name")]
        public string FirstName { get; set; }

        [Required]
        [StringLength(20, MinimumLength=2)]
        [Display(Name = "Last Name")]
        public string LastName { get; set; }

        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString="{0:yyyy-MM-dd}", ApplyFormatInEditMode=true)]
        [Display(Name="Enrollment Date")]
        public DateTime EnrollmentDate { get; set; }

        public string Name
        {
            get { return FirstName + "," + LastName; }
        }

        public virtual ICollection<Enrollment> Enrollments { get; set; }
    }
}
```

```
}
```

```
}
```

In the code above, we've applied the *Required* attribute for the columns that are required for the database. This attribute is not needed for the value types, such as int, double and DateTime. The value type fields are by default as required fields because they cannot have the null value. The *Display* attribute is used to specify the caption for the text boxes.

Chapter 10: Promoting MVC 4 and Web API Project to MVC 5 and Web API 2

10.1 Introduction

This chapter will introduce you to enhancing MVC 4 and Web API projects based Web Application to MVC 5 and Web API 2. As you already know, MVC 5 and Web API 2 have many new features like attribute routing, authentication filters and so on. I am giving you a brief description to upgrade the application to the most recent version.

So, let's proceed with the following sections:

- Working with MVC 4 Project
- Working with Visual Studio 2013
- Removing the MVC 4 Project GUID

10.2 Working with MVC 4 Project

Step 1: Please create a backup of your project. There are many changes made in the application with this walkthrough.

Step 2: If you want to upgrade the Web API to Web API 2, then open the Global.asax file and replace the following code:

```
WebApiConfig.Register(GlobalConfiguration.Configuration);
```

with the code below:

```
GlobalConfiguration.Configure(WebApiConfig.Register);
```

Step 3: You need to be sure that the packages that you are using are suitable with MVC 5 and Web API 2. I am presenting the following table that shows that the MVC 4 and Web API related packages than need to be changed. A few of the packages with their old version and new version are below:

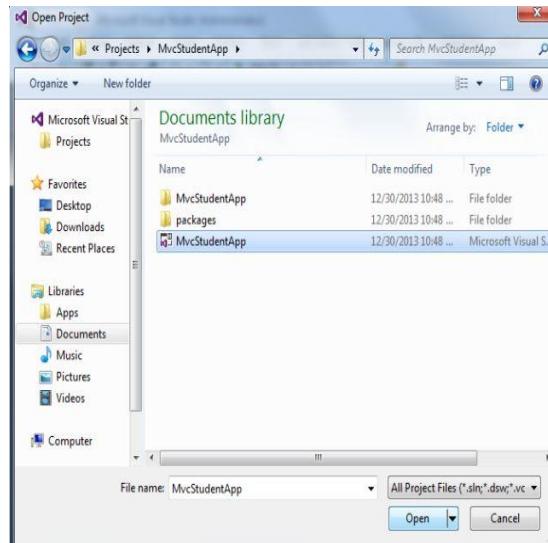
Package Id	Old Version	New Version
Microsoft.AspNet.Razor	2.0.x.x	3.0.0
Microsoft.AspNet.WebPages	2.0.x.x	3.0.0
Microsoft.AspNet.WebPages.OAuth	2.0.x.x	3.0.0
Microsoft.AspNet.Mvc	4.0.x.x	5.0.0
Microsoft.AspNet.Mvc.Facebook	4.0.x.x	5.0.0
Microsoft.AspNet.WebApi.Core	4.0.x.x	5.0.0
Microsoft.AspNet.WebApi	4.0.x.x	5.0.0
Microsoft.AspNet.Mvc.FixedDisplayModes		Removed
Microsoft.AspNet.WebPages.Administration		Removed

Note: MVC 5 is only compatible with Razor 3.

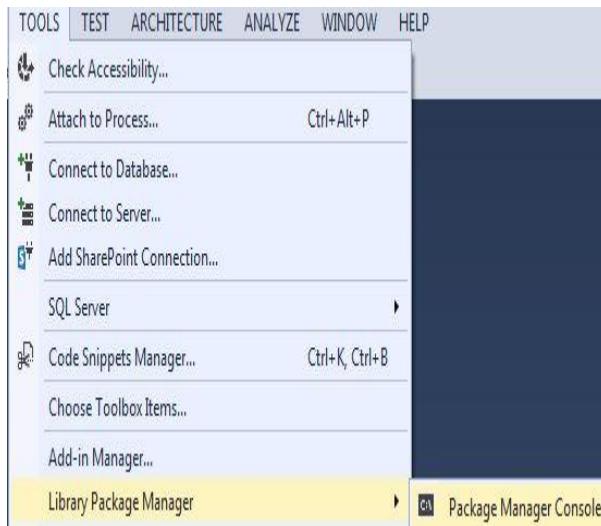
10.3 Working with Visual Studio 2013

Now we'll work with the Visual Studio 2013 (RTM Version). Please use the following procedure.

Step 1: Click on "Open Project" and select the existing MVC 4 application.



Step 2: Open the Package Manager Console.



Step 3: You remove the following NuGet Packages from the application. Your project might not include all of these.

1. *Microsoft.AspNet.WebPages.Administration*

This package is added, when upgrading from MVC 3 to MVC 4. To remove it enter the following command:

```
Uninstall-Package -Id Microsoft.AspNet.WebPages.Administration
```

2. *Microsoft-Web-Helpers*

This package is replaced with *Microsoft.AspNet.WebHelpers*. To remove it enter the following command:

```
Uninstall-Package -Id Microsoft-Web-Helpers
```

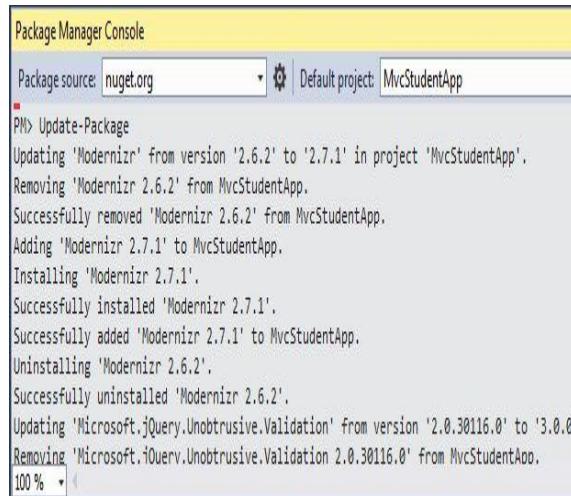
3. *Microsoft.AspNet.Mvc.FixedDisplayModes*

This package is used for working around a bug in MVC 4 that is fixed in MVC 5. To remove it enter the following command:

```
Uninstal-Package -Id Microsoft.AspNet.Mvc.FixedDisplayModes
```

Step 4: Now you update all current packages in your application. Enter the following command in the Package Manager Console (PMC):

Update-Package



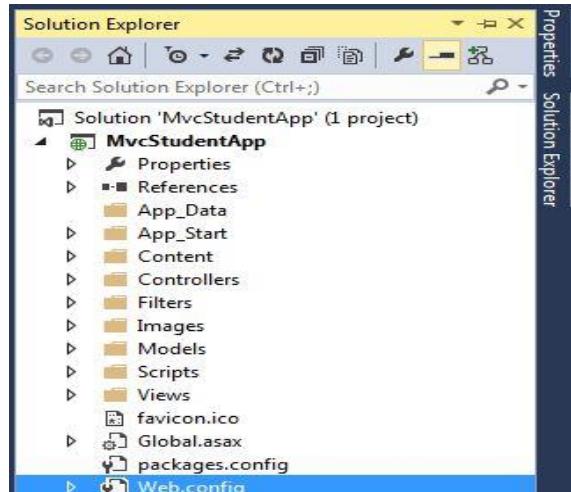
```
Package Manager Console
Package source: nuget.org | Default project: MvcStudentApp
PM> Update-Package
Updating 'Modernizr' from version '2.6.2' to '2.7.1' in project 'MvcStudentApp'.
Removing 'Modernizr 2.6.2' from MvcStudentApp.
Successfully removed 'Modernizr 2.6.2' from MvcStudentApp.
Adding 'Modernizr 2.7.1' to MvcStudentApp.
Installing 'Modernizr 2.7.1'.
Successfully installed 'Modernizr 2.7.1'.
Successfully added 'Modernizr 2.7.1' to MvcStudentApp.
Uninstalling 'Modernizr 2.6.2'.
Successfully uninstalled 'Modernizr 2.6.2'.
Updating 'Microsoft.jQuery.Unobtrusive.Validation' from version '2.0.30116.0' to '3.0.0'.
Removing 'Microsoft.jQuery.Unobtrusive.Validation 2.0.30116.0' from MvcStudentApp.
100 %
```

Note: You can update the package by giving the specific package name using the ID argument.

10.4 Updating the application Web.Config File

Now we'll make some changes in the *Web.Config* file of the application. So, proceed with the following procedure.

Step 1: Open the *Web.Config* file present in the root folder.



Step 2: Please change the old version to the new version of elements. So please update the code as shown below with the highlighted code:

```
<runtime>
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
<dependentAssembly>
<assemblyIdentity name="DotNetOpenAuth.Core" publicKeyToken="2780ccd10d57b246" />
<bindingRedirect oldVersion="0.0.0.0-4.3.0.0" newVersion="4.3.0.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity name="DotNetOpenAuth.AspNet" publicKeyToken="2780ccd10d57b246" />
<bindingRedirect oldVersion="0.0.0.0-4.3.0.0" newVersion="4.3.0.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
<bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
<bindingRedirect oldVersion="0.0.0.0-5.0.0.0" newVersion="5.0.0.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
<bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
</dependentAssembly>
```

Step 3: Update the AppSettings as shown below with the highlighted code:

```
<appSettings>
<add key="webpages:Version" value="3.0.0.0" />
<add key="webpages:Enabled" value="false" />
<add key="PreserveLoginUrl" value="true" />
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
```

Step 4: Remove any trust levels other than "Full". As an example:

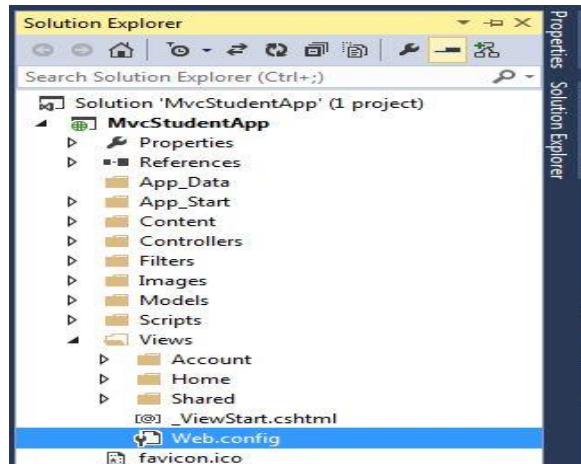
```
<securityPolicy>
<!--<trustLevel name="Medium" policyFile="web_mediumtrust.config"/><!--&gt;
&lt;/securityPolicy&gt;</pre>

```

11.5 Updating the Web.Config File in Views

Now we'll make some changes in the *Web.Config* file that exists in the Views Folder. So, proceed with the following procedure.

Step 1: Open the *Web.Config* file present in the Views folder



Step 2: Please change the old version to the new version of elements that contains *System.Web.Mvc*. So please update the code as shown below with the highlighted code:

```
<system.web.webPages.razor>
<host factoryType="System.Web.Mvc.MvcWebRazorHostFactory, System.Web.Mvc, Version=5.0.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
<pages pageBaseType="System.Web.Mvc.WebViewPage">
<namespaces>
```

and

```
<pages>
  <validateRequest="false" />
  <pageParserFilterType="System.Web.Mvc.ViewTypeParserFilter, System.Web.Mvc, Version=5.0.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
```

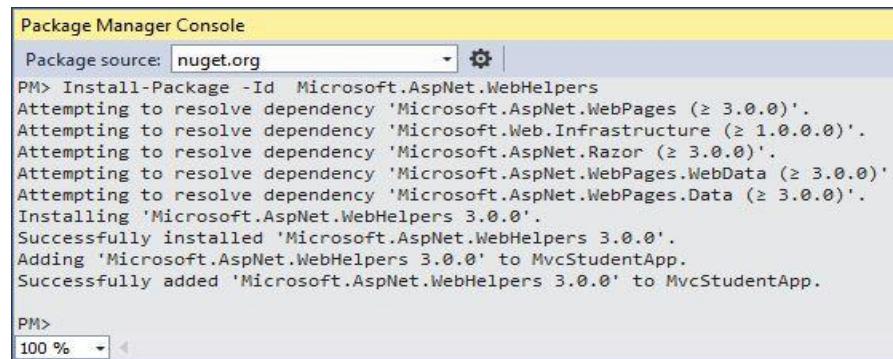
```
pageBaseType="System.Web.Mvc.ViewPage, System.Web.Mvc, Version=5.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"
    userControlBaseType="System.Web.Mvc.ViewUserControl, System.Web.Mvc, Version=5.0.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35">
<controls>
    <add assembly="System.Web.Mvc, Version=5.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" namespace="System.Web.Mvc" tagPrefix="mvc" />
</controls>
```

Step 3: Please change the old version to the new version of elements that contains *System.Web.WebPages.Razor*. So please update the code as shown below with the highlighted code:

```
<configuration>
    <configSections>
        <sectionGroup name="system.web.webPages.razor" type="System.Web.WebPages.Razor.Configuration.RazorWebSectionGroup, System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">
            <section name="host" type="System.Web.WebPages.Razor.Configuration.HostSection,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false" />
            <section name="pages" type="System.Web.WebPages.Razor.Configuration.RazorPagesSection,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false" />
        </sectionGroup>
    </configSections>
```

Step 4: Install the following NuGet Package, if you removed it:

```
Install-Package -Id Microsoft.AspNet.WebHelpers
```



```
Package Manager Console
Package source: nuget.org
PM> Install-Package -Id Microsoft.AspNet.WebHelpers
Attempting to resolve dependency 'Microsoft.AspNet.WebPages (>= 3.0.0)'.
Attempting to resolve dependency 'Microsoft.Web.Infrastructure (>= 1.0.0.0)'.
Attempting to resolve dependency 'Microsoft.AspNet.Razor (>= 3.0.0)'.
Attempting to resolve dependency 'Microsoft.AspNet.WebPages.WebData (>= 3.0.0)'.
Attempting to resolve dependency 'Microsoft.AspNet.WebPages.Data (>= 3.0.0)'.
Installing 'Microsoft.AspNet.WebHelpers 3.0.0'.
Successfully installed 'Microsoft.AspNet.WebHelpers 3.0.0'.
Adding 'Microsoft.AspNet.WebHelpers 3.0.0' to MvcStudentApp.
Successfully added 'Microsoft.AspNet.WebHelpers 3.0.0' to MvcStudentApp.

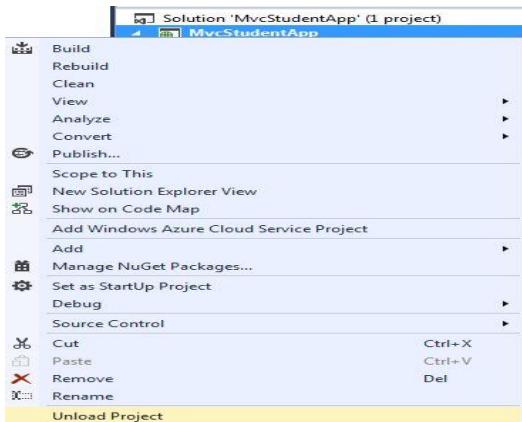
PM>
100 %
```

10.6 Removing the MVC 4 Project GUID

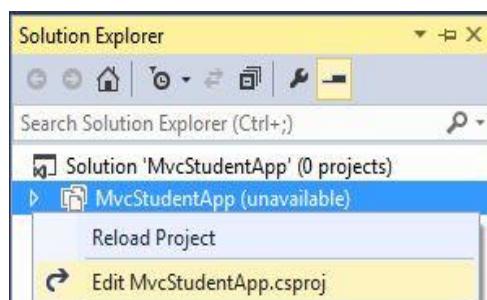
please build your solution.

Now to remove the MVC 4 Project GUID, use the following procedure.

Step 1: Just right-click on your project and select "Unload Project by Solution Explorer".



Step 2: Now, right-click to select "Edit ProjectName.csproj"



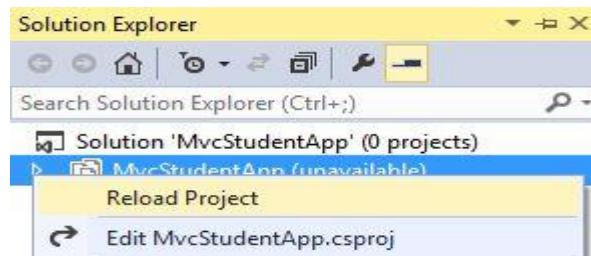
Step 3: Find the *ProjectTypeGuids* element and remove the MVC 4 project GUID as shown below with the highlighted text:

```
<ProjectTypeGuids>{E3E379DF-F4C6-4180-9B81-6769533ABE47};{349c5851-65df-11da-9384-  
00065b846f21};{fae04ec0-301f-11d3-bf4b-00c04f79efbc}</ProjectTypeGuids>
```

Remove the highlighted text in the element.

Step 4: Please save it and close the opened file.

Step 5: Finally right-click on the project to select "Reload Project".



Chapter 11: Getting Started with Enum Support in MVC 5 View

11.1 Introduction

Microsoft has released an updated version of Visual Studio 2013, Visual Studio 2013 Update 1. There are various updated features in this version. I have also described them in the [New Release Notes](#) of Visual Studio 2013.

In that context, let's see some new release notes of MVC 5.1 and Web API 2.1 in the following points:

- ASP.NET MVC 5.1
 - Attribute Routing
 - Bootstrap Support for editor templates
 - Enum Support in Views
 - Unobtrusive validation for Min/MaxLength Attributes
- ASP.NET Web API 2.1
 - Global error handling
 - Attribute routing
 - IgnoreRoute support
 - Better support for async filters
- ASP.NET Web Pages 3.1 Bug fixes

Getting Started

Now you are about to start to install this release and I've already mentioned in my previous chapter how to update your Visual Studio to the latest version. There are some features that require you to update your Visual Studio too but most of them are updated from the NuGet Package Manager. You might however update your Visual Studio. Its better to always update, right?

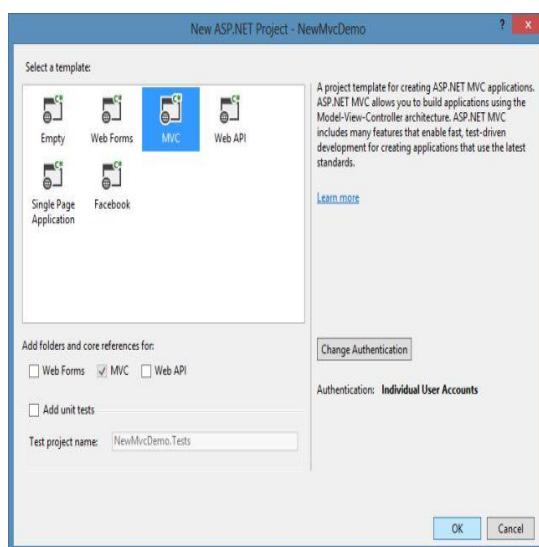
So, go to my previous chapter as I mentioned above in this chapter to update your Visual Studio, whether it is Visual Studio 2012 or Visual Studio 2013.

11.2 Working With MVC App

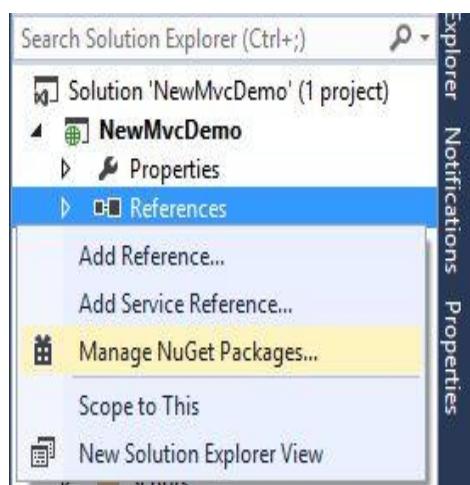
Now, I am developing an ASP.NET application using MVC Project Template to work with the Enum support in the Views. We can check out both aspects or say with Enum support or without Enum support.

So, let's get started with the following procedure.

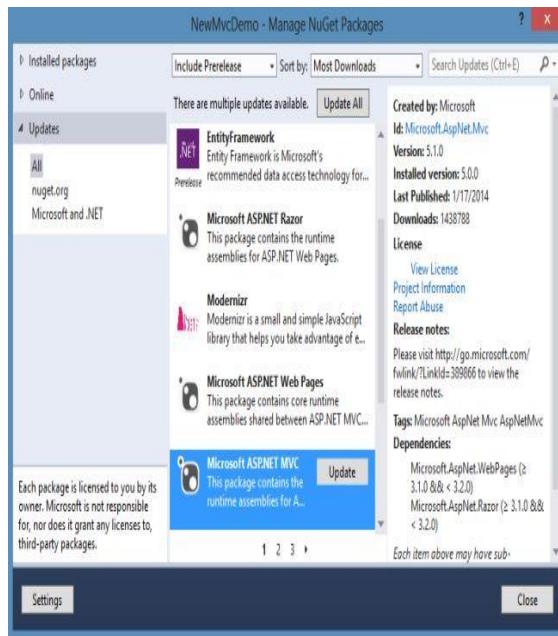
Step 1: Create an application with the MVC Project Template.



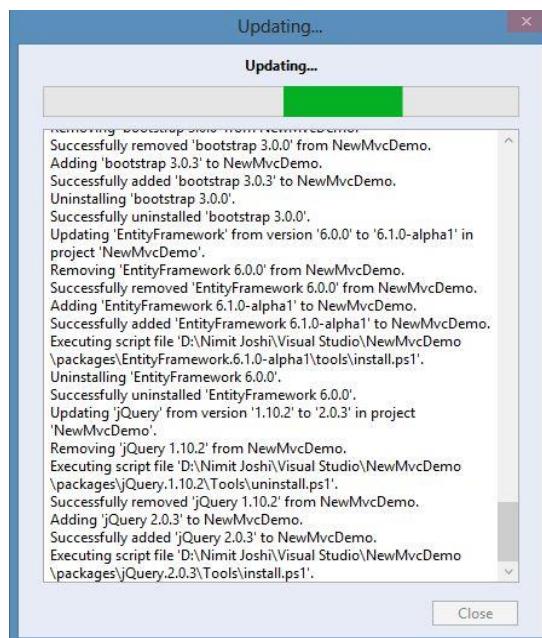
Step 2: In Solution Explorer, right-click on the project to open the Manage NuGet Packages.



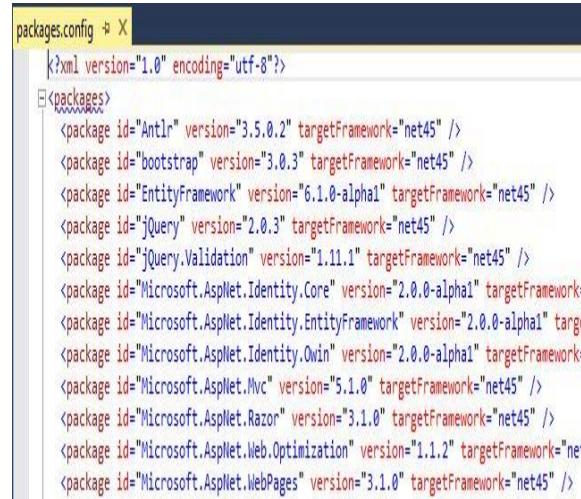
Step 3: There are various types of updates available here.



Note: I am updating all that is to be updated because this is not the real project. You can select only those updates that are relevant to your project to update.



After installing, you can see your *Packages.config* is also updated:



```

packages.config # X
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Antlr" version="3.5.0.2" targetFramework="net45" />
  <package id="bootstrap" version="3.0.3" targetFramework="net45" />
  <package id="EntityFramework" version="6.1.0-alpha1" targetFramework="net45" />
  <package id="jQuery" version="2.0.3" targetFramework="net45" />
  <package id="jQuery.Validation" version="1.11.1" targetFramework="net45" />
  <package id="Microsoft.AspNet.Identity.Core" version="2.0.0-alpha1" targetFramework="net45" />
  <package id="Microsoft.AspNet.Identity.EntityFramework" version="2.0.0-alpha1" targetFramework="net45" />
  <package id="Microsoft.AspNet.Identity.Owin" version="2.0.0-alpha1" targetFramework="net45" />
  <package id="Microsoft.AspNet.Mvc" version="5.1.0" targetFramework="net45" />
  <package id="Microsoft.AspNet.Razor" version="3.1.0" targetFramework="net45" />
  <package id="Microsoft.AspNet.Web.Optimization" version="1.1.2" targetFramework="net45" />
  <package id="Microsoft.AspNet.WebPages" version="3.1.0" targetFramework="net45" />

```

11.3 Creating Enum Support on Model

Step 1: Now create a class in the model named *Student*. Use the code below:

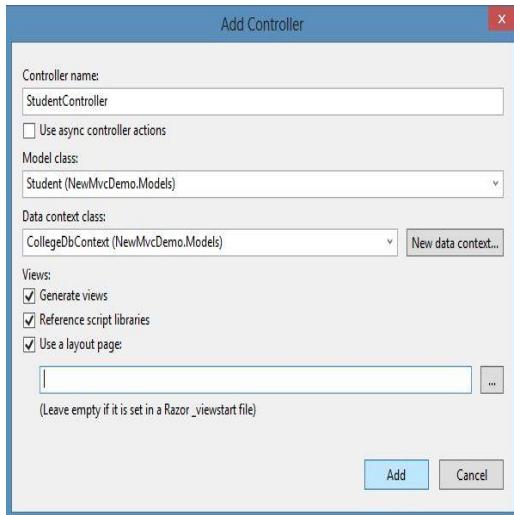
```

namespace NewMvcDemo.Models
{
    public enum Gender
    {
        Male,
        Female
    }
    public class Student
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public Gender Sex { get; set; }
        public string City { get; set; }
        public string State { get; set; }
    }
}

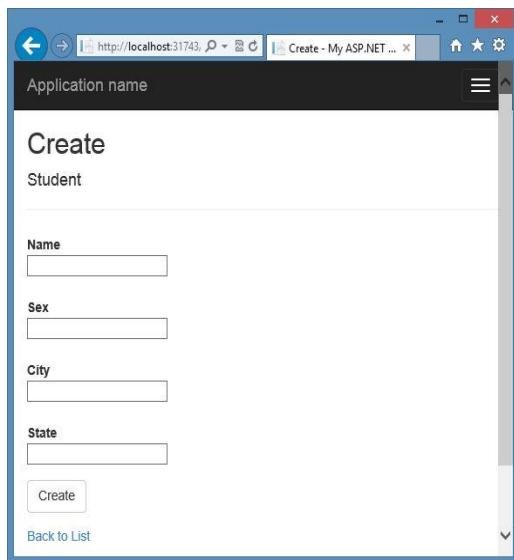
```

In the code above, you can see that I've created some properties related to the Student entity and also generated an Enum property for the Gender for the Sex property.

Step 2: Scaffold the model with a new controller.



Step 3: Now run the application and open the controller.

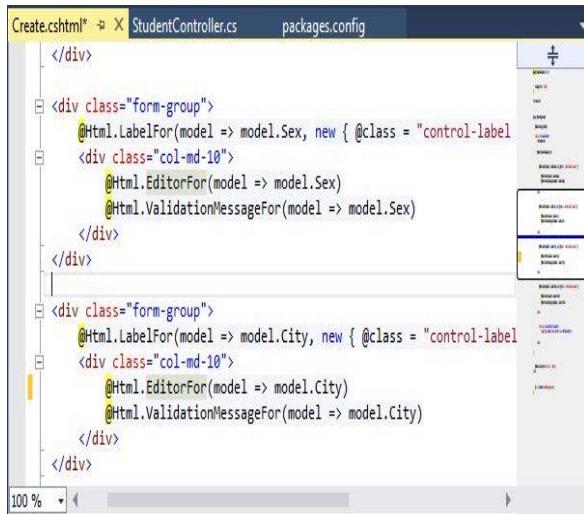


Hey, where are the Enum values (Male, Female)? Just hang on. We do not update the view(*Create.cshtml*). Proceed to the next section

11.4 Creating Enum Support on a View

In this section we'll create the Enum support for the view using the following procedure.

Step 1: As you can see that the scaffolded view uses the general code for the Sex Property, in other words `@Html.EditorFor`.

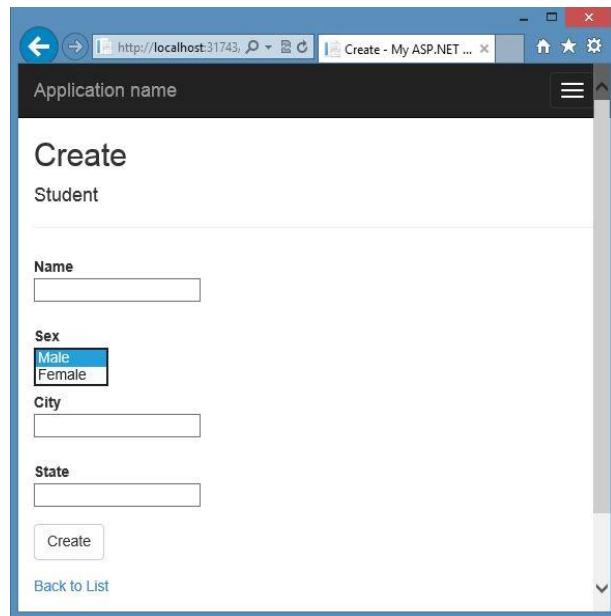


Step 2: Now we'll change it with the following code:

```
<div class="col-md-10">
    @Html.EnumDropDownListFor(model => model.Sex)
    @Html.ValidationMessageFor(model => model.Sex)
</div>
```

That's it. We've added Enum support for the View

Step 3: Ok, now run the application and open the controller



The screenshot shows a Windows-style application window titled "Create - My ASP.NET ...". The main title bar says "Application name". The window content is titled "Create" and "Student". It contains the following fields:

- Name: An input field.
- Sex: A dropdown menu with "Male" selected (highlighted in blue) and "Female" as an option.
- City: An input field.
- State: An input field.
- Create: A button.
- Back to List: A link.

Chapter 12: Working With DropDownList in MVC 5 Using jQuery

12.1 Introduction

Today we'll learn to work with the DropDownList in the MVC 5 using jQuery. Suppose you need to display the states stored in the DropDownList that are dependent upon on the selected value in another DropDownList named Country.

In that context, here we'll create this scenario in the ASP.NET Web Application based on the MVC 5 Project Template using jQuery. In this chapter you will learn to create the MVC application and work with the controller and view in it. We'll also use the Razor syntax in the MVC Views.

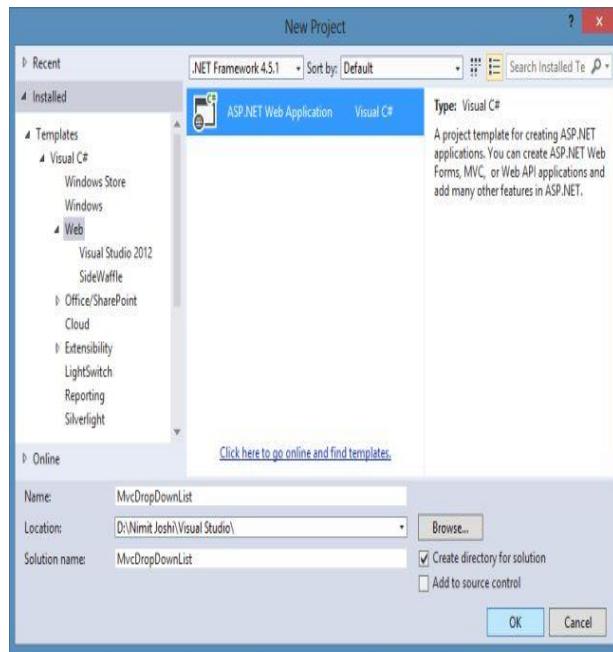
So, let's start with the following sections:

- Creating MVC application
- Working with a Controller
- Working with a View
- Running the application

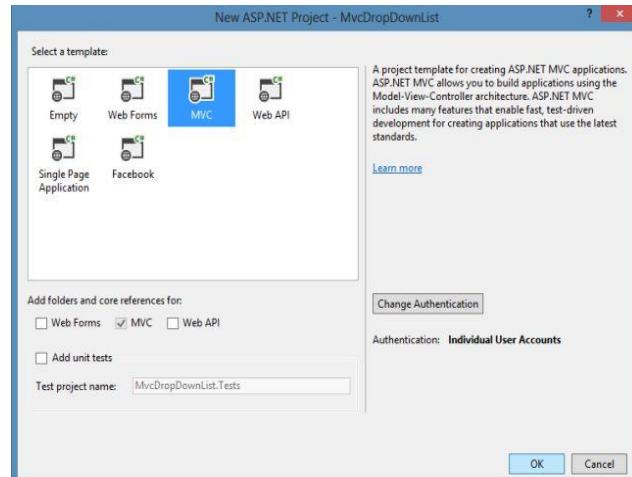
12.2 Creating MVC Application

In this section we'll create the ASP.NET MVC application using the following procedure.

Step 1: Open Visual Studio and click on "New Project" and enter the application name as "MvcUsers".



Step 2: Select the MVC Project Template.



That's it for the MVC application creation. Visual Studio automatically creates the MVC 5 application and adds files and folders to it.

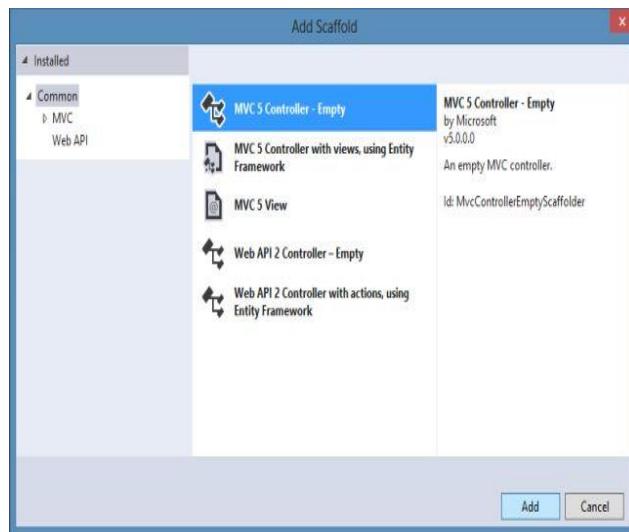
Note: I am using Visual Studio 2013 to develop the MVC 5 application. You can create this using MVC 4 also.

12.3 Working with Controller

Now we'll add the new scaffolded MVC empty controller here using the following procedure.

Step 1: Just right-click on the Controllers folder and add a New Scaffolded item.

Step 2: Select MVC Controller- Empty.



Step 3: Enter the controller name as *SampleController* and modify the *Index()* with the code below:

```
public ActionResult Index()
{
    List<string> ListItems = new List<string>();
    ListItems.Add("Select");
    ListItems.Add("India");
    ListItems.Add("Australia");
    ListItems.Add("America");
    ListItems.Add("South Africa");
    SelectList Countries = new SelectList(ListItems);
    ViewData["Countries"] = Countries;
    return View();
}
```

In the code above you can see that the `ListItems` is a generic string that holds the country names. The `DropDownList` Html helper in MVC View displays its data in the form of a `SelectList` object. The object of `SelectList` is passed to the view using the Countries ViewData variable.

Step 4: Add another method named `States()` in the same controller with the following code:

```
public JsonResult States(string Country)
{
    List<string> StatesList = new List<string>();
    switch (Country)
    {
        case "India":
            StatesList.Add("New Delhi");
            StatesList.Add("Mumbai");
            StatesList.Add("Kolkata");
            StatesList.Add("Chennai");
            break;
        case "Australia":
            StatesList.Add("Canberra");
            StatesList.Add("Melbourne");
            StatesList.Add("Perth");
            StatesList.Add("Sydney");
            break;
        case "America":
            StatesList.Add("California");
            StatesList.Add("Florida");
            StatesList.Add("New York");
            StatesList.Add("Washington");
            break;
        case "South Africa":
            StatesList.Add("Cape Town");
            StatesList.Add("Centurion");
            StatesList.Add("Durban");
            StatesList.Add("Jahannesburg");
            break;
    }
}
```

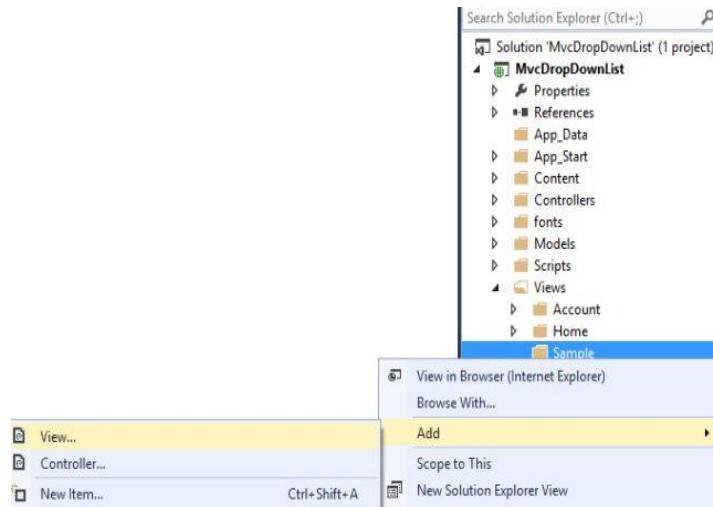
```
    return Json(StatesList);
}
```

In the preceding *States()* you can see that this accepts the country name and returns the StatesList as JsonResult. This method returns the JsonResult because this will be called using the jQuery. This method returns the states that are based on the country value. Finally, the states generic list is returned to the caller using the *Json()* method.

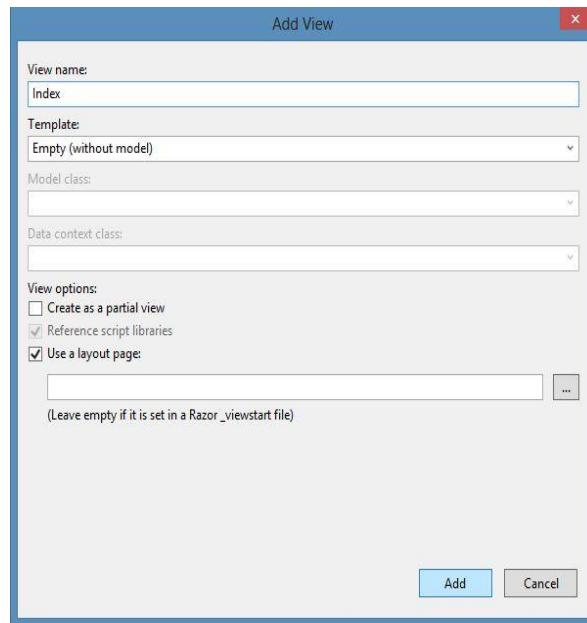
12.4 Working With View

We've completed the work with the controller, now its time to work with the View. Follow the procedure described below.

Step 1: At first we've add the View for the *Index()* as defined in the *SampleController.cs*. So, right-click on the Sample folder to add a View.



Step 2: Enter the View name as Index.



Step 3: Replace the code with the code below:

```

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>
<script src="~/Scripts/jquery-1.10.2.js"></script>
<script>
    $(document).ready(function () {
        $("#State").prop("disabled", true);
        $("#Country").change(function () {
            if ($("#Country").val() != "Select") {
                var CountryOptions = {};
                CountryOptions.url = "/Sample/states";
                CountryOptions.type = "POST";
                CountryOptions.data = JSON.stringify({ Country: $("#Country").val() });
                CountryOptions.datatype = "json";
                CountryOptions.contentType = "application/json";
                CountryOptions.success = function (StatesList) {
                    $("#State").empty();
                    for (var i = 0; i < StatesList.length; i++) {
                        ...
                    }
                }
            }
        })
    })
}

```

```

        $($("#State").append("<option>" + StatesList[i] + "</option>");
    }
    $($("#State")).prop("disabled", false);
};

CountryOptions.error = function () { alert("Error in Getting States!!"); };
$.ajax(CountryOptions);
}

else {
    $($("#State")).empty();
    $($("#State")).prop("disabled", true);
}
});

});

</script>

```

```

@using (Html.BeginForm("Index", "Sample", FormMethod.Post))
{
    @Html.AntiForgeryToken()
<h4>Select Country & States</h4>
<hr />
@Html.ValidationSummary()
<div class="form-group">
    @Html.Label("Select Country:", new { @class = "col-md-2 control-label" })
    <div class="col-md-10">
        @Html.DropDownList("Country", ViewData["Countries"] as SelectList, new { @class = "form-control" })
    </div>
</div><br />
<div class="form-group">
    @Html.Label("Select States:", new { @class = "col-md-2 control-label" })
    <div class="col-md-10">
        <select id="State"></select>
    </div>
</div>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" class="btn btn-default" value="Submit" />
    </div>
</div>

```

```
</div>
}
```

In the code above, I used the Razor syntax to show the content in the View. It includes the DropDownList named Country that is rendered by the DropDownList Html helper. The first parameter of this helper represents the name of the DropDownList, the second is the *SelectList* object that contains the DropDownList values which ID is State.

The jQuery reference is added here manually and inside the script tag the *ready()* handler the first code will disable the DropDownList using the *prop()*. The *change()* handler function at first checks whether the value is selected in the Country and if it is other than "Please Select", the code creates an *CountryOptions* object. The *CountryOptions* object holds various settings for the Ajax request to be made to the server for retrieving the State values. There are various properties defined here such as URL that points to the *States()* method, the type is set to post that indicates that a post method is used whereas the request and data contains the JSON representation of the Country.

12.5 Running the Application

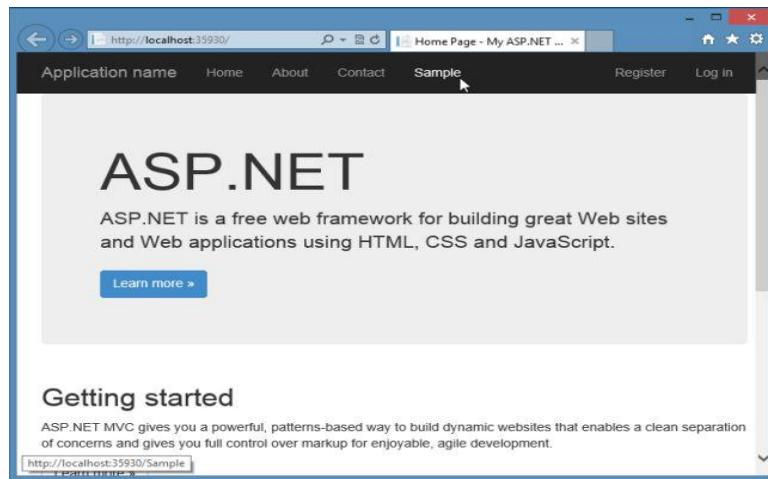
In this section we'll run the application and view the Sample Index view using the following procedure.

Step 1: At first open the *Views\Shared_Layout.cshtml* and modify the code with the highlighted code below:

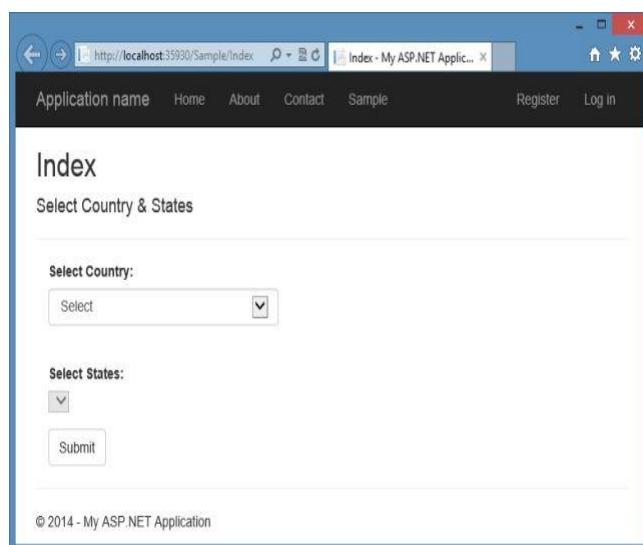
```
<li>@Html.ActionLink("Home", "Index", "Home")</li>
<li>@Html.ActionLink("About", "About", "Home")</li>
<li>@Html.ActionLink("Contact", "Contact", "Home")</li>
<li>@Html.ActionLink("Sample", "Index", "Sample")</li>
```

In the code above, we've added the Sample link for the SampleController in the Home Page of the application.

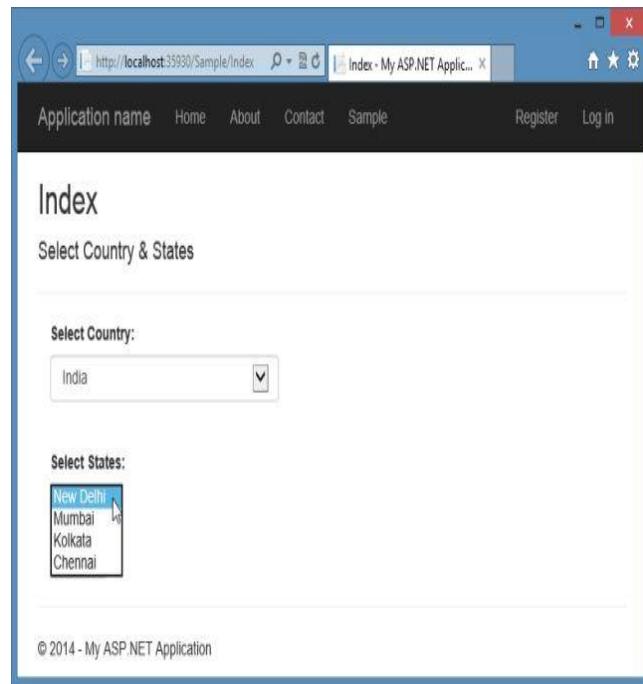
Step 2: Press Ctrl+F5 or F5 to run the application.



Step 3: You can see the Index page containing the DropDownList



Step 4: Now select the country and then the states from the next DropDownList as shown below:



After clicking on "Submit" the default values will display.

Chapter 13: Getting Started with Wizard in ASP.Net MVC 5: Part 1

13.1 Introduction

Today we'll learn to work with the wizard interface in ASP.NET MVC Web Applications. A wizard is used to allow to logically divide and group the data so that any user can enter the data or information gently in a step-by-step manner. In this chapter you'll see that the implementation of the wizard is so easy in the ASP.NET MVC Application and you can also develop it in the Web Forms application. There are various way to create the wizard in the MVC application and here you can learn the one among them.

In that context, I am creating the wizard in the MVC Web Application in Part 1 of this chapter and here you will learn to develop the wizard that stores the data in the ASP.NET Session and wizard works on the traditional form submission.

You will see the following approach to develop the wizard in the application:

- Each step of the wizard has the action method and the corresponding view.
- The data is stored in a view model class in each step.
- All action methods have the three parameters.
- The Action method gets the data and stores it in the Session until the final step.
- The action method returns the view for the next step if the "Next" button is clicked.
- The validations are checked after the "Next" button is clicked.

You now have some idea of the development of the application, let's develop the application using the following sections:

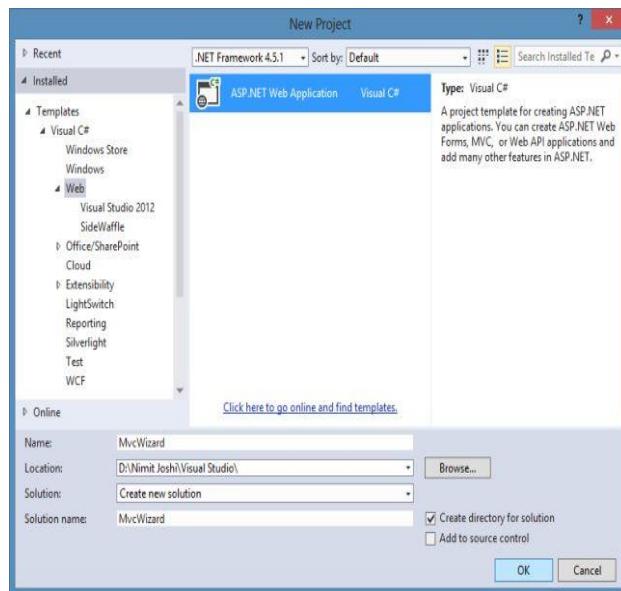
- Create MVC Web Application
- Working with Entity Data Model
- Working with Model
- Working with Controller
- Working with View
- Run the application

13.2 Create MVC Web Application

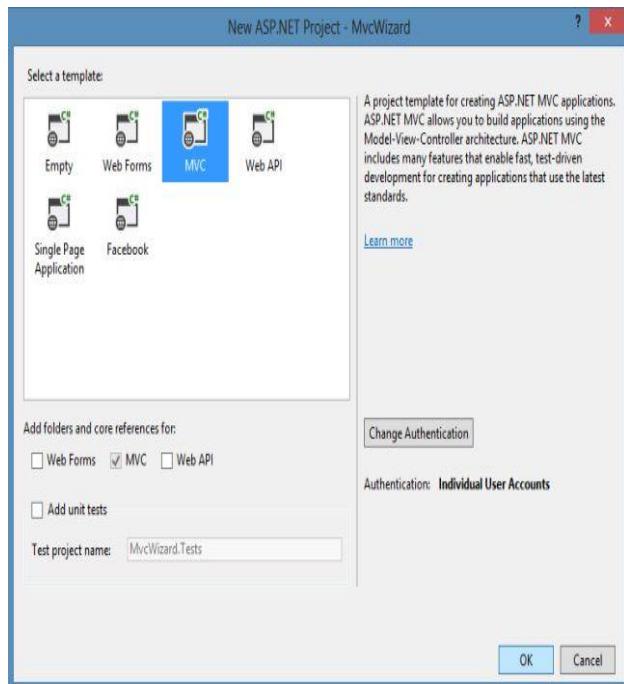
I am creating the ASP.NET Web Application in Visual Studio 2013 using MVC 5. Proceed with the following procedure.

Step 1: Open Visual Studio 2013 and click on "New Project".

Step 2: Select the ASP.NET Web Application and enter the name as "MVC Wizard".



Step 2: In the "One ASP.NET" wizard, select the "MVC" Project Template.

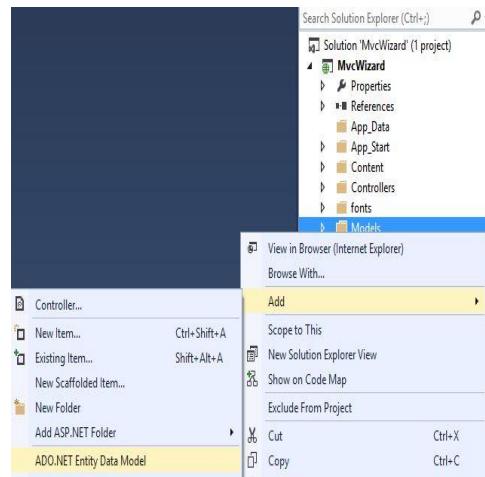


Visual Studio creates the application automatically and adds the files and folders to the application. Now proceed with the next section.

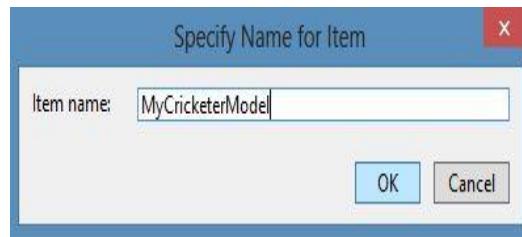
13.3 Working with Entity Data Model

In this section, we'll generate the entity data model from the database. I've created the table in the database and I assume that you have the table for generating the model or you can create a new one. So, let's proceed with the following procedure.

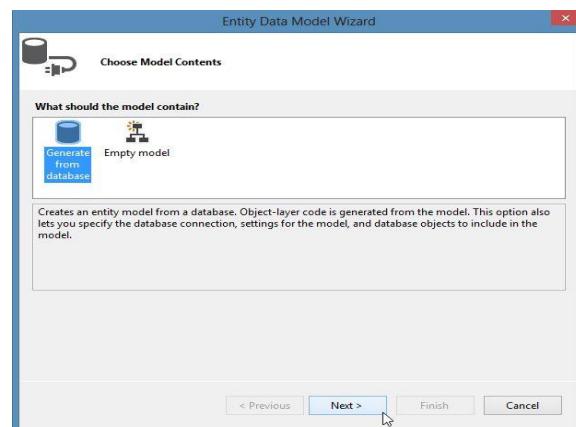
Step 1: In Solution Explorer, just right-click on the *Models* folder to add a ADO.NET Entity Data Model.



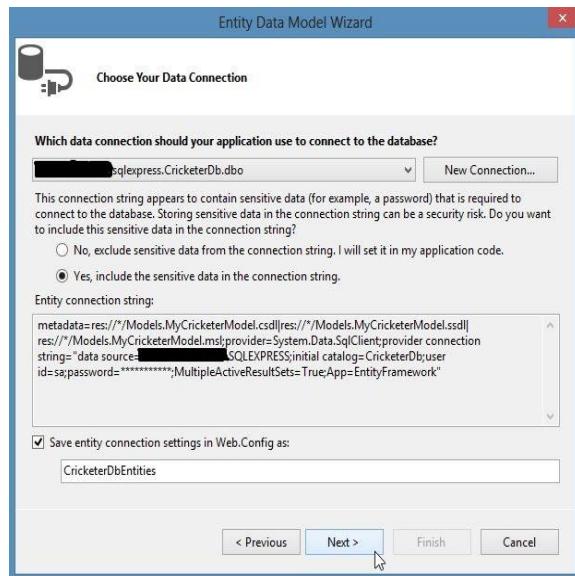
Step 2: Enter the name for the model as "*MyCricketerModel*".



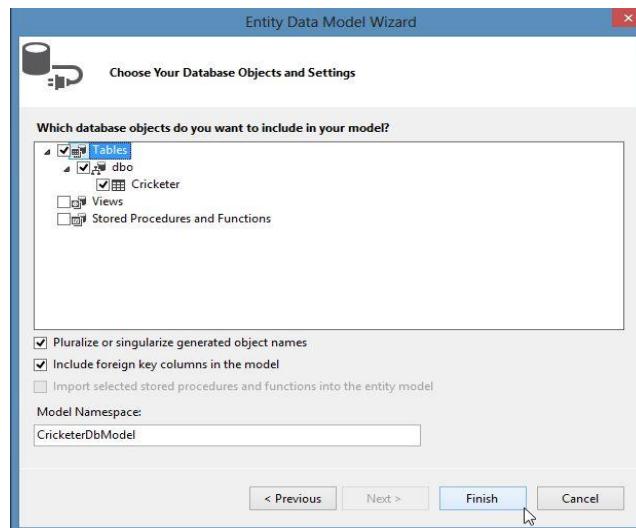
Step 3: Select the "Generate from Database" in the Entity Data Model Wizard.



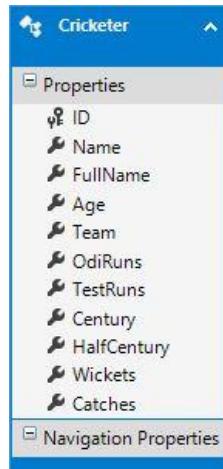
Step 4: Create a "New Connection" to connect with the database and click on "Next".



Step 5: Select the table of the database.



Visual Studio creates the Entity Data Model and generates the diagram of the table. As you can see in the "*Cricketer*" entity diagram below:



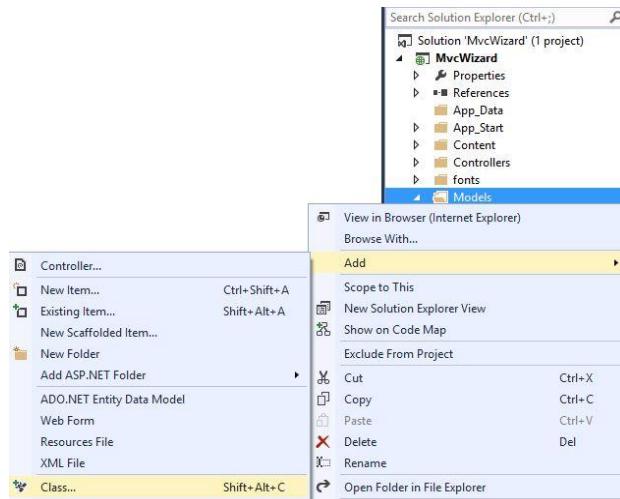
13.4 Working with the Model

Now as you can see in the preceding screenshot that there are various properties defined in the entity and for the sake of creating the wizard let's group them into the two steps as follows:

- Cricketer Details: ID, Name, FullName, Age and Team
- Cricketer Statistics: OdiRuns, TestRuns, Century, HalfCentury, Wickets and Catches

So, you need to create the model class for each of the wizard steps defined above. So let's work with the model with the following procedure.

Step 1: In Solution Explorer, just right-click on the *Models* folder to add a class as in the following:



Step 2: Enter the class name as "*CricketerDetails*" and replace the code with the code below:

```
using System.ComponentModel.DataAnnotations;
namespace MvcWizard.Models
{
    public class CricketerDetails
    {
        [Required]
        public int ID { get; set; }
        [Required]
        public string Name { get; set; }
        [Required]
        public string FullName { get; set; }
        [Required]
        public int Age { get; set; }
        [Required]
        public string Team { get; set; }
    }
}
```

Step 3: Add another class named "*CricketerStatistics*" and replace the code with the code below:

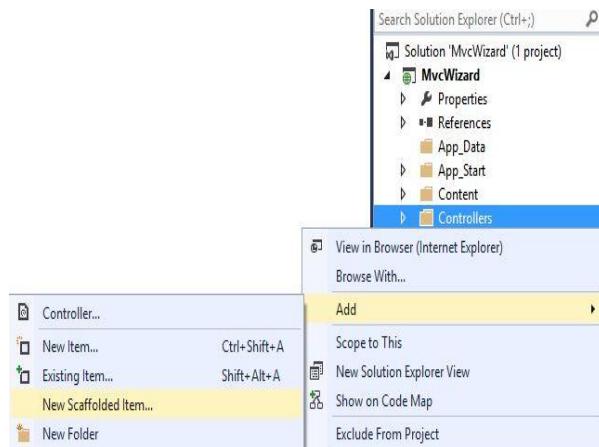
```
using System.ComponentModel.DataAnnotations;
namespace MvcWizard.Models
{
    public class CricketerStatistics
    {
        [Required]
        public int OdiRuns { get; set; }
        [Required]
        public int TestRuns { get; set; }
        [Required]
        public int Century { get; set; }
        [Required]
        public int HalfCentury { get; set; }
        [Required]
        public int Wickets { get; set; }
        [Required]
        public int Catches { get; set; }
    }
}
```

In both steps, we defined the properties for creating the steps in the wizard. We've also defined the *DataAnnotations* for the properties to validate. That's all for the model. Now we'll proceed with the next section.

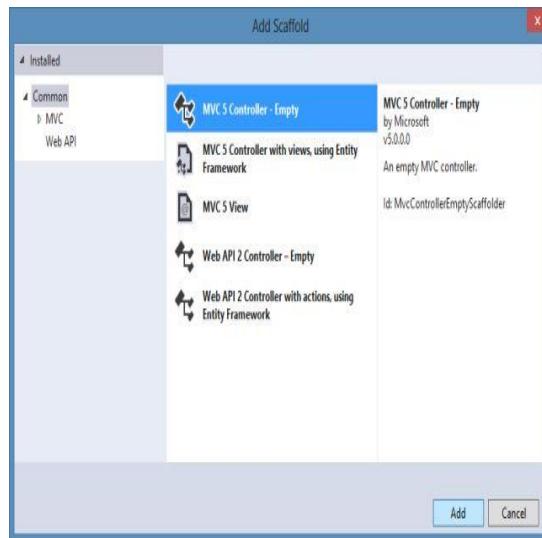
13.5 Working with Controller

In this section, we'll create the controller and create various action methods using the following procedure.

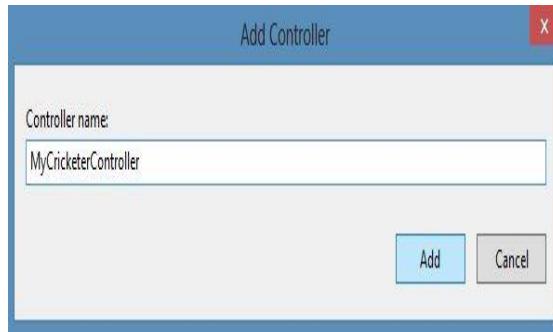
Step 1: In Solution Explorer, just right-click on the *Controllers* folder to add a "New Scaffolded Item".



Step 2: Select the "MVC Controller - Empty".



Step 3: Enter the controller name as "*MyCricketerController*".



Step 4: Just replace the code with the following code:

```
using MvcWizard.Models;
using System.Web.Mvc;
namespace MvcWizard.Controllers
{
    public class MyCricketerController : Controller
    {
        //
        // GET: /MyCricketer/
        public ActionResult Index()
```

```

{
    return View("CricketerDetails");
}
private Cricketer GetCricketer()
{
    if (Session["cricketer"] == null)
    {
        Session["cricketer"] = new Cricketer();
    }
    return (Cricketer)Session["cricketer"];
}
private void RemoveCricketer()
{
    Session.Remove("cricketer");
}
[HttpPost]
public ActionResult CricketerDetails(CricketerDetails DetailsData, string BtnPrevious, string BtnNext)
{
    if (BtnNext != null)
    {
        if (ModelState.IsValid)
        {
            Cricketer CricObj = GetCricketer();
            CricObj.ID = DetailsData.ID;
            CricObj.Name = DetailsData.Name;
            CricObj.FullName = DetailsData.FullName;
            CricObj.Age = DetailsData.Age;
            CricObj.Team = DetailsData.Team;
            return View("CricketerStatistics");
        }
    }
    return View();
}
[HttpPost]
public ActionResult CricketerStatistics(CricketerStatistics StatisticsData, string BtnPrevious, string BtnNext)
{
    Cricketer CricObj = GetCricketer();
}

```

```

if (BtnPrevious != null)
{
    CricketerDetails DetailsObj = new CricketerDetails();
    DetailsObj.ID = CricObj.ID;
    DetailsObj.Name = CricObj.Name;
    DetailsObj.FullName = CricObj.FullName;
    DetailsObj.Age = CricObj.Age;
    DetailsObj.Team = CricObj.Team;
    return View("CricketerDetails", DetailsObj);
}
if (BtnNext != null)
{
    if (ModelState.IsValid)
    {
        CricObj.OdiRuns = StatisticsData.OdiRuns;
        CricObj.TestRuns = StatisticsData.TestRuns;
        CricObj.Century = StatisticsData.Century;
        CricObj.HalfCentury = StatisticsData.HalfCentury;
        CricObj.Wickets = StatisticsData.Wickets;
        CricObj.Catches = StatisticsData.Catches;
        CricketerDbEntities db = new CricketerDbEntities();
        db.Cricketers.Add(CricObj);
        db.SaveChanges();
        RemoveCricketer();
        return View("Success");
    }
}
return View();
}
}
}

```

In the code above the *Index()* method simply returns a view that represents the CricketerDetails that is the first step of the wizard. The *GetCricketer()* method is used to check the stored cricketer in the Session and if the cricketer object existsthen that object is returned, otherwise a new cricketer object is created and stored in the Session with a key. The *RemoveCricketer()*simply removes the cricketer key and Session object.

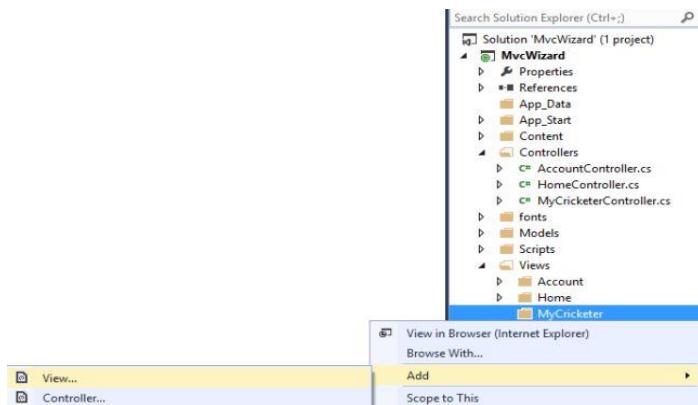
The *CricketerDetails()* action method has three parameters named *CricketerDetails* object, *BtnNext* and *BtnPrevious*. The *CricketerDetails* object contains the value of the properties defined in *CricketerDetails*. If *BtnNext* and *BtnPrevious* is not null then that indicates that the button was clicked.

In the *CricketerDetails()* action method, the *ModelState.IsValid* property determines that the model contains the valid data and returns true and *GetCustomer()* retrieves the cricketer object from the Session. The code then returns to the next view named *CricketerStatistics* and if there are no validation errors, then the Entity Framework context is instantiated and the cricketer object is added to the Cricketers DbSet. *SaveChanges()* method to save the data in the database. Finally, the Success view is returned from the method.

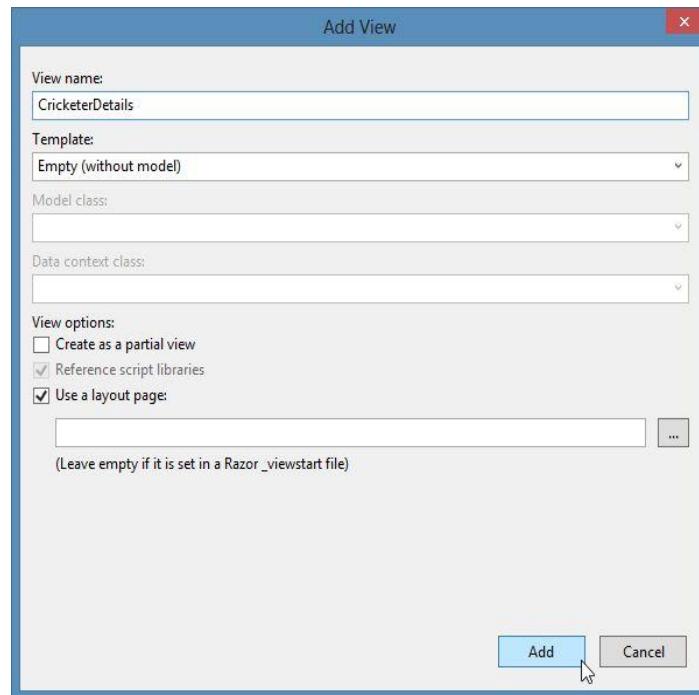
13.6 Working with View

As I said earlier that I am using the MVC 5 so that here I am using the Razor syntax to define the view. Let's proceed with the following procedure.

Step 1: The *MyCricketer* folder is created automatically in the Views folder because you've created a New Scaffolded Item with the same name in the Controller, so just right-click on it to add the view.



Step 2: Enter the name as "*CricketerDetails*".



Step 3: Replace the code with the following code:

```

@model MvcWizard.Models.CricketerDetails
 @{
    ViewBag.Title = "Cricketer Details";
}

<h2>Cricketer Details</h2>

@using (Html.BeginForm("CricketerDetails", "MyCricketer", FormMethod.Post))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Step 1: Cricketer Details</h4>
        <hr />
        @Html.ValidationSummary(true)

        <div class="form-group">
            @Html.LabelFor(model => model.ID, new { @class = "control-label col-md-2" })

```

```
<div class="col-md-10">
    @Html.TextBoxFor(model => model.ID)
    @Html.ValidationMessageFor(model => model.ID)
</div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Name, new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextBoxFor(model => model.Name)
        @Html.ValidationMessageFor(model => model.Name)
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.FullName, new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextBoxFor(model => model.FullName)
        @Html.ValidationMessageFor(model => model.FullName)
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Age, new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextBoxFor(model => model.Age)
        @Html.ValidationMessageFor(model => model.Age)
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Team, new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextBoxFor(model => model.Team)
        @Html.ValidationMessageFor(model => model.Team)
    </div>
</div>
```

```

<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" name="BtnNext" value="Next" class="btn btn-default" />
    </div>
</div>
</div>
}

```

In the code above, the model has been set to the *CricketerDetails*. The view renders the form using the *BeginForm()* HTML helper that posts the *CricketerDetails* action method of *MyCricketerController*. The fields for ID, Name and so on are rendered from the *LabelFor()* and *TextBoxFor()* helpers.

Validations are emitted from the *ValidationMessageFor()* helper.

Step 4: Add another view named "*CricketerStatistics*" and replace the code with the following code:

```

@model MvcWizard.Models.CricketerStatistics
 @{
    ViewBag.Title = "Cricketer Statistics";
}

<h2>Cricketer Statistics</h2>

@using (Html.BeginForm("CricketerStatistics", "MyCricketer", FormMethod.Post))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Step 2: Cricketer Statistics</h4>
        <hr />
        @Html.ValidationSummary(true)
        <div class="form-group">
            @Html.LabelFor(model => model.OdiRuns, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.TextBoxFor(model => model.OdiRuns)
                @Html.ValidationMessageFor(model => model.OdiRuns)
            </div>
        </div>
    </div>
}

```

```
</div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.TestRuns, new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextBoxFor(model => model.TestRuns)
        @Html.ValidationMessageFor(model => model.TestRuns)
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.Century, new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextBoxFor(model => model.Century)
        @Html.ValidationMessageFor(model => model.Century)
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.HalfCentury, new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextBoxFor(model => model.HalfCentury)
        @Html.ValidationMessageFor(model => model.HalfCentury)
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.Wickets, new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextBoxFor(model => model.Wickets)
        @Html.ValidationMessageFor(model => model.Wickets)
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.Catches, new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextBoxFor(model => model.Catches)
        @Html.ValidationMessageFor(model => model.Catches)
    </div>
```

```

</div>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" name="BtnPrevious" value="Previous" class="btn btn-default" />
        <input type="submit" name="BtnNext" value="Finish" class="btn btn-default" />
    </div>
</div>
</div>
}

```

Step 5: Add another view named "Success" and replace the code with the following code:

```

@{
    ViewBag.Title = "Success";
}

<h3>Cricketer Saved Successfully! Thanks</h3>

<div>
    @Html.ActionLink("Add Another Cricketer", "Index", "MyCricketer")
</div>

```

Step 6: Open the *Views\Shared_Layout.cshtml* file to modify the code as shown in the highlighted code below:

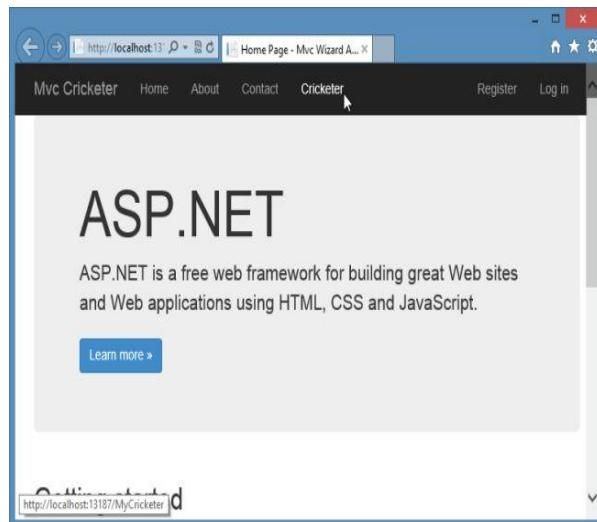
- <title>@ViewBag.Title - Mvc Wizard App</title>
- <ul class="nav navbar-nav">
 @Html.ActionLink("Home", "Index", "Home")
 @Html.ActionLink("About", "About", "Home")
 @Html.ActionLink("Contact", "Contact", "Home")
 @Html.ActionLink("Cricketer", "Index", "MyCricketer")

- <footer>
 <p>© @DateTime.Now.Year - Mvc Wizard Application</p>
 </footer>

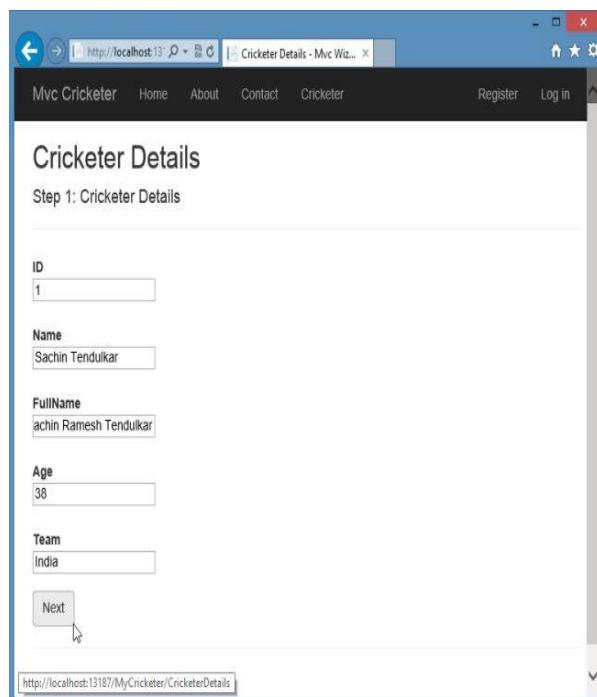
That's all for the Model, View and Controller.

13.7 Run the Application

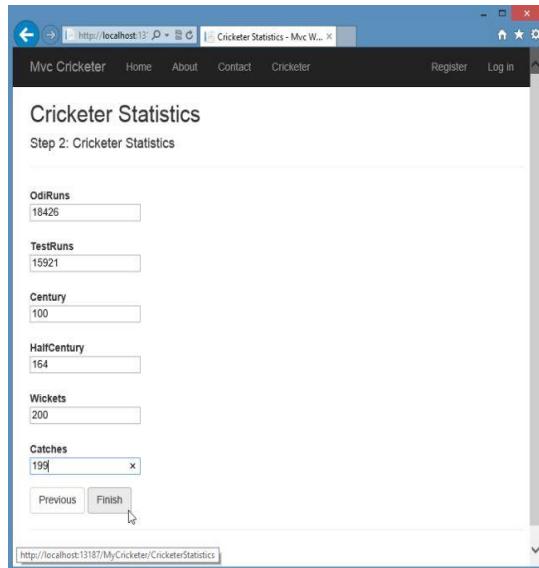
Step 1: Press Ctrl+F5 or F5 to run the application and click on the *Cricketer* link.



Step 2: Enter the details of cricketer in the first step of the wizard and click on "Next".



Step 3: Fill in the rest of the statistics of cricketer in the second step of the wizard and click on "Finish".



OdiRuns
18426

TestRuns
15921

Century
100

HalfCentury
164

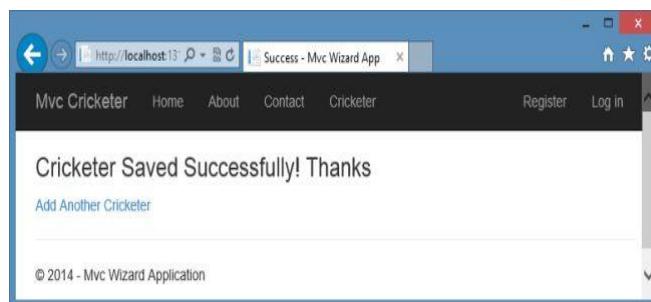
Wickets
200

Catches
199

Previous Finish

You can also click on the previous button to edit in the Details view.

Step 4: The Success View opens and the data saved successfully. You can add another cricketer also from here.



You can also see the Validation Message, if the wrong data was inserted.

Cricketer Details
Step 1: Cricketer Details

ID The ID field is required.

Name asdf

FullName asdf

Age asdf The value 'asdf' is not valid for Age.

Team The Team field is required.

© 2014 - Mvc Wizard Application

Chapter 14: Introducing Mobile Site in MVC 5 and jQuery Mobile

14.1 Introduction

As you know there are various types of emulators available for viewing applications. We can use the iPhone and Windows Phone simulators for browsing the application. You can open the website on a phone that is so much more satisfying than a Desktop.

In that context, when you create the MVC 4 application you can use a Mobile template to design the application. Here, I am using Visual Studio 2013 and a MVC 5 project template to design the application. We'll use the jQuery Mobile application for displaying it in the phones and tablets.

Let's create an application on Visual Studio using MVC 5 Project template and perform some CRUD operations and add jQuery Mobile and ViewSwitcher. I am using the [iPhone](#) simulator to browse the application.

14.2 Creating CRUD Operations

Step 1: Add a model class named *Cricketer* and add the following code:

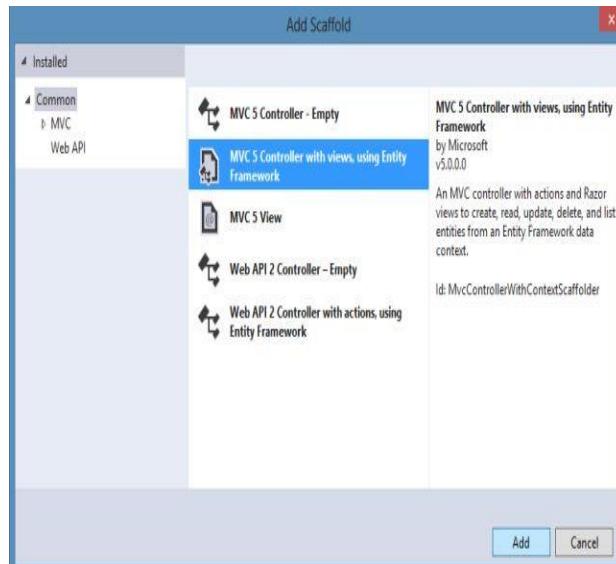
```
public enum Grade
{
    A,B,C
}

public class Cricketer
{
    public int ID { get; set; }
    public string Name { get; set; }
    public string Team { get; set; }
    public Grade Grade { get; set; }
}

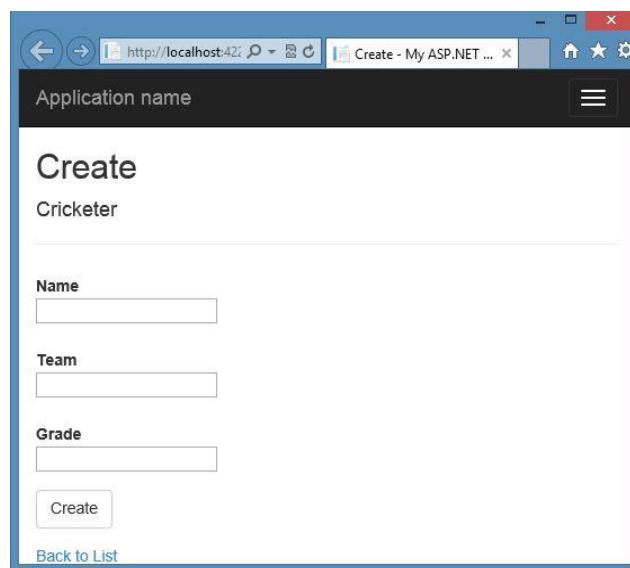
public class CricketerDbContext : DbContext
{
    public DbSet<Cricketer> Cricketers { get; set; }
}
```

In the code above, we've created an Enum property and we'll add the Enum support to the View in this chapter later.

Step 2: Scaffold a New Controller



Step 3: Unfortunately scaffolding does not do the Enums in the *Create.cshtml* and *Edit.cshtml* pages. You can see in the following screenshot:



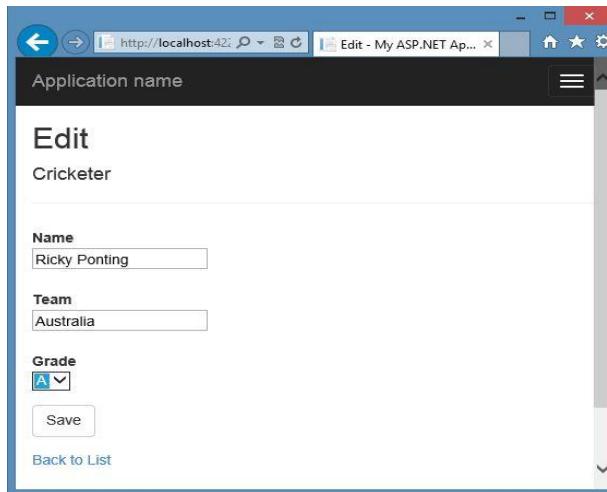
Step 4: So we need to update it using the following highlighted code:

```

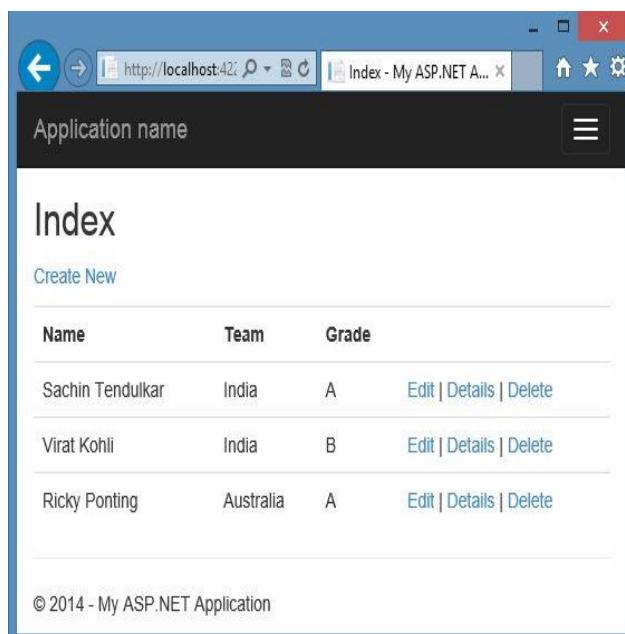
<div class="form-group">
    @Html.LabelFor(model => model.Grade, new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EnumDropDownListFor(model => model.Grade)
        @Html.ValidationMessageFor(model => model.Grade)
    </div>
</div>

```

Step 4: Ok. Now it's time to run the application and perform the CRUD operations.



The screenshot shows an 'Edit' view for a 'Cricketer'. The page title is 'Edit - My ASP.NET Ap...'. The main heading is 'Edit' and the sub-heading is 'Cricketer'. There are three input fields: 'Name' (Ricky Ponting), 'Team' (Australia), and 'Grade' (A). A 'Save' button is at the bottom, and a 'Back to List' link is at the bottom left.



The screenshot shows an 'Index' view displaying a list of cricketers. The table has columns for Name, Team, and Grade. The data is as follows:

Name	Team	Grade
Sachin Tendulkar	India	A
Virat Kohli	India	B
Ricky Ponting	Australia	A

At the bottom, there is a copyright notice: © 2014 - My ASP.NET Application.

14.3 Adding Mobile Support

You can develop it while creating the MVC 4 application with the Mobile Template but you can add the jQuery Mobile along with the MVC 5 application. You can simply have it from the NuGet Pacakges or entering the following command in the Pacakge Manager Console:

Install-Package jQuery.Mopible.MVC

This package will add various things such as:

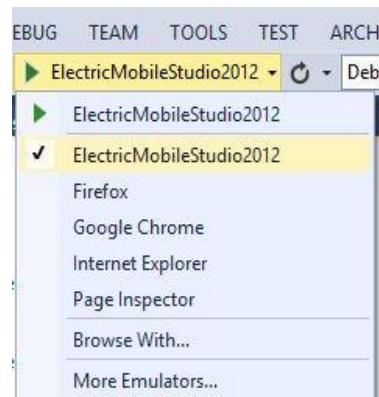
- A *ViewSwitcher* partial view and supporting Controller
- Basic *_Layout.Mobile.cshtml* and supporting CSS
- Newly added *BundleMobileConfig.cs*

Note: You can use jQuery in any mobile framework.

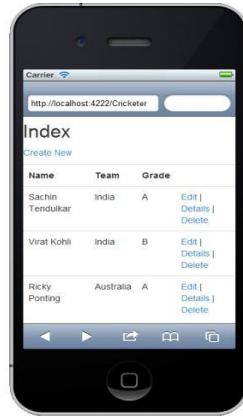
Now you need to modify the *Global.asax* file with the following highlighted code:

```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
    BundleMobileConfig.RegisterBundles(BundleTable.Bundles);
}
```

When you run the application on the iPhone simulator as in the following:



The following screenshot will open:



Now change the table in the *Index.cshtml* page with the following code:

```
<ul data-role="listview" data-filter="true" class="my_class_list">
    @foreach (var item in Model)
    {
        <li>
            <a href="@Url.Action("Details", new {item.ID})">
                <h3>@Html.DisplayFor(modelItem => item.Name)</h3>
                <p>@Html.DisplayFor(modelItem => item.Team) - @Html.DisplayFor(modelItem => item.Grade) Grade</p>
            </a>
        </li>
    }
</ul>
```

Now just refresh the page.



Chapter 15: Working with Yahoo External Provider in MVC 5

15.1 Introduction

I described earlier how to do [External Authentication in MVC](#) like Google, Facebook and Twitter. So here I am describing to connect with Yahoo. You can use Yahoo as an External Authentication Provider to login with the MVC application.

Here in this chapter, I am developing the ASP.NET Web Application using the latest MVC 5 in the Visual Studio 2013. So, let's proceed with the following sections:

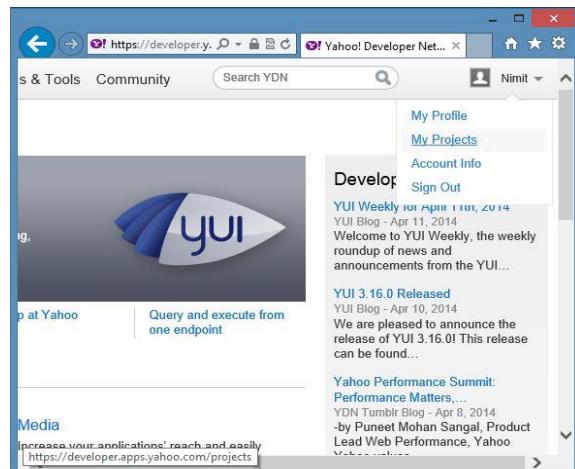
- Creating Yahoo App
- Adding NuGet Package
- Working with MVC Application

15.2 Creating Yahoo App

Use the following procedure create the Yahoo app.

Step 1: Open the [developers.yahoo.com](https://developer.yahoo.net)

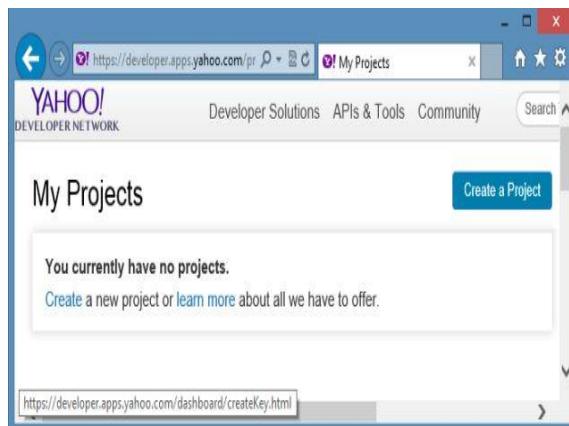
Step 2: Login and on the top right hover on your name to select My Projects.



Step 3: Now accept the terms and conditions by Yahoo.



Step 4: It might be possible that you didn't have a single project. If a project exists then you can use that one otherwise click on Create New Project.



Step 5: Enter the details of the app.

Create a new project

Tell us about your application

Application Name: *

Application Type: Web-based

Description: (250 characters or less) *

Home Page URL: *

Access Scopes:

- This app will only access public APIs, Web Services, or RSS feeds.
- This app requires access to private user data.

Callback Domain: *

Yahoo! APIs use the OAuth protocol for secure validation of API usage and end-user authentication. Please specify the domain to which your application will be returning after successfully authenticating.

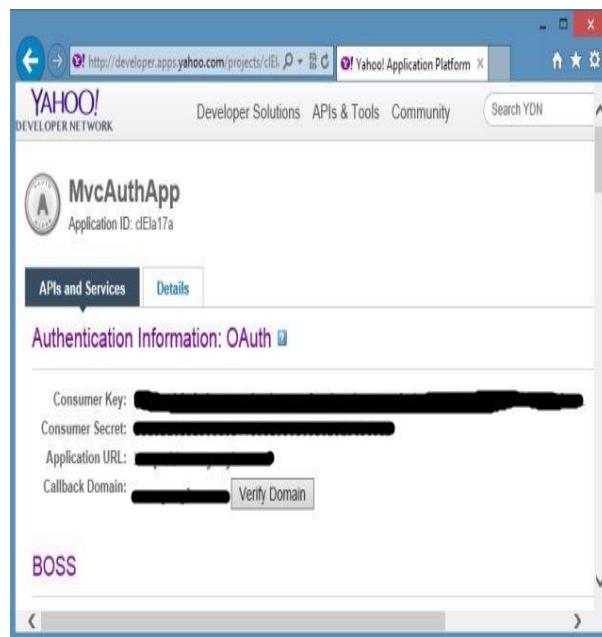
Select APIs for private user data access:

- Contacts
- Read - With your user's permission, you can read their Contacts information.
- ReadWrite - With your user's permission, you can read and write their Contacts information.

Step 6: Check the check box to accept the terms and conditions and click on Create Project



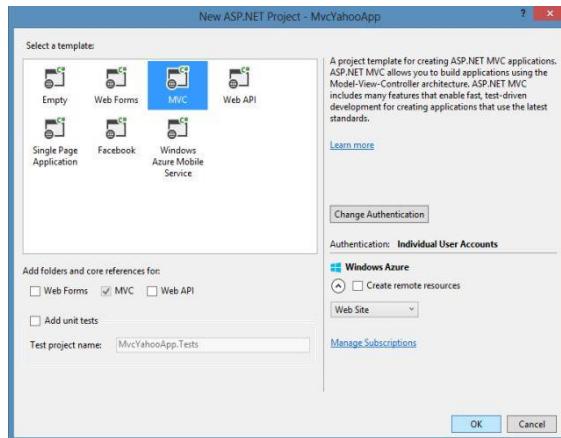
Step 7: Now copy the App ID and App Secret of the Yahoo app.



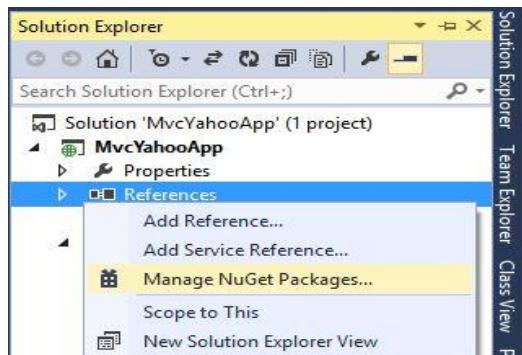
15.3 Adding NuGet Package

In this section we'll add the NuGet Package for the MVC application. Proceed with the procedure below.

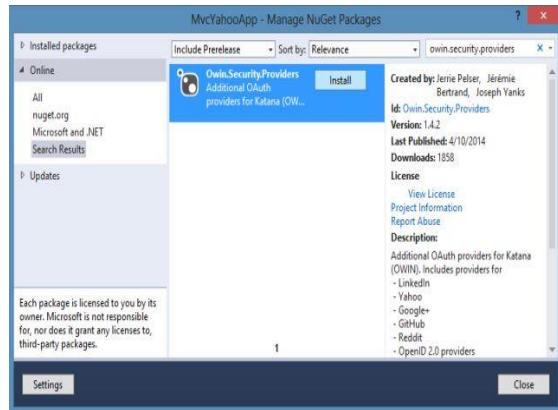
Step 1: Create the MVC Application



Step 2: Manage NuGet Packages of the application.



Step 3: Add the OWIN.security.providers package.



15.4 Working with MVC Application

In this section we'll add the code to the Authentication file that is used to authenticate the Yahoo app provider in the MVC application. So you just update the *App_Start\Startup.Auth.cs* file with the following code.

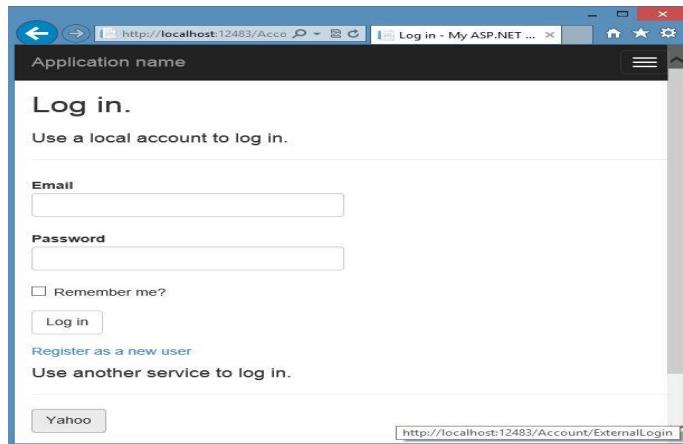
Add the following namespace:

```
using Owin.Security.Providers.Yahoo;
```

Add the following code to the class:

```
app.UseYahooAuthentication("Your App ID", "App Secret");
```

That's it. Now when you run the application login into the application with Yahoo.

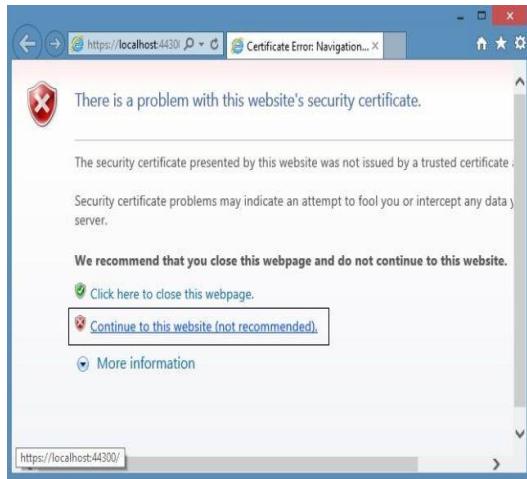


Chapter 16: Working With SSL Certificate Warning in MVC 5 Application

16.1 Introduction

This chapter explains the accessibility of OAuth 2.0 in the ASP.NET MVC 5 Web Application. We can use various types of external authentication providers like Facebook, Google, Microsoft and Twitter in MVC 5 applications to log in.

In the same context, I am very excited to describe that now you will not receive any type of SSL Certificate warnings when you enable SSL for the MVC application. Previously while enabling the SSL in the MVC application, the browser like IE, Chrome and Firefox showed a SSL Certificate warning. You can refer to Working with Facebook in MVC and check out in the Step 8 of Creating and [Working with Facebook App](#) as in the following screenshot:



With the use of the following chapter you will **not** receive the SSL Certificate Warning in most browsers including IE and Chrome. Firefox **will** show the warning because Firefox uses its own certificate store, so it will display the warning. We'll see it later in this chapter.

So, let's develop the application to work with the external providers. Here I am also describing how to develop the app on Google and authenticating the MVC application by the app on Google. Please use the following procedure:

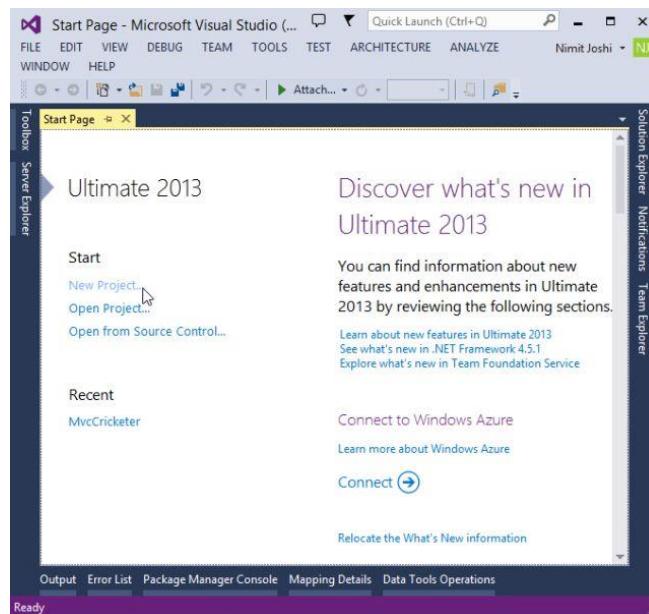
- Creating ASP.NET MVC 5 Web Application

- Enabling and Setting Up the SSL

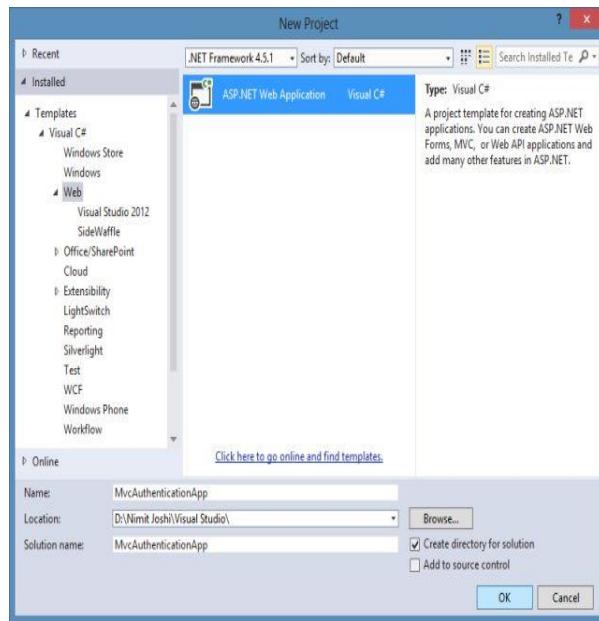
16.2 Creating ASP.NET MVC 5 Web Application

In this section you will create the web application in the Visual Studio 2013 using the following procedure.

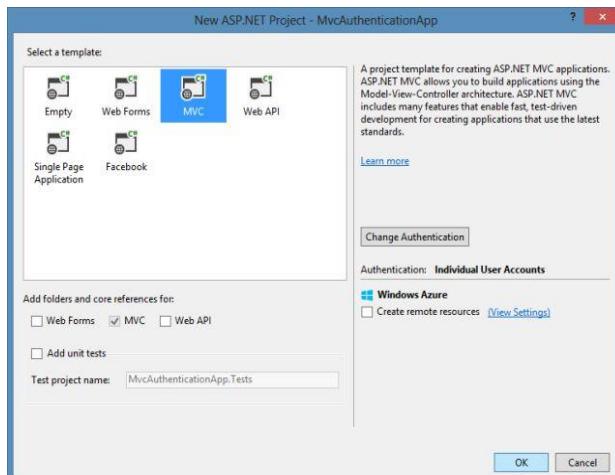
Step 1: Open Visual Studio 2013 and click on "New Project".



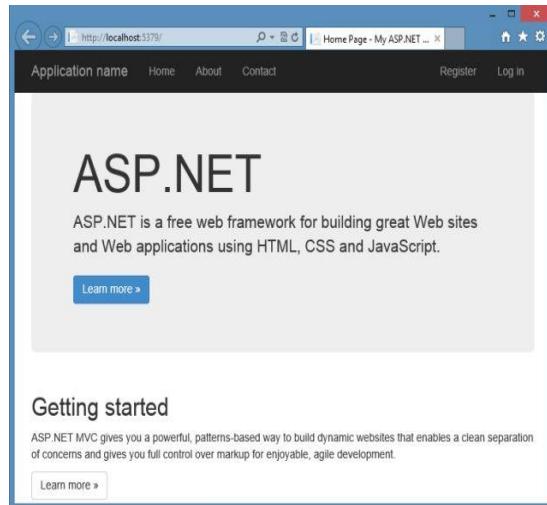
Step 2: Select the ASP.NET Web Application and enter the application name.



Step 3: Select the MVC Project Template to develop the application.



Visual Studio automatically creates the MVC 5 application and you can execute the application by pressing Ctrl+ F5 or F5.



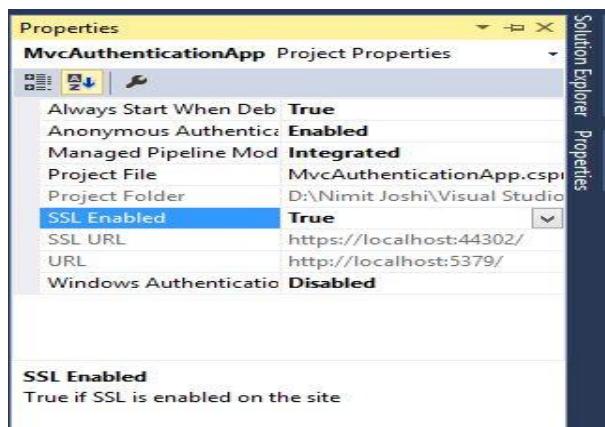
16.3 Enabling and Setting Up the SSL

We need to set up the IIS Express to use SSL to connect with the external authentication providers like Google and Facebook. It's important to continue using SSL after login and not drop back to HTTP, your login cookie is just as secret as your username and password and without using SSL you're sending it in clear-text across the wire.

Use the following procedure.

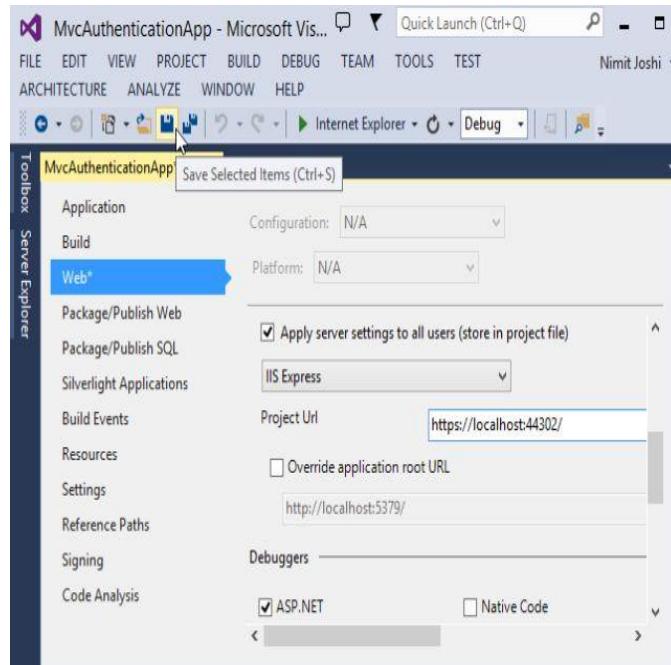
Step 1: In the Solution Explorer, select the application and press the F4 key.

Step 2: Enable the SSL Enabled option and copy the SSL URL.



Step 3: Now just right-click on the application and select the Properties option.

Step 4: Select the Web tab from the left pane, and paste the SSL URL into the Project URL box.



Step 5: Select the *HomeController.cs* from the Controllers folder and add the following highlighted code to edit:

```
namespace MvcAuthenticationApp.Controllers
{
    [RequireHttps]
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

In the code above the *RequireHttps* attribute is used to require that all requests must use HTTPS.

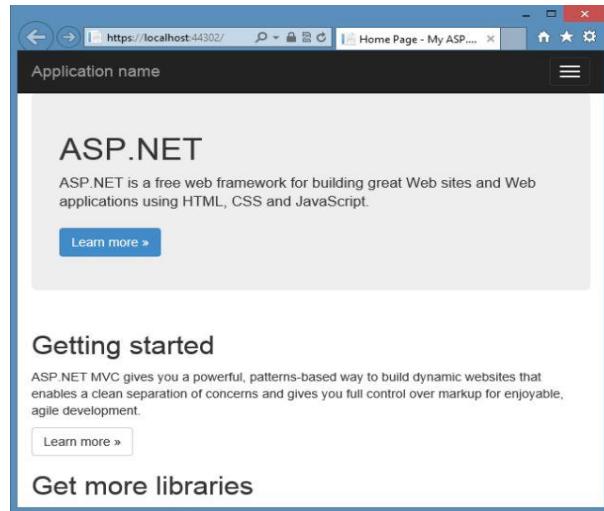
Step 6: Now press Ctrl+F5 to run the application and follow the instructions to trust the self-signed certificate generated by IIS Express.



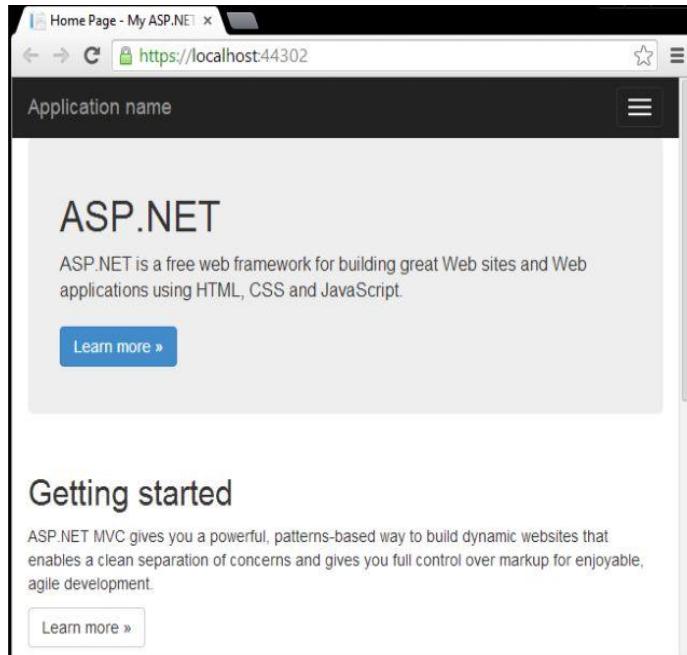
Step 7: After clicking on Yes, the Security Warning wizard opens and click Yes to install the certificate representing the localhost.



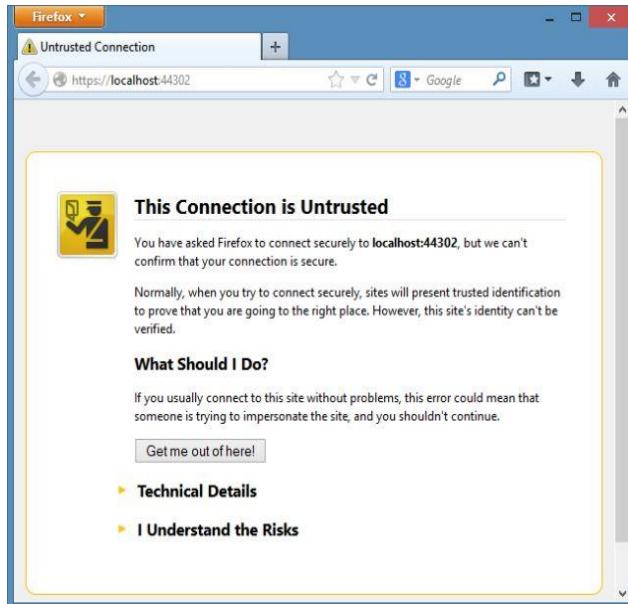
Step 8: Now, when you run the application using Internet Explorer (IE), it shows the Home Page of the application and there is no SSL warning.



The same thing for Google Chrome as in the following:



Firefox will show the warning because, as I said earlier, Firefox uses its own certificate store, so it will display the warning.



Chapter 17: CRUD Operations Using Asynchronous Programming in MVC 5

17.1 Introduction

In ASP.NET MVC 5 you have used the Scaffolding and worked with the Entity Framework. I also have used that [scaffolding in the MVC 5](#). To use scaffolding we can apply any approach of Entity Framework 6.

We've used the synchronous programming to perform the basic CRUD (Create, Read, Update and Delete) functionality in ASP.NET MVC 5 Web Applications. We'll now use the asynchronous programming for the CRUD functionality in the MVC 5 application.

Why Asynchronous Programming

Generally there are very reserved worker processes available in the web server. If the load increases in the web server then it may be possible that all processes are in use. In that situation, the server cannot handle the new request until the processes are freed up. In the synchronous programming , the worker processes may be tied up while they aren't actually doing any work because they are waiting for the I/O operations to complete. When we use asynchronous programming, when the I/O operations are not complete, the worker processes are freed up for the further use. So, asynchronous programming enables the server resources to be used more efficiently.

The .NET Framework 4.5 makes the asynchronous programming very easier.

Prerequisites

There are some prerequisites before starting with this chapter, given below:

- Visual Studio 2013
- MVC 5.1.1 (Latest)

However the preceding prerequisites are only applicable for performing the same as described in this chapter. You can perform this on MVC 5 applications also.

Getting Started

In this section, we'll create the MVC 5 application using the following sections:

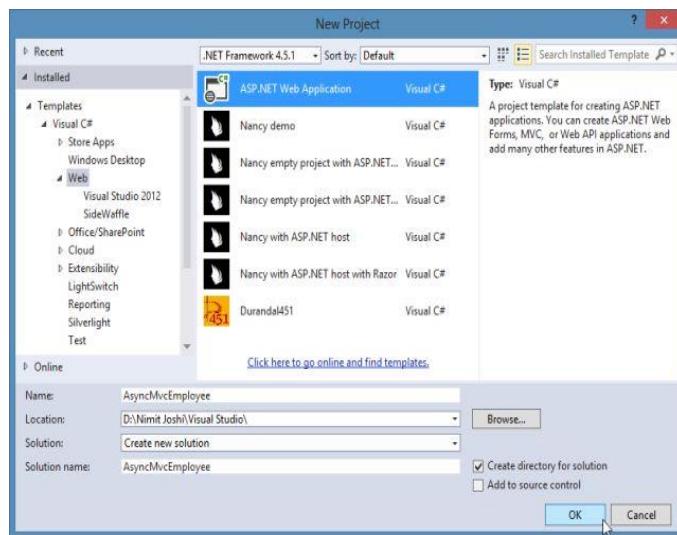
- Create ASP.NET MVC Application
- Perform CRUD Functionality Asynchronously
- Running MVC application

Create ASP.NET MVC Application

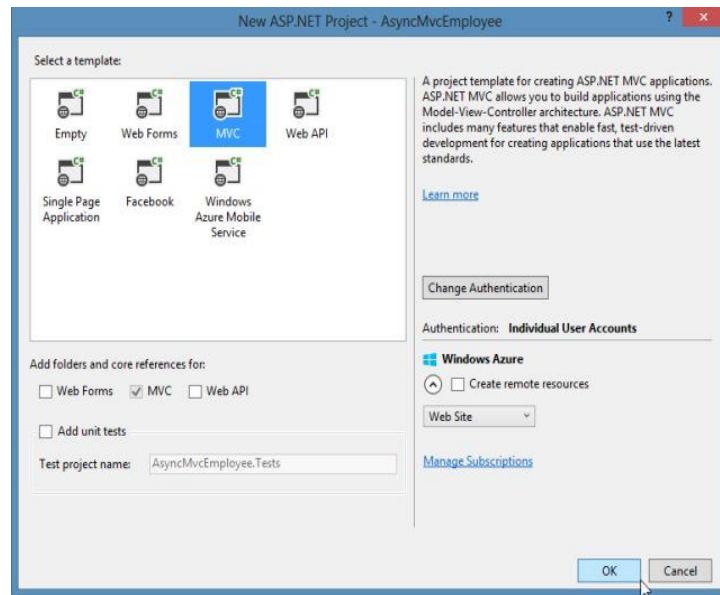
In this section we'll create the application in Visual Studio 2013. Take a look at the following procedure:

Step 1: Open Visual Studio 2013 and click on "New Project".

Step 2: Select the ASP.NET Web Application and enter the name.



Step 3: Select the MVC Project Template from the One ASP.NET Wizard.



Visual Studio will automatically create the MVC application.

17.2 Perform CRUD Functionality Asynchronously

In this section we'll perform the CRUD functionality in this application. Use the following procedure.

Step 1: In the Solution Explorer, add a new class named "*Employee*" in the Models folder and update the code with the following code:

```
using System.Data.Entity;
namespace AsyncMvcEmployee.Models
{
    public class Employee
    {
        public enum Gender
        {
            Male,
            Female
        }
        public int ID { get; set; }
        public string Name{ get; set; }
        public Gender Sex { get; set; }
```

```

public string Post { get; set; }
public decimal Salary { get; set; }
public string City { get; set; }
}
public class EmployeeDbContext : DbContext
{
    public DbSet<Employee> Employees { get; set; }
}
}

```

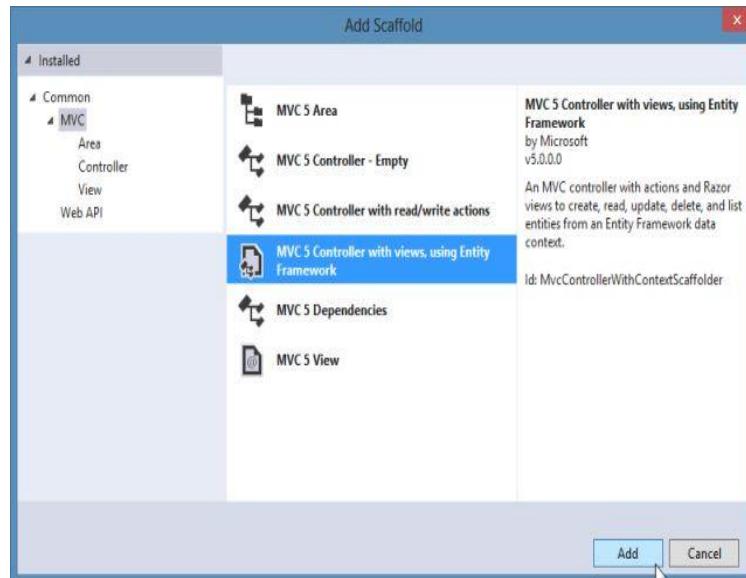
Step 2: Save the *EmployeeDbContext* as a Database Context in the *Connection Strings* in the *Web.Config* file:

```

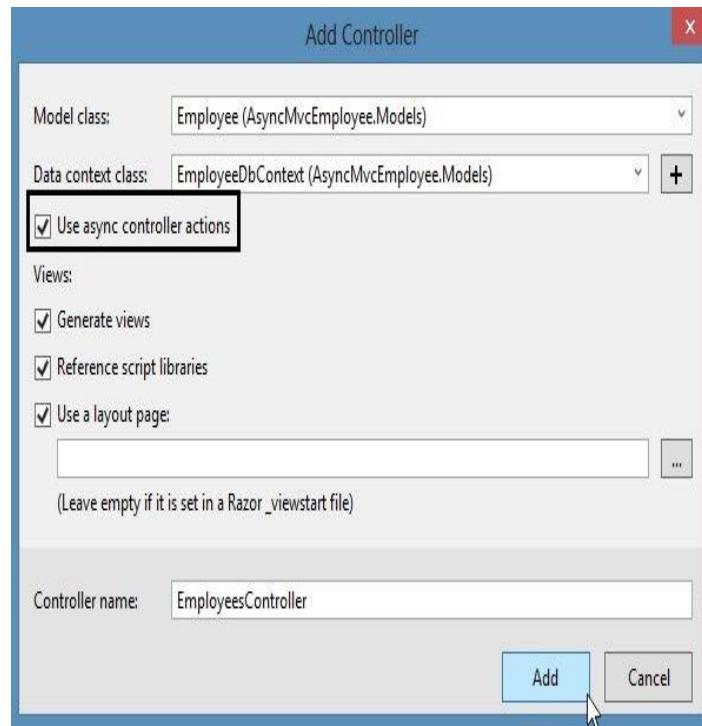
<add name="EmployeeDbContext" connectionString="Data Source=(LocalDb)\v11.0;
AttachDbFilename=|DataDirectory|\Employee.mdf;
initial Catalog=Employee;Integrated Security=True" providerName="System.Data.SqlClient" />

```

Step 3: Now right-click on the Controllers folder to add a new Scaffolded item, select the MVC Scaffolding using Entity Framework with Views as in the following:



Step 4: Select the Model class, Database Context class and select the check box of Use async controllers as in the following:



As you see in the preceding screenshot that I've checked the use async controller. After this the new scaffolded controller is generated and there were some changes applied when Entity Framework executed asynchronously. Those are given below:

- Check out the following code:

```
public async Task<ActionResult> Index()
{
    return View(await db.Employees.ToListAsync());
}
```

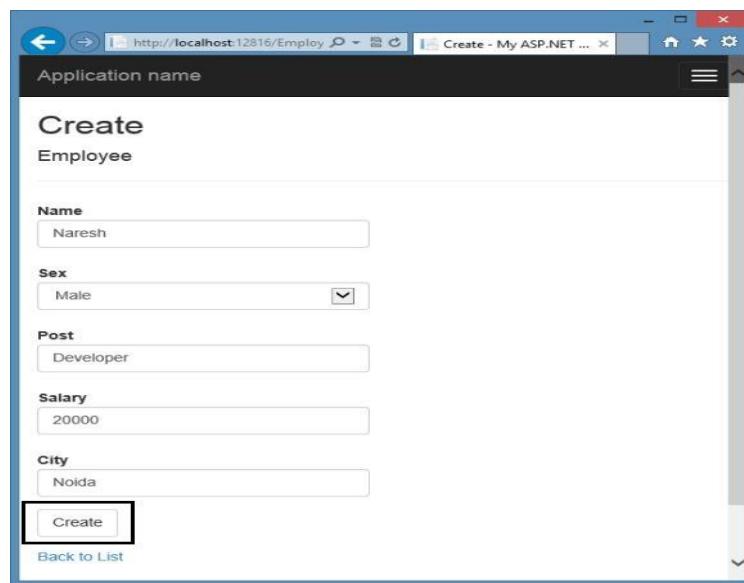
The `Index()` method is now an asynchronous method that tells the compiler to generate callbacks for parts of the method body and create automatically a `Task<ActionResult>` object that is returned.

- You can also check out that the method return type is `Task<ActionResult>` in which the `Task<T>` shows the current work with the result of the same type (T).

- The *await* keyword calls the service. In the background , the method splits into two parts. The first part ends with the operations that is started asynchronously and the second part is put into a callback method that is called when the operation completes.

17.3 Running MVC Application

Now run the application, that works the same as previously. Run the Employees Controller as in the following:



The screenshot shows a web browser window with the URL `http://localhost:12816/Employ`. The page title is "Create - My ASP.NET ...". The main content is a "Create" form for an "Employee". The form has the following fields:

- Name:** Naresh
- Sex:** Male
- Post:** Developer
- Salary:** 20000
- City:** Noida

A red box highlights the "Create" button at the bottom of the form. Below the form is a link "Back to List".

Chapter 18: Stored Procedures in Entity Framework 6 in MVC 5

18.1 Introduction

As you know we can apply any Entity Framework approach such as Code First and Database First in the ASP.NET MVC Application, so in this chapter I am explaining the use of Stored Procedures when we use the Code First Approach of Entity Framework in MVC 5 applications.

We can implement the Stored Procedures in Entity Framework 6 and perform only the Insert, Delete and Update operations. One more feature is that it only works for those applications that use the Code First Approach; that is, we first create the structure of the class and next accomplish the application and the database is created when running the application.

You will also learn the Code First Migrations use in here and perform the operations in the database. We use the functionality of Stored Procedures with the Fluent API. Any implementation occurs through a feature called Fluent API.

Prerequisites

Visual Studio 2013 is the prerequisite to work with this chapter.

So, let's just use the following sections with which we can implement the functionality:

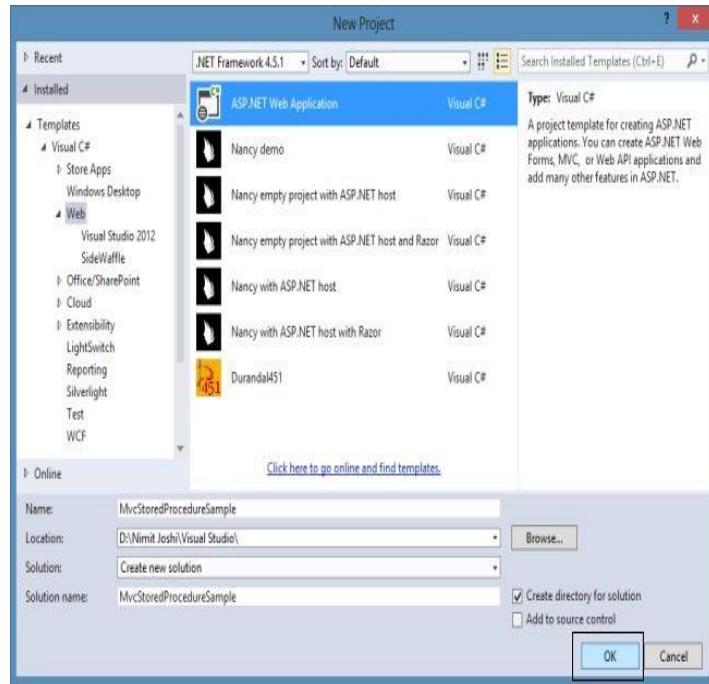
- Create ASP.NET MVC 5 Application
- Adding Model
- Scaffolding in MVC 5
- View in MVC 5
- LOG in Entity Framework
- Working with Stored Procedures

18.2 Create ASP.NET MVC 5 Application

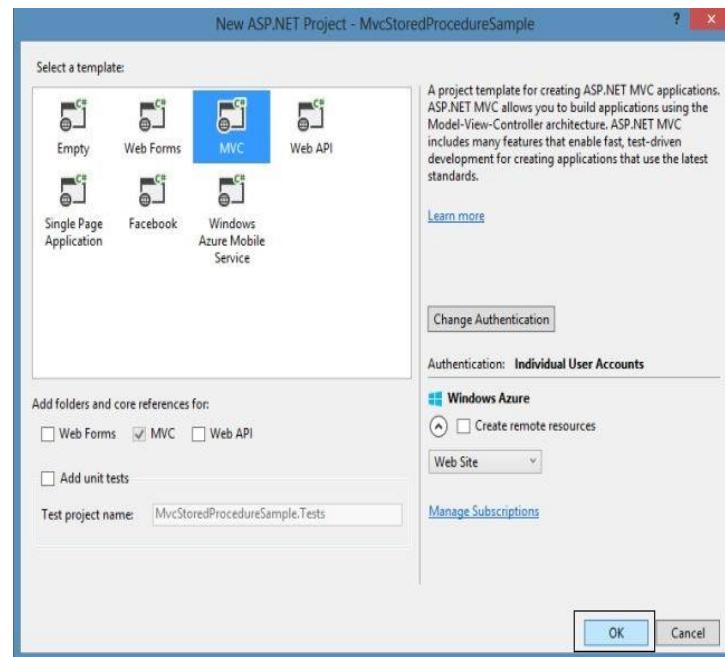
In this section we'll create the ASP.NET Web Application with the MVC 5 Project Template. Use the following procedure.

Step 1: Open the Visual Studio 2013 and click on the "New Project".

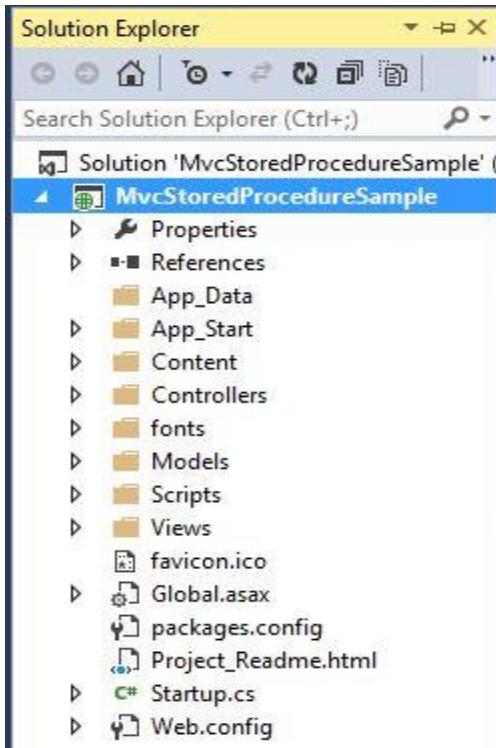
Step 2: Select the Web from the left pane and create the ASP.NET Web Application.



Step 3: Select the MVC Project Template in the next One ASP.NET Wizard.



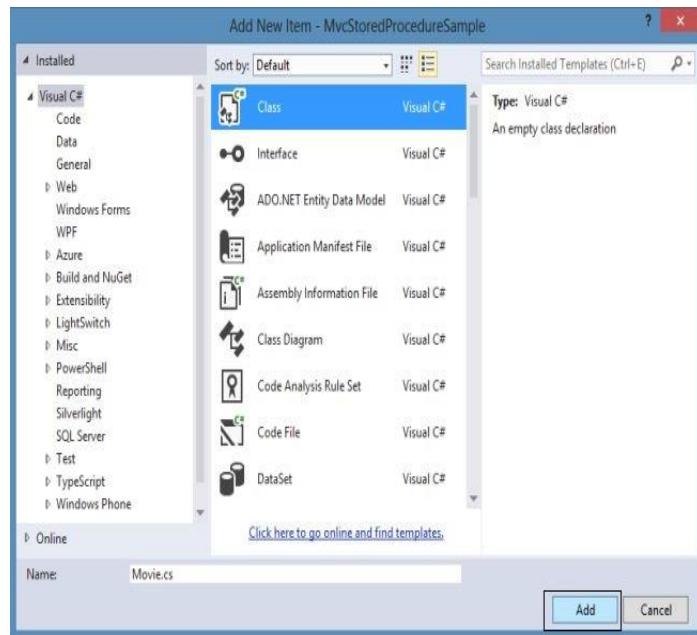
Visual Studio automatically creates the MVC 5 application and adds some files and folders to the solution. Have a look:



18.3 Adding Model

In this section, we'll add the class in the models folder. Use the following procedure.

Step 1: Right-click on the Models folder and Add a new Class, "*Movie*".



Step 2: Edit the code with the following code:

```

using System;
using System.ComponentModel.DataAnnotations;

namespace MvcStoredProcedureSample.Models
{
    public class Movie
    {
        public int ID { get; set; }

        [Required]
        public string Name { get; set; }

        [Required]
        [Display (Name="Release Date")]
        public DateTime ReleaseDate { get; set; }

        [Required]
        public string Category { get; set; }
    }
}

```

In the code above, the properties are defined in a class. You can also notice that there is no entity key is defined in the preceding code because we are using the Entity Framework 6 and it is not necessary since the key property is composed by class name + ID. As we have the class named *Movie*, so the ID property is identified automatically as the primary key. You can also add other properties to the class.

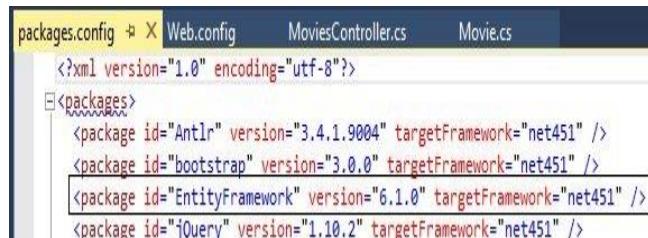
Step 3: Build the solution.

Working with Entity Framework

Generally when we create the latest MVC project, the Entity Framework is installed as a default. If it is not available in the *packages.config* file; you can install it from the Package Manager Console by entering the following command:

Install-Package EntityFramework

In my solution, the latest version of Entity Framework, EntityFramework 6.1.0, is installed as the default. Have a look:



```

<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Antlr" version="3.4.1.9004" targetFramework="net451" />
  <package id="bootstrap" version="3.0.0" targetFramework="net451" />
  <package id="EntityFramework" version="6.1.0" targetFramework="net451" />
  <package id="jQuery" version="1.10.2" targetFramework="net451" />

```

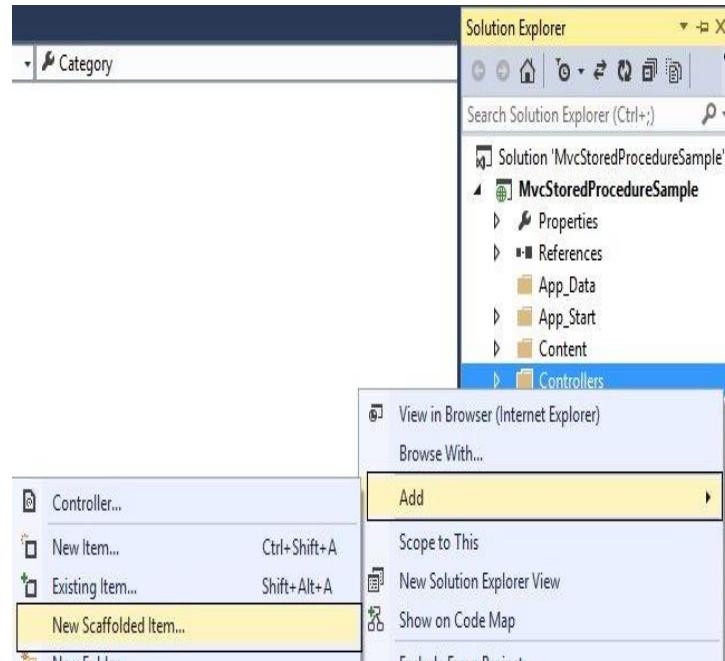
You can also update the package by entering the following command in the Package Manager Console:

Update-Package EntityFramework

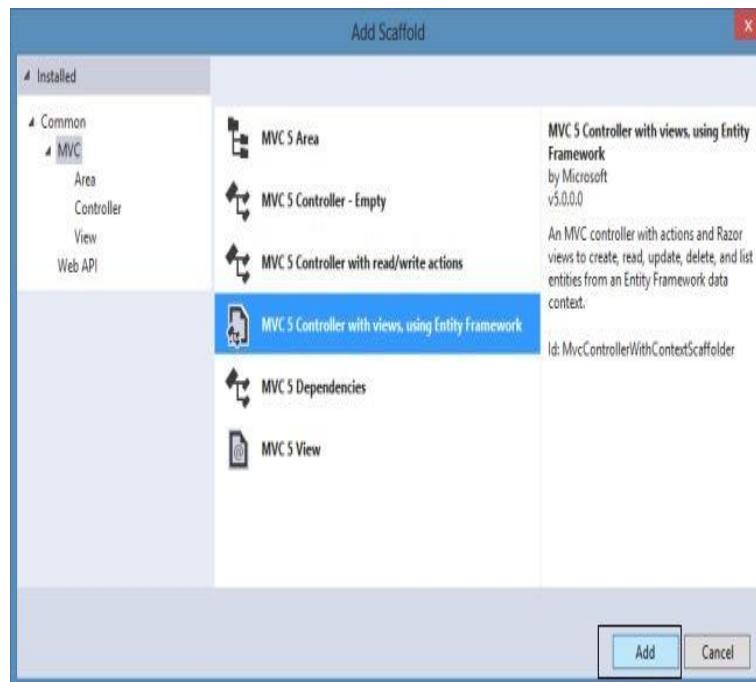
18.4 Scaffolding in MVC 5

In this section we'll add a new scaffolded controller using Entity Framework. So, follow the procedure below.

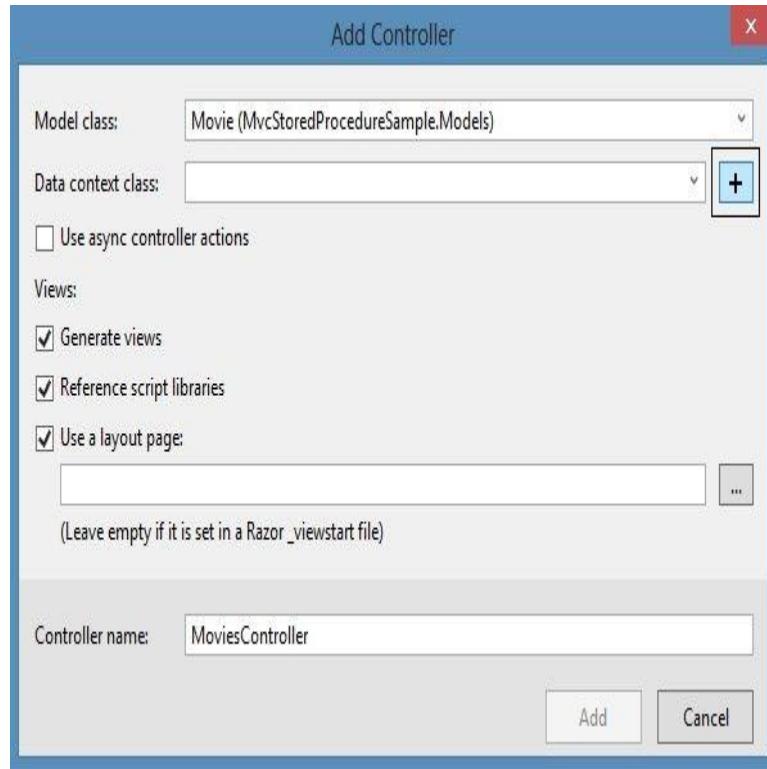
Step 1: Just right-click on the Controllers folder and click on the Add-> New Scaffolded Item



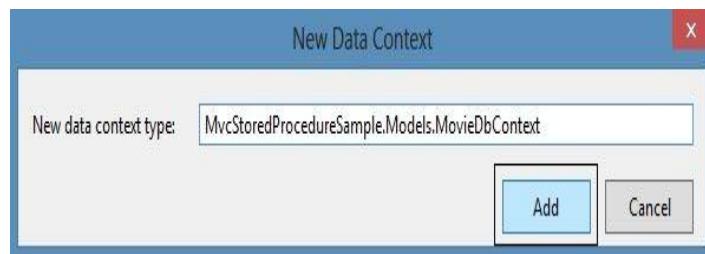
Step 2: In the next Add Scaffold wizard, select the MVC 5 Controller with views as in the following:



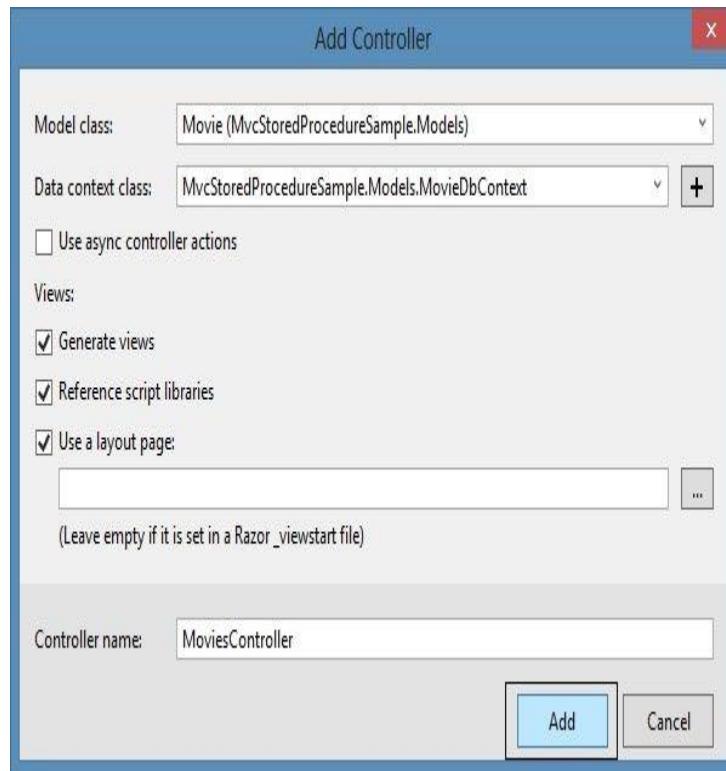
Step 3: In the next Add Controller wizard, select the Model Class and to use the Data Context class we need to add new.



Step 4: Enter the Data Context class as in the following:



Step 5: Now Add the Controller by clicking the Add button



Step 6: Now we have the *MovieDbContext* class and *MoviesController* class after scaffolding the controller. Check it out:

MovieDbContext class:

```
using System.Data.Entity;
namespace MvcStoredProcedureSample.Models
{
    public class MovieDbContext : DbContext
    {
        public MovieDbContext() : base( "name = MovieDbContext" )
        {
        }
        public DbSet<Movie> Movies { get; set; }
    }
}
```

MoviesController class:

```
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web.Mvc;
using MvcStoredProcedureSample.Models;
namespace MvcStoredProcedureSample.Controllers
{
    public class MoviesController : Controller
    {
        private MovieDbContext db = new MovieDbContext();

        // GET: Movies
        public ActionResult Index()
        {
            return View(db.Movies.ToList());
        }

        // GET: Movies/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Movie movie = db.Movies.Find(id);
            if (movie == null)
            {
                return HttpNotFound();
            }
            return View(movie);
        }

        // GET: Movies/Create
        public ActionResult Create()
```

```

{
    return View();
}

// POST: Movies/Create
// To protect from overposting attacks, please enable the specific properties you want to bind to, for
// more details see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "ID,Name,ReleaseDate,Category")] Movie movie)
{
    if (ModelState.IsValid)
    {
        db.Movies.Add(movie);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(movie);
}

// GET: Movies/Edit/5
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Movie movie = db.Movies.Find(id);
    if (movie == null)
    {
        return HttpNotFound();
    }
    return View(movie);
}

// POST: Movies/Edit/5

```

```
// To protect from overposting attacks, please enable the specific properties you want to bind to, for
// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "ID,Name,ReleaseDate,Category")] Movie movie)
{
    if (ModelState.IsValid)
    {
        db.Entry(movie).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(movie);
}

// GET: Movies/Delete/5
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Movie movie = db.Movies.Find(id);
    if (movie == null)
    {
        return HttpNotFound();
    }
    return View(movie);
}

// POST: Movies/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    Movie movie = db.Movies.Find(id);
    db.Movies.Remove(movie);
}
```

```

        db.SaveChanges();
        return RedirectToAction("Index");
    }

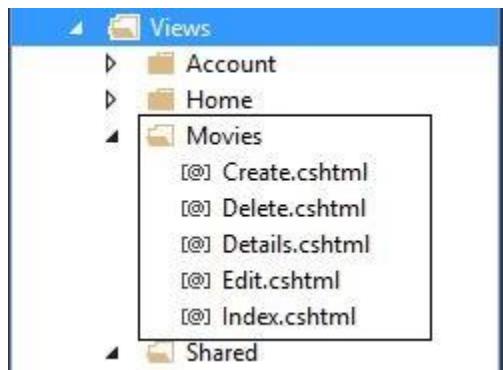
protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        db.Dispose();
    }
    base.Dispose(disposing);
}
}
}
}

```

In the code above, the *MoviesController* is defined that inherits from the Controller. All database accessing methods like Create(), Edit(), Delete() are defined automatically in this controller class.

View in MVC 5

When we use the scaffolding using the MVC 5 Controller with Views using Entity Framework, the Movies folder is automatically created in the Views folder. Check it out:



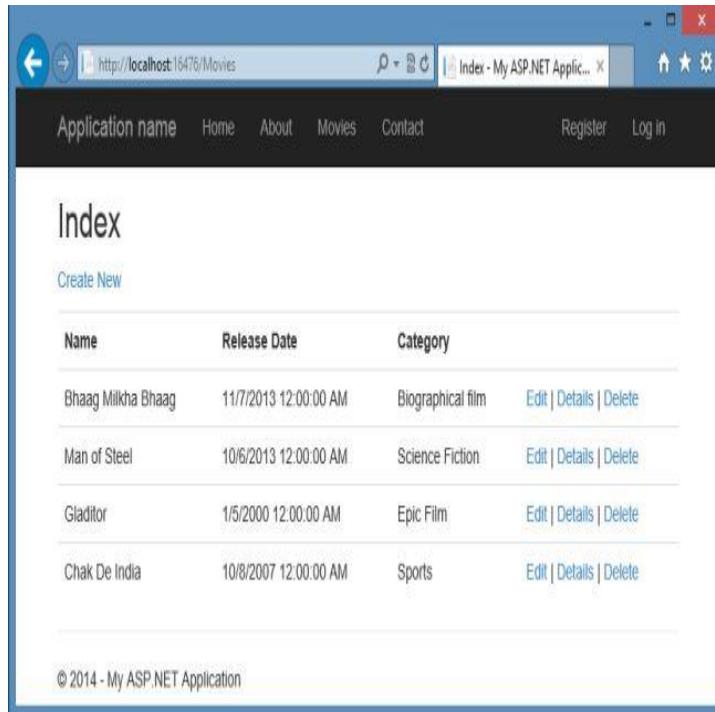
Now we add an ActionLink in the main layout page of our application to connect with the new controller. So, open the *_Layout.cshtml* file in the *Views/Shared* folder and edit the code with the following highlighted code:

```

<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("About", "About", "Home")</li>
        <li>@Html.ActionLink("Movies", "Index", "Movies")</li>
        <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
    </ul>
    @Html.Partial("_LoginPartial")
</div>

```

Now we run the application. Press F5 to run the application and open the Movies Controller and add some movies. When it done, the Index page will look such as follows:



Name	Release Date	Category	
Bhaag Milkha Bhaag	11/7/2013 12:00:00 AM	Biographical film	Edit Details Delete
Man of Steel	10/6/2013 12:00:00 AM	Science Fiction	Edit Details Delete
Gladitor	1/5/2000 12:00:00 AM	Epic Film	Edit Details Delete
Chak De India	10/8/2007 12:00:00 AM	Sports	Edit Details Delete

© 2014 - My ASP.NET Application

As you can see that the data is inserted into the table but we do not know which technique the Entity Framework is inserting the data, whether using the Stored Procedure or by T-SQL Statements? Well generally, it uses the T-SQL statements to insert the data because we didn't specify for it to use the Stored Procedure. Just proceed to the next section to use this.

18.5 LOG in Entity Framework

Now in this section we'll track what Entity Framework does behind the scenes. We'll add the `System.Diagnostics` so that we can see the result on the window Output Visual Studio at runtime.

Step 1: Update the controller with the following highlighted code:

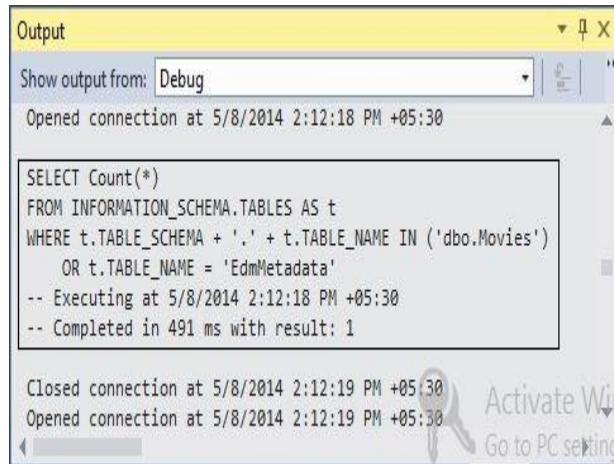
```
using System.Diagnostics;

namespace MvcStoredProcedureSample.Controllers
{
    public class MoviesController : Controller
    {
        private MovieDbContext db = new MovieDbContext();

        public MoviesController()
        {
            db.Database.Log = l => Debug.WriteLine(l);
        }

        // GET: Movies
        public ActionResult Index()
        {
            return View(db.Movies.ToList());
        }
    }
}
```

Step 2: Now run the project and open the controller again. When you are viewing the list of movies, do not close the browser and switch to your Visual Studio. Open the Output window and check out the SQL statement. Have a look:



The screenshot shows the 'Output' window in Visual Studio. It displays a SQL query being executed:

```

SELECT Count(*)
FROM INFORMATION_SCHEMA.TABLES AS t
WHERE t.TABLE_SCHEMA + '.' + t.TABLE_NAME IN ('dbo.Movies')
    OR t.TABLE_NAME = 'EdmMetadata'
-- Executing at 5/8/2014 2:12:18 PM +05:30
-- Completed in 491 ms with result: 1

```

Below the query, status messages indicate the connection state and execution time.

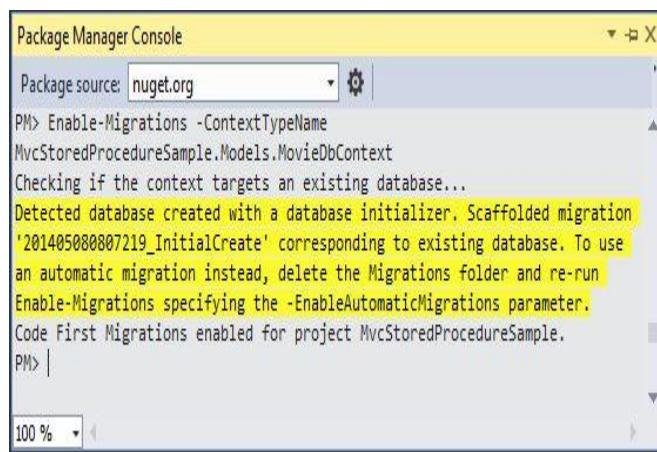
So now we'll use Stored Procedure in the next section.

Working with Stored Procedures

If we want to work with the Stored Procedure then we need to use the Code First Migrations that is very safe, smooth and productive. So use the following procedure.

Step 1: Open the Tools-> NuGet Package Manager->Package Manager Console and enter the following command:

Enable-Migrations



The screenshot shows the 'Package Manager Console' window. The command 'Enable-Migrations -ContextTypeName Mvc.StoredProcedureSample.Models.MovieDbContext' is entered. The output indicates that migrations are being scaffolded for an existing database named '201405080807219_InitialCreate'. It also notes that automatic migrations can be used instead by deleting the Migrations folder and re-running the command with the '-EnableAutomaticMigrations' parameter.

Step 2: Now the data context class will use the Stored Procedure. Open the Context class and update the code as shown below:

```

namespace Mvc.StoredProcedureSample.Models
{
    public class MovieDbContext : DbContext
    {
        public MovieDbContext() : base( "name = MovieDbContext" )
        {

        }

        public DbSet<Movie> Movies { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Movie>().MapToStoredProcedures();
        }
    }
}

```

Step 3: Build the solution. Now in the Package Manager Console enter the following command:

Add-Migration MyMovieSP

You can use any name in the place of *MyMovieSP*.

It creates the *201405080929139_MyMovieSP.cs* file and in which you can see the following code:

```

namespace Mvc.StoredProcedureSample.Migrations
{
    using System;
    using System.Data.Entity.Migrations;

    public partial class MyMovieSP : DbMigration
    {
        public override void Up()
        {
            CreateStoredProcedure(
                "dbo.Movie_Insert",

```

```

p => new
{
    Name = p.String(),
    ReleaseDate = p.DateTime(),
    Category = p.String(),
},
body:
@"INSERT [dbo].[Movies]([Name], [ReleaseDate], [Category])
VALUES (@Name, @ReleaseDate, @Category)

DECLARE @ID int
SELECT @ID = [ID]
FROM [dbo].[Movies]
WHERE @@ROWCOUNT > 0 AND [ID] = scope_identity()

SELECT t0.[ID]
FROM [dbo].[Movies] AS t0
WHERE @@ROWCOUNT > 0 AND t0.[ID] = @ID"
);

CreateStoredProcedure(
    "dbo.Movie_Update",
p => new
{
    ID = p.Int(),
    Name = p.String(),
    ReleaseDate = p.DateTime(),
    Category = p.String(),
},
body:
@"UPDATE [dbo].[Movies]
SET [Name] = @Name, [ReleaseDate] = @ReleaseDate, [Category] = @Category
WHERE ([ID] = @ID)"
);

CreateStoredProcedure(
    "dbo.Movie_Delete",

```

```

p => new
{
    ID = p.Int(),
},
body:
@"DELETE [dbo].[Movies]
WHERE ([ID] = @ID)"

};

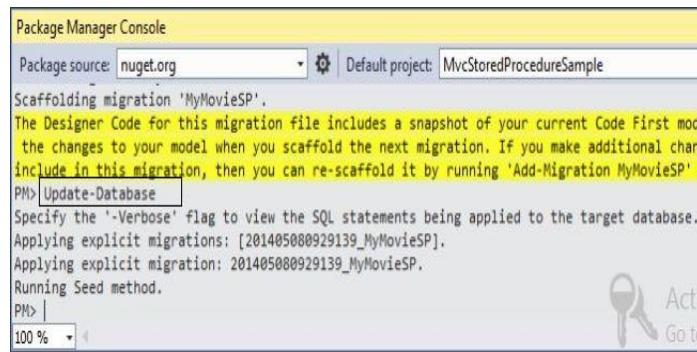
}

public override void Down()
{
    DropStoredProcedure("dbo.Movie_Delete");
    DropStoredProcedure("dbo.Movie_Update");
    DropStoredProcedure("dbo.Movie_Insert");
}
}
}
}

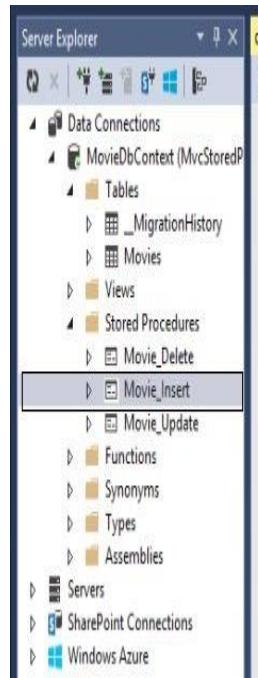
```

Step 4: We need to tell the database to create the *MyMovieSP*. So just enter the following command in the Package Manager Console:

Update-Database



Step 5: You can also check out the Stored Procedure from the Server Explorer. Check out the following screenshot:



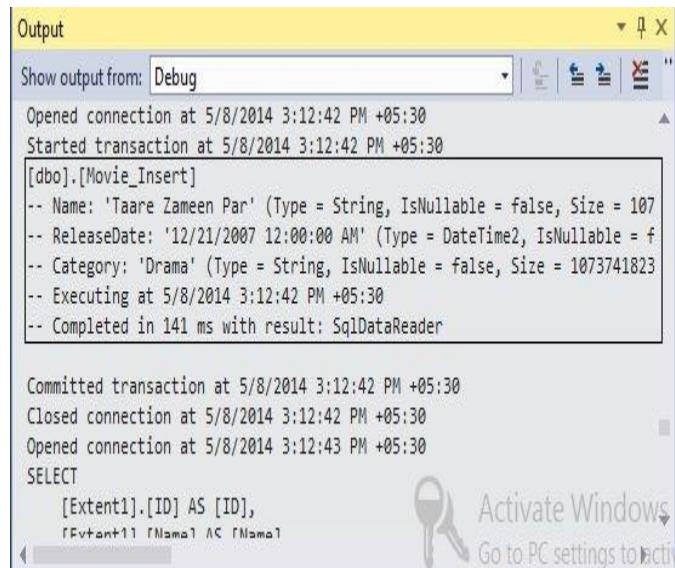
```

CREATE PROCEDURE [dbo].[Movie_Insert]
    @Name [nvarchar](max),
    @ReleaseDate [datetime],
    @Category [nvarchar](max)
AS
BEGIN
    INSERT [dbo].[Movies]([Name], [ReleaseDate], [Category])
    VALUES (@Name, @ReleaseDate, @Category)

    DECLARE @ID int
    SELECT @ID = [ID]
    FROM [dbo].[Movies]
    WHERE @@ROWCOUNT > 0 AND [ID] = scope_identity()

    SELECT t0.[ID]
    FROM [dbo].[Movies] AS t0
    WHERE @@ROWCOUNT > 0 AND t0.[ID] = @ID
END
  
```

Step 6: If you want to check out whether or not the Entity Framework is now using the Stored Procedure, run the application again and add some movies and at the same time check out the Output window:



```

Opened connection at 5/8/2014 3:12:42 PM +05:30
Started transaction at 5/8/2014 3:12:42 PM +05:30
[dbo].[Movie_Insert]
-- Name: 'Taare Zameen Par' (Type = String, IsNullable = false, Size = 107
-- ReleaseDate: '12/21/2007 12:00:00 AM' (Type = DateTime2, IsNullable = f
-- Category: 'Drama' (Type = String, IsNullable = false, Size = 1073741823
-- Executing at 5/8/2014 3:12:42 PM +05:30
-- Completed in 141 ms with result: SqlDataReader

Committed transaction at 5/8/2014 3:12:42 PM +05:30
Closed connection at 5/8/2014 3:12:42 PM +05:30
Opened connection at 5/8/2014 3:12:43 PM +05:30
SELECT
    [Extent1].[ID] AS [ID],
    [Extent1].[Name] AS [Name]
  
```



Chapter 19: ASP.Net MVC 5 Using Visual Basic: Getting Started

19.1 Introduction

There are two technologies available to create ASP.NET Web Applications that use the MVC Project Template: The first one is Visual C# and the second one is Visual Basic. Since I have created many chapter regarding MVC in Visual C#, you can also refer to them by [Creating MVC in Visual C#.](#)

In here, I am creating a series of chapter in which you'll learn to work with the ASP.NET MVC 5 Application using Visual Basic. In the first chapter, we'll learn to create the ASP.NET MVC Application in Visual Basic.

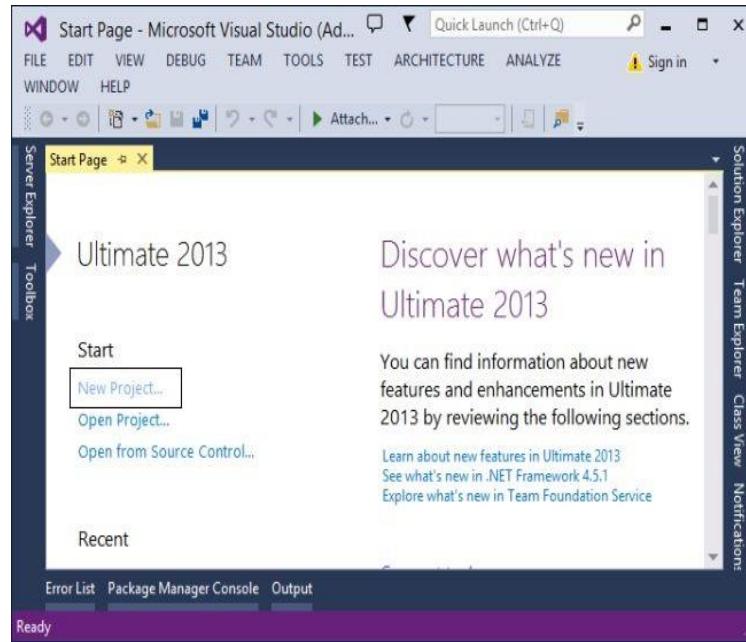
So, let's start with the following procedure:

- Creating MVC 5 Application
- Working with MVC Application
- Running the Application

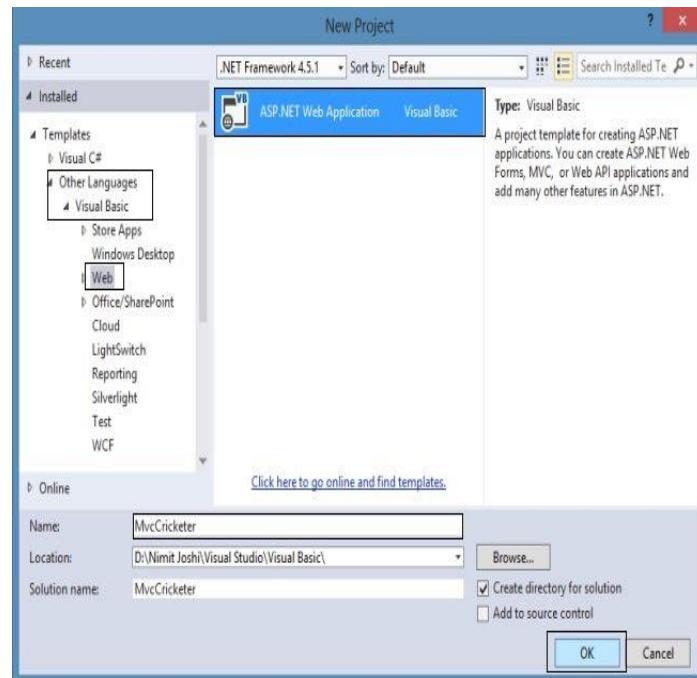
19.2 Creating MVC 5 Application

In this section, we'll create the ASP.NET Web Application using MVC Project Template. We'll create it using Visual Basic. Use the following procedure.

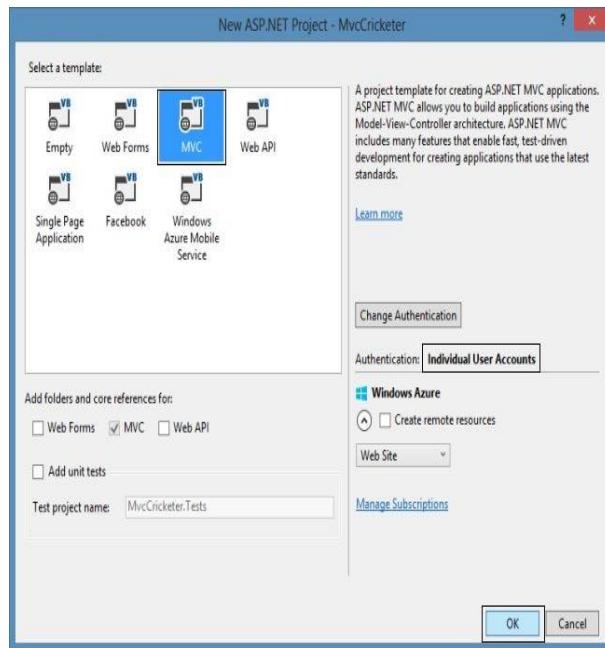
Step 1: Open Visual Studio 2013 and click on "New Project"



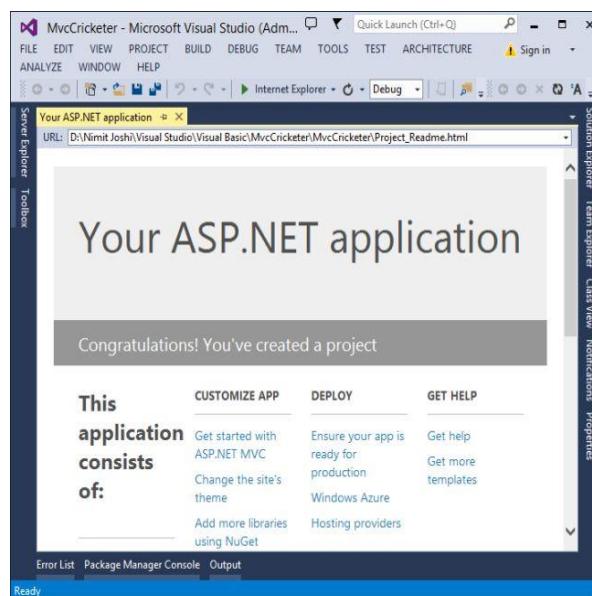
Step 2: In the left pane, expand the Other Templates and select the Web tab after expanding the Visual Basic tab. Just select the "ASP.NET Web Application" and enter the name for it.



Step 3: Select the MVC Project Template in the next "One ASP.NET" Wizard.



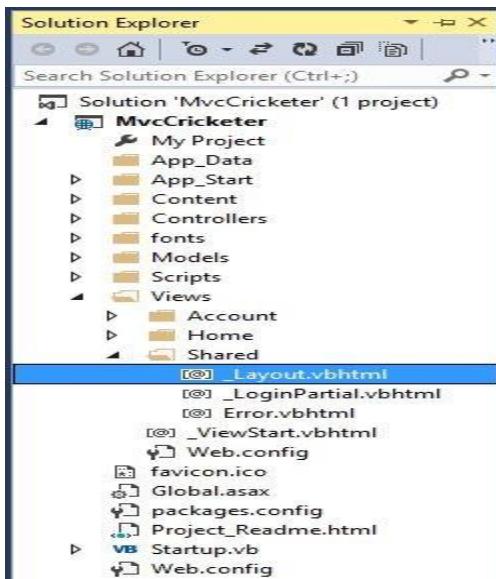
Visual Studio creates the application with the MVC and adds some files and folders in the solution. After creating the entire project, it provides the *Project_Readme.html* file with which you can find various chapter and blogs related to MVC.



19.3 Working with MVC Application

In this section, we'll edit the MVC application as per our project. By default a MVC application uses its default settings and if we want to develop the application as per our requirements then we need to change the application as we need. Use the following procedure to do that.

Step 1: Open the `_Layout.cshtml` file that provides the master layout for the application



Step 2: Change the code using the following highlighted code:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My Cricketer App</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
```

```

<div class="navbar-header">
    <button type="button" class="navbar-toggle"
data-toggle="collapse" data-target=".navbar-collapse">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
    </button>
    @Html.ActionLink("Mvc Cricketer", "Index", "Home",
New With {.area = ""}, New With {.class = "navbar-brand"})
</div>
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("About", "About", "Home")</li>
        <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
    </ul>
    @Html.Partial("_LoginPartial")
</div>
</div>
<div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
        <p>© @DateTime.Now.Year - Mvc Cricketer Application</p>
    </footer>
</div>

@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required:=False)
</body>
</html>

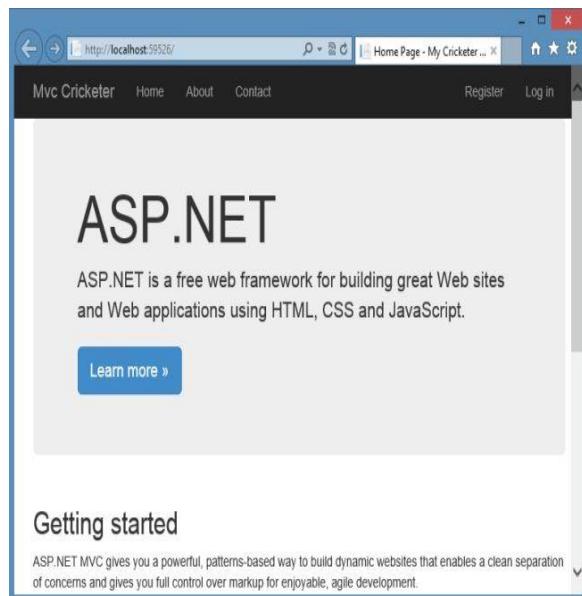
```

In the code, we have done some changes as per our project requirements.

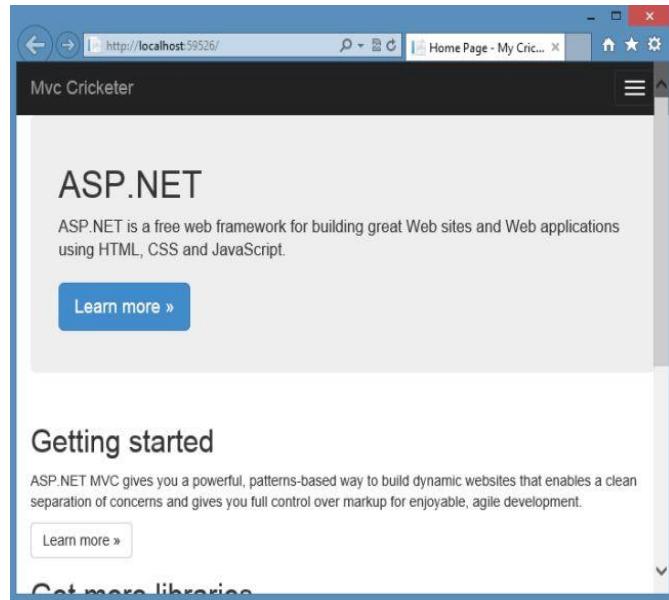
Step 3: Save the application.

19.4 Running the Application

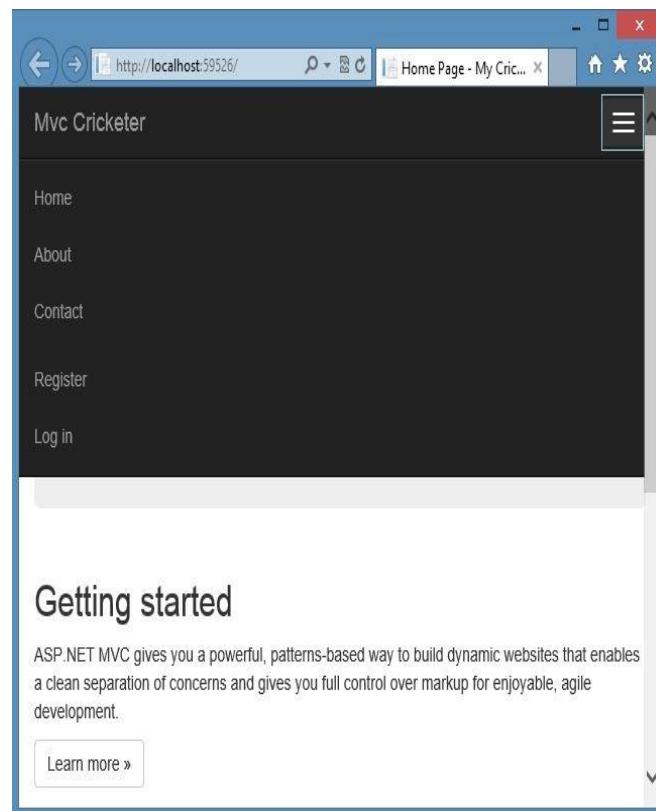
Now press Ctrl+F5 or F5 to debug the application. It starts IIS Express that will run the application in your local computer. You can check the browser address bar that shows the `http://localhost:port number/` because the localhost always points to the local computer. The port number is different and chosen randomly.



MVC 5 uses the Bootstrap template that provides the responsiveness to the application. Your MVC application is a responsive application, which means that even though you resize your browser, the content also resizes and fits the browser screen.



Now you can click on the icon to check the menu bar.



Chapter 20: ASP.Net MVC 5 Using Visual Basic: Adding Controller

20.1 Introduction

Today, I am creating the second chapter of my ASP.NET MVC 5 using Visual Basic Series and in this chapter we'll learn to add a new controller. In the previous chapter we learned [Getting Started with MVC 5 using Visual Basic](#).

Overview

Let me provide you a basic introduction to MVC. MVC is an acronym of Model View Controller. It is the design pattern for developing ASP.NET Web Applications and used to do the decoupled architecture. There are mainly three main characteristics used here, given below:

- **Model:** It represents the data (entities) defined in the web application and use the validation to implement the business rules for the data.
- **View:** It generates dynamically the HTML response for the application. These are the template files.
- **Controller:** It handles the request of the browser, communicate with the model and designates the view that is returned as a response to the browser.

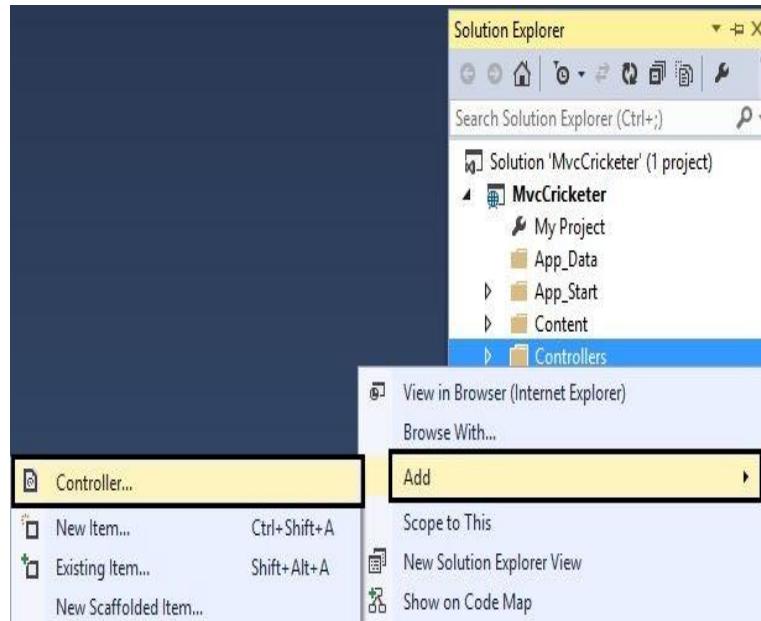
So, let's start to work with this and add a new controller with the following procedure:

- Adding Controller
- Working with Controller
- Running the Controller

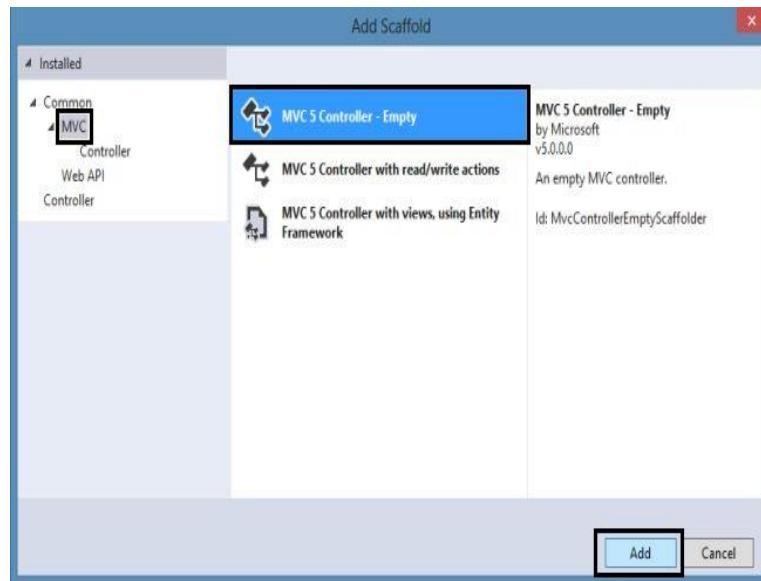
20.2 Adding Controller

In this section, we'll add the new controller to the Controllers folder in the application. Use the following procedure to do that.

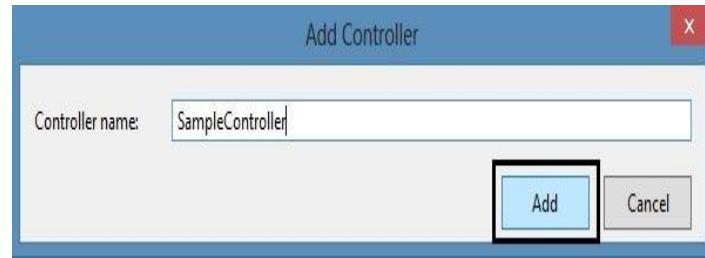
Step 1: Just right-click on the Controllers folder and click on the Controller



Step 2: Choose the MVC 5 Empty Controller in the Add Scaffold wizard.



Step 3: Enter the controller name as "*SampleController*".



Step 4: Edit the new controller code with the following code:

```

Imports System.Web.Mvc

Namespace Controllers
    Public Class SampleController
        Inherits Controller

        ' GET: Sample
        Function Index() As ActionResult
            Return View()
        End Function

        Function SampleCricketer() As String
            Return "This is Sample Controller of <b>Crickter</b> App"
        End Function

    End Class
End Namespace

```

In the code above, a new *SampleCricketer()* function is created. This function returns a string of HTML. Let's request the new controller method named *SampleCricketer()* in the browser.

Step 5: Run the application and enter the URL in the browser like: <http://localhost:59526/Sample/SampleCricketer>. The browser will look at the *SampleCricketer* method in the Sample Controller and it'll look like this:



20.3 Working with Controller

As you see in the URL, the browser is invoked by ASP.NET MVC. It can request many controllers that depend upon the URL. The default URL routing format used by MVC is as in the following:

/[Controller]/[ActionName]/[Parameters]

The default routing URL for the MVC application is as follows:

Public Module RouteConfig

```
Public Sub RegisterRoutes(ByVal routes As RouteCollection)
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}")

    routes.MapRoute(
        name:="Default",
        url:="{controller}/{action}/{id}",
        defaults:=New With {.controller = "Home",
                           .action = "Index", .id = UrlParameter.Optional}
    )
End Sub
End Module
```

You can find it in the *App_Start/RouteConfig* file. As you can see, the default controller route is the Home controller and the default action method is index.

The index action is the default method of the controller. If we want to invoke the Index action method in the browser, we do not need to write it in the browser URL.

You can proceed with the following procedure.

Step 1: Change the *RouteConfig* code as follows:

```
Public Module RouteConfig
    Public Sub RegisterRoutes(ByVal routes As RouteCollection)
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}")

        routes.MapRoute(
            name:="Default",
            url:="{controller}/{action}/{id}",
            defaults:=New With {.controller = "Sample",
                .action = "Index", .id = UrlParameter.Optional}
        )
    End Sub
End Module
```

In the code above, we change the controller to Sample. This would cause the Sample controller to run as a default.

Step 2: Change the Controller code as follows:

```
Imports System.Web.Mvc

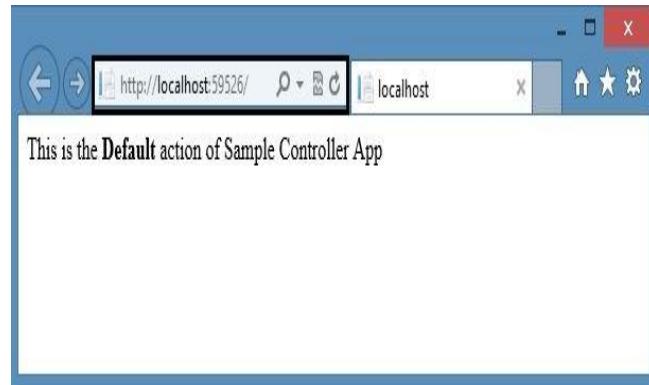
Namespace Controllers
    Public Class SampleController
        Inherits Controller

        ' GET: Sample
        Function Index() As String
            Return "This is the <b>Default</b> action of Sample Controller App"
        End Function

        Function SampleCricketer() As String
            Return "This is Sample Controller of <b>Crickter</b> App"
        End Function

    End Class
End Namespace
```

Step 3: Run the application and this time you do not need to enter the URL in the browser to run your controller. You have already defined it as a default URL.



20.4 Running the Controller

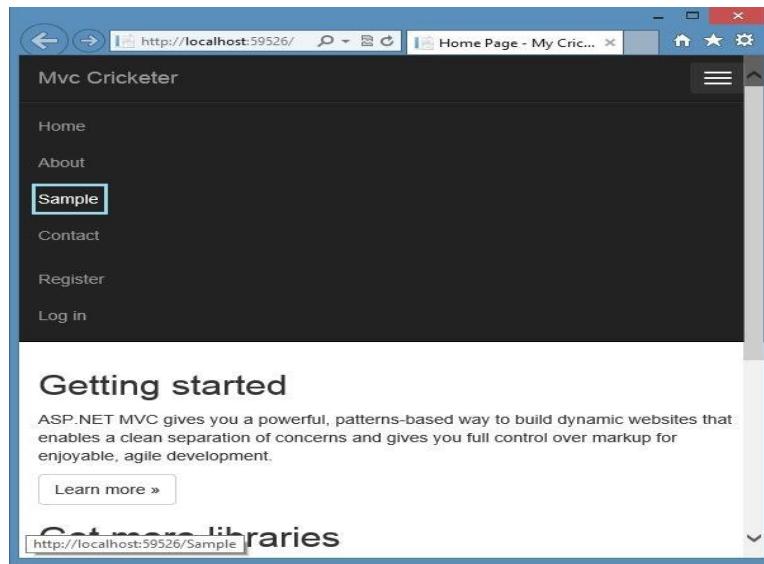
In this section we'll run the controller from the main MVC home page. So just restore the editing in the *RouteConfig* file. Follow the procedure below.

Step 1: Restore the default code in the *RouteConfig* file.

Step 2: Open the *Views/Shared/_Layout.cshtml* file and update the code from the highlighted code below:

```
<ul class="nav navbar-nav">
    <li>@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("About", "About", "Home")</li>
    <li>@Html.ActionLink("Sample", "Index", "Sample")</li>
    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
</ul>
```

Step 3: Run the application. From the Home Page open the controller named Sample.



Step 4: Now you can view your *Index()* method that returns the HTML string as shown below:



See, this time the URL of the browser shows the controller name. You do not need to pass the Index in the URL also.

21 New Release Candidate Update For ASP.Net MVC 5, Web API 2 and Web Pages 3

21.1 Introduction

Last week the Microsoft ASP.NET team released the latest updates for ASP.NET MVC 5, Web API 2 and Web Pages 3. Now the ASP.NET MVC 5.2, Web API 2.2 and Web Pages 3.2 are available on the NuGet gallery.

Prerequisites

There are the following prerequisites to use these updates:

- For Visual Studio 2013 users, [Visual Studio 2013 Update 1](#)
- For Visual Studio 2012 users, [ASP.NET Web Tools 2013.1 for VS 2012](#)

Download

You can install this releases or update your versions to ASP.NET MVC 5.2, ASP.NET Web API 2.2 and ASP.NET Web Pages 3.2 using the Package Manager Console.

Please enter the following commands to install:

- Install-Package Microsoft.AspNet.Mvc -Version 5.2.0-rc -Pre
- Install-Package Microsoft.AspNet.WebApi -Version 5.2.0-rc -Pre
- Install-Package Microsoft.AspNet.WebPages -Version 3.2.0-rc -Pre

As an example I've installed the ASP.NET Web API rc update on the sample application:

```

Package Manager Console
Package source: [nuget.org] - Default project: WebApplication24
Type 'get-help NuGet' to see all available NuGet commands.

PM> Install-Package Microsoft.AspNet.WebApi -Version 5.2.0-rc -Pre
Attempting to resolve dependency 'Microsoft.AspNet.WebApi.WebHost (>= 5.2.0-rc && < 5.2.1)'.
Attempting to resolve dependency 'Microsoft.AspNet.WebApi.Core (>= 5.2.0-rc && < 5.2.1)'.
Attempting to resolve dependency 'Microsoft.AspNet.WebApi.Client (>= 5.2.0-rc)'.
Attempting to resolve dependency 'Newtonsoft.Json (>= 4.5.11)'.
Successfully installed 'Newtonsoft.Json 4.5.11'.
Successfully added 'Newtonsoft.Json 4.5.11' to WebApplication24.
Installing 'Microsoft.AspNet.WebApi.Client 5.2.0-rc'.
You are downloading Microsoft.AspNet.WebApi.Client from Microsoft, the license agreement can be found at http://www.microsoft.com/web/eula/net_library_eula_ENU.htm. Check the package dependencies, which may come with their own license agreement(s). Your use of the constituents of this package constitutes your acceptance of their license agreements. If you do not accept the license agreement(s), you may delete the relevant components from your device.
Successfully installed 'Microsoft.AspNet.WebApi.Client 5.2.0-rc'.
Installing 'Microsoft.AspNet.WebApi.Core 5.2.0-rc'.
You are downloading Microsoft.AspNet.WebApi.Core from Microsoft, the license agreement can be found at http://www.microsoft.com/web/eula/net_library_eula_ENU.htm. Check the package dependencies, which may come with their own license agreement(s). Your use of the constituents of this package constitutes your acceptance of their license agreements. If you do not accept the license agreement(s), you may delete the relevant components from your device.
Successfully installed 'Microsoft.AspNet.WebApi.Core 5.2.0-rc'.
Installing 'Microsoft.AspNet.WebApi.WebHost 5.2.0-rc'.
You are downloading Microsoft.AspNet.WebApi.WebHost from Microsoft, the license agreement can be found at http://www.microsoft.com/web/eula/net_library_eula_ENU.htm. Check the package dependencies, which may come with their own license agreement(s). Your use of the constituents of this package constitutes your acceptance of their license agreements. If you do not accept the license agreement(s), you may delete the relevant components from your device.
Successfully installed 'Microsoft.AspNet.WebApi.WebHost 5.2.0-rc'.
Successfully added 'Microsoft.AspNet.WebApi 5.2.0-rc'.
Adding 'Newtonsoft.Json 4.5.11' to WebApplication24.
Adding 'Microsoft.AspNet.WebApi.Client 5.2.0-rc' to WebApplication24.
Adding 'Microsoft.AspNet.WebApi.Core 5.2.0-rc' to WebApplication24.
Adding 'Microsoft.AspNet.WebApi.WebHost 5.2.0-rc' to WebApplication24.
Successfully added 'Microsoft.AspNet.WebApi.WebHost 5.2.0-rc' to WebApplication24.
Adding 'Microsoft.AspNet.WebApi 5.2.0-rc' to WebApplication24.
Successfully added 'Microsoft.AspNet.WebApi 5.2.0-rc' to WebApplication24.

100% - 

```

Overview of Release

There are various new features and bug fixes included in this release that bring many more virtues to MVC, the Web API and Web Pages. Now we'll see the features that are included in this release in the following.

21.2 ASP.NET MVC 5.2 Release Candidate

There are the following features included in this release.

Attribute Routing

Now it provides an extensibility point called `IDirectRouteProvider` that allows full control over how attribute routes are discovered and configured. This provides a list of actions and controllers and the route information is also associated, that specifies what routing configuration is desired for those actions. The customization of the `IDirectRouteProvider` is easy by enhancing the default implementation of `DefaultDirectRouteProvider`.

Feature Updates

There are many types of feature updates available now in this release candidate version. A few of them are given below:

- `IDirectRouteProvider`
- `ValidationSummary()`

- HTML DropDownList
- jQuery Unobtrusive Validation

There are also many types of bug fixes in this release.

21.3 ASP.NET Web API 2.2 Release Candidate

There are the following features included in this release.

OData v4 support

There are the following features added in this OData v4 protocol:

- Support for aliasing properties in OData Model
- Added OData function support
- Integrate with ODL UriParser
- Support for open complex type
- Added Attribute routing support

Attribute Routing

Now it provides an extensibility point called IDirectRouteProvider, that allows full control over how attribute routes are discovered and configured. This provides a list of actions and controllers and the route information is also associated that specifies what routing configuration is desired for those actions. The customization of the IDirectRouteProvider is easy by enhancing the default implementation of DefaultDirectRouteProvider.

For example of DefaultDirectRouteProvider:

```
using System.Collections.Generic;
using System.Web.Http;
using System.Web.Http.Controllers;
using System.Web.Http.Routing;
namespace WebApplication24
{
    public class BaseController : ApiController
```

```
{
```

```
//statements;  
}  
  
[RoutePrefix("api/values")]  
public class ValuesController : BaseController  
{  
    //statements for values  
}  
  
public class CustomDirectRoute : DefaultDirectRouteProvider  
{  
    protected override IReadOnlyList<IDirectRouteFactory> GetActionRouteFactories(HttpActionDescriptor action  
)  
    {  
        return action.GetCustomAttributes<IDirectRouteFactory>  
(inherit: true);  
    }  
}
```

21.4 Web API Client Support for Windows Phone 8.1

Now we can use the Web API client in Windows Phone 8.1 applications. We can easily get the Web API Client from the NuGet Gallery.

Feature Updates

There are a few types of feature updates available now in this release candidate version. A few of them are given below:

- ApiController should accept related URI
- Support added for Windows Phone 8.1

ASP.NET Web Pages 3.2 Release Candidate

There is a slight update in this section. The MakeTypeHelper.ObjectToDictionary feature is updated on this release.

Chapter 22: Implement ASP.Net Web API 2 in ASP.Net MVC 5

22.1 Introduction

In this chapter we'll learn how to access the Web API 2 in the ASP.NET MVC 5 application. Previously we have already learned how to work with the ASP.NET Web API 2.

Therefore in this chapter we'll create the Web API and access the Web API using MVC 5.

So, let's proceed with the following procedure:

- Create ASP.NET Web API Application
- Working with Web API Application
- Working with MVC Controller
- Adding MVC View

Prerequisites

To create this application, there are the following prerequisites:

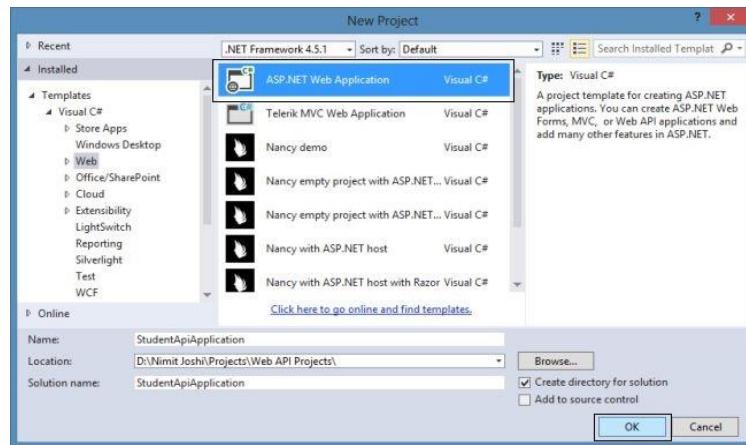
- Visual Studio 2013
- ASP.NET MVC 5

22.2 Create ASP.NET Web API Application

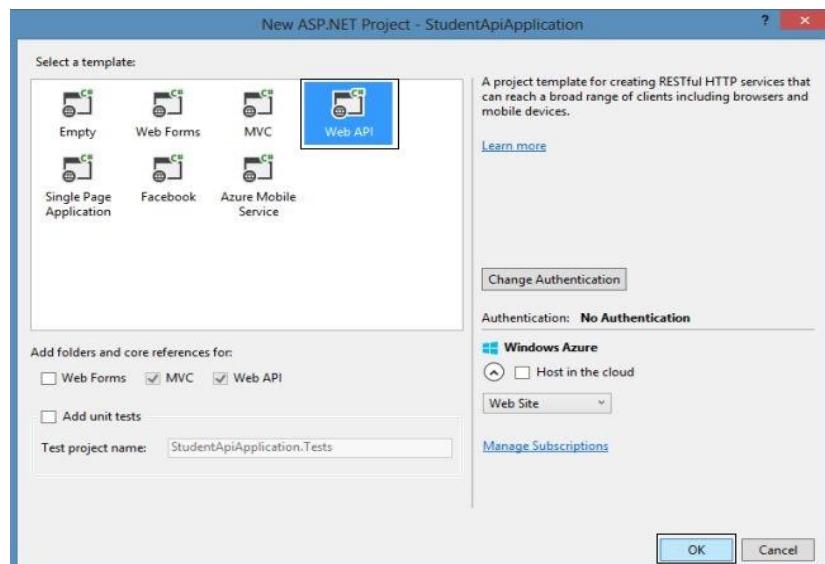
In this section we'll create the ASP.NET Web Application using the Web API Project Template. So, start with the following procedure.

Step 1: Open Visual Studio and click on New Project.

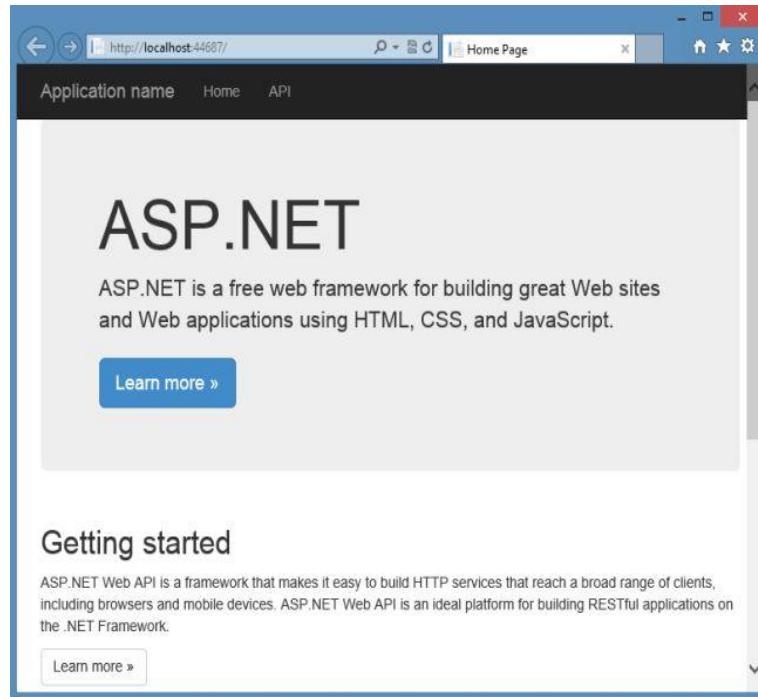
Step 2: Select the ASP.NET Web Application and enter the name for the application.



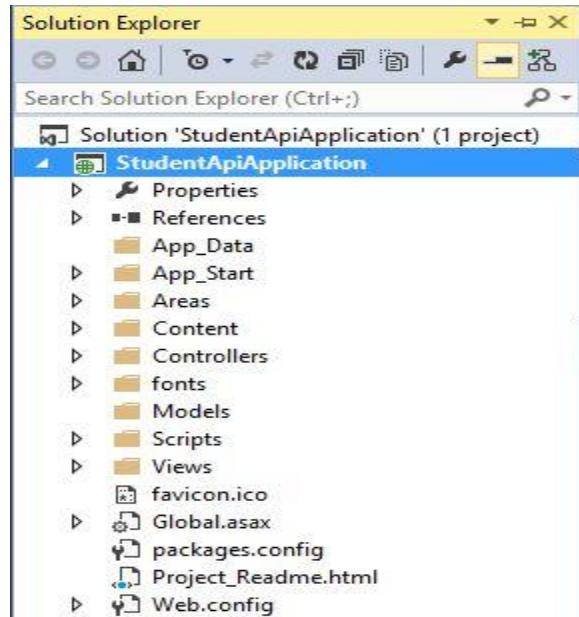
Step 3: Select Web API Project Template and tick the check box of MVC and click OK.



Visual Studio automatically creates the Web API application using the MVC 5 based projects. When you run the application, the following is the home page of your application.



You can also click on the Web API option to view the default API structure in the browser. The application Solution Explorer contains all the files and folders associated with the Web API application.



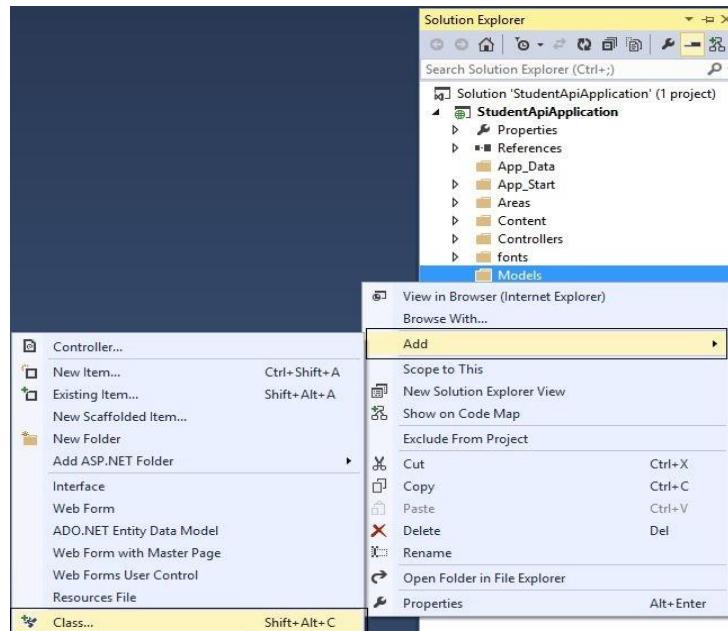
Working with Web API Application

Now, we have created the Web API application that uses the MVC 5 based architecture. Now we'll proceed with the following procedure.

22.3 Adding Model

In this section, we'll add a class and an interface for creating the model. Use the procedure described below.

Step 1: In the Solution Explorer, right-click on the Models folder and click on Add to add class named *Student*.



Step 2: Replace the code with the following code:

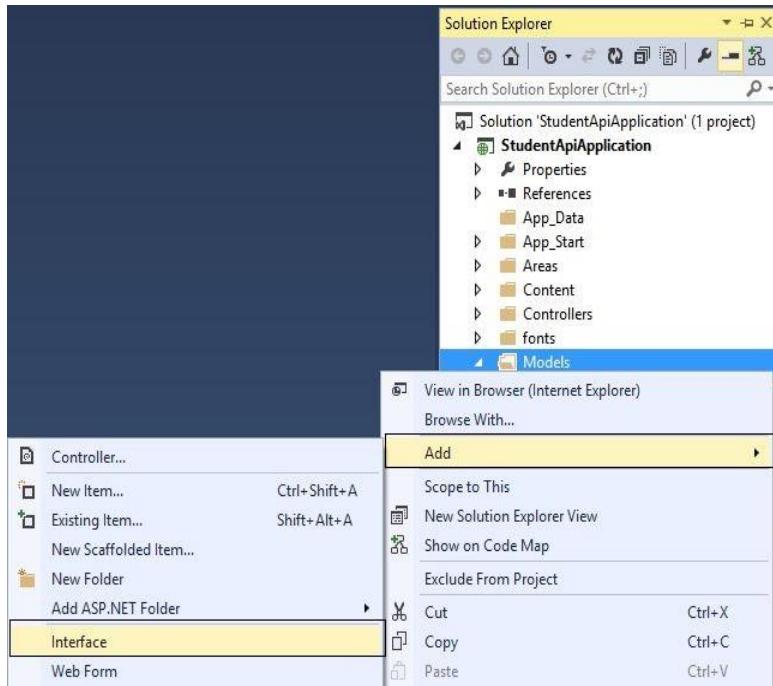
```

1. using System;
2.
3. namespace StudentApiApplication.Models
4. {
5.     public class Student
6.     {
7.         public int ID { get; set; }
8.         public string Name { get; set; }
9.         public string City { get; set; }
  
```

```

10.     public string Course { get; set; }
11. }
12. }
```

Step 3: Now add an interface to the models folder named *IStudentRepository*.



Step 4: Replace the code with the following code:

```

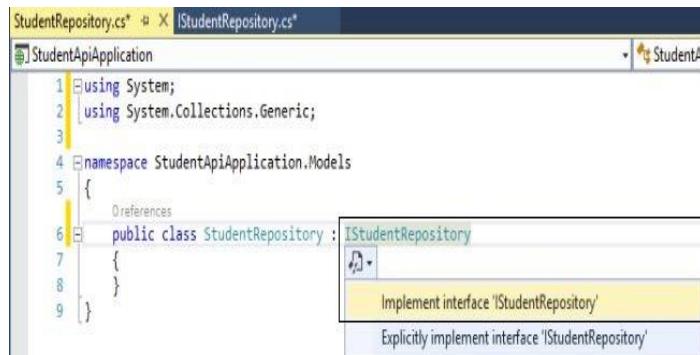
1. using System;
2. using System.Collections.Generic;
3.
4. namespace StudentApiApplication.Models
5. {
6.     interface IStudentRepository
7.     {
8.         IEnumerable<Student> GetAllStudents();
9.         Student AddStudent(Student student);
10.    }
11. }
```

Step 5: Now we'll add a repository class in the models folder and after adding it implement the interface in this class using the following code:

```

1. public class StudentRepository : IStudentRepository
2. {
3. }
```

Step 6: Now implement the interface using the following screenshot:



Now modify the code with the following code:

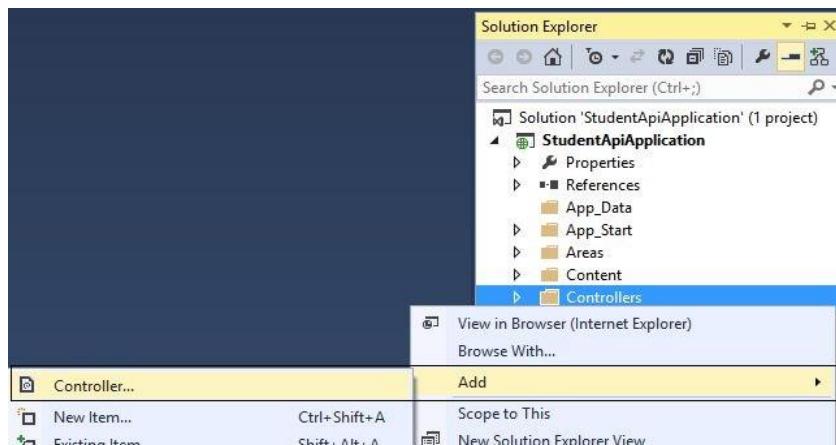
```

1. using System;
2. using System.Collections.Generic;
3.
4. namespace StudentApiApplication.Models
5. {
6.     public class StudentRepository : IStudentRepository
7.     {
8.         private List<Student> items = new List<Student>();
9.         private int next = 1;
10.        public StudentRepository()
11.        {
12.            AddStudent(new Student { ID = 1, Name = "Ashish", City = "New Delhi", Course = "B.Tech" });
13.            AddStudent(new Student { ID = 2, Name = "Nimit", City = "Noida", Course = "MCA" });
14.            AddStudent(new Student { ID = 3, Name = "Prawah", City = "Dehradun", Course = "B.Tech" });
15.            AddStudent(new Student { ID = 4, Name = "Sumit", City = "Nainital", Course = "MCA" });
16.        }
17.
18.        public IEnumerable<Student> GetAllStudents()
19.        {
20.            return items;
21.        }
22.
23.        public Student AddStudent(Student student)
24.        {
25.            if (items == null)
26.            {
27.                throw new ArgumentNullException("student");
28.            }
29.
30.            student.ID = next++;
31.            items.Add(student);
32.            return student;
33.        }
34.    }
35. }
```

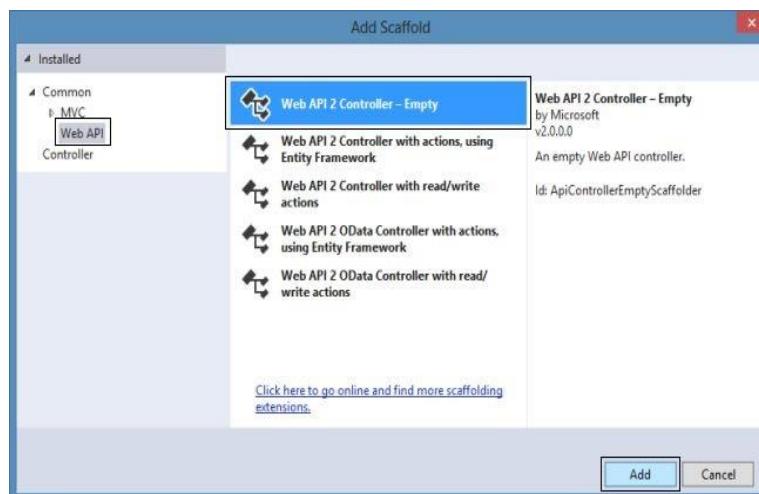
22.4 Adding Controller

So far we have created the model, now we'll create the controller to use the model. Use the following procedure.

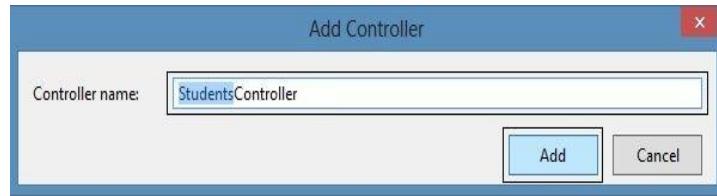
Step 1: In the Solution Explorer, right-click on the Controllers folder and click on Add to add the Controller.



Step 2: In the Add Scaffold wizard, select the Web API 2 Empty Controller.



Step 3: Specify the name of the controller.



Step 4: Replace the code with the following code:

```

1. using StudentApiApplication.Models;
2. using System.Collections.Generic;
3. using System.Web.Http;
4.
5. namespace StudentApiApplication.Controllers
6. {
7.     public class StudentsController : ApiController
8.     {
9.         static readonly IStudentRepository studentRepository = new StudentRepository();
10.
11.        public IEnumerable<Student> GetAll()
12.        {
13.            return studentRepository.GetAllStudents();
14.        }
15.    }
16. }
```

Working with MVC Controller

In this section, we'll add an action method to the existing controller. You can also create a new controller. So now open the *HomeController.cs* in the Controllers folder and modify it with the following code:

```

1. using System.Web.Mvc;
2.
3. namespace StudentApiApplication.Controllers
4. {
5.     public class HomeController : Controller
6.     {
7.         public ActionResult Index()
8.         {
9.             ViewBag.Title = "Home Page";
10.
11.            return View();
12.        }
13.
14.        public ActionResult Data()
15.        {
16.            ViewBag.Title = "Student Details";
17.            return View();
18.        }
19. }
```

```

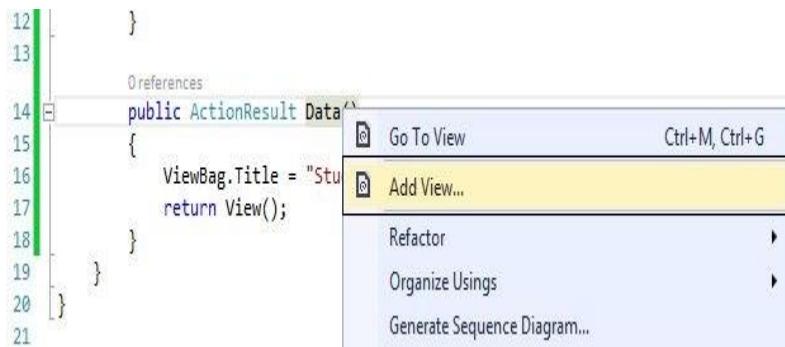
18.    }
19.    }
20. }
```

In the code above, I've added an extra *ActionResult* method that is returning the View.

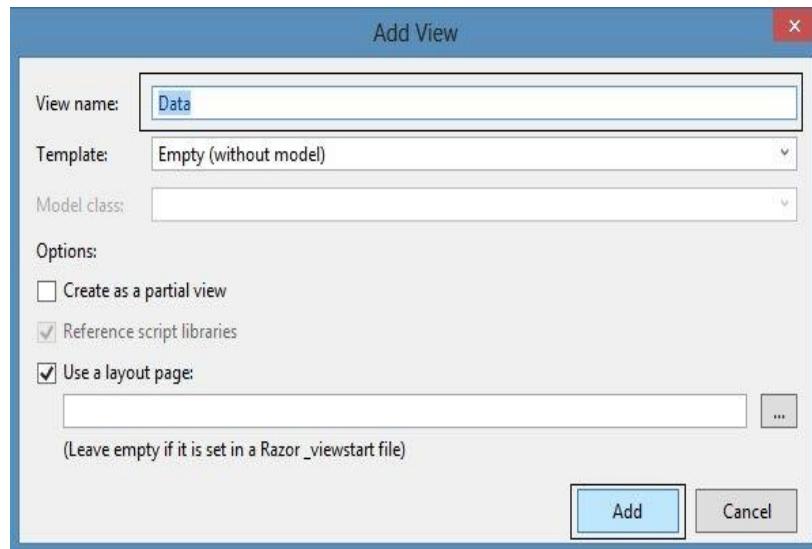
22.5 Adding MVC View

We'll add the MVC view here and work with the newly added view. Use the following procedure.

Step 1: In the *HomeController* class, right-click on the *Data()* and click on Add View.



Step 2: Enter Data for view name and click on Add.



Step 3: Replace the code with the following code:

```

1.  @{
2.    ViewBag.Title = "Data";
3. }
4.
5. <script src="~/Scripts/jquery-1.10.2.min.js"></script>
6. <script type="text/javascript">
7.   $(document).ready(function () {
8.     $.ajax({
9.       url: "http://localhost:44687/api/students",
10.      type: "Get",
11.      success: function (data) {
12.        for (var i = 0; i < data.length; i++) {
13.          $("<tr><td>" + data[i].Name + "</td><td>" + data[i].Course + "</td><td>" + data[i].City + "</td></tr>").ap
14.          pendTo("#students");
15.        }
16.      },
17.      error: function (msg) { alert(msg); }
18.    });
19.  </script>
20.
21. <h2>Index</h2>
22.
23. <div id="body">
24.   <section class="content-wrapper main-content">
25.     <table id="students" class="table">
26.       <thead>
27.         <tr>
28.           <th>@Html.DisplayName("Name")</th>
29.
30.           <th>@Html.DisplayName("Course")</th>
31.
32.           <th>@Html.DisplayName("City")</th>
33.         </tr>
34.       </thead>
35.     </table>
36.   </section>
37. </div>

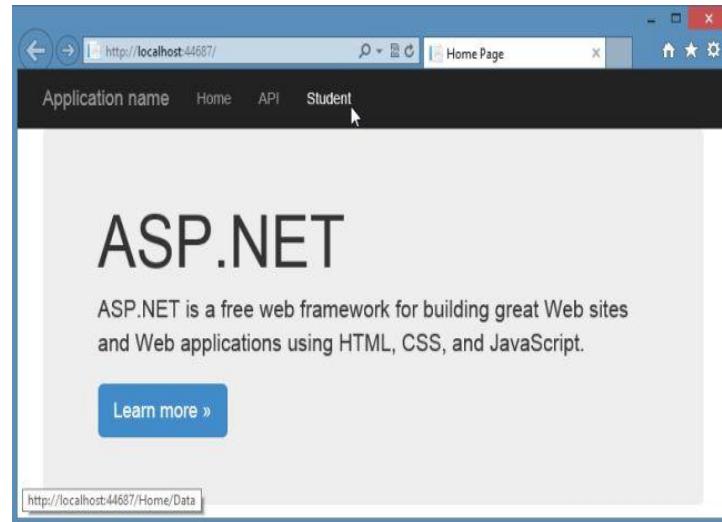
```

In the code above, I've used the jQuery to call the Web API.

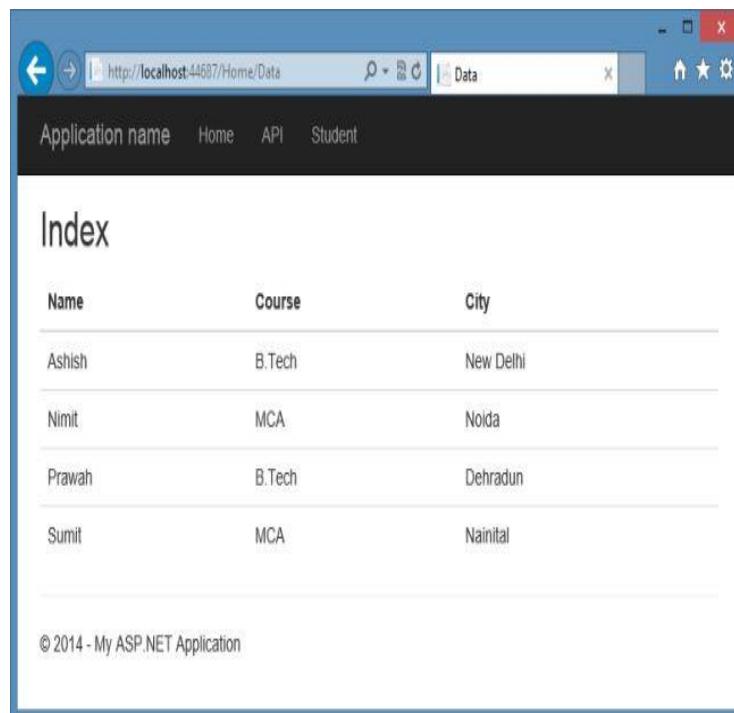
Step 4: Open the *Views/Shared/_Layout.cshtml* page and add an ActionLink to redirect to your View.

```
1.  <li>@Html.ActionLink("Student", "Data", "Home", new { area = "" }, null)</li>
```

Step 5: Now run the application. In the Home Page, click on the Student link.



You can see in the following screenshot that, the data is coming by the Web API.

A screenshot of a web browser window showing a table of student data. The title bar says "http://localhost:44687/Home/Data". The page has a dark header with "Application name" and three links: "Home", "API", and "Student". The "Student" link is highlighted with a cursor. The main content area has a heading "Index" and a table with three columns: "Name", "Course", and "City". There are four rows of data:

Name	Course	City
Ashish	B.Tech	New Delhi
Nimit	MCA	Noida
Prawah	B.Tech	Dehradun
Sumit	MCA	Nainital

At the bottom of the page, there's a copyright notice: "© 2014 - My ASP.NET Application".

Chapter 23: Model First Approach in ASP.Net MVC 5

23.1 Introduction

As we know there are various approaches for the Entity Framework available with which we can connect with the database from the ASP.NET web application. There are generally three approaches available, given below:

- Code First Approach
- Database First Approach
- Model First Approach

In this chapter we will work with the Model First Approach using the Entity Framework 6 and using this approach we will connect with the database from the web application. We will create the ASP.NET web application using the MVC Project Template. I am using the Visual Studio 2013 and in this IDE, we use the MVC 5 Project Template to create the web application. I have created an application in which I used the [Database First Approach](#) to connect with the database.

Entity Framework 6 Overview

The Entity Framework 6.1 is the latest version of Entity Framework. We will work here in the version 6 of Entity Framework. The following are some features of Entity Framework 6:

- Async Query and Save
- Connection Resiliency
- Code Based Configuration
- Dependency Resolution
- Multiple Contexts per Database

You can get the full description of Entity Framework 6 from [here](#).

I am using the Visual Studio 2013 to work with the Entity Framework 6. We can create the application using the Entity Framework 6 that targets .NET 4.0 or .NET 4.5. I am using .NET 4.5.1.

Getting Started

We will now create the MVC 5 application using the Model First Approach. Use the following procedure:

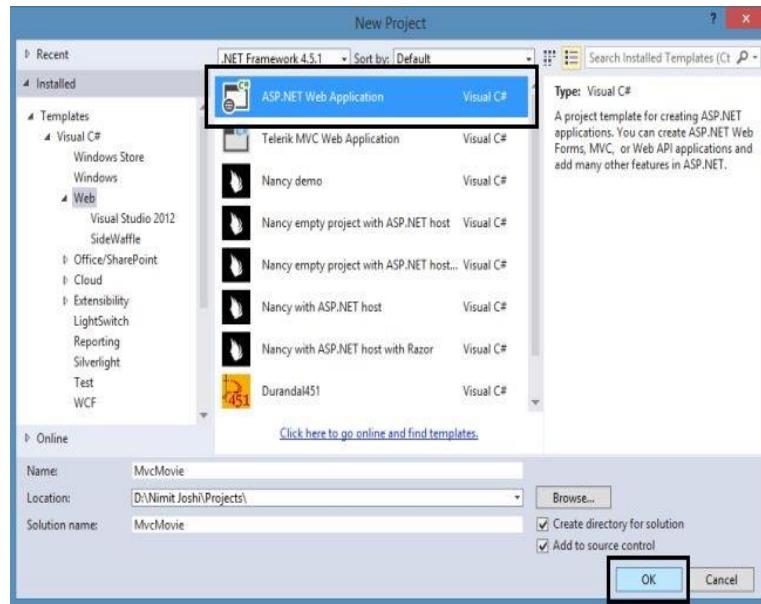
- Creating Web Application
- Adding Entity Data Model
- Working with Entity Data Model Designer
- Working with Controller and Views
- do CRUD Operations
- Working with Database

23.2 Creating Web Application

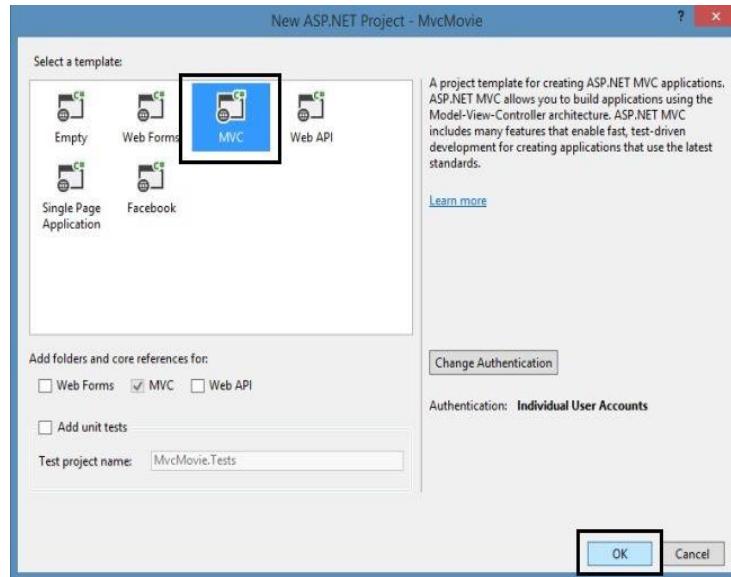
In this section, we will create the ASP.NET web application using the MVC Project Template. Follow the steps given below.

Step 1: Open Visual Studio 2013 and click on New Project.

Step 2: Select the Web tab from the left pane and select the ASP.NET web application and specify the application as "MvcMovie"



Step 3: Select the MVC Project Template and click on OK.

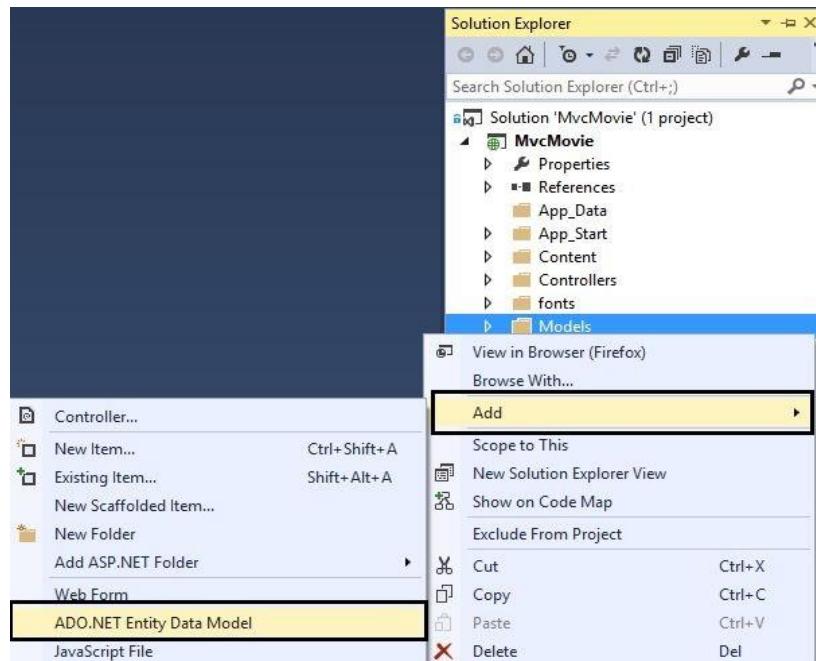


Visual Studio automatically creates the MVC application and adds many files and folders to the project. You can check it out in the Solution Explorer.

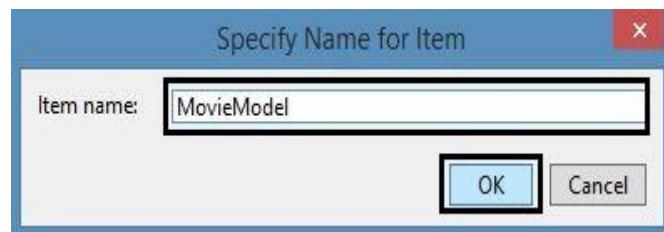
23.3 Adding Entity Data Model

In this section, we will add the ADO.NET Entity Data Model to the application. We will create the empty model in here. Use the following procedure.

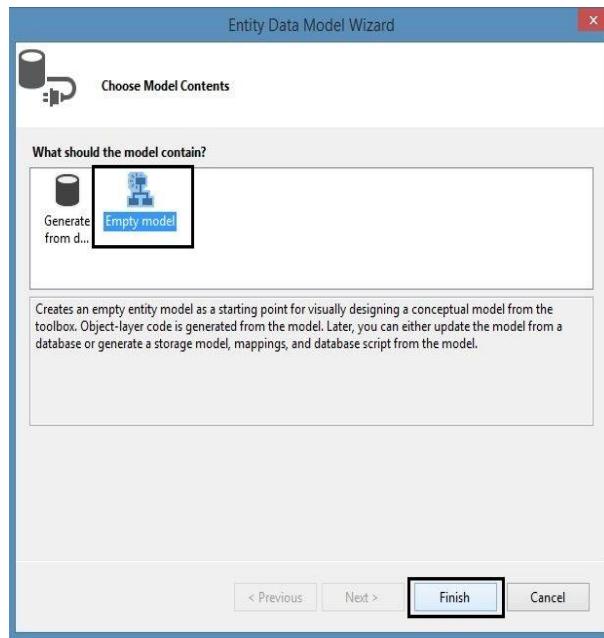
Step 1: In the Solution Explorer, right-click on the Models folder and click on ADO.NET Entity Data Model.



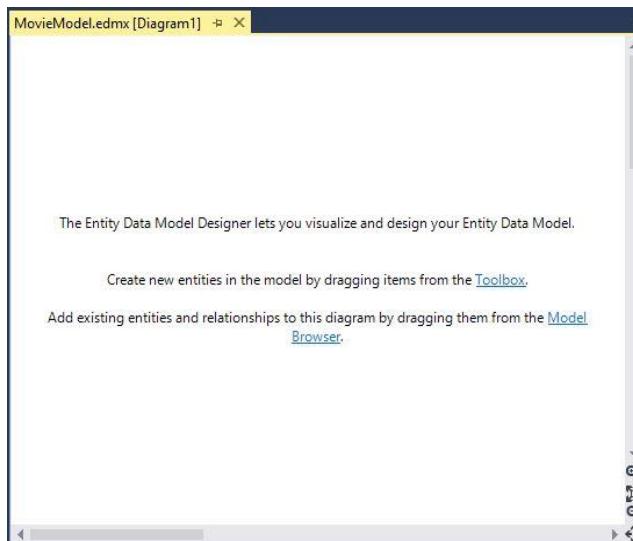
Step 2: Specify the model name.



Step 3: In the next Entity Data Model Wizard, select the Empty Model and click on "Finish".



Now you can see the Entity Data Model Designer.



23.4 Working with Entity Data Model Designer

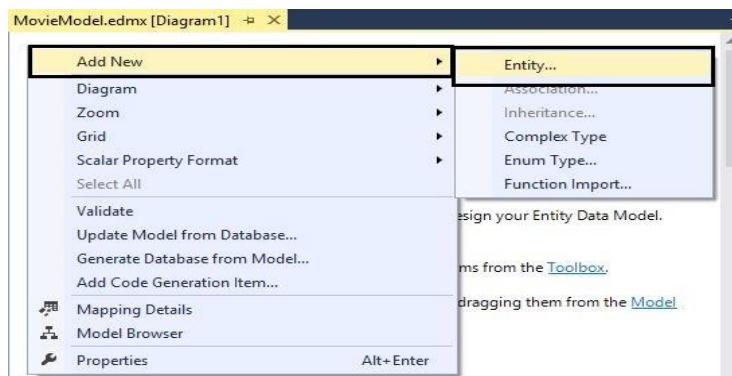
In this section, we'll add the entity in the Entity Data Model Designer. We will proceed in this section using following two sections:

- Adding Entity Model
- Generate Database from Model

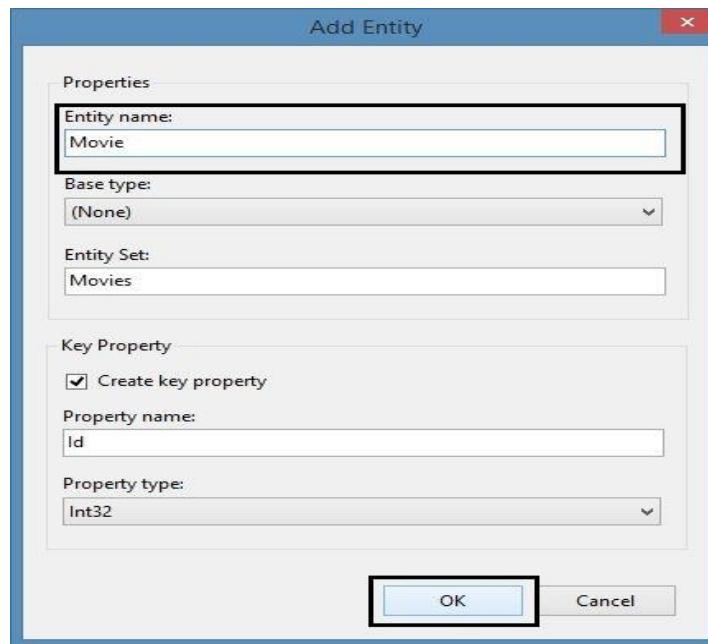
Adding Entity Model

In this section, we'll create the entity model using the following procedure.

Step 1: Right-click on the Data Model Designer and go to add new entity as shown below:

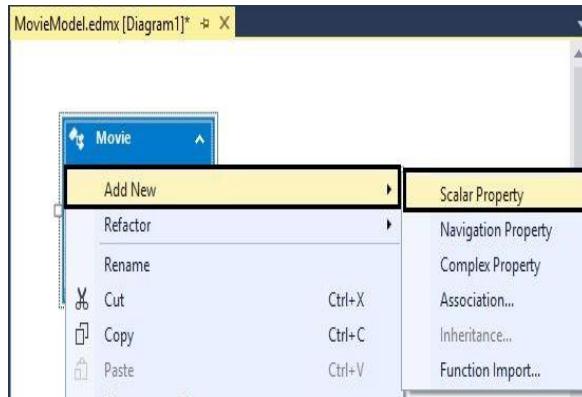


Step 2: In the next Add Entity wizard, specify the entity name.

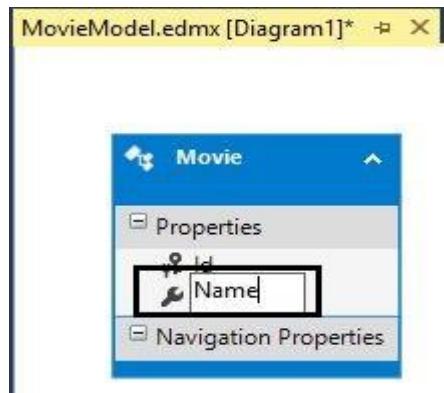


Note: You can change the name of Entity Set that is generated automatically.

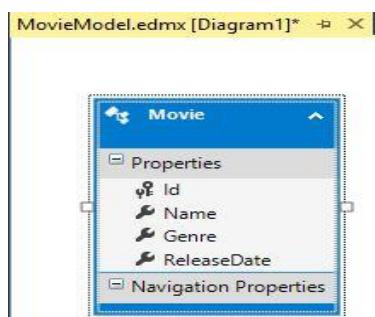
Step 3: Now you can see that the entity is created. So, it is time to add a scalar property to the entity. Right-click on the entity and go to add new scalar property.



Change the name of the property.



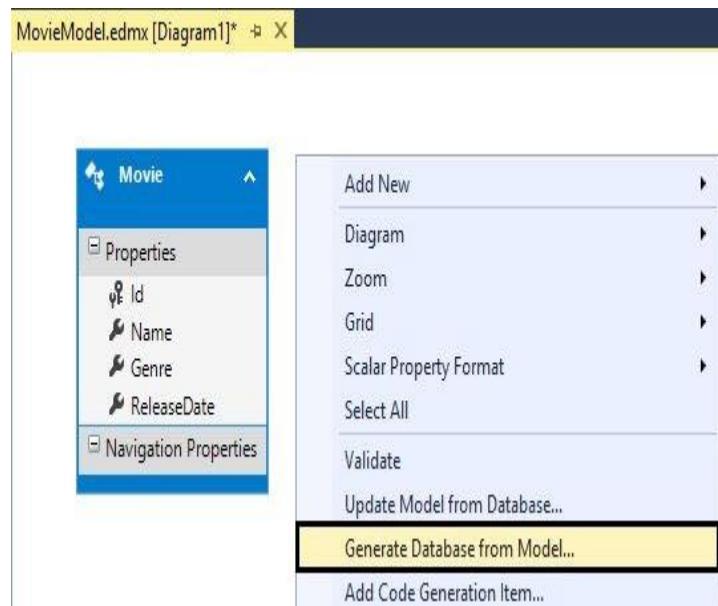
Step 4: Add more scalar properties to the entity.



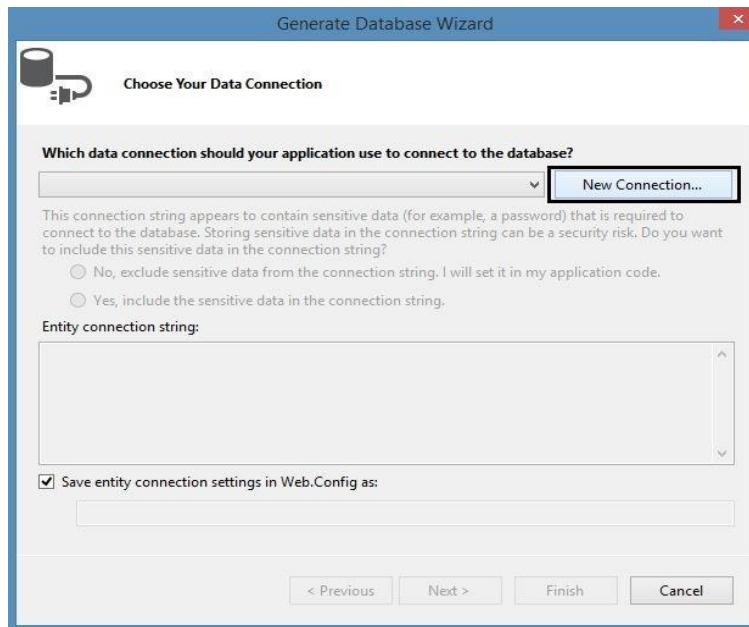
Generate Database from Model

Now we have generated an entity model. You can also create more entity models in the Data Model Designer. We will now generate the database from the model. Use the following procedure.

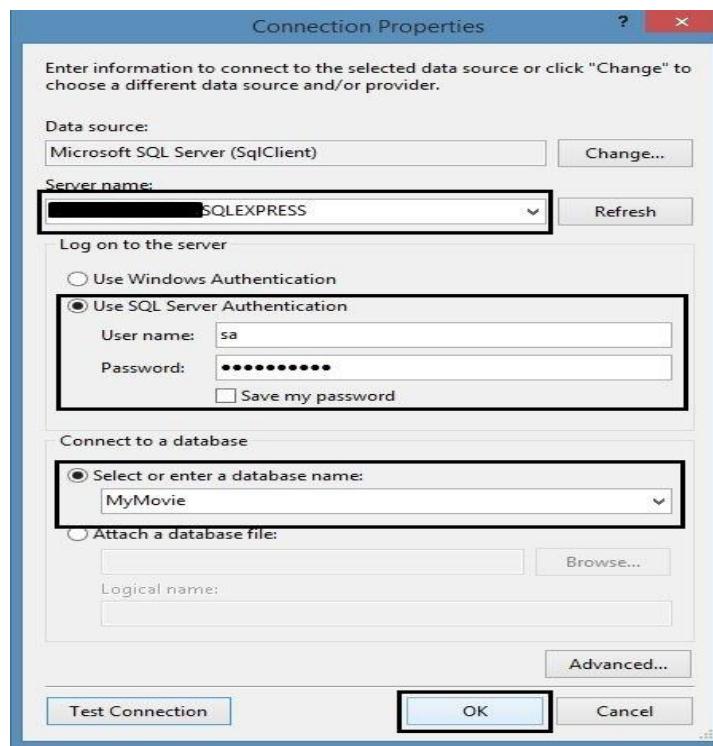
Step 1: Right-click on the Data Model Designer surface and select the Generate Database from Model.



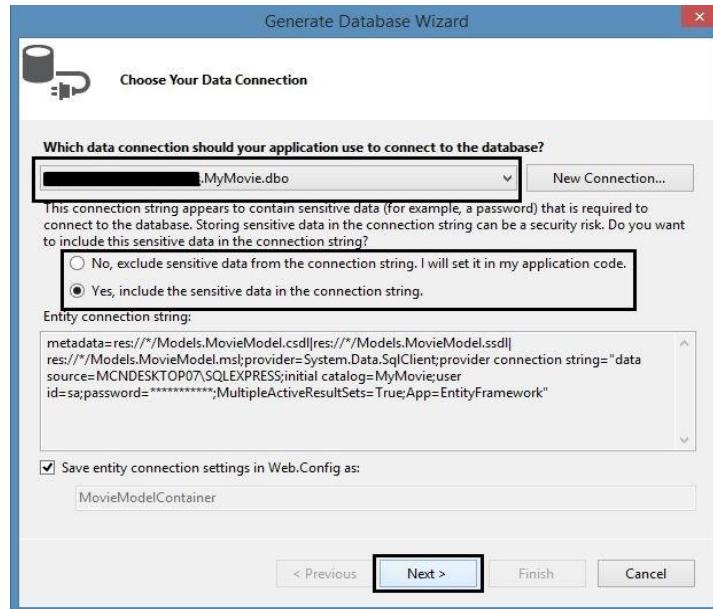
Step 2: Now in the Generate Database Wizard, click on New Connection to connect with the database.



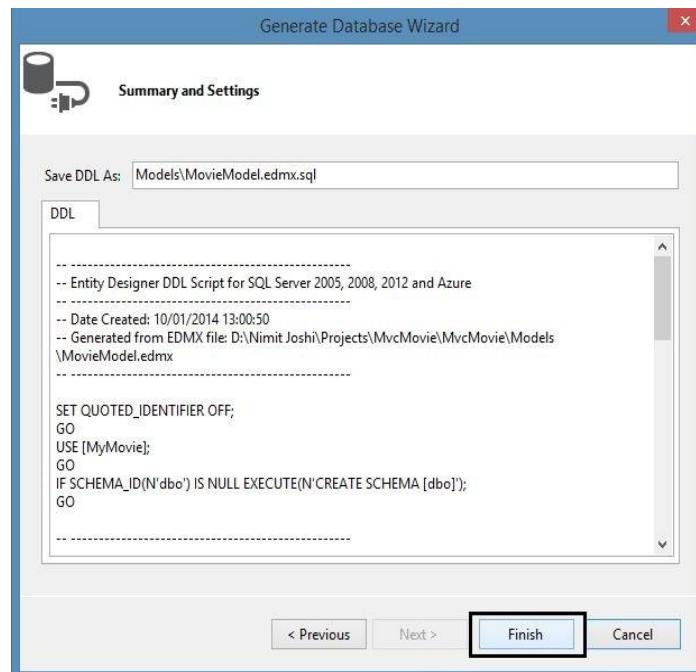
Step 3: In the next Connection Properties wizard, specify the server name and the specify the database name.



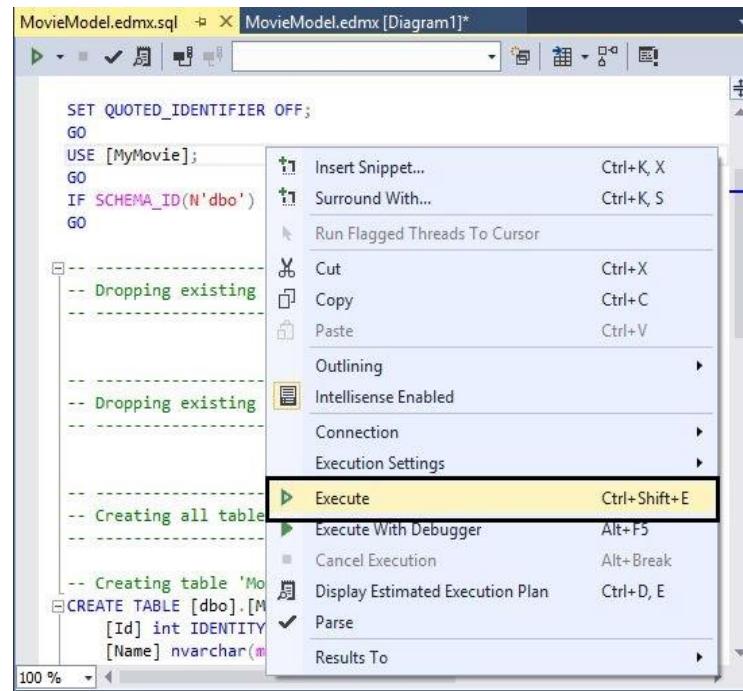
Step 4: Now in the Generate Database Wizard, you can see that the connection is created. Click on the Next button.



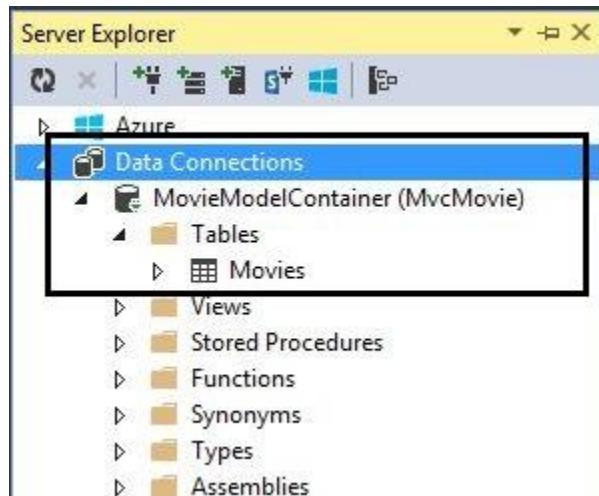
Step 5: The database summary and script is generated and click on Finish.



Step 6: Now right-click on the script and click on Execute.



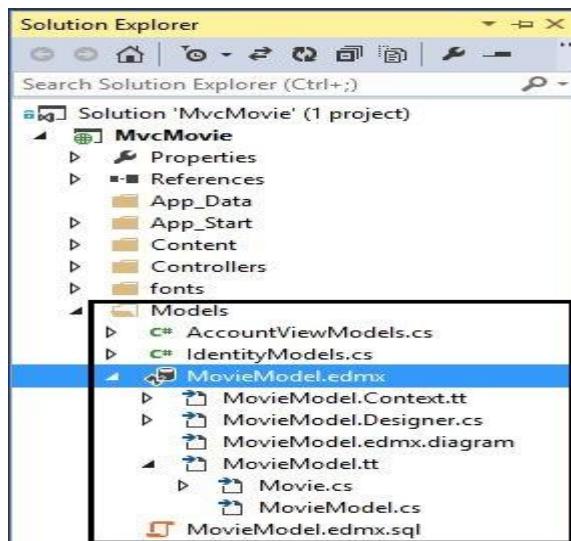
After executing the database will have been created. You can see your database in the Server Explorer.



Working with Controller and Views

In this section we will work with the controller and views in MVC 5. So far we have done the work with the database. We will now add the MVC 5 controller and view using the Entity Framework. So let's get started with the following procedure.

Step 1: Build the solution at first. When you build the solution then you can find the model class in the Models folder.



The Movie class is generated in the models folder. Edit the code with the following highlighted code:

```

1. //-----
2. //<auto-generated>
3. // This code was generated from a template.
4. //
5. // Manual changes to this file may cause unexpected behavior in your application.
6. // Manual changes to this file will be overwritten if the code is regenerated.
7. //</auto-generated>
8. //-----
9.
10. namespace MvcMovie.Models
11. {
12.     using System;
13.     using System.Collections.Generic;
14.     using System.ComponentModel.DataAnnotations;
15.
16.     public partial class Movie
17.     {
18.         public int Id { get; set; }

```

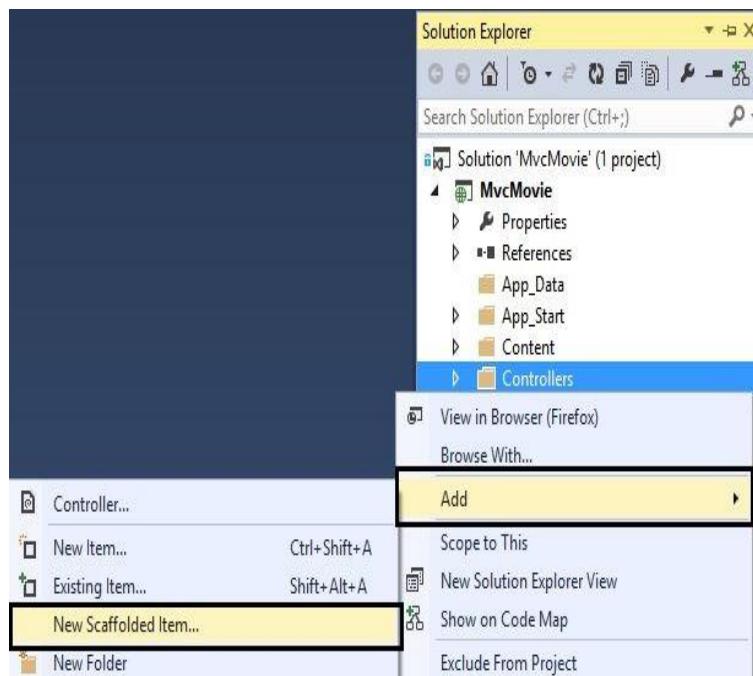
```

19.     public string Name { get; set; }
20.     public string Genre { get; set; }
21.     [Display(Name="Release Date")]
22.     [DataType(DataType.Date)]
23.     public System.DateTime ReleaseDate { get; set; }
24. }
25.

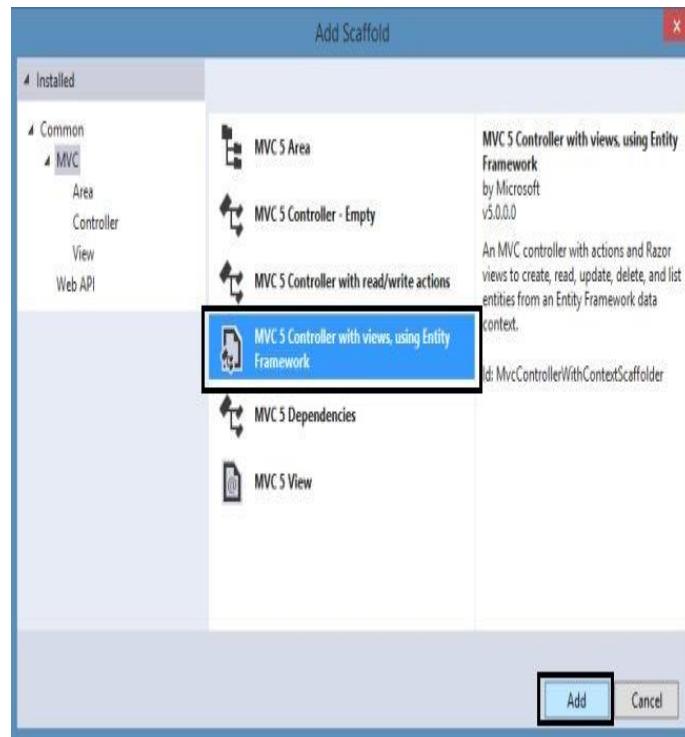
```

In the code above, I've added the Data Annotation in which I am changing the display name of the property and date property initialized.

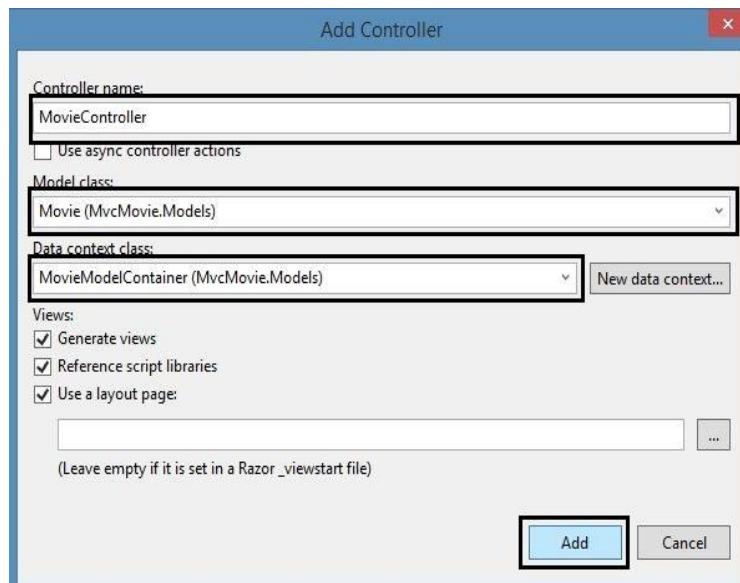
Step 2: In the Solution Explorer, right-click on the Controller and go to Add and then New Scaffolded Item.



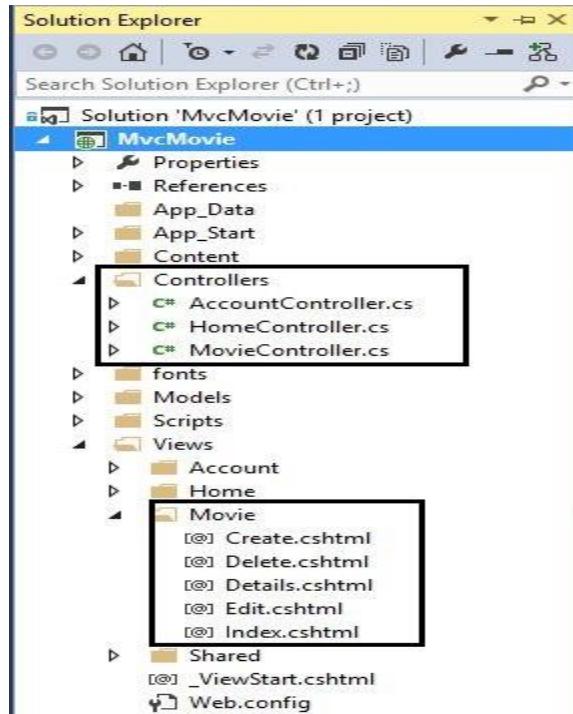
Step 3: In the Add Scaffold wizard, select the MVC 5 Controller using the Entity Framework with Views.



Step 4: In the Add Controller Wizard, specify the controller name, select the model class and select the Data Context class and then click Add.



When you click on Add, the *MovieController* class is generated in the Controllers folder and the Movie folder is generated in the Views folder. Have a look:



23.5 do CRUD Operations

In this section we will do the CRUD (Create, Read, Update and Delete) operations. So let's start the following procedure.

At first we will create an Action Link to our View. Go to the *Views->Shared->_Layout.cshtml* file and edit the code with the following highlighted code:

```

1. <html>
2. <head>
3.   <meta charset="utf-8" />
4.   <meta name="viewport" content="width=device-width, initial-scale=1.0">
5.   <title>@ViewBag.Title - Movie App</title>
6.   @Styles.Render("~/Content/css")
7.   @Scripts.Render("~/bundles/modernizr")
8.
9. </head>
10. <body>
```

```

11. <div class="navbar navbar-inverse navbar-fixed-top">
12.   <div class="container">
13.     <div class="navbar-header">
14.       <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
15.         <span class="icon-bar"></span>
16.         <span class="icon-bar"></span>
17.         <span class="icon-bar"></span>
18.       </button>
19.       @Html.ActionLink("Best Movies", "Index", "Home", null, new { @class = "navbar-brand" })
20.     </div>
21.     <div class="navbar-collapse collapse">
22.       <ul class="nav navbar-nav">
23.         <li>@Html.ActionLink("Home", "Index", "Home")</li>
24.         <li>@Html.ActionLink("About", "About", "Home")</li>
25.         <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
26.         <li>@Html.ActionLink("Movies", "Index", "Movie")</li>
27.       </ul>
28.       @Html.Partial("_LoginPartial")
29.     </div>
30.   </div>
31. </div>
32. <div class="container body-content">
33.   @RenderBody()
34.   <hr />
35.   <footer>
36.     <p>© @DateTime.Now.Year - My Movie Application</p>
37.   </footer>
38. </div>
39.
40. @Scripts.Render("~/bundles/jquery")
41. @Scripts.Render("~/bundles/bootstrap")
42. @RenderSection("scripts", required: false)
43. </body>
44. </html>

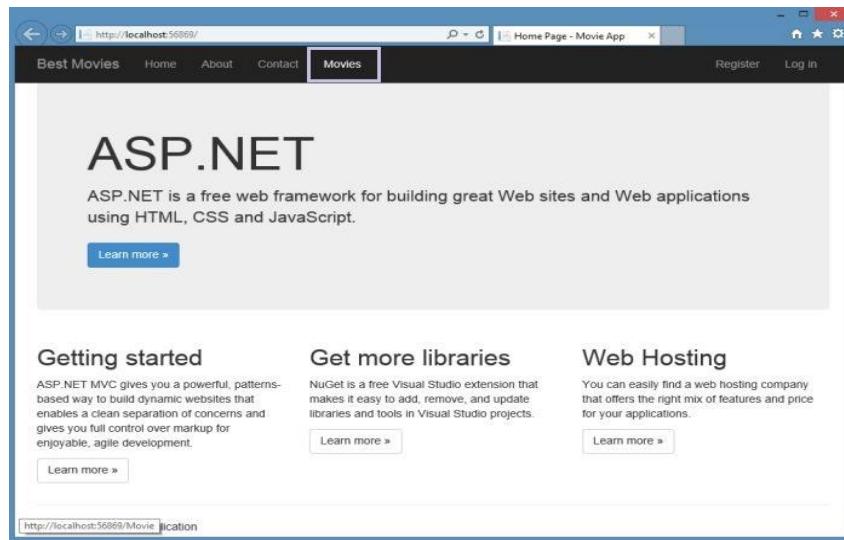
```

In the code above, we have created an action link and edited the app name and title.

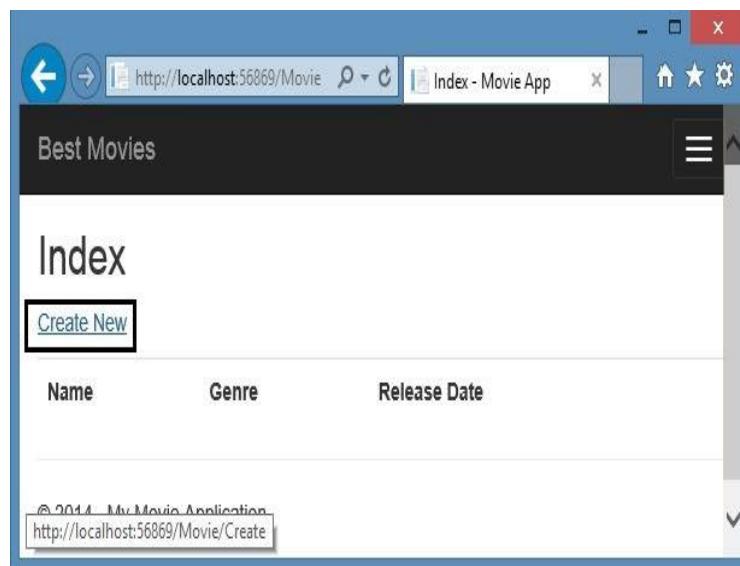
Create

In this section, we will do the Create operation.

Step 1: Run the application and in the home page of the application click on Movies.



Step 2: In the next window, click on the Create New link to create the record.



Step 3: Add a record as shown below:

Create

Movie

Name
Bhaag Milkha Bhaag

Genre
Biographic Film

Release Date
07/11/2013

[Back to List](#)

Add a few more records using the same procedure.

Read

Whenever you click on the Movies link, you can read the corresponding records.

Index

[Create New](#)

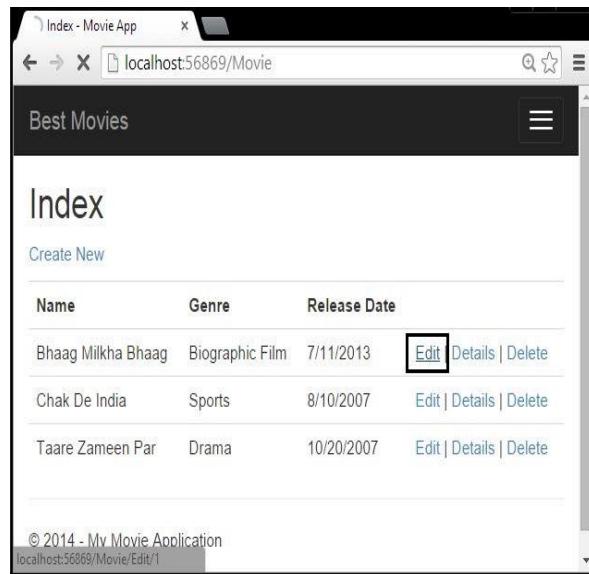
Name	Genre	Release Date	
Bhaag Milkha Bhaag	Biographic Film	7/11/2013	Edit Details Delete
Chak De India	Sports	8/10/2007	Edit Details Delete
Taare Zameen Par	Drama	10/20/2007	Edit Details Delete

© 2014 - My Movie Application

Update

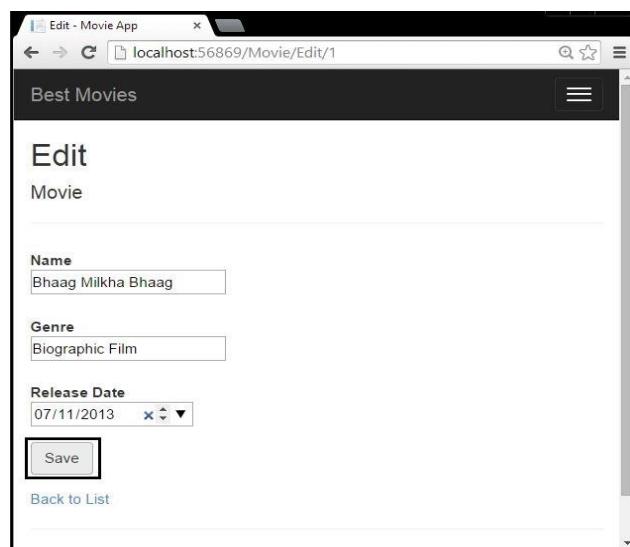
We can update the corresponding record using the following procedure.

Step 1: Click on the Edit link in that record.



Name	Genre	Release Date	
Bhaag Milkha Bhaag	Biographic Film	7/11/2013	Edit Details Delete
Chak De India	Sports	8/10/2007	Edit Details Delete
Taare Zameen Par	Drama	10/20/2007	Edit Details Delete

Step 2: Edit the record and click on the Save button.



Edit
Movie

Name
Bhaag Milkha Bhaag

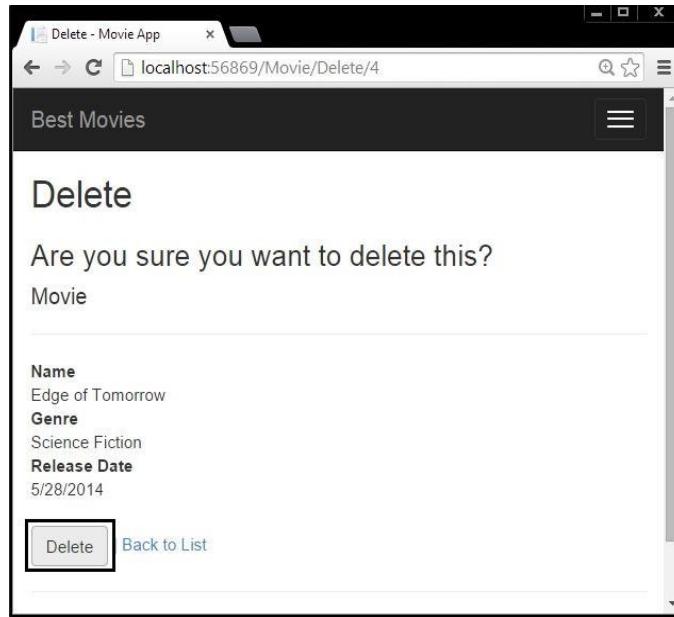
Genre
Biographic Film

Release Date
07/11/2013

[Back to List](#)

Delete

You can delete the record by clicking on the Delete link. Then you can see the windows as in the following:

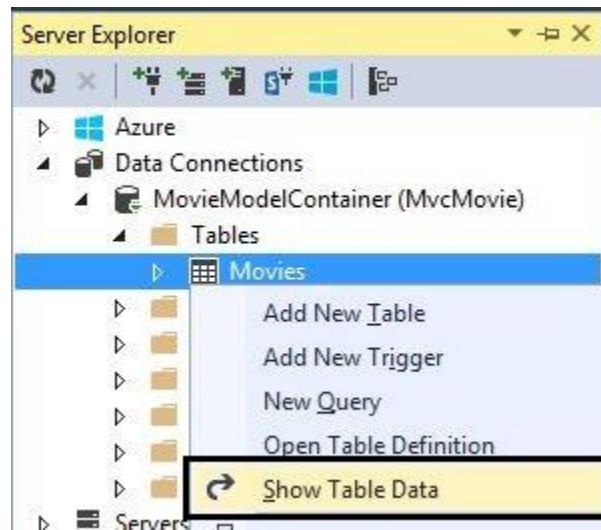


Working with Database

The table and records are updated in the database. We can check out this using the Server Explorer. Use the following procedure.

Step 1: Open the Model in the Server Explorer.

Step 2: Expand the Tables folder and right-click on the table and click on the Show Table Data.



Step 3: The records will display on the screen.

A screenshot of the 'Movies' table data grid. The columns are 'Id', 'Name', 'Genre', and 'ReleaseDate'. There are four rows: 1. Id: 1, Name: 'Bhaag Milkha B...', Genre: 'Biographic Film', ReleaseDate: '7/11/2013 12:00...'. 2. Id: 2, Name: 'Chak De India', Genre: 'Sports', ReleaseDate: '8/10/2007 12:00...'. 3. Id: 3, Name: 'Taare Zameen ...', Genre: 'Drama', ReleaseDate: '10/20/2007 12:0...'. 4. An empty row with Id: NULL, Name: NULL, Genre: NULL, and ReleaseDate: NULL.

Chapter 24: Cascading DropDownList in MVC 5 Using Web API 2

I have a requirement to show a cascading dropdownlist based upon the selection of a different state. I am binding all the states of India with a state dropdownlist and on the change of the listed state, all the cities of that state will change and we can choose the city from the next dropdownlist. I need to do this entire operation using the Web API in the MVC.

24.1 Overview

In the ASP.Net MVC 5 project template there are various types of techniques by which we can do the database operations like Create, Read, Update and Delete (CRUD) in the web application. We can use the Entity Framework approaches in the controller using an Entity Data Model and display it from the View. We can create the service for doing data operations and using a service reference we can call the operations using JavaScript like Angular, Knockout or jQuery in the web application.

In the previous chapter [Integrating ASP.Net Web API With MVC](#) we have seen how to integrate the API with the MVC project template based an ASP.Net application and we have also done some operations like fetch the data from the database using the API call and using the MVC controller we call the API and return to the View. In this chapter we will add some more GET operations and display the cities based upon the states. We will fetch the data from the database using the API call and using the MVC controller we will call the API and return the data to the view.

Note

Please go through [here](#) before working with this chapter. This chapter is not the second part of the previous chapter. This is the extension part of the previous one.

Prerequisites

In this chapter the web applications are created with the new Project Templates like ASP.Net Web API 2 and ASP.Net MVC 5 in the Visual Studio 2013, so there are the following prerequisites before getting started:

- Visual Studio 2012 or later
- ASP.NET Web API 2 Project Template
- ASP.Net MVC 5 Project Template

Getting Started

let's begin with the following sections:

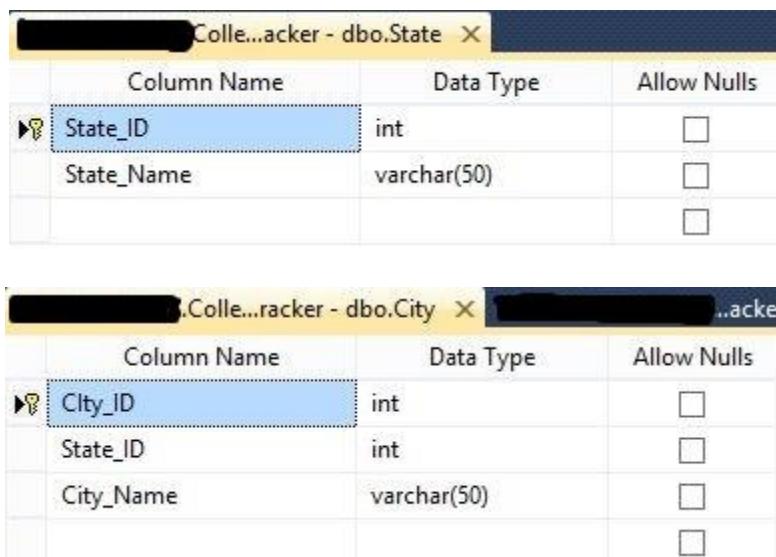
- Working with Database
- Creating Model
- Working with Data Access Layer
- Working with API Project
- Working with User Interface Layer

24.2 Working with Database

In this section we will create the table of state and city and insert the state and city in the both tables. We will also create some Stored Procedures for fetching the data from both tables.

Step 1

Create the State and City tables and insert values into both of the tables.

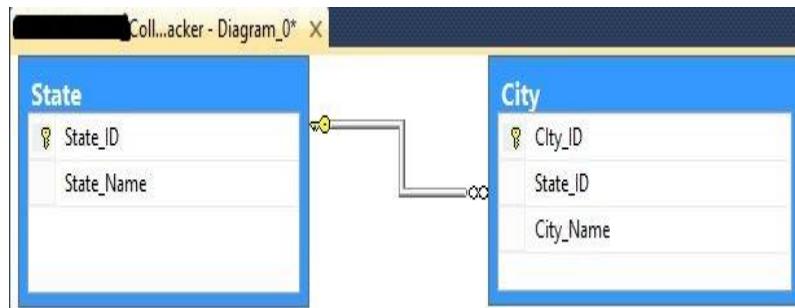


The screenshot shows two tables being created in a database:

Table	Column Name	Data Type	Allow Nulls
dbo.State	State_ID	int	<input type="checkbox"/>
dbo.State	State_Name	varchar(50)	<input type="checkbox"/>
dbo.City	City_ID	int	<input type="checkbox"/>
dbo.City	State_ID	int	<input type="checkbox"/>
dbo.City	City_Name	varchar(50)	<input type="checkbox"/>

Step 2

In the City table create the reference between City and State as shown in the following diagram.



Step 3

Create the Stored Procedure to select the state with the following code:

```

1. Create Proc [dbo].[CT_State_Select]
2. As
3. Begin
4. Set NOCOUNT ON
5. Select State_ID, State_Name
6. From State (NOLOCK)
7. Order By State_Name ASC
8. End
  
```

Step 4

Create the Stored Procedure to select the city with the following code:

```

1. CREATE PROCEDURE [dbo].[CT_City_Select] (
2.     @StateId Int)
3. AS
4. BEGIN
5.     SET NOCOUNT ON
6.     SELECT
7.         City_ID, State_ID, City_Name
8.     FROM City (NOLOCK)
9.     WHERE State_ID=@StateId
10.    ORDER BY City_Name ASC
11. END
  
```

Step 5

In the following code, we will update the previous Stored Procedure to fetch the College Details data:

```

1. ALTER Proc [dbo].[CT_CollegeDetails_Select]
2. As
  
```

```

3. Begin
4. Select
5.   c.CollegeID,
6.   c.[CollegeName],
7.   c.[CollegeAddress],
8.   c.[CollegePhone],
9.   c.[CollegeEmailID],
10.  c.[ContactPerson],
11.  c.[State],
12.  c.[City],
13.  s.State_Name,
14.  ci.City_Name
15. From CollegeDetails c inner join State s on c.State = s.State_ID inner join City ci on c.City = ci.City_ID
16. End

```

24.3 Creating Model

Since we have created the model in the previous application as said in the previous chapter, in this section we will add some more classes to add a new model. Start with the following procedure.

Step 1

Right-click on the "CollegeTrackerModels" project and add a new class as the named state with the following code:

```

1. namespace CollegeTrackerModels
2. {
3.   /// <summary>
4.   /// class for state
5.   /// </summary>
6.   public class State
7.   {
8.     #region Properties
9.     /// <summary>
10.    /// get and set the State ID
11.    /// </summary>
12.    public int State_ID { get; set; }
13.    /// <summary>
14.    /// get and set the State Name
15.    /// </summary>
16.    public string State_Name { get; set; }
17.    #endregion
18.  }
19. }

```

Step 2

Add a new class as the named city with the following code:

```

1.  namespace CollegeTrackerModels
2.  {
3.      /// <summary>
4.      /// class for City
5.      /// </summary>
6.      public partial class City
7.      {
8.          #region Properties
9.          /// <summary>
10.         /// get and set the city id
11.         /// </summary>
12.         public int City_ID { get; set; }
13.         /// <summary>
14.         /// get and set the state id
15.         /// </summary>
16.         public int State_ID { get; set; }
17.         /// <summary>
18.         /// get and set the city name
19.         /// </summary>
20.         public string City_Name { get; set; }
21.         /// <summary>
22.         /// get and set the value
23.         /// </summary>
24.         public int Value { get; set; }
25.         /// <summary>
26.         /// get and set the text
27.         /// </summary>
28.         public string Text { get; set; }
29.     #endregion
30. }
31. }
```

Step 3

Update the CollegeDetails class with the following code:

```

1.  namespace CollegeTrackerModels
2.  {
3.      /// <summary>
4.      /// class for College Details
5.      /// </summary>
6.      public class CollegeDetails
7.      {
8.          #region Constructor
9.          public CollegeDetails()
10.         {
```

```

11.     this.States = new List<State>();
12.     this.Cities = new List<City>();
13. }
14. #endregion
15.
16. #region Properties
17. ///<summary>
18. ///get and set the College ID
19. ///</summary>
20. public int CollegeID { get; set; }
21. ///<summary>
22. ///get and set the College Name
23. ///</summary>
24. [Display(Name = "College Name")]
25. public string CollegeName { get; set; }
26. ///<summary>
27. ///get and set the College Address
28. ///</summary>
29. [Display(Name = "College Address")]
30. public string CollegeAddress { get; set; }
31. ///<summary>
32. ///get and set the College Phone
33. ///</summary>
34. [Display(Name = "College Phone")]
35. public string CollegePhone { get; set; }
36. ///<summary>
37. ///get and set the College Email ID
38. ///</summary>
39. [Display(Name = "College EmailID")]
40. public string CollegeEmailID { get; set; }
41. ///<summary>
42. ///get and set the Contact Person
43. ///</summary>
44. [Display(Name = "Contact Person")]
45. public string ContactPerson { get; set; }
46. ///<summary>
47. ///get and set the State Id
48. ///</summary>
49. public Int16 StatId { get; set; }
50. ///<summary>
51. /// get and set the States
52. ///</summary>
53. public IList<State> States { get; set; }
54. ///<summary>
55. /// get and set the Cities
56. ///</summary>
57. public IList<City> Cities { get; set; }
58. ///<summary>
59. ///get and set the City Id
60. ///</summary>
61. public Int16 CityId { get; set; }
62. ///<summary>
63. ///get and set the status
64. ///</summary>

```

```

65.    public int Status { get; set; }
66.    /// <summary>
67.    /// get and set the state name
68.    /// </summary>
69.    [Display(Name="State")]
70.    public string State_Name { get; set; }
71.    /// <summary>
72.    /// get and set the city name
73.    /// </summary>
74.    [Display(Name="City")]
75.    public string City_Name { get; set; }
76.    #endregion
77. }
78. }
```

24.4 Working with Data Access Layer

In this section we will create some more classes to access the state and city table data.

Step 1

Right-click on the DAL folder and add a new class named "StateDAL" and provide the following code:

```

1.  using System;
2.  using System.Collections.Generic;
3.  using CollegeTrackerModels;
4.  using System.Configuration;
5.  using System.Data;
6.  using System.Data.SqlClient;
7.
8.  namespace CollegeTrackerCore.DAL
9.  {
10.     public class StateDAL
11.     {
12.         #region Variables
13.         SqlConnection con;
14.         SqlCommand cmd;
15.         SqlDataAdapter adap;
16.         DataTable dt;
17.         DataSet ds;
18.         string connectionstring = ConfigurationManager.ConnectionStrings["CollegeTrackerConnectionString"].Connectio
nString;
19.         #endregion
20.
21.         #region Constructor
22.         public StateDAL()
23.         {
24.             con = new SqlConnection(this.connectionstring);
25.         }
26.     }
27. }
```

```

26. #endregion
27.
28. #region Public Method
29. ///<summary>
30. /// This method is used to get all State.
31. ///</summary>
32. ///<returns></returns>
33. public List<State> GetState()
34. {
35.     List<State> objState = new List<State>();
36.     using (cmd = new SqlCommand("CT_State_Select", con))
37.     {
38.         try
39.         {
40.             cmd.CommandType = CommandType.StoredProcedure;
41.             con.Open();
42.             adap = new SqlDataAdapter();
43.             adap.SelectCommand = cmd;
44.             dt = new DataTable();
45.             adap.Fill(dt);
46.
47.             foreach (DataRow row in dt.Rows)
48.             {
49.                 State state = new State();
50.                 state.State_ID = Convert.ToInt32(row["State_ID"]);
51.                 state.State_Name = row["State_Name"].ToString();
52.                 objState.Add(state);
53.             }
54.         }
55.         catch (Exception ex)
56.         {
57.             con.Close();
58.         }
59.         return objState;
60.     }
61. }
62. #endregion
63. }
64. }
```

Step 2

Right-click on the DAL folder and add a new class named "CityDAL" and provide the following code:

```

1. using System;
2. using System.Collections.Generic;
3. using CollegeTrackerModels;
4. using System.Configuration;
5. using System.Data;
6. using System.Data.SqlClient;
```

```

7.
8. namespace CollegeTrackerCore.DAL
9. {
10. public class CityDAL
11. {
12.     #region Variables
13.     SqlConnection con;
14.     SqlCommand cmd;
15.     SqlDataAdapter adap;
16.     DataTable dt;
17.     DataSet ds;
18.     string connectionstring = ConfigurationManager.ConnectionStrings["CollegeTrackerConnectionString"].Connectio
nString;
19.     #endregion
20.
21.     #region Constructor
22.     public CityDAL()
23.     {
24.         con = new SqlConnection(this.connectionstring);
25.     }
26.     #endregion
27.
28.     #region Public Method
29.     ///<summary>
30.     /// This method is used to get all cities.
31.     ///</summary>
32.     ///<returns></returns>
33.     public List<City> GetCity(int stateld)
34.     {
35.         List<City> objCity = new List<City>();
36.         using (cmd = new SqlCommand("CT_City_Select", con))
37.         {
38.             cmd.CommandType = CommandType.StoredProcedure;
39.             dt = new DataTable();
40.             cmd.Parameters.AddWithValue("@Stateld", stateld);
41.             adap = new SqlDataAdapter();
42.             adap.SelectCommand = cmd;
43.             try
44.             {
45.                 con.Open();
46.                 adap.Fill(dt);
47.                 foreach (DataRow row in dt.Rows)
48.                 {
49.                     City city = new City();
50.                     city.City_ID = Convert.ToInt32(row["City_ID"]);
51.                     city.State_ID = Convert.ToInt32(row["State_ID"]);
52.                     city.City_Name = row["City_Name"].ToString();
53.                     objCity.Add(city);
54.                 }
55.             }
56.             catch (Exception ex)
57.             {
58.                 throw ex;
59.             }
}

```

```

60.    finally
61.    {
62.        con.Close();
63.    }
64.    return objCity;
65. }
66. }
67. #endregion
68. }
69. }
```

Step 3

Update the "CollegeDAL" method named "GetAllCollegeDetails()" with the following code:

```

1. public List<CollegeDetails> GetAllCollegeDetails()
2. {
3.     List<CollegeDetails> objCollegeDetails = new List<CollegeDetails>();
4.     using (cmd = new SqlCommand("CT_CollegeDetails_Select", con))
5.     {
6.         try
7.         {
8.             cmd.CommandType = CommandType.StoredProcedure;
9.             con.Open();
10.            adap = new SqlDataAdapter();
11.            adap.SelectCommand = cmd;
12.            dt = new DataTable();
13.            adap.Fill(dt);
14.
15.            foreach (DataRow row in dt.Rows)
16.            {
17.                CollegeDetails col = new CollegeDetails();
18.                col.CollegeID = Convert.ToInt32(row["CollegeID"]);
19.                col.CollegeName = row["CollegeName"].ToString();
20.                col.CollegeAddress = row["CollegeAddress"].ToString();
21.                col.CollegePhone = row["CollegePhone"].ToString();
22.                col.CollegeEmailID = row["CollegeEmailID"].ToString();
23.                col.ContactPerson = row["ContactPerson"].ToString();
24.                col.State_Name = row["State_Name"].ToString();
25.                col.City_Name = row["City_Name"].ToString();
26.                objCollegeDetails.Add(col);
27.            }
28.        }
29.        catch (Exception ex)
30.        {
31.            con.Close();
32.        }
33.        return objCollegeDetails;
34.    }
35. }
```

Step 4

Add new classes named “StateBLCore” and “CityBLCore” in the BLL folder. Begin with following procedure.

- **StateBLCore Class**

```

1. namespace CollegeTrackerCore.BLL
2. {
3.   public abstract class StateBLCore
4.   {
5.     #region public method
6.     ///<summary>
7.     /// This method is used to get the State Details
8.     ///</summary>
9.     ///<returns></returns>
10.    protected List<State> GetState()
11.    {
12.      List<State> objState = null;
13.      try
14.      {
15.        objState = new StateDAL().GetState();
16.      }
17.      catch (Exception ex)
18.      {
19.        throw ex;
20.      }
21.      return objState;
22.    }
23.    #endregion
24.  }
25. }
```

- **CityBLCore Class**

```

26. namespace CollegeTrackerCore.BLL
27. {
28.   public abstract class CityBLCore
29.   {
30.     #region public method
31.     ///<summary>
32.     /// This method is used to get the City Details
33.     ///</summary>
34.     ///<returns></returns>
35.     protected List<City> GetCity(int stateId)
36.     {
37.       List<City> objCity = null;
38.       try
39.       {
40.         objCity = new CityDAL().GetCity(stateId);
```

```

41.    }
42.    catch (Exception ex)
43.    {
44.        throw ex;
45.    }
46.    return objCity;
47. }
48. #endregion
49. }
50. }

```

24.5 Working with API Project

In this section we will add some more classes to access the data layer. Start with the following procedure:

Step 1

In the "CollegeTrackerAPI" project go to the Areas folder and right-click on the BLL folder to add two more classes named "CityBL" and "StateBL". Replace the code with the following code:

- **CityBL Class**

```

1. using CollegeTrackerCore.BLL;
2. using CollegeTrackerModels;
3. using System.Collections.Generic;
4.
5. namespace CollegeTrackerAPI.Areas.BLL
6. {
7.     internal sealed class CityBL : CityBLCore
8.     {
9.         /// <summary>
10.        /// This method is used to get the CityDetails
11.        /// </summary>
12.        /// <return></return>
13.        internal new List<City> GetCity(int stateId)
14.        {
15.            return base.GetCity(stateId);
16.        }
17.    }
18. }

```

- **StateBL Class**

```

1. using CollegeTrackerCore.BLL;
2. using CollegeTrackerModels;
3. using System.Collections.Generic;
4.

```

```

5. namespace CollegeTrackerAPI.Areas.BLL
6. {
7.   internal sealed class StateBL : StateBLCore
8.   {
9.     /// <summary>
10.    /// This method is used to get the states
11.    /// </summary>
12.    /// <return></return>
13.    internal new List<State> GetState()
14.    {
15.      return base.GetState();
16.    }
17.  }
18. }
```

Step 2

Go to the Controllers folder and add a new Web API 2 Controller named "StateDetails". Replace the code with the following code:

```

1. using CollegeTrackerAPI.Areas.BLL;
2. using CollegeTrackerModels;
3. using System;
4. using System.Collections.Generic;
5. using System.Net;
6. using System.Net.Http;
7. using System.Web.Http;
8.
9. namespace CollegeTrackerAPI.Areas.Controller
10. {
11.   public class StateDetailsController : ApiController
12.   {
13.     #region variable
14.     ///<summary>
15.     ///variable for StateBL
16.     ///</summary>
17.     private StateBL objStateBL;
18.     ///<summary>
19.     /// variable for httpResponseMessage
20.     /// </summary>
21.     HttpResponseMessage response;
22.     #endregion
23.
24.     #region Response Method
25.     ///summary
26.     /// This method is used to fetch StateDetails
27.     /// <summary>
28.     /// <return></return>
29.     [HttpGet, ActionName("GetAllStateDetails")]
```

```

30.    public HttpResponseMessage GetAllStatesDetails()
31.    {
32.        objStateBL = new StateBL();
33.        HttpResponseMessage response;
34.        try
35.        {
36.            var detailsResponse = objStateBL.GetState();
37.            if (detailsResponse != null)
38.                response = Request.CreateResponse<List<State>>(HttpStatusCode.OK, detailsResponse);
39.            else
40.
41.                response = new HttpResponseMessage(HttpStatusCode.NotFound);
42.            }
43.            catch (Exception ex)
44.            {
45.                response = Request.CreateErrorResponse(HttpStatusCode.InternalServerError, ex.Message);
46.
47.            }
48.            return response;
49.        }
50.        #endregion
51.    }
52. }
```

Step 3

Add a new Controller named "CityDetails" and add the following code:

```

1. using CollegeTrackerAPI.Areas.BLL;
2. using CollegeTrackerModels;
3. using System;
4. using System.Collections.Generic;
5. using System.Net;
6. using System.Net.Http;
7. using System.Web.Http;
8.
9. namespace CollegeTrackerAPI.Areas.Controller
10. {
11.     public class CityDetailsController : ApiController
12.     {
13.         #region variable
14.         ///<summary>
15.         ///variable for CityBL
16.         ///</summary>
17.         private CityBL objCityBL;
18.         ///<summary>
19.         /// variable for httpResponseMessage
20.         /// </summary>
21.         HttpResponseMessage response;
22.         #endregion
23.
24.         #region Response Method
```

```

25.    ///<summary>
26.    /// This method is used to fetch CityDetails
27.    ///<summary>
28.    ///<return></return>
29.    [HttpGet, ActionName("GetAllCityDetails")]
30.    public HttpResponseMessage GetAllCityDetails(int statId)
31.    {
32.        objCityBL = new CityBL();
33.        HttpResponseMessage response;
34.        try
35.        {
36.            var detailsResponse = objCityBL.GetCity(statId);
37.            if (detailsResponse != null)
38.                response = Request.CreateResponse<List<City>>(HttpStatusCode.OK, detailsResponse);
39.            else
40.                response = new HttpResponseMessage(HttpStatusCode.NotFound);
41.        }
42.        catch (Exception ex)
43.        {
44.            response = Request.CreateErrorResponse(HttpStatusCode.InternalServerError, ex.Message);
45.        }
46.    }
47.    return response;
48. }
49. }
50. #endregion
51. }
52. }
```

24.6 Working with User Interface Layer

In this section we will update the controller and add some new views to create the new view and display the dropdownlist in the view. Use the following procedure.

Step 1

Update the "CollegeDetails" controller and add the following code to it:

```

1.  public ActionResult CreateCollegeDetails()
2.  {
3.      CollegeDetails objcollege = new CollegeDetails();
4.      objcollege.States = GetAllStates();
5.      return View("~/Views/CollegeDetails/CreateCollegeDetails.cshtml", objcollege);
6.  }
7.
8.  public List<State> GetAllStates()
9.  {
10.     var list = new List<CollegeTrackerModels.State>();
```

```

11. var httpClient = new HttpClient();
12. httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
13. HttpResponseMessage response;
14. response = httpClient.GetAsync("http://localhost:59866/api/" + "StateDetails/GetAllStateDetails/").Result;
15. response.EnsureSuccessStatusCode();
16. List<State> stateList = response.Content.ReadAsAsync<List<State>>().Result;
17. if (!object.Equals(stateList, null))
18. {
19.     var states = stateList.ToList();
20.     return states;
21. }
22. else
23. {
24.     return null;
25. }
26. }
27.
28. [HttpGet]
29. public JsonResult GetJsonCity(int stateld)
30. {
31.     var list = new List<CollegeTrackerModels.City>();
32.     try
33.     {
34.         var httpClient = new HttpClient();
35.         httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
36.         HttpResponseMessage response;
37.         response = httpClient.GetAsync("http://localhost:59866/api/" + "cityDetails/GetAllCityDetails?stateld=" + stateld +
38.         "").Result;
39.         response.EnsureSuccessStatusCode();
40.         List<City> cityList = response.Content.ReadAsAsync<List<City>>().Result;
41.         if (!object.Equals(cityList, null))
42.         {
43.             var cities = cityList.ToList();
44.             foreach (var item in cities)
45.             {
46.                 list.Add(new City
47.                 {
48.                     Value = item.City_ID,
49.                     Text = item.City_Name
50.                 });
51.             }
52.         }
53.         catch (HttpException ex)
54.         {
55.             throw new HttpException(ex.Message);
56.         }
57.         catch (Exception ex)
58.         {
59.             throw new Exception(ex.Message);
60.         }
61.         return Json(new SelectList(list, "Value", "Text"), JsonRequestBehavior.AllowGet);
62.     }

```

Step 2

Add the following code before the table in the Views/CollegeDetails/CollegeDetails page:

```

1. <p>
2.     @Html.ActionLink("Create New", "CreateCollegeDetails", "CollegeDetails")
3. </p>
```

Step 3

Go to Views/CollegeDetails and add an empty view named "CreateCollegeDetails" and replace the code with the following code:

```

1. @model CollegeTrackerModels.CollegeDetails
2. @{
3.     ViewBag.Title = "Create CollegeDetails";
4. }
5.
6. <h2>Create</h2>
7.
8. <script type="text/javascript">
9.     function getCities() {
10.         var stateID = $('#DropDownListStates').val();
11.         if (stateID > 0)
12.         {
13.             $.ajax{
14.                 url: "@Url.Action("GetJsonCity", "CollegeDetails")",
15.                 data: { statid: stateID },
16.                 dataType: "json",
17.                 type: "GET",
18.                 success: function (data) {
19.                     var items = "";
20.                     items = "<option value='>--Choose City--</option>";
21.                     $.each(data, function (i, item) {
22.                         items += "<option value=\"" + item.Value + "\">" + item.Text + "</option>";
23.                     });
24.                     $('#DropDownListCities').html(items);
25.                 }
26.             });
27.         }
28.     }
29. </script>
30.
31. @using (Ajax.BeginForm("CreateUpdateCollegeDetails", "CollegeDetails", null))
32. {
33.     @Html.AntiForgeryToken()
34.
35.     <div class="form-horizontal">
36.         <hr />
37.         @Html.ValidationSummary(true)
```

```

38.
39.    <div class="form-group">
40.        @Html.LabelFor(model => model.CollegeName, new { @class = "control-label col-md-2" })
41.        <div class="col-md-10">
42.            @Html.EditorFor(model => model.CollegeName)
43.            @Html.ValidationMessageFor(model => model.CollegeName)
44.        </div>
45.    </div>
46.
47.    <div class="form-group">
48.        @Html.LabelFor(model => model.CollegeAddress, new { @class = "control-label col-md-2" })
49.        <div class="col-md-10">
50.            @Html.EditorFor(model => model.CollegeAddress)
51.            @Html.ValidationMessageFor(model => model.CollegeAddress)
52.        </div>
53.    </div>
54.
55.    <div class="form-group">
56.        @Html.LabelFor(model => model.CollegePhone, new { @class = "control-label col-md-2" })
57.        <div class="col-md-10">
58.            @Html.TextBoxFor(model => model.CollegePhone)
59.            @Html.ValidationMessageFor(model => model.CollegePhone)
60.        </div>
61.    </div>
62.
63.    <div class="form-group">
64.        @Html.LabelFor(model => model.CollegeEmailID, new { @class = "control-label col-md-2" })
65.        <div class="col-md-10">
66.            @Html.EditorFor(model => model.CollegeEmailID)
67.            @Html.ValidationMessageFor(model => model.CollegeEmailID)
68.        </div>
69.    </div>
70.
71.    <div class="form-group">
72.        @Html.LabelFor(model => model.ContactPerson, new { @class = "control-label col-md-2" })
73.        <div class="col-md-10">
74.            @Html.EditorFor(model => model.ContactPerson)
75.            @Html.ValidationMessageFor(model => model.ContactPerson)
76.        </div>
77.    </div>
78.
79.    <div class="form-group">
80.        @Html.LabelFor(model => model.StateId, new { @class = "control-label col-md-2" })
81.        <div class="col-md-10">
82.            @Html.DropDownListFor(model => model.StateId, new SelectList(Model.States, "State_ID", "State_Name"),
83. "Choose State", new { onchange = "getCities()", @id = "DropDownListStates" })
84.            @Html.ValidationMessageFor(model => model.StateId)
85.        </div>
86.    </div>
87.
88.    <div class="form-group">
89.        @Html.LabelFor(model => model.CityId, new { @class = "control-label col-md-2" })
90.        <div class="col-md-10">

```

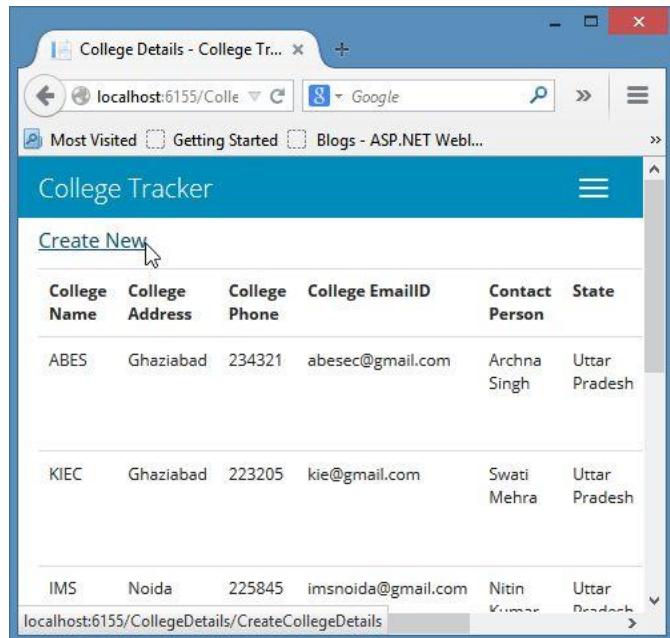
```

91.      @Html.DropDownListFor(model => model.CityId, new SelectList(Model.Cities, "City_ID", "State_ID", "City_N
ame"), "Choose City", new { @id = "DropDownListCities" })
92.      @Html.ValidationMessageFor(model => model.CityId)
93.  </div>
94. </div>
95.
96. <div class="form-group">
97.   <div class="col-md-offset-2 col-md-10">
98.     <input type="submit" value="Create" class="btn btn-success" />
99.   </div>
100. </div>
101. </div>
102. }
103.
104. <div>
105.   @Html.ActionLink("Back to List", "GetCollegeList")
106. </div>
107.
108. @section Scripts{
109.   @Scripts.Render("~/bundles/jqueryval");
110. }

```

Step 4

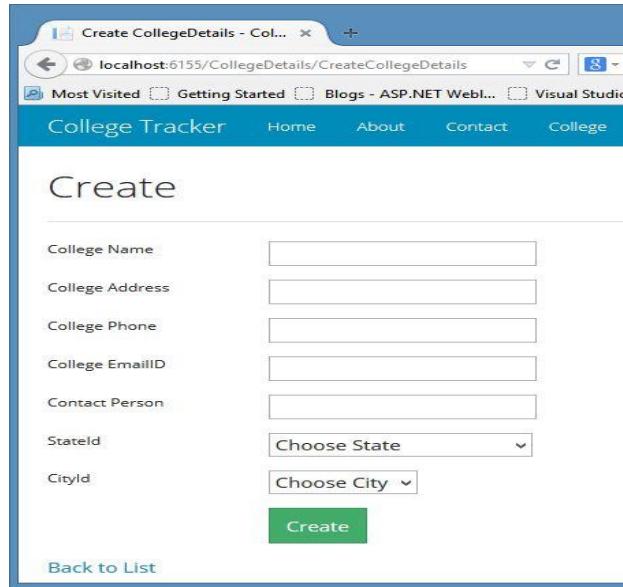
Now run the project and click on the Create New Action Link as shown below:



College Name	College Address	College Phone	College EmailID	Contact Person	State
ABES	Ghaziabad	234321	abesec@gmail.com	Archna Singh	Uttar Pradesh
KIEC	Ghaziabad	223205	kie@gmail.com	Swati Mehra	Uttar Pradesh
IMS	Noida	225845	imsnoida@gmail.com	Nitin Kumar	Uttar Pradesh

Step 5

In the next page you can check that the states are listed in the States dropdownlist.



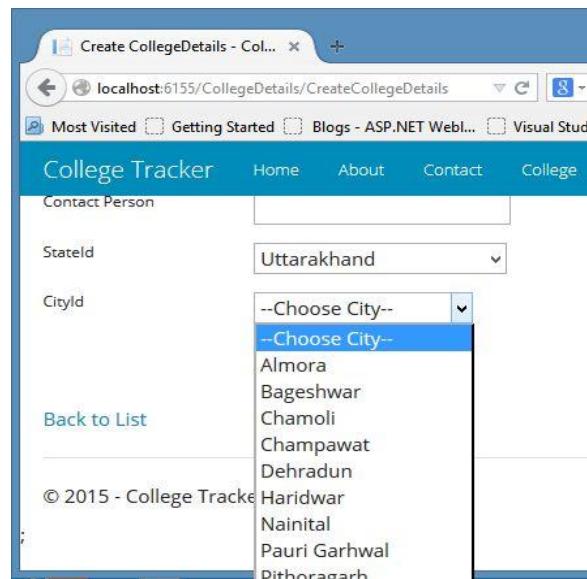
The screenshot shows a web browser window with the title "Create CollegeDetails - Col...". The address bar shows "localhost:6155/CollegeDetails/CreateCollegeDetails". The page header includes "College Tracker" and navigation links for "Home", "About", "Contact", and "College". The main content area is titled "Create" and contains the following form fields:

- College Name: [text input]
- College Address: [text input]
- College Phone: [text input]
- College EmailID: [text input]
- Contact Person: [text input]
- StateId: [dropdown menu] currently showing "Choose State"
- CityId: [dropdown menu] currently showing "Choose City"
- Create** button (green)

At the bottom left is a "Back to List" link.

Step 6

Choose the state and the city dropdownlist will update.



The screenshot shows the same "Create" page as above, but the "StateId" dropdown is now set to "Uttarakhand". The "CityId" dropdown menu is open, displaying a list of cities:

- Choose City--
- Almora
- Bageshwar
- Chamoli
- Champawat
- Dehradun
- Haridwar
- Nainital
- Pauri Garhwal
- Dittoragarh

Chapter 25: Introducing Google Chart in ASP.NET MVC 5

Google provides the Google API for applying chart representation of data. The Google Chart tool is very dominant and smooth to use and it is free. Google Chart API provides many types of chart in which we can represent the data in the application. Some of them are given below:

1. Geo Chart
2. Column Chart
3. Bar Chart
4. Line Chart
5. Pie Chart
6. Area Chart

So, let's create an ASP.NET MVC application and fetch the data from the database and in the application we will use the Google Chart API to represent the data in the Google Chart. We will proceed with the following sections:

1. Getting Started
2. Perform Database Operation
3. Creating Library
4. Represent Data

25.1 Getting Started

In this section we will create MVC application in the Visual Studio 2013 with the following steps:

Step 1: Open Visual Studio 2013 and click on “New Project”,

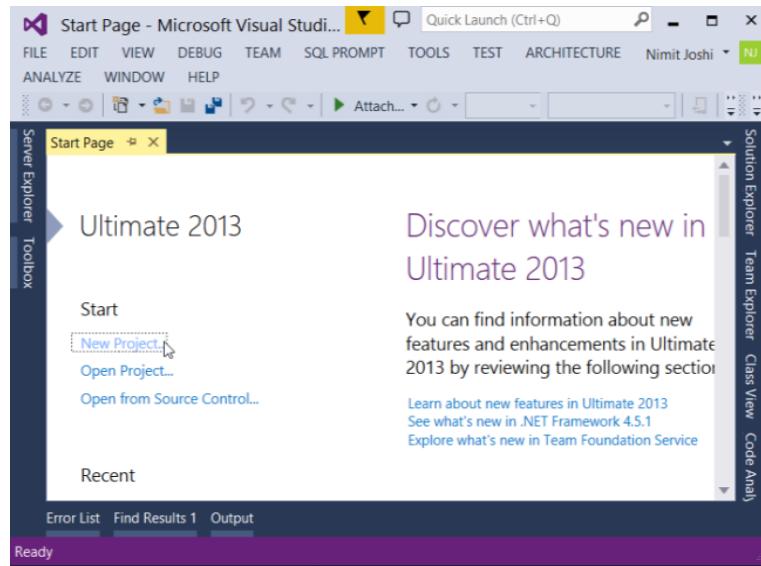


Figure 1: Start Page of VS 2013

Step 2: Select Web tab from the left pane and select the “ASP.NET Web Application”,

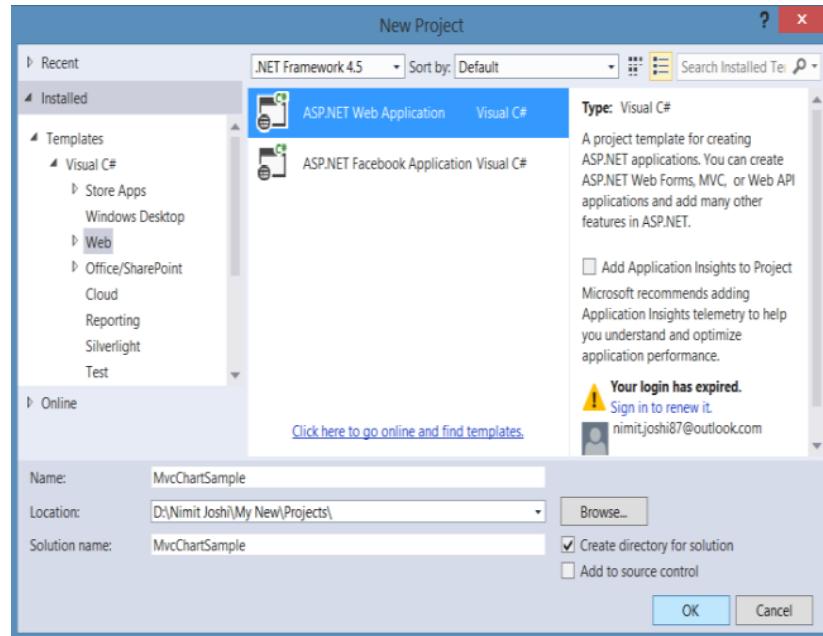


Figure 2: Creating Web Application in VS 2013

Step 3: Select “MVC project template” from the ASP.Net Wizard to create MVC app,

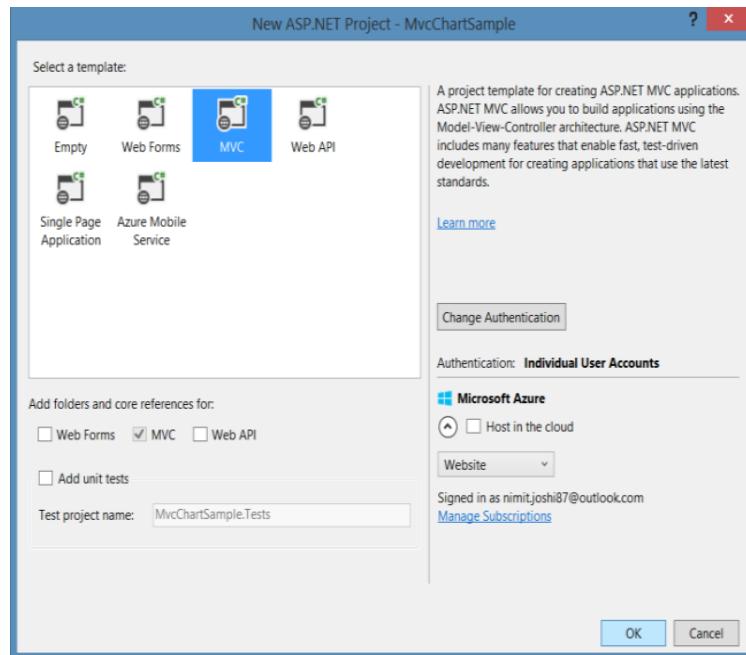


Figure 3: MVC Project Template

This will create your MVC Application and after some changes in default layout page run the application.

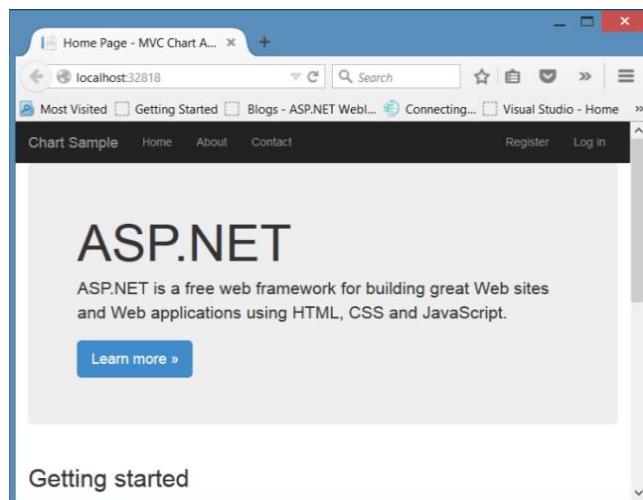


Figure 4: Mvc App Start Page

25.2 Perform Database Operation

In this section we will perform database operation in which we will get the data from the database. So, let's start with the following procedure:

Step 1: Create database and database tables from the following query,

```

1. CREATE DATABASE ChartSample
2.
3. USE [ChartSample]
4. GO
5.
6. SET ANSI_NULLS ON
7. GO
8. SET QUOTED_IDENTIFIER ON
9. GO
10. SET ANSI_PADDING ON
11. GO
12. CREATE TABLE [dbo].[CS_Player](
13.     [PlayerId] [int] IDENTITY(1,1) NOT NULL,
14.     [PlayerName] [varchar](50) NULL,
15.     CONSTRAINT [PK_CS_Player] PRIMARY KEY CLUSTERED
16. (
17.     [PlayerId] ASC
18. )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
19. ) ON [PRIMARY]
20.
21. GO
22. SET ANSI_PADDING OFF
23. GO
24. /****** Object: Table [dbo].[CS_PlayerRecord] Script Date: 11/19/2015 4:00:01 PM *****/
25. SET ANSI_NULLS ON
26. GO
27. SET QUOTED_IDENTIFIER ON
28. GO
29. CREATE TABLE [dbo].[CS_PlayerRecord](
30.     [ID] [int] IDENTITY(1,1) NOT NULL,
31.     [PlayerId] [int] NULL,
32.     [Year] [int] NULL,
33.     [TotalRun] [int] NULL,
34.     [TotalWickets] [int] NULL,
35.     [ODIMatches] [int] NULL,
36.     [TestMatches] [int] NULL,
37.     CONSTRAINT [PK_CS_PlayerRecord] PRIMARY KEY CLUSTERED
38. (
39.     [ID] ASC
40. )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
41. ) ON [PRIMARY]
42.
```

```

43. GO
44. ALTER TABLE [dbo].[CS_PlayerRecord] WITH CHECK ADD CONSTRAINT [FK_CS_PlayerRecord_CS_Player] FOREIGN KEY([PlayerId])
45. REFERENCES [dbo].[CS_Player] ([PlayerId])
46. GO
47. ALTER TABLE [dbo].[CS_PlayerRecord] CHECK CONSTRAINT [FK_CS_PlayerRecord_CS_Player]
48. GO

```

Step 2: Insert records in the tables.

Step 3: Now we will create procedures with the following code,

```

1. USE [ChartSample]
2. GO
3. /****** Object: StoredProcedure [dbo].[SC_GetPlayers] Script Date: 11/19/2015 4:21:29 PM *****/
4. SET ANSI_NULLS ON
5. GO
6. SET QUOTED_IDENTIFIER ON
7. GO
8. ALTER PROC [dbo].[SC_GetPlayers]
9. AS
10. BEGIN
11.     SELECT *
12.     FROM CS_Player
13. END
14.
15. USE [ChartSample]
16. GO
17. /****** Object: StoredProcedure [dbo].[SC_GetPlayerRecordsBtPlayerId] Script Date: 11/19/2015 4:21:26 PM *****/
18. SET ANSI_NULLS ON
19. GO
20. SET QUOTED_IDENTIFIER ON
21. GO
22. ALTER PROC [dbo].[SC_GetPlayerRecordsBtPlayerId] @PlayerId INT
23. AS
24. BEGIN
25.     SELECT PlayerId ,
26.             Year ,
27.             TotalRun ,
28.             TotalWickets ,
29.             ODIMatches ,
30.             TestMatches
31.     FROM CS_PlayerRecord
32.     WHERE PlayerId = @PlayerId
33. END

```

25.3 Creating Library

In this section we will add class library project in the same solution and by using the reference we will get data from the database. So, let's start with the following steps:

Step 1: Right click on the solution and add a new project,

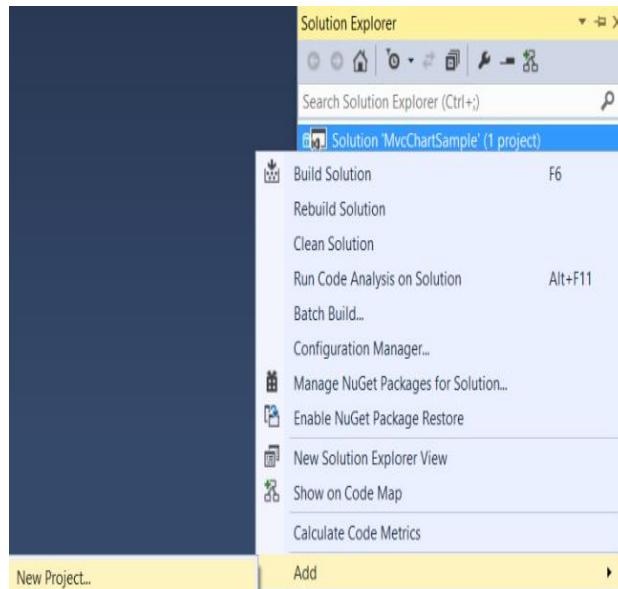


Figure 5: Adding New Project

Step 2: Select the “**Class Library**” and specify the name as “**Chart Library**”,

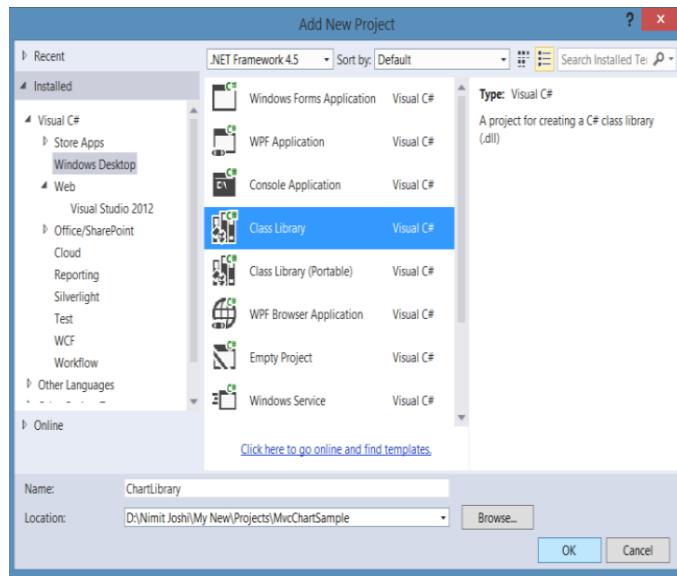


Figure 6: Creating Class Library

Step 3: Now add a class in the library project,

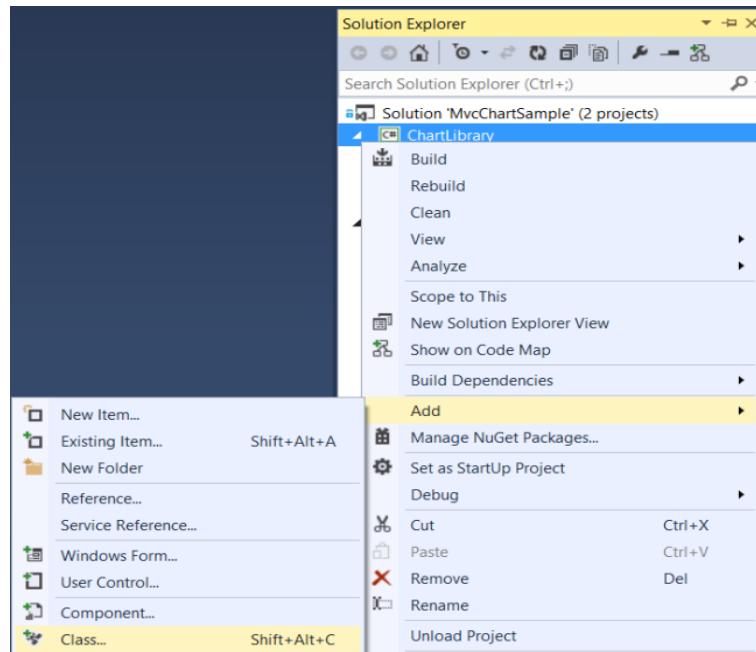


Figure 7: Adding New Class

Step 4: Replace the class code with the following code,

```
1. namespace ChartLibrary
2. {
3.     public class Players
4.     {#
5.         region Properties
6.         /// <summary>
7.         /// get and set the PlayerId
8.         /// </summary>
9.         public int PlayerId
10.        {
11.            get;
12.            set;
13.        }
14.        /// <summary>
15.        /// get and set the PlayerName
16.        /// </summary>
17.        public string PlayerName
18.        {
19.            get;
20.            set;
21.        }
22.        /// <summary>
23.        /// get and set the PlayerList
24.        /// </summary>
25.        public List < Players > PlayerList
26.        {
27.            get;
28.            set;
29.        }
30.        /// <summary>
31.        /// get and set the PlayerRecordList
32.        /// </summary>
33.        public List < PlayerRecord > PlayerRecordList
34.        {
35.            get;
36.            set;
37.        }#
38.    endregion
39. }
40. public class PlayerRecord
41. {#
42.     region Properties
43.     /// <summary>
44.     /// get and set the PlayerId
45.     /// </summary>
46.     public int PlayerId
47.     {
48.         get;
49.         set;
50.     }
51.     /// <summary>
```

```
52.    /// get and set the Year
53.    /// </summary>
54.    public int Year
55.    {
56.        get;
57.        set;
58.    }
59.    /// <summary>
60.    /// get and set the TotalRun
61.    /// </summary>
62.    public int TotalRun
63.    {
64.        get;
65.        set;
66.    }
67.    /// <summary>
68.    /// get and set the TotalWickets
69.    /// </summary>
70.    public int TotalWickets
71.    {
72.        get;
73.        set;
74.    }
75.    /// <summary>
76.    /// get and set the ODIMatches
77.    /// </summary>
78.    public int ODIMatches
79.    {
80.        get;
81.        set;
82.    }
83.    /// <summary>
84.    /// get and set the TestMatches
85.    /// </summary>
86.    public int TestMatches
87.    {
88.        get;
89.        set;
90.    }#
91.    endregion
92. }
93. }
```

Step 5: Right click on References in the class library project and click on “**Manage NuGet Packages**”,

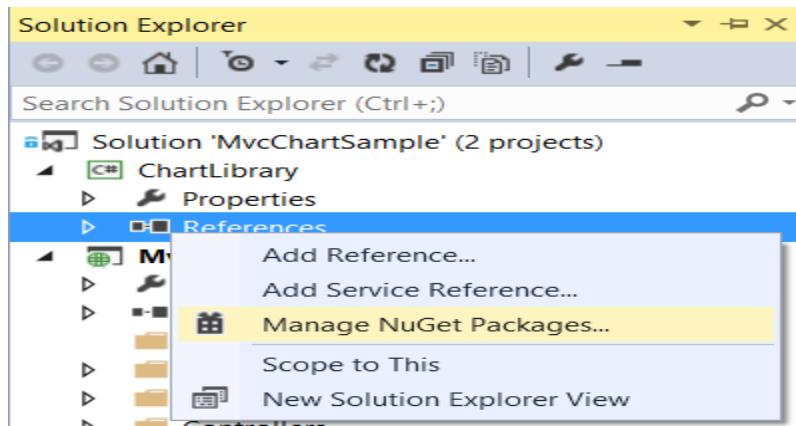


Figure 8: Adding NuGet Package

Step 6: Search for Enterprise Library and install it in the library project,

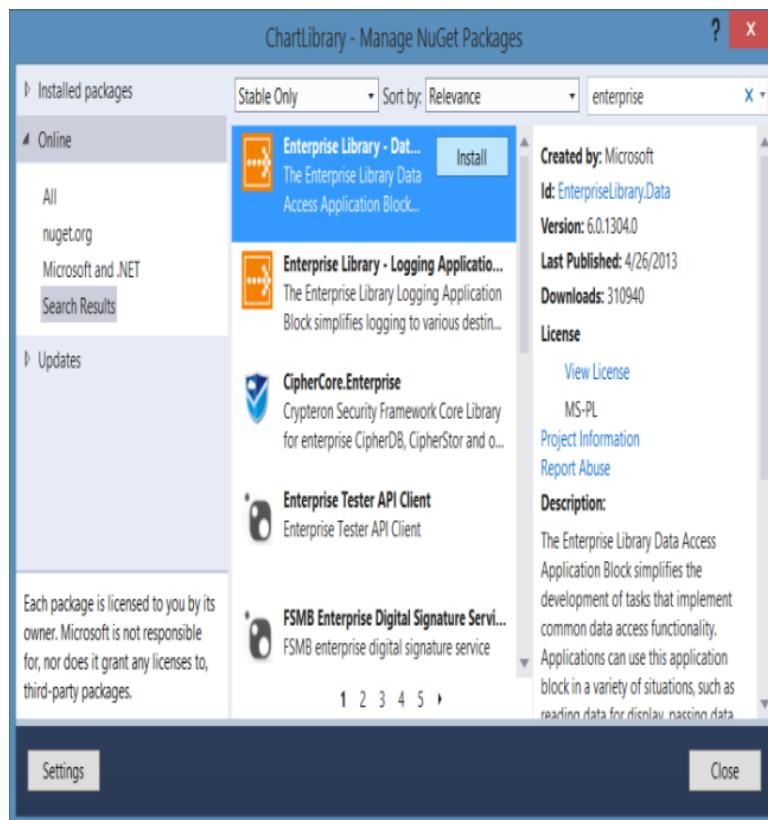


Figure 9: Adding Enterprise Library Package

Step 7: Add another class named “**PlayerDAL**” and replace the code with the following code,

```

1. using Microsoft.Practices.EnterpriseLibrary.Data;
2. using Microsoft.Practices.EnterpriseLibrary.Data.Sql;
3. using System;
4. using System.Collections.Generic;
5. using System.Configuration;
6. using System.Data;
7. using System.Data.Common;
8. using System.Linq;
9. using System.Reflection;
10. namespace ChartLibrary
11. {
12.     public class PlayerDAL
13.     {#
14.         region Variable
15.         ///<summary>
16.         /// Specify the Database variable
17.         ///</summary>
18.         Database objDB;
19.         ///<summary>
20.         /// Specify the static variable
21.         ///</summary>
22.         string ConnectionString;#
23.         endregion# region Database Method
24.         public List < T > ConvertTo < T > (DataTable datatable) where T: newnew List < T > ();
27.         try
28.         {
29.             List < string > columnsNames = new List < string > ();
30.             foreach(DataColumn DataColumn in datatable.Columns)
31.                 columnsNames.Add(DataColumn.ColumnName);
32.             Temp = datatable.AsEnumerable().ToList().ConvertAll < T > (row => getObject < T > (row, columnsNames));
33.             return Temp;
34.         }
35.         catch
36.         {
37.             return Temp;
38.         }
39.     }
40.     public T getObject < T > (DataRow row, List < string > columnsName) where T: newnew T();
43.         try
44.         {
45.             string columnname = "";
46.             string value = "";
47.             PropertyInfo[] Properties;
48.             Properties = typeof (T).GetProperties();
49.             foreach(PropertyInfo objProperty in Properties)

```

```

50.      {
51.          columnname = columnsName.Find(name => name.ToLower() == objProperty.Name.ToLower());
52.          if (!string.IsNullOrEmpty(columnname))
53.          {
54.              value = row[columnname].ToString();
55.              if (!string.IsNullOrEmpty(value))
56.              {
57.                  if (Nullable.GetUnderlyingType(objProperty.PropertyType) != null)
58.                  {
59.                      value = row[columnname].ToString().Replace("$", "").Replace(",", "");
60.                      objProperty.SetValue(obj, Convert.ChangeType(value, Type.GetType(Nullable.GetUnderlyingType(o
bjProperty.PropertyType).ToString()))), null);
61.                  }
62.                  else
63.                  {
64.                      value = row[columnname].ToString();
65.                      objProperty.SetValue(obj, Convert.ChangeType(value, Type.GetType(objProperty.PropertyType.ToS
tring()))), null);
66.                  }
67.              }
68.          }
69.      }
70.      return obj;
71.  }
72.  catch (Exception ex)
73.  {
74.      return obj;
75.  }
76. }#
77. endregion# region Constructor
78. /// <summary>
79. /// This constructor is used to get the connectionstring from the config file
80. /// </summary>
81. public PlayerDAL()
82. {
83.     ConnectionString = ConfigurationManager.ConnectionStrings["PlayerConnectionString"].ToString();
84. }#
85. endregion# region Player Details
86. /// <summary>
87. /// This method is used to get all players
88. /// </summary>
89. /// <returns></returns>
90. public List < Players > GetPlayerDetails()
91. {
92.     List < Players > objPlayers = null;
93.     objDB = new SqlDatabase(ConnectionString);
94.     using (DbCommand objcmd = objDB.GetStoredProcCommand("SC_GetPlayers"))
95.     {
96.         try
97.         {
98.             using (DataTable dataTable = objDB.ExecuteDataSet(objcmd).Tables[0])
99.             {
100.                 objPlayers = ConvertTo < Players > (dataTable);
101.             }
102.         }
103.     }
104. }

```

```

102.        }
103.        catch (Exception ex)
104.        {
105.            throw ex;
106.            return null;
107.        }
108.    }
109.    return objPlayers;
110.}
111./// <summary>
112./// This method is used to get player records on the basis of player id
113./// </summary>
114./// <param name="playerId"></param>
115./// <returns></returns>
116. public List < PlayerRecord > GetPlayerRecordByPlayerId(Int16 ? playerId)
117.{
118.    List < PlayerRecord > objPlayerRecords = null;
119.    objDB = new SqlDatabase(ConnectionString);
120.    using(DbCommand objcmd = objDB.GetStoredProcCommand("SC_GetPlayerRecordsBtPlayerId"))
121.    {
122.        objDB.AddInParameter(objcmd, "@PlayerId", DbType.Int16, playerId);
123.        try
124.        {
125.            using(DataTable dataTable = objDB.ExecuteDataSet(objcmd).Tables[0])
126.            {
127.                objPlayerRecords = ConvertTo < PlayerRecord > (dataTable);
128.            }
129.        }
130.        catch (Exception ex)
131.        {
132.            throw ex;
133.            return null;
134.        }
135.    }
136.    return objPlayerRecords;
137.}
138. #endregion
139.}
140.}

```

Step 8: Add the following connection string in the Web.Config file in the MVC Application,

1. <add name="PlayerConnectionString" connectionString="Data Source=Your Server Name; Initial Catalog=ChartSample ; User Id=uid; Password=Your password" providerName="System.Data.SqlClient" />

Step 9: Build the solution

25.4 Represent Data

In this section we will get the data from the library reference and represent data in the newly added view in the chart form. So, let's start with the following steps:

Step 1: Right click on the reference in the MVC Application project and click on “**Add Reference**”,

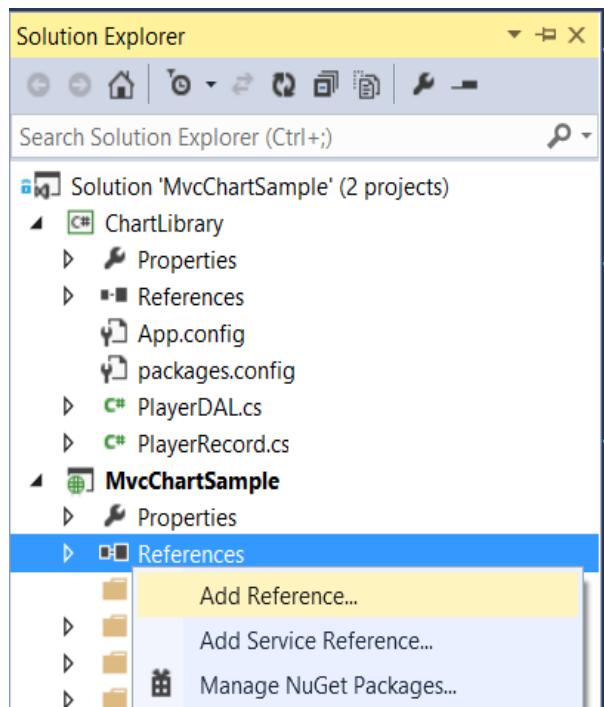


Figure 10: Adding Reference

Step 2: Add the library reference in the MVC application,

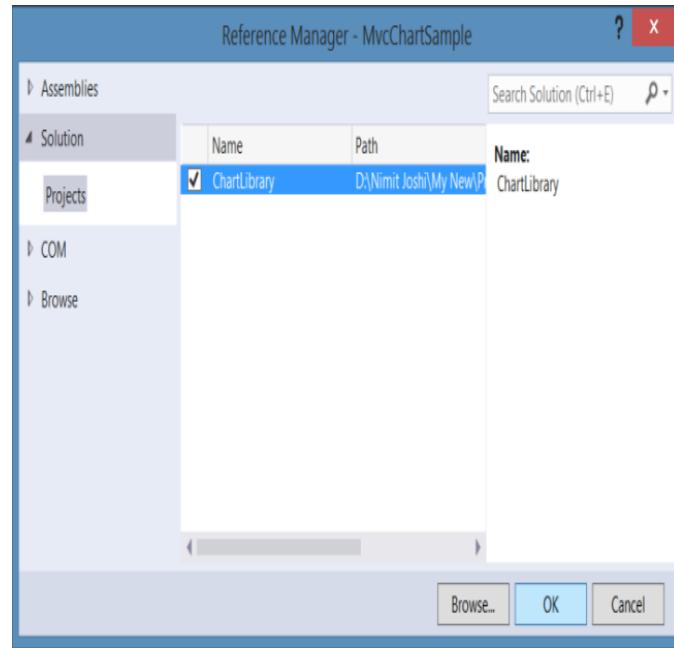


Figure 11: Adding Library Reference

Step 3: Now right click on the Controllers folder and click on “**Add Controller**”,

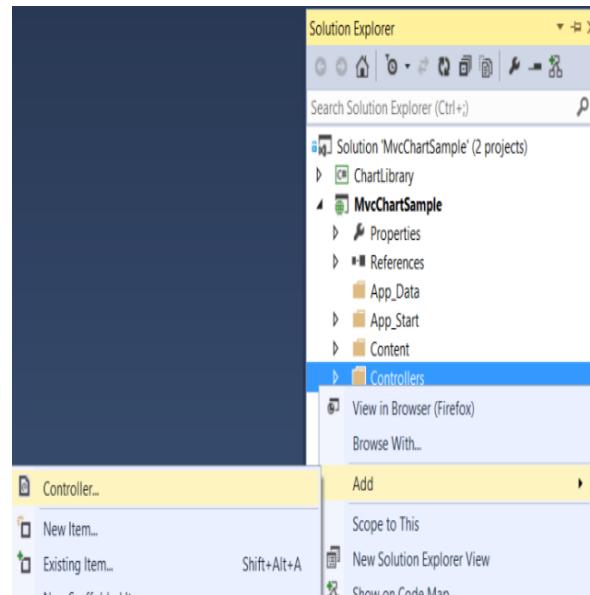


Figure 12: Adding Controller in MVC App

Step 3: Select MVC 5 Empty Controller from the wizard,

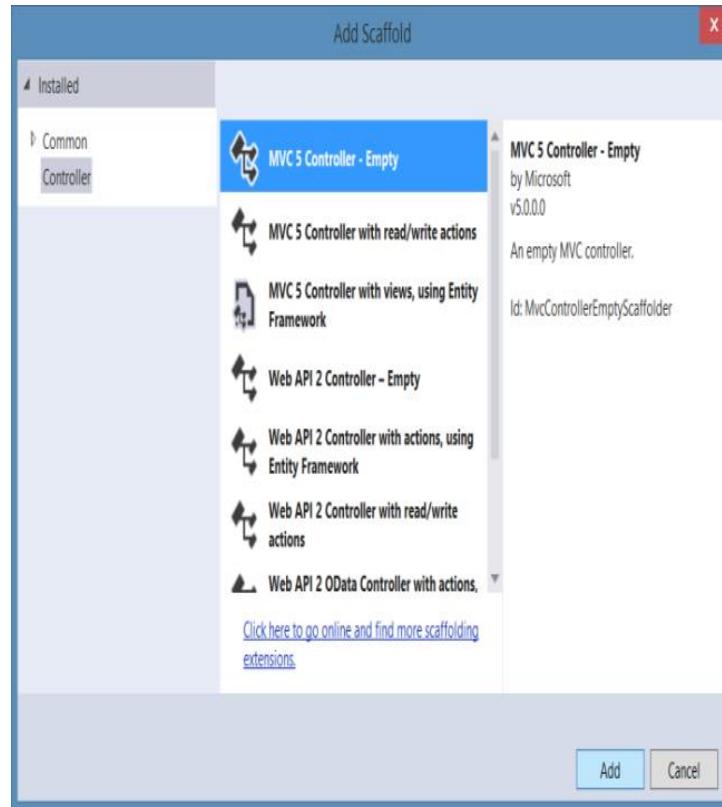


Figure 13: Adding MVC 5 Empty Controller

And specify the **Controller name**,



Figure 14: Specifying Controller Name

Step 4: Add the following code in the controller class,

```

1. public class PlayerController: Controller
2. {
3.     // GET: Player
4.     public ActionResult Index()
5.     {
6.         return View();
7.     }
8.     public ActionResult PlayerChart()
9.     {
10.        Players objPlayers = new Players();
11.        PlayerDAL objPlayerDAL = new PlayerDAL();
12.        try
13.        {
14.            objPlayers.PlayerList = objPlayerDAL.GetPlayerDetails();
15.            return View("~/Views/Player/PlayerChart.cshtml", objPlayers);
16.        }
17.        catch (Exception ex)
18.        {
19.            throw;
20.        }
21.    }
22.    public JsonResult PlayerDashboardList(Int16 ? playerId)
23.    {
24.        Players objPlayers = new Players();
25.        PlayerDAL objPlayerDAL = new PlayerDAL();
26.        if (object.Equals(playerId, null))
27.        {
28.            playerId = 1;
29.        }
30.        try
31.        {
32.            var response = objPlayerDAL.GetPlayerRecordByPlayerId(playerId);
33.            if (!object.Equals(response, null))
34.            {
35.                objPlayers.PlayerRecordList = response.ToList();
36.            }
37.        }
38.        catch (Exception ex)
39.        {
40.            throw;
41.        }
42.        return Json(objPlayers.PlayerRecordList, JsonRequestBehavior.AllowGet);
43.    }
44. }
```

Step 5: Right click on the Views/Player folder and click on the Add -> View and specify the view name as “**Player Chart**”,

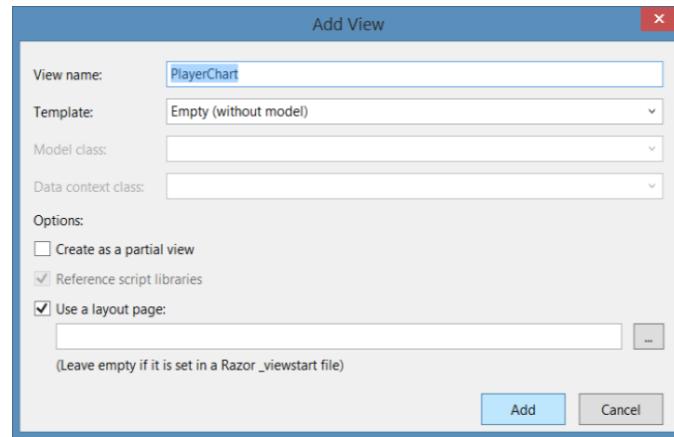


Figure 15: Adding View

Step 6: Add the following code in the view page,

```

1. @model ChartLibrary.Players
2. @{
3. {
4.     ViewBag.Title = "PlayerChart";
5. }
< script src = "~/Scripts/jquery-1.10.2.js" > </ script >
< script type = "text/javascript" src = "https://www.google.com/jsapi" > </ script >
6. < style > label
7. {
8.     font - size: 18 px;
9.     font - weight: lighter;
10. }
11. select
12. {
13.     width: 250 px; height: 40 px; padding: 0 14 px; font - size: 16 px;
14. }
< / style > <
h2 style = "margin:25px 0; color:#5a5a5a;" > Player Chart < / h2 > < div class = "clear" > < / div >
< div class = "row" > < div class = "col-md-8" >
< section id = "loginForm" style = "margin-bottom:25px;" >
< span style = "margin-right:15px; font-size:15px; font-weight:lighter;" > @Html.LabelFor(m => m.PlayerName, "Player Name") < / span >
15. @Html.DropDownListFor(m => m.PlayerName, new SelectList(Model.PlayerList, "PlayerId", "PlayerName"), new
16. {
17.     @onchange = "drawChart()", @id = "playerNameList"
18. })
< / section >
< / div >
< / div >
< div class = "clear" > < / div >
```

```

< div >< div id = "Player_Chart" style = "width: 100%; height: 500px" >
< /div>
< /div>
< div id = "divProcessing" class = "processingButton" style = "display: none; text-align: center" >
< img src = "~/Images/ajaxloader_small.gif" width = "16" height = "11" />
< /div>
< div id = "divLoading" class = "loadingCampus" >
< div class = "LoadingImageForActivity" >
< img width = "31" height = "31" alt = "" src = "~/Images/ajax-loader-round-dashboard.gif" />
< /div>
< /div>
< script type = "text/javascript" >
    google.load("visualization", "1",
19. {
20.     packages: ["corechart"]
21. });
22. google.setOnLoadCallback(drawChart);
23.
24. function drawChart()
25. {
26.     var playerId = $('#playerNameList :selected').val();
27.     $.ajax(
28.     {
29.         url: '@Url.Action("PlayerDashboardList","Player")',
30.         dataType: "json",
31.         data:
32.         {
33.             playerId: playerId
34.         },
35.         type: "GET",
36.         error: function (xhr, status, error)
37.         {
38.             var err = eval("(" + xhr.responseText + ")");
39.             toastr.error(err.message);
40.         },
41.         beforeSend: function ()
42.         {
43.             $("#divLoading").show();
44.         },
45.         success: function (data)
46.         {
47.             PlayerDashboardChart(data);
48.             return false;
49.         },
50.         error: function (xhr, status, error)
51.         {
52.             var err = eval("(" + xhr.responseText + ")");
53.             toastr.error(err.message);
54.         },
55.         complete: function ()
56.         {
57.             $("#divLoading").hide();
58.         }
59.     });
}

```

```
60.    return false;
61. }
62. //This function is used to bind the user data to chart
63. function PlayerDashboardChart(data)
64. {
65.     $("#Player_Chart").show();
66.     var dataArray = [
67.         ['Years', 'Total Runs', 'Total Wickets', 'ODI Matches', 'Test Matches']
68.     ];
69.     $.each(data, function (i, item)
70.     {
71.         dataArray.push([item.Year, item.TotalRun, item.TotalWickets, item.ODIMatches, item.TestMatches]);
72.     });
73.     var data = google.visualization.arrayToDataTable(dataArray);
74.     var options = {
75.         pointSize: 5,
76.         legend:
77.         {
78.             position: 'top',
79.             textStyle:
80.             {
81.                 color: '#f5f5f5'
82.             }
83.         },
84.         colors: ['#34A853', 'ff6600', '#FBBC05'],
85.         backgroundColor: '#454545',
86.         hAxis:
87.         {
88.             title: 'Years',
89.             titleTextStyle:
90.             {
91.                 italic: false,
92.                 color: '#00BBF1',
93.                 fontSize: '20'
94.             },
95.             textStyle:
96.             {
97.                 color: '#f5f5f5'
98.             }
99.         },
100.        vAxis:
101.        {
102.            baselineColor: '#f5f5f5',
103.            title: 'Statistics',
104.            titleTextStyle:
105.            {
106.                color: '#00BBF1',
107.                italic: false,
108.                fontSize: '20'
109.            },
110.            textStyle:
111.            {
112.                color: '#f5f5f5'
113.            },
114.            title: 'Statistics',
115.            titleTextStyle:
116.            {
117.                color: '#00BBF1',
118.                italic: false,
119.                fontSize: '20'
120.            },
121.            textStyle:
122.            {
123.                color: '#f5f5f5'
124.            }
125.        }
126.    }
127. }
```

```

114.     viewWindow:
115.     {
116.         min: 0,
117.         format: 'long'
118.     }
119. },
120. };
121. var chart = new google.visualization.LineChart(document.getElementById('Player_Chart'));
122. chart.draw(data, options);
123. return false;
124. };</script>

```

Step 7: Edit the Views/Shared/_Layout page from the following code,

```

1.  <!DOCTYPE html>
2.  <html>
3.
4.  <head>
5.      <meta charset="utf-8" />
6.      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7.      <title>@ViewBag.Title -
MVC Chart App</title> @Styles.Render("~/Content/css") @Scripts.Render("~/bundles/modernizr") </head>
8.
9.  <body>
10. <div class="navbar navbar-inverse navbar-fixed-top">
11.     <div class="container">
12.         <div class="navbar-header">
13.             <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-
collapse"><span class="icon-bar"></span><span class="icon-bar"></span><span class="icon-
bar"></span> </button> @Html.ActionLink("Chart Sample", "Index", "Home", new { area = "" }, new { @class = "navb
ar-brand" }) </div>
14.         <div class="navbar-collapse collapse">
15.             <ul class="nav navbar-nav">
16.                 <li>@Html.ActionLink("Home", "Index", "Home")</li>
17.                 <li>@Html.ActionLink("About", "About", "Home")</li>
18.                 <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
19.                 <li>@Html.ActionLink("Players", "PlayerChart", "Player")</li>
20.             </ul> @Html.Partial("_LoginPartial") </div>
21.         </div>
22.     </div>
23.     <div class="container body-content"> @RenderBody()
24.         <hr />
25.         <footer>
26.             <p>© @DateTime.Now.Year - MVC Chart Sample</p>
27.         </footer>
28.     </div> @Scripts.Render("~/bundles/jquery") @Scripts.Render("~/bundles/bootstrap") @RenderSection("scripts", r
equired: false) </body>
29.
30. </html>

```

Step 8: Now run the application and click on the Players list from the start page,

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

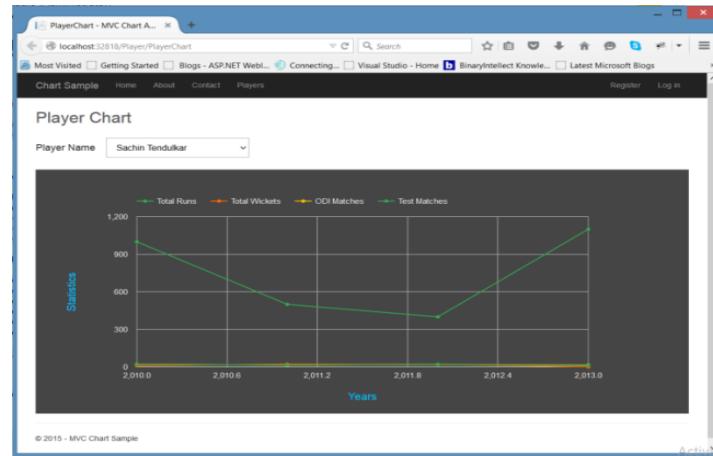


Figure 16: Chart Page

Now you can change the player name from the list box and find out the difference. When you hover the point it will show the following statistics:

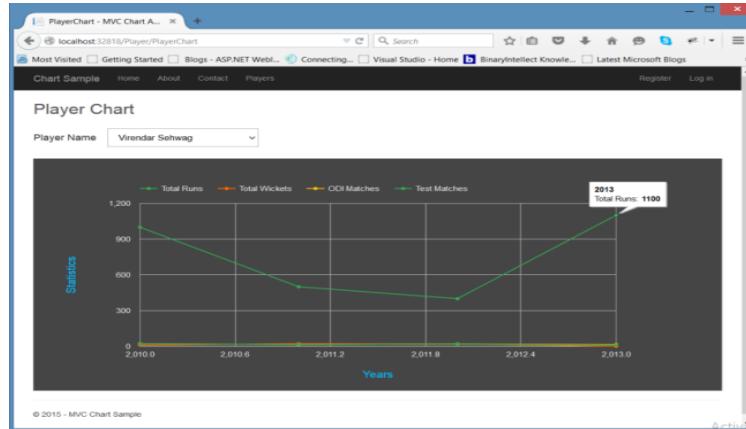


Figure 17: Notations in Chart Page

Chapter 26: Binding View With Model Using Enterprise Library In ASP.NET MVC 5

I am using Microsoft Enterprise Library to perform the database operation. Microsoft provides this library to perform the database operations which is very flexible to use with the web application.

So, let's start with the following procedure:

- Creating Solution
- Perform Database Operation
- Working with Microsoft Enterprise Library
- Binding Model with View
- Binding Data with View using jQuery

Prerequisites

There are the following prerequisite before getting started with the scenario,

- Visual Studio 2013 or Visual Studio 2015

Note: I am creating this project in Visual Studio 2015.

26.1 Creating Solution

In this section we will create a blank solution and in the blank solution we will categorize the project into different categories as “**Web**”, “**Models**” and “**Infrastructure**”. In the Web folder we will create the web application with the help of ASP.NET MVC 5 Project Template and in the rest two solution folders we will create the “**Class Library**” for Models and “**Core**” for database operations. So let's start with the following steps:

Step 1: Open Visual Studio 2015 and click on “**New Project**”,

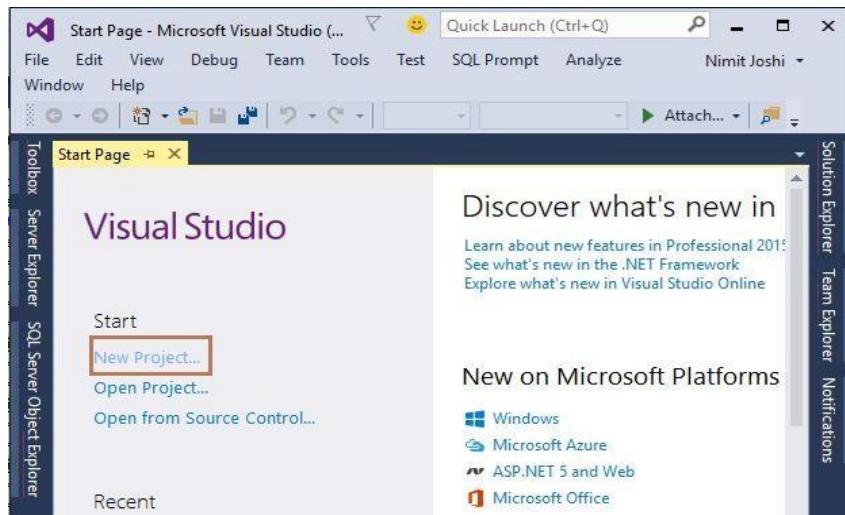


Figure 1: Visual Studio Start Page

Step 2: Select Visual Studio Solutions from the left pane and select the “**Blank Solution**” named “**CricketMasters**”.

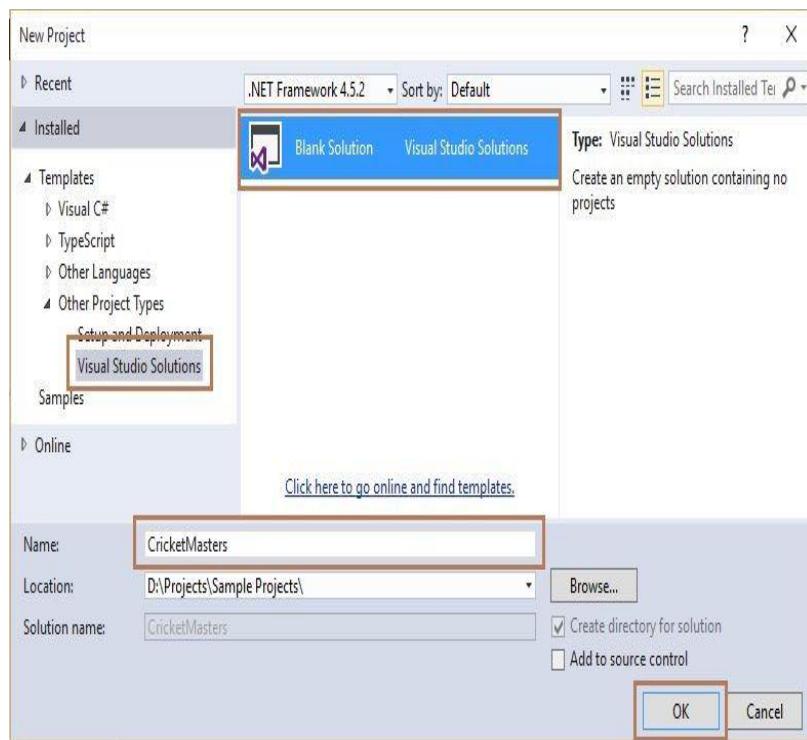


Figure 2: Creating Blank Solution

Step 3: Right click on the “CricketMasters” from the Solution Explorer and add a “New Solution Folder” named “Web”,

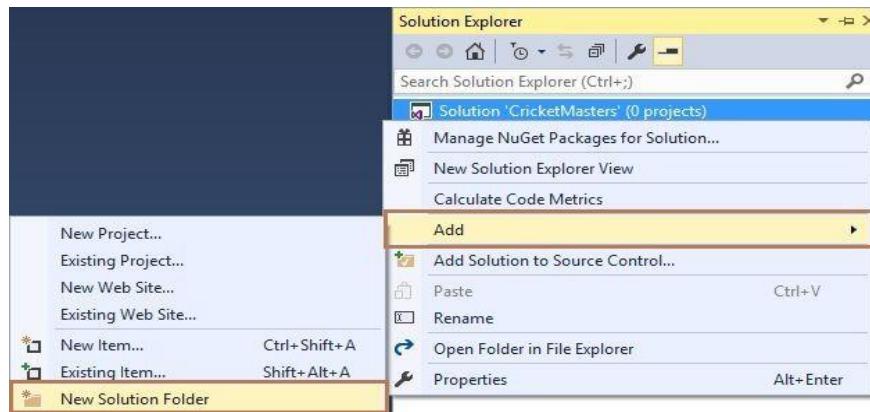


Figure 3: Adding New Solution Folder

Create two more solutions folder named “**Infrastructure**” and “**Models**”.

Step 4: Right click on the “**Web**” folder and select “**New**” and click “**Add New Project**”,

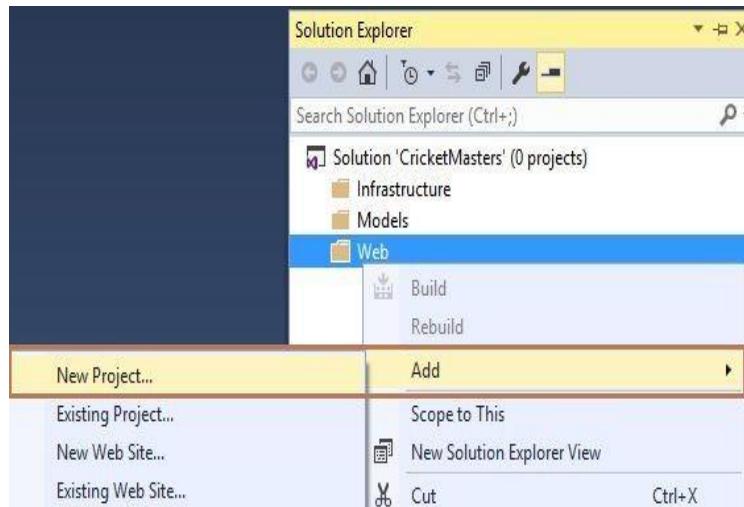


Figure 4: Adding New Project In Solution

Step 5: Select the Web from the left pane in the next wizard and click on the “**ASP.NET Web Application**” to create web application named “**CricketMastersWeb**”

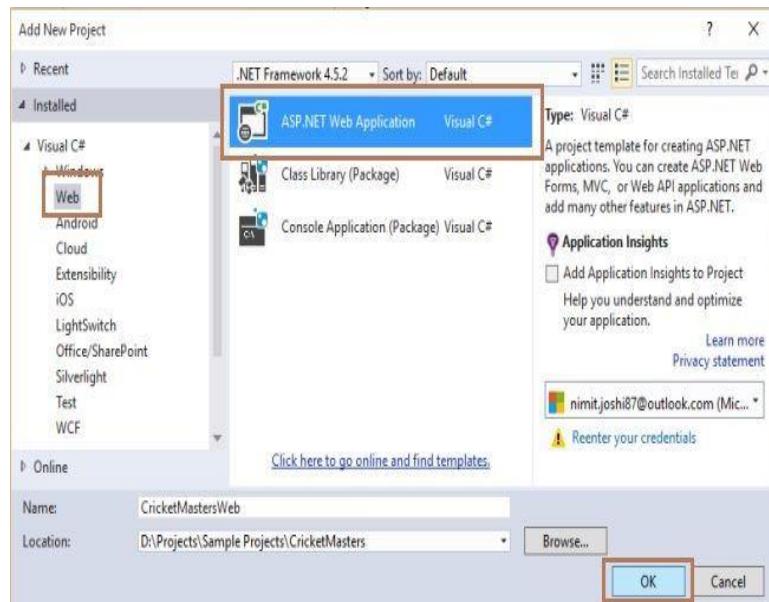


Figure 5: Adding Web Application Project

Step 6: In the next “ASP.NET” wizard select the “**MVC Project Template**” to create ASP.NET MVC Application.

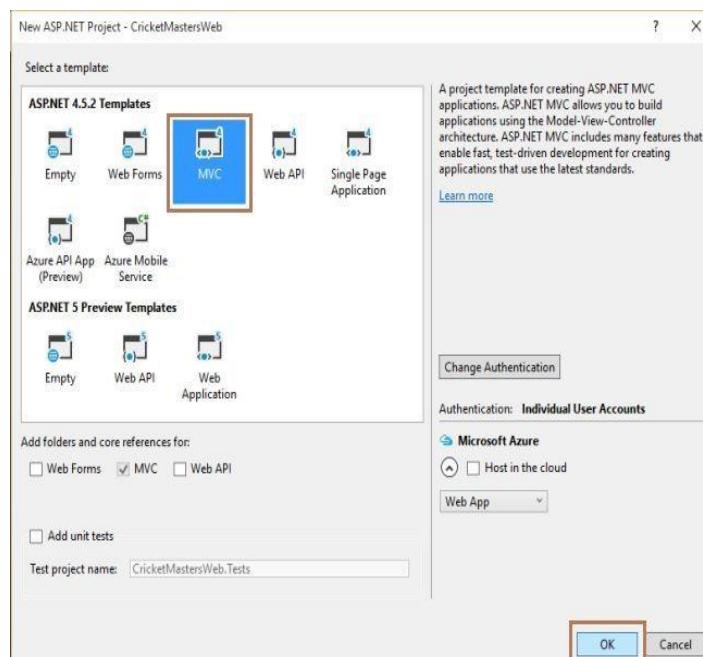


Figure 6: MVC Project Template

Now your web project created successfully.

26.2 Perform Database Operation

In this section we will create the database for the application and tables with the records. We will create stored procedures for getting records from the database. Begin with the following steps:

Step 1: Create a database with the following query:

```
1. CREATE DATABASE Cricketer
```

Step 2: Now, let's execute the following SQL Script for creating table with the records and stored procedures:

```
1. USE [Cricketer]
2. GO
3. /****** Object: Table [dbo].[CricketerProfile]  Script Date: 12/13/2015 1:33:41 PM *****/
4. SET ANSI_NULLS ON
5. GO
6. SET QUOTED_IDENTIFIER ON
7. GO
8. SET ANSI_PADDING ON
9. GO
10. IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[CricketerProfile]') AND type in (N'U'))
11. BEGIN
12. CREATE TABLE [dbo].[CricketerProfile](
13.     [ID] [int] IDENTITY(1,1) NOT NULL,
14.     [Name] [varchar](50) NULL,
15.     [ODI] [int] NULL,
16.     [Tests] [int] NULL,
17.     [ODIRuns] [int] NULL,
18.     [TestRuns] [int] NULL,
19.     CONSTRAINT [PK_CricketerProfile] PRIMARY KEY CLUSTERED
20.     (
21.         [ID] ASC
22.     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
23. ) ON [PRIMARY]
24. END
25. GO
26. SET ANSI_PADDING OFF
27. GO
28. SET IDENTITY_INSERT [dbo].[CricketerProfile] ON
29.
30. GO
31. INSERT [dbo].[CricketerProfile] ([ID], [Name], [ODI], [Tests], [ODIRuns], [TestRuns]) VALUES (1, N'Sachin Tendulkar', 4
63, 200, 18426, 15921)
```

```

32. GO
33. INSERT [dbo].[CricketerProfile] ([ID], [Name], [ODI], [Tests], [ODIRuns], [TestRuns]) VALUES (2, N'Saurav Ganguly', 311
, 113, 11363, 7212)
34. GO
35. INSERT [dbo].[CricketerProfile] ([ID], [Name], [ODI], [Tests], [ODIRuns], [TestRuns]) VALUES (3, N'Rahul Dravid', 344, 1
64, 10889, 13228)
36. GO
37. INSERT [dbo].[CricketerProfile] ([ID], [Name], [ODI], [Tests], [ODIRuns], [TestRuns]) VALUES (4, N'V.V.S. Laxman', 86, 1
34, 2338, 8781)
38. GO
39. INSERT [dbo].[CricketerProfile] ([ID], [Name], [ODI], [Tests], [ODIRuns], [TestRuns]) VALUES (5, N'Virender Sehwag', 2
51, 104, 8273, 8586)
40. GO
41. SET IDENTITY_INSERT [dbo].[CricketerProfile] OFF
42. GO
43. /****** Object: StoredProcedure [dbo].[CC_GetCricketerDetailsById] Script Date: 12/13/2015 1:33:41 PM *****/
44. SET ANSI_NULLS ON
45. GO
46. SET QUOTED_IDENTIFIER ON
47. GO
48. IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[CC_GetCricketerDetailsById]') AND
type in (N'P', N'PC'))
49. BEGIN
50. EXEC dbo.sp_executesql @statement = N'CREATE PROCEDURE [dbo].[CC_GetCricketerDetailsById] AS'
51. END
52. GO
53. ALTER Proc [dbo].[CC_GetCricketerDetailsById]
54. @ID int
55. AS
56. Begin
57.   select * from CricketerProfile (NOLOCK) where ID = @Id
58. End
59.
60. GO
61. /****** Object: StoredProcedure [dbo].[CC_GetCricketerList] Script Date: 12/13/2015 1:33:41 PM *****/
62. SET ANSI_NULLS ON
63. GO
64. SET QUOTED_IDENTIFIER ON
65. GO
66. IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[CC_GetCricketerList]') AND type in
(N'P', N'PC'))
67. BEGIN
68. EXEC dbo.sp_executesql @statement = N'CREATE PROCEDURE [dbo].[CC_GetCricketerList] AS'
69. END
70. GO
71. ALTER Proc [dbo].[CC_GetCricketerList]
72. AS
73. Begin
74.   select ID,Name from CricketerProfile (NOLOCK)
75. End
76. GO

```

3.3 Working with Microsoft Enterprise Library

In this section we will install the Microsoft Enterprise Library from the Manage NuGet Packages in the “**CricketMasters.Core**” project and create the class library for the web application. So, let’s start with the following steps:

Step 1: In the Solution Explorer, right click on the Models and add a “**Class Library Project**” by clicking on “**New Project**” in the Add sub menu as “**CricketMasters.Models**”

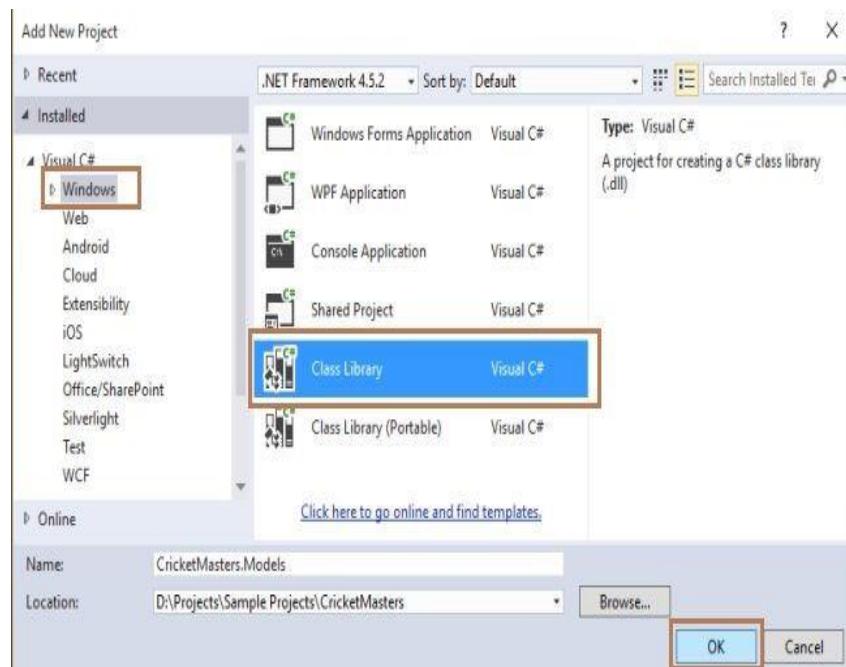


Figure 7: Adding Class Library Project

Step 2: Add a new class in the Models project.

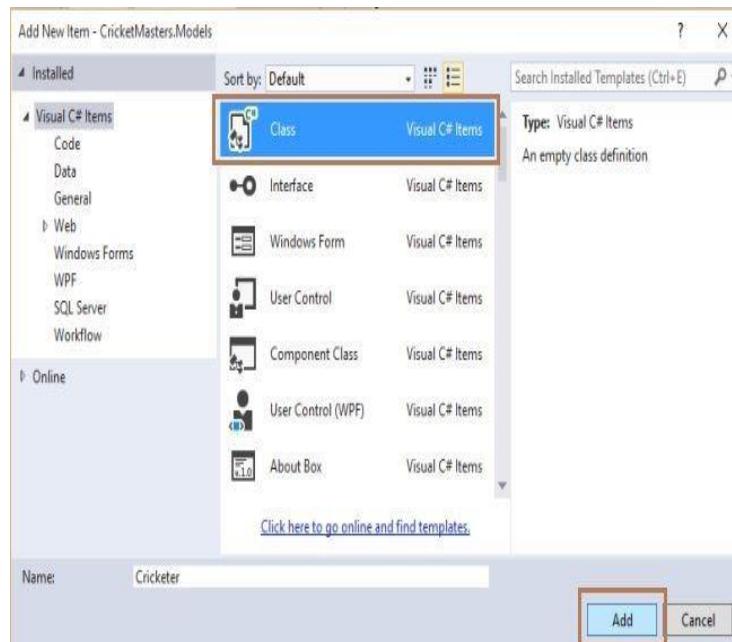


Figure 8: Adding New Class in Project

Step 3: Replace the class with the following code of class:

```

1.  namespace CricketMasters.Models
2.  {
3.      #region Cricketer Class
4.      ///<summary>
5.      /// This class is used for the cricketers
6.      ///</summary>
7.      public class Cricketer
8.      {
9.          #region Properties
10.         ///<summary>
11.         /// get and set the ID
12.         ///</summary>
13.         public int ID { get; set; }
14.         ///<summary>
15.         /// get and set the Name
16.         ///</summary>
17.         public string Name { get; set; }
18.         ///<summary>
19.         /// get and set the ODI
20.         ///</summary>
21.         public int ODI { get; set; }
22.         ///<summary>
23.         /// get and set the Tests
24.         ///</summary>
25.         public int Tests { get; set; }

```

```

26. /// <summary>
27. /// get and set the OdiRuns
28. /// </summary>
29. public int OdiRuns { get; set; }
30. /// <summary>
31. /// get and set the TestRuns
32. /// </summary>
33. public int TestRuns { get; set; }
34. /// <summary>
35. /// get and set the Cricketers
36. /// </summary>
37. public List<Cricketer> Cricketers { get; set; }
38. #endregion
39. }
40. #endregion
41. }

```

Note: Build the solution.

Step 4: In the Solution Explorer, right click on the Infrastructure folder and add a “**New Project**” named “**CricketMasters.Core**”.

Step 5: Right click on the project in the Infrastructure folder and add two new folders names “**BLL**” and “**DAL**”.

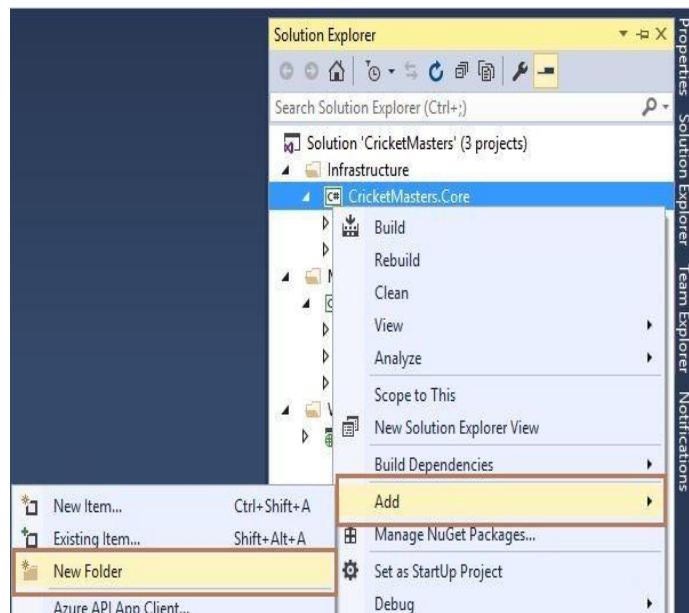


Figure 9: Adding New Folder in Project

Step 6: Right click on the “**References**” in the core project and add a reference of the Models project.

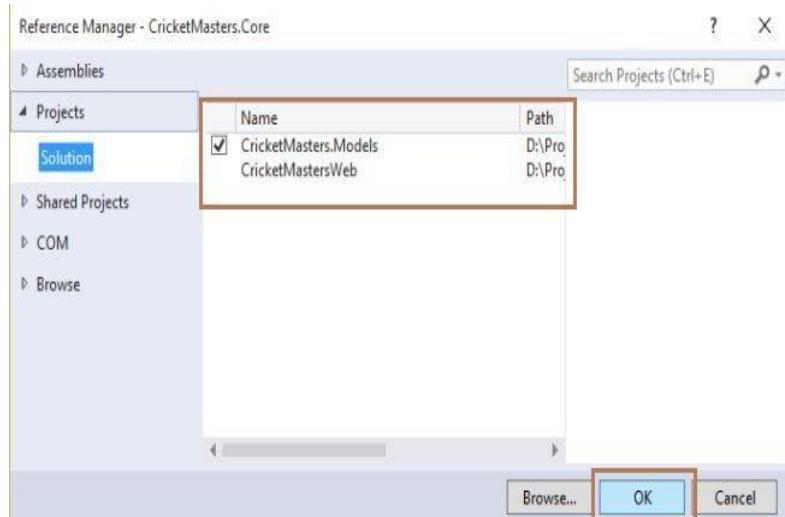


Figure 10: Adding Model Reference

Step 7: Right click on the core project and click on “**Manage NuGet Packages**”

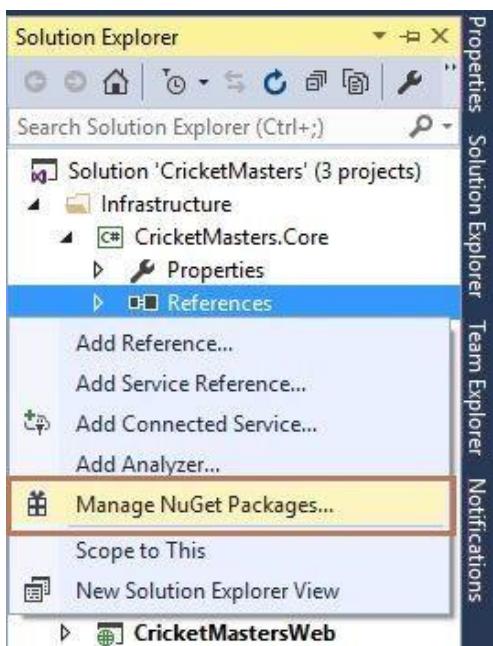


Figure 11: Adding NuGet Package

Step 8: Search “Enterprise Library” and install the library,

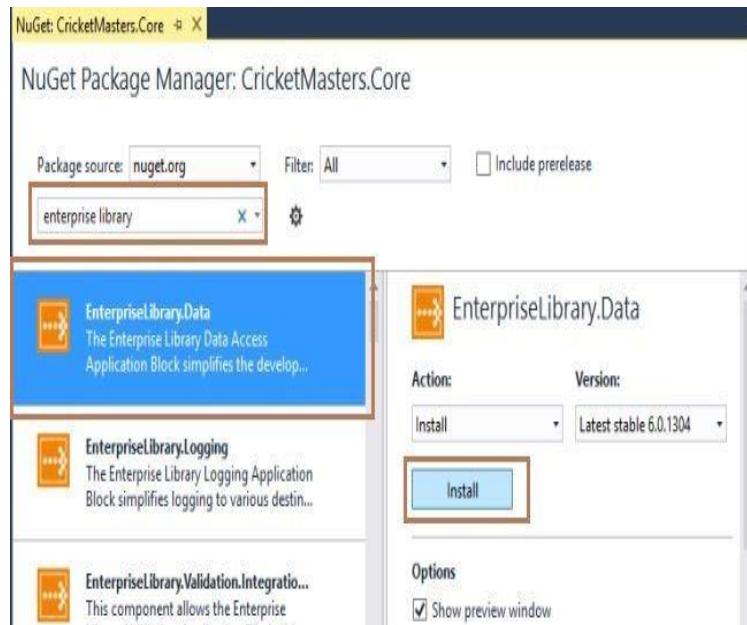


Figure 12: Adding Microsoft Enterprise Library

Step 9: Add a class in the DAL folder.

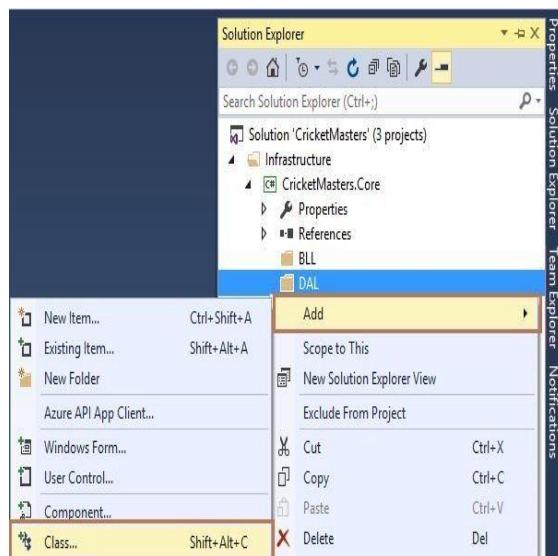


Figure 13: Adding New Class in Folder

Step 10: Replace the DAL class code with the following code:

```

1. using CricketMasters.Models;
2. using Microsoft.Practices.EnterpriseLibrary.Data;
3. using Microsoft.Practices.EnterpriseLibrary.Data.Sql;
4. using System;
5. using System.Collections.Generic;
6. using System.Configuration;
7. using System.Data;
8. using System.Data.Common;
9. using System.Linq;
10. using System.Reflection;
11.
12. namespace CricketMasters.Core.DAL
13. {
14.     #region Cricketer DAL
15.     ///<summary>
16.     /// This class is used for Cricketer Data Access Class
17.     ///</summary>
18.     public class CricketerDAL
19.     {
20.         #region Variable
21.         ///<summary>
22.         /// Specify the Database variable
23.         ///</summary>
24.         Database objDB;
25.         ///<summary>
26.         /// Specify the static variable
27.         ///</summary>
28.         static string ConnectionString;
29.         #endregion
30.
31.         #region Constructor
32.         ///<summary>
33.         /// This constructor is used to get the connectionstring from the config file
34.         ///</summary>
35.         public CricketerDAL()
36.         {
37.             ConnectionString = ConfigurationManager.ConnectionStrings["CricketerConnectionString"].ToString();
38.         }
39.         #endregion
40.
41.         #region Database Method
42.         public List<T> ConvertTo<T>(DataTable datatable) where T : newnew List<T>();
45.             try
46.             {
47.                 List<string> columnsNames = new List<string>();
48.                 foreach ( DataColumn DataColumn in datatable.Columns)
49.                     columnsNames.Add(DataColumn.ColumnName);
50.                 Temp = datatable.AsEnumerable().ToList().ConvertAll<T>(row => getObject<T>(row, columnsNames));
51.             return Temp;

```

```

52.    }
53.    catch
54.    {
55.        return Temp;
56.    }
57. }
58. public T getObject<T>(DataRow row, List<string> columnsName) where T : newnew T();
61.     try
62.     {
63.         string columnname = "";
64.         string value = "";
65.         PropertyInfo[] Properties;
66.         Properties = typeof(T).GetProperties();
67.         foreach (PropertyInfo objProperty in Properties)
68.         {
69.             columnname = columnsName.Find(name => name.ToLower() == objProperty.Name.ToLower());
70.             if (!string.IsNullOrEmpty(columnname))
71.             {
72.                 value = row[columnname].ToString();
73.                 if (!string.IsNullOrEmpty(value))
74.                 {
75.                     if (Nullable.GetUnderlyingType(objProperty.PropertyType) != null)
76.                     {
77.                         value = row[columnname].ToString().Replace("$", "").Replace(",", "");
78.                         objProperty.SetValue(obj, Convert.ChangeType(value, Type.GetType(Nullable.GetUnderlyingType(o
bjProperty.PropertyType).ToString()))), null);
79.                     }
80.                 else
81.                 {
82.                     value = row[columnname].ToString();
83.                     objProperty.SetValue(obj, Convert.ChangeType(value, Type.GetType(objProperty.PropertyType.ToS
tring()))), null);
84.                 }
85.             }
86.         }
87.     }
88.     return obj;
89. }
90. catch (Exception ex)
91. {
92.     return obj;
93. }
94. }
95. #endregion
96.
97. #region College
98. /// <summary>
99. /// This method is used to get the cricketer data
100. /// </summary>
101. /// <returns></returns>
102. public List<Cricketer> GetCricketerList()
103. {

```

```

104.    List<Cricketer> objGetCricketers = null;
105.    objDB = new SqlDatabase(ConnectionString);
106.    using (DbCommand objcmd = objDB.GetStoredProcCommand("CC_GetCricketerList"))
107.    {
108.        try
109.        {
110.            using (DataTable dataTable = objDB.ExecuteDataSet(objcmd).Tables[0])
111.            {
112.                objGetCricketers = ConvertTo<Cricketer>(dataTable);
113.            }
114.        }
115.        catch (Exception ex)
116.        {
117.            throw ex;
118.            return null;
119.        }
120.    }
121.    return objGetCricketers;
122. }
123.
124. ///<summary>
125. /// This method is used to get cricketers details by cricketer id
126. ///</summary>
127. ///<returns></returns>
128. public List<Cricketer> GetCricketerDetailsById(int Id)
129. {
130.    List<Cricketer> objCricketerDetails = null;
131.    objDB = new SqlDatabase(ConnectionString);
132.    using (DbCommand objcmd = objDB.GetStoredProcCommand("CC_GetCricketerDetailsById"))
133.    {
134.        try
135.        {
136.            objDB.AddInParameter(objcmd, "@ID", DbType.Int32, Id);
137.
138.            using (DataTable dataTable = objDB.ExecuteDataSet(objcmd).Tables[0])
139.            {
140.                objCricketerDetails = ConvertTo<Cricketer>(dataTable);
141.            }
142.        }
143.        catch (Exception ex)
144.        {
145.            throw ex;
146.            return null;
147.        }
148.    }
149.    return objCricketerDetails;
150. }
151. #endregion
152. }
153. #endregion
154. }

```

Step 11: Now add a class in the BLL folder named “**CricketerBL**” and add the following code in the class:

```
1. using CricketMasters.Core.DAL;
2. using CricketMasters.Models;
3. using System;
4. using System.Collections.Generic;
5.
6. namespace CricketMasters.Core.BLL
7. {
8.     public class CricketerBL
9.     {
10.         public List<Cricketer> GetCricketerList()
11.         {
12.             List<Cricketer> ObjCricketers = null;
13.             try
14.             {
15.                 ObjCricketers = new CricketerDAL().GetCricketerList();
16.             }
17.             catch (Exception)
18.             {
19.
20.                 throw;
21.             }
22.             return ObjCricketers;
23.         }
24.         /// <summary>
25.         /// This method is used to get cricketers details by cricketer id
26.         /// </summary>
27.         /// <returns></returns>
28.         public List<Cricketer> GetCricketerDetailsById(int Id)
29.         {
30.             List<Cricketer> ObjCricketerDetails = null;
31.             try
32.             {
33.                 ObjCricketerDetails = new CricketerDAL().GetCricketerDetailsById(Id);
34.             }
35.             catch (Exception)
36.             {
37.
38.                 throw;
39.             }
40.             return ObjCricketerDetails;
41.         }
42.     }
43. }
```

Step 12: Build the solution.

26.4 Binding Model with View

In this section we will create the empty MVC 5 controller and add a view for displaying the details. We will also bind the view from the data by passing the data from the model to the controller. So, let's begin with the following steps:

Step 1: In the Web Project, right click on the Controllers folder go to Add and click on the “New Scaffolded Item”

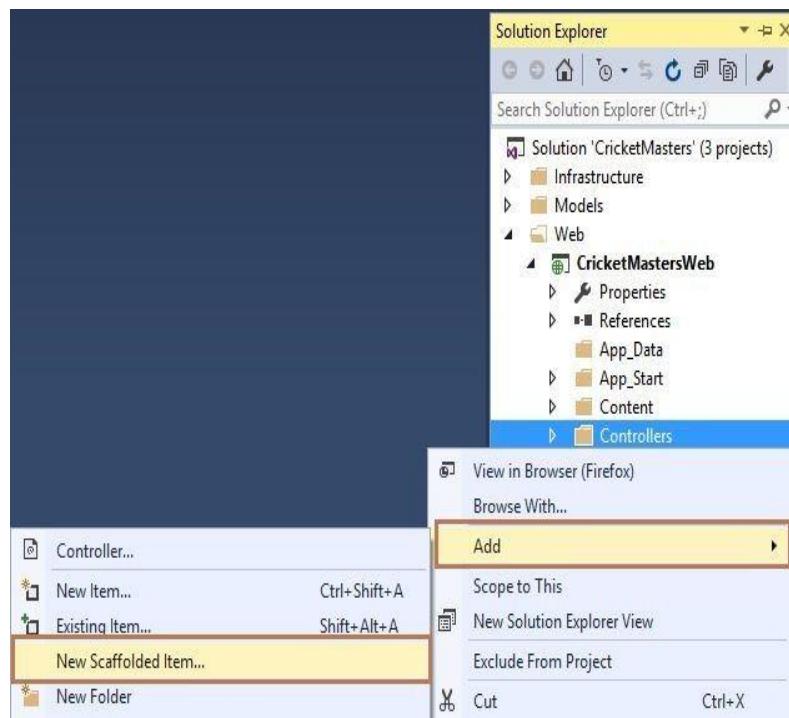


Figure 14: Adding New Scaffolded Item

Step 2: In the next wizard, select the “MVC 5 Empty Controller”.

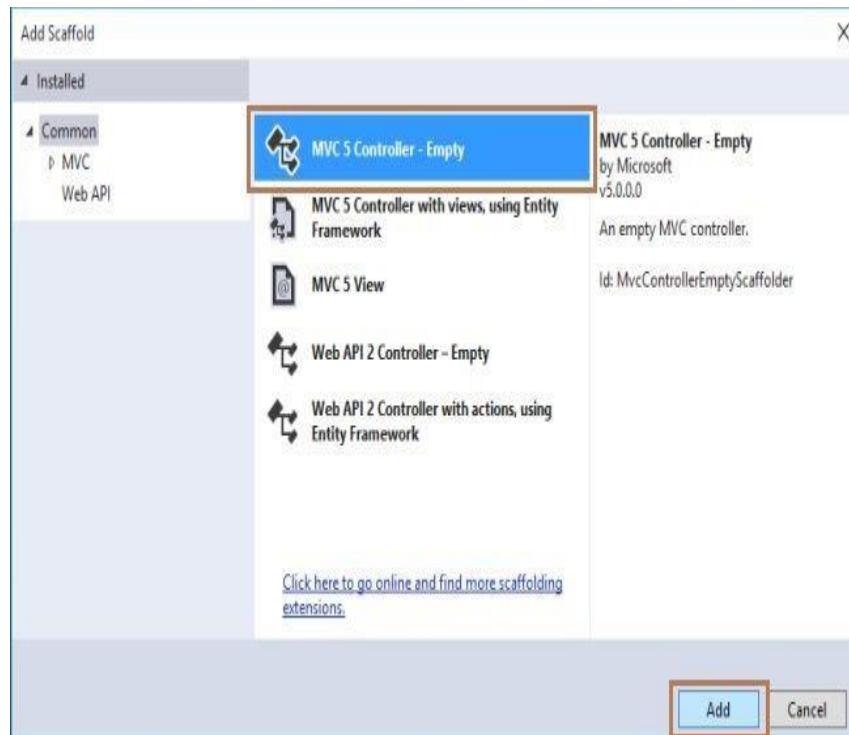


Figure 15: Add Scaffold Wizard

Specify the controller name as “**CricketersController**”

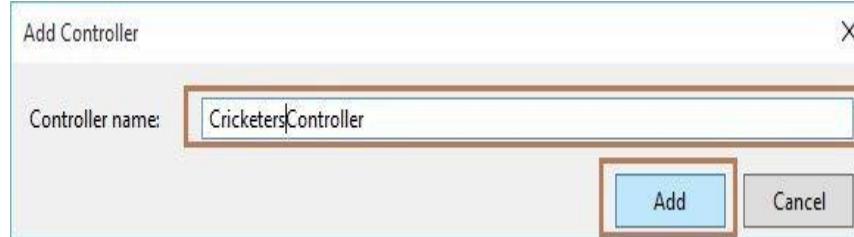


Figure 16: Adding New Controller

Step 3: Add a reference of Models and Core Project in the Web Project.

Step 4: In the CricketersController, replace the code with the following code:

```

1. using CricketMasters.Core.BLL;
2. using CricketMasters.Models;
3. using System;
4. using System.Collections.Generic;
5. using System.Linq;
6. using System.Web.Mvc;
7.
8. namespace CricketMastersWeb.Controllers
9. {
10.     public class CricketersController : Controller
11.     {
12.         #region Variable
13.         ///<summary>
14.         /// This variable is used for Cricketer BL
15.         ///</summary>
16.         public CricketerBL cricketerBL;
17.         #endregion
18.
19.         #region Cricketer
20.         ///<summary>
21.         /// This method is used to get all cricketer names
22.         ///</summary>
23.         ///<returns></returns>
24.         [HttpGet, ActionName("GetAllCricketer")]
25.         public ActionResult GetAllCricketer()
26.         {
27.             List<Cricketer> objCricketer = new List<Cricketer>();
28.
29.             var response = new CricketerBL().GetCricketerList();
30.             if (!object.Equals(response, null))
31.             {
32.                 objCricketer = response.ToList();
33.             }
34.             return View("~/Views/Cricketers/Cricketer.cshtml", new Cricketer { Cricketers = objCricketer });
35.         }
36.         #endregion
37.     }
38. }
```

In the above code, there is an action method which is used to get all cricketer names from the database. You can see that there is a highlighted code, that code is used to bind the model with the cricketer names which is further used to bind the cricketer names in the dropdownlist.

Step 5: Now add a view, by right click on the Views, then Cricketers

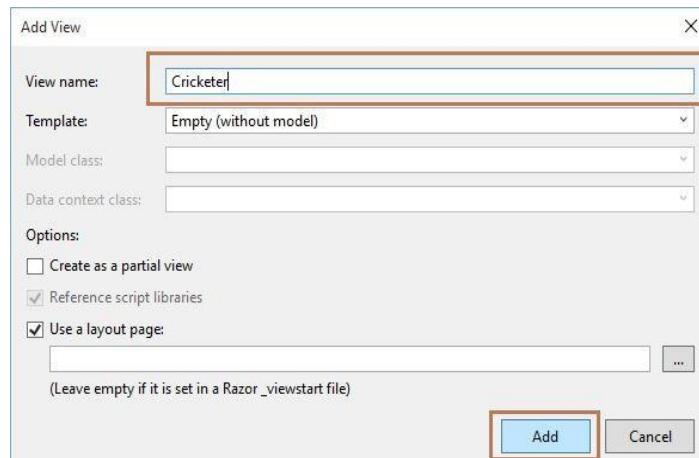


Figure 17: Adding View

Step 6: Replace the view code with the following code:

```

1. @model CricketMasters.Models.Cricketer
2. @{
3.     ViewBag.Title = "Cricketer";
4. }
5.
6. <h2>Cricketer Statistics</h2>
7.
8. <div class="row">
9.     <div class="col-md-8">
10.        <hr />
11.        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
12.        <div class="form-group">
13.            @Html.LabelFor(m => m.Name, new { @class = "col-md-2 control-label" })
14.            <div class="col-md-10">
15.                @Html.DropDownListFor(m => m.Name, new SelectList(Model.Cricketers, "ID", "Name"), new { @id = "playerNameList", @class = "form-control" })
16.            </div>
17.        </div>
18.    </div>
19. </div>

```

Step 7: Go to the Web.Config file of the Web application and add the following connection string in the Connection Strings tab:

```

1. <add name="CricketerConnectionString" connectionString="Data Source=Your Server Name;Initial Catalog=Cricketer;
User ID=Your User ID;Password=Your Password" providerName="System.Data.SqlClient" />

```

Note: Please replace the highlighted code with your server credentials.

Step 8: Build the solution and now open Views, Shared, then _Layout.cshtml file and change the code with the highlighted code below:

```

1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.      <meta charset="utf-8" />
5.      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6.      <title>@ViewBag.Title - Cricket Masters Application</title>
7.      @Styles.Render("~/Content/css")
8.      @Scripts.Render("~/bundles/modernizr")

9.
10. </head>
11. <body>
12.     <div class="navbar navbar-inverse navbar-fixed-top">
13.         <div class="container">
14.             <div class="navbar-header">
15.                 <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
16.                     <span class="icon-bar"></span>
17.                     <span class="icon-bar"></span>
18.                     <span class="icon-bar"></span>
19.                 </button>
20.                 @Html.ActionLink("Cricket Masters", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
21.             </div>
22.             <div class="navbar-collapse collapse">
23.                 <ul class="nav navbar-nav">
24.                     <li>@Html.ActionLink("Home", "Index", "Home")</li>
25.                     <li>@Html.ActionLink("About", "About", "Home")</li>
26.                     <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
27.                     <li>@Html.ActionLink("Cricketers", "GetAllCricke", "Cricketers")</li>
28.                 </ul>
29.                 @Html.Partial("_LoginPartial")
30.             </div>
31.         </div>
32.     </div>
33.     <div class="container body-content">
34.         @RenderBody()
35.         <hr />
36.         <footer>
37.             <p>© @DateTime.Now.Year - Cricket Masters</p>
38.         </footer>
39.     </div>
40.
41.     @Scripts.Render("~/bundles/jquery")
42.     @Scripts.Render("~/bundles/bootstrap")
43.     @RenderSection("scripts", required: false)
44. </body>
45. </html>

```

Step 9: Now run the project. Click on the **Cricketers** link to open the view as in the following,

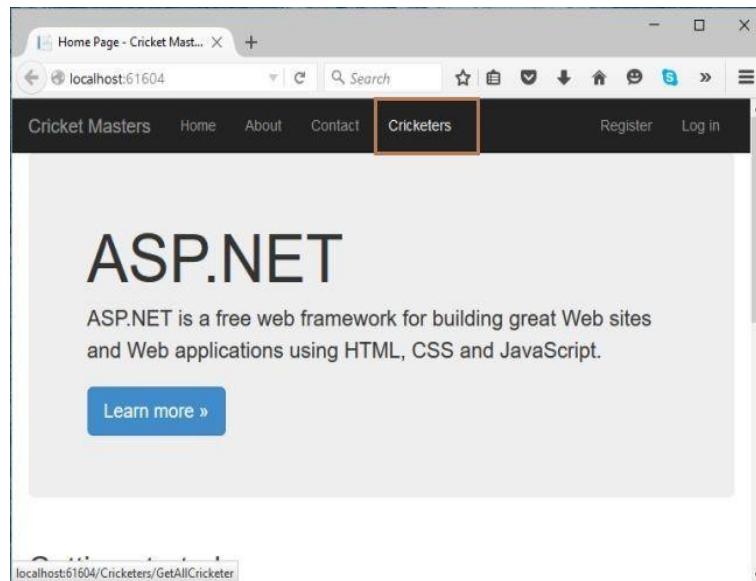


Figure 18: Opening View in MVC 5

When you click on the “**Cricketers**” link, you can see that your DropDownList has the values which has been passed using the model. Take a look.

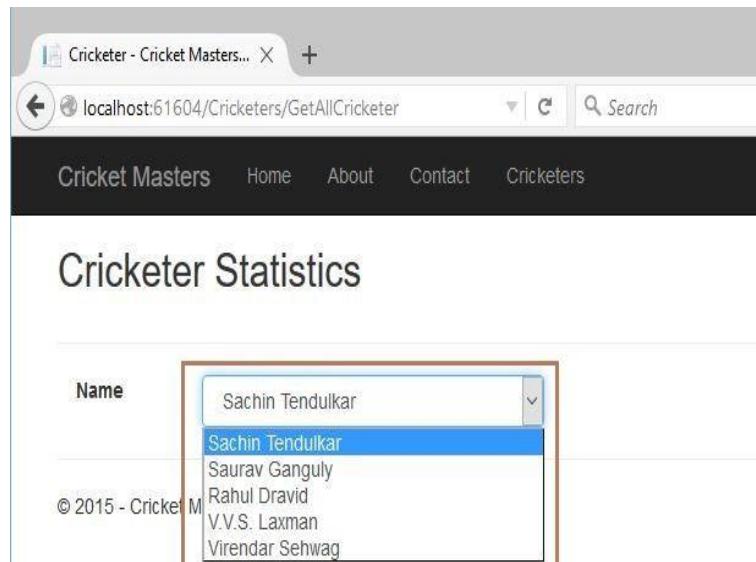


Figure 19: Binding Data in DropDownList Using Model

26.5 Binding Data with View using jQuery

In this section we will bind the details of the particular player in the table. We will use jQuery to send the id to the database to fetch the details of the player. So, let's begin with the following steps:

Step 1: Add the following method in the “**CricketersController**”.

```

1. /// <summary>
2. /// This method is used to get all cricketer details based on the cricketer id
3. /// </summary>
4. /// <param name="CricketerId"></param>
5. /// <returns></returns>
6. [HttpGet, ActionName("GetCricketerDetailsById")]
7. public JsonResult GetCricketerDetailsById(int CricketerId)
8. {
9.     List<Cricketer> objCricketerDetails = new List<Cricketer>();
10.
11.    var response = new CricketerBL().GetCricketerDetailsById(CricketerId);
12.    if (!object.Equals(response, null))
13.    {
14.        objCricketerDetails = response.ToList();
15.    }
16.    return Json(objCricketerDetails, JsonRequestBehavior.AllowGet);
17. }
```

In the above method we get the details of the individual player and pass the data to the view in the JSON format.

Step 2: By adding the elements of table data your final view code is as in the following,

```

1. @model CricketerApp.Model.Cricketer
2. @{
3.     ViewBag.Title = "Cricketer";
4. }
5. <script src="~/Scripts/jquery-1.10.2.min.js"></script>
6. <h2>Cricketer Statistics</h2>
7.
8. <div class="row">
9.     <div class="col-md-8">
10.         <hr />
11.         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
12.         <div class="form-group">
13.             @Html.LabelFor(m => m.Name, new { @class = "col-md-2 control-label" })
14.             <div class="col-md-10">
15.                 @Html.DropDownListFor(m => m.Name, new SelectList(Model.Cricketers, "ID", "Name"), new { @id = "playerNameList", @class = "form-control", @Onchange = "return GetCricketerDetails();;" })
```

```

16.      </div>
17.    </div>
18.  </div>
19. </div><br />
20. <div class="row">
21.   <div class="form-group">
22.     <div id="CricketList" class="col-md-8"></div>
23.   </div>
24. </div>
25.
26. <script type="text/javascript">
27. $(document).ready(function () {
28.   GetCricketDetails();
29. });
30.
31. function GetCricketDetails() {
32.   var cricketId = $('#playerNameList option:selected').val();
33.   $.ajax({
34.     url: '@Url.Action("GetCricketDetailsById", "Cricketers")',
35.     type: "GET",
36.     dataType: "json",
37.     data: { CricketId: cricketId },
38.     success: function (data) {
39.       $('#CricketList').html(" ");
40.       var html = "";
41.       html += "<table class=\"table\">";
42.       html += "<tr>";
43.       html += "<th>";
44.       html += "@Html.DisplayNameFor(model=>model.Name)";
45.       html += "</th>";
46.       html += "<th>";
47.       html += "@Html.DisplayNameFor(model=>model.ODI)";
48.       html += "</th>";
49.       html += "<th>";
50.       html += "@Html.DisplayNameFor(model=>model.Tests)";
51.       html += "</th>";
52.       html += "<th>";
53.       html += "@Html.DisplayNameFor(model=>model.OdiRuns)";
54.       html += "</th>";
55.       html += "<th>";
56.       html += "@Html.DisplayNameFor(model=>model.TestRuns)";
57.       html += "</th>";
58.       html += "</tr>";
59.       $.each(data, function (index, item) {
60.         html += "<tr>";
61.         html += "<td>";
62.         html += "<lable>" + item.Name + "</lable>"
63.         html += "</td>";
64.         html += "<td>";
65.         html += "<lable>" + item.ODI + "</lable>"
66.         html += "</td>";
67.         html += "<td>";
68.         html += "<lable>" + item.Tests + "</lable>"
69.         html += "</td>";

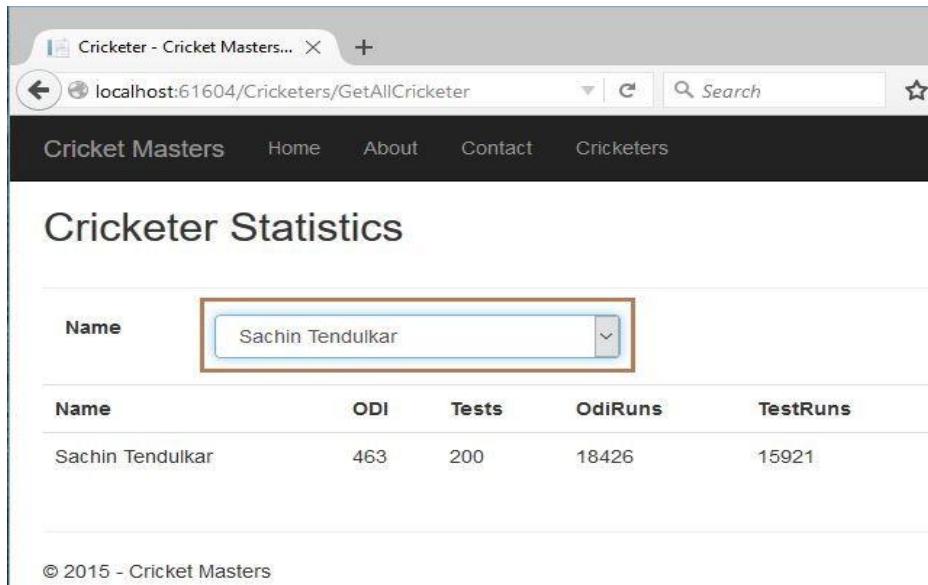
```

```

70.         html += "<td>";
71.         html += "<table>" + item.OdiRuns+ "</table>";
72.         html += "</td>";
73.         html += "<td>";
74.         html += "<table>" + item.TestRuns + "</table>";
75.         html += "</td>";
76.         html += "</tr>";
77.         html += "</table>";
78.     });
79.     $('#CricketerList').append(html);
80. }
81. });
82. return false;
83. }
84. </script>

```

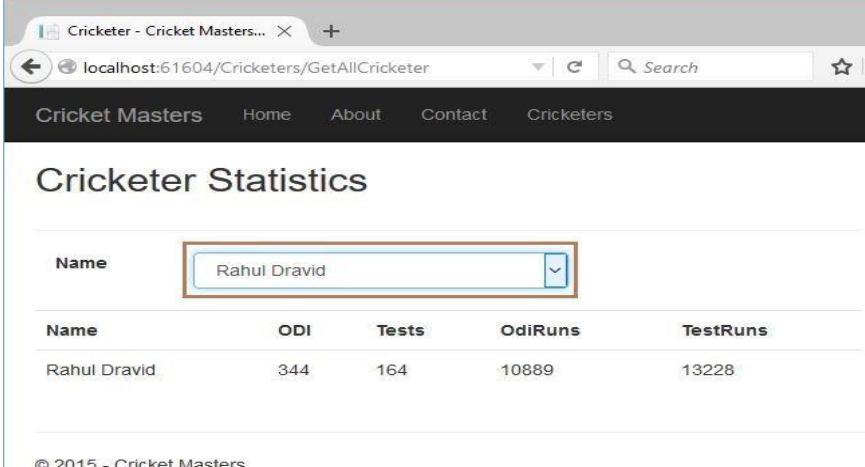
Step 3: Save the page and run the project. You can see in the following screenshot that the data is bind to the table.



Name	ODI	Tests	OdiRuns	TestRuns
Sachin Tendulkar	463	200	18426	15921

Figure 20: Binding Data in View using jQuery

You can now change the selection and you will get the output.



The screenshot shows a web browser window with the title "Cricketer - Cricket Masters...". The address bar displays "localhost:61604/Cricketers/GetAllCricketer". The page header includes links for "Cricket Masters", "Home", "About", "Contact", and "Cricketers". The main content area is titled "Cricketer Statistics". A dropdown menu is open, showing "Rahul Dravid" as the selected item. Below the dropdown is a table with the following data:

Name	ODI	Tests	OdiRuns	TestRuns
Rahul Dravid	344	164	10889	13228

At the bottom of the page, there is a copyright notice: "© 2015 - Cricket Masters".

Figure 21: Binding Data in View in MVC 5

Chapter 27: Paging, Sorting, And Filtering With Partial View In ASP.NET MVC 5

27.1 Description

In my last series on Restful API in ASP.NET, we have successfully created an ASP.NET Web Application with the ASP.NET Web API Project Template. Now, in this chapter I am describing the use of Partial view in MVC 5 with the help of my old API application. We will change up the application with the new requirement. So, let's have a look at Partial View. Partial View performs as user controls (.ascx). With the help of Partial view, we can create a view which will be brought to pass in a parent view. Here in this chapter, we will perform Paging, Sorting and Filtering with the use of Partial View in the MVC 5.

Now, I am assuming that you have gone through my previous chapter so that I can easily create a new section to perform the current requirement.

Prerequisites

You must have Visual Studio 2013 or later version to work on MVC 5 based web applications.

Getting Started

In this section, we will categorize our application into the following three parts:

- Paging
- Sorting
- Filtering

27.2 Adding Web Project

First we have to add a Web Project in our solution. Please follow the steps below to perform:

Step 1: In the Solution Explorer, right click on the Solution and add a new Solution Folder named "Web".

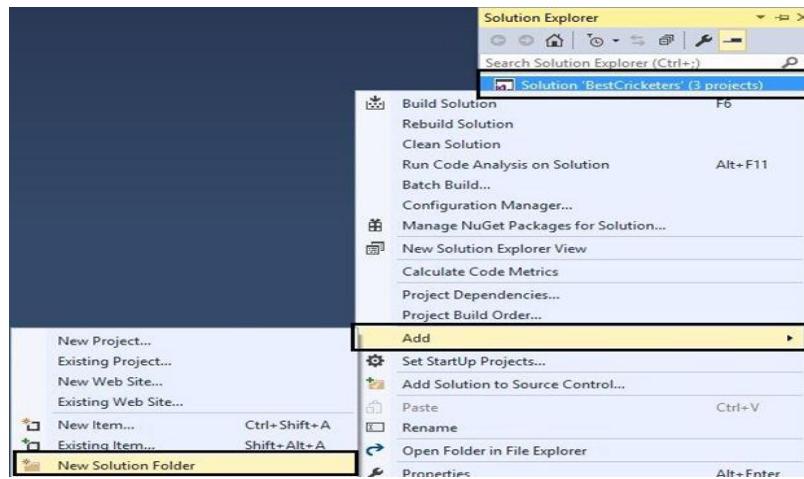


Figure 1: Adding Solution Folder

Step 2: Now right click on the **Web** folder and click on "Add New Project".

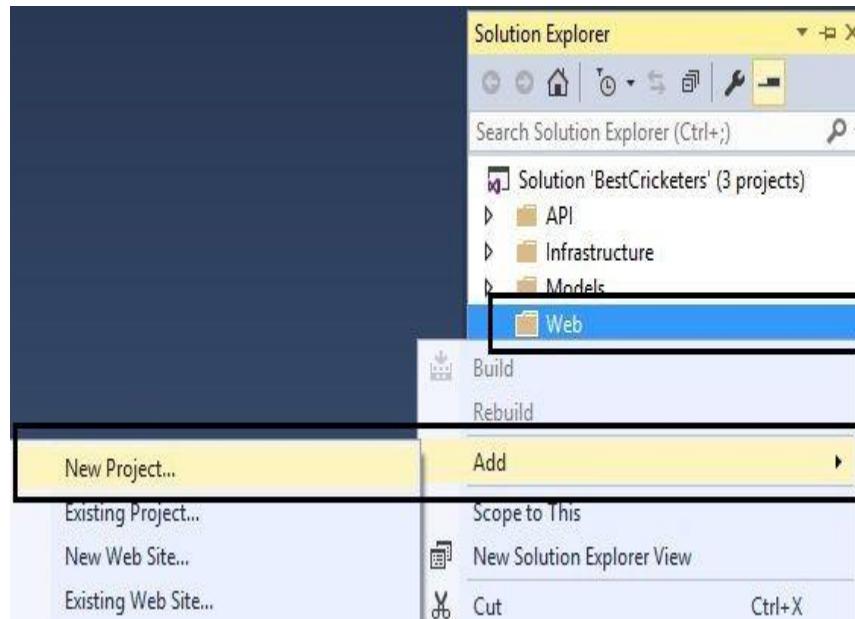


Figure 2: Adding New Project

Step 3: Select the "**Web**" option from the left pane in the next wizard and click on "**ASP.NET Web application**".

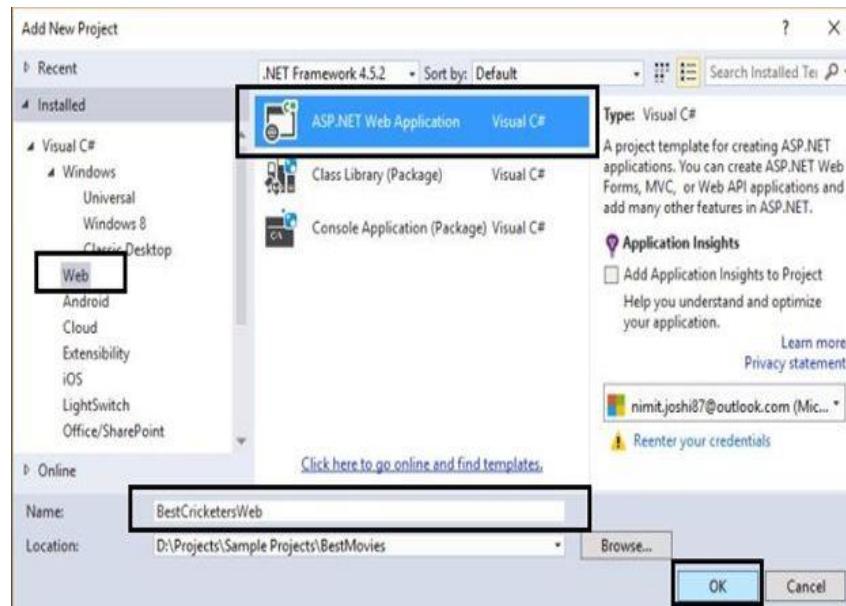


Figure 3: Adding Web Application

Step 4: In the next "ASP.NET" wizard, select the **MVC** Project Template.

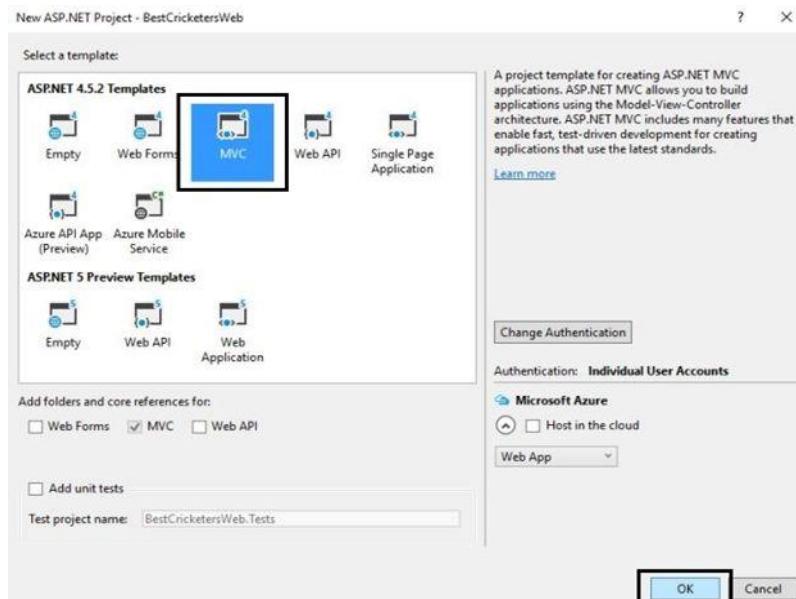


Figure 4: One ASP.NET Wizard

That's it. You have successfully created an MVC Project.

27.3 Paging

In this section we have to change the structure of the stored procs, methods in API; so that we can create a view to get the records. So, follow the following procedure to perform this section:

Step 1: Modify the Stored Procedure as the Paging requirement from the following script:

```

1. ALTER PROC [dbo].[CC_GetCricketerList]
2. @PageNumber INT ,
3. @PageSize INT
4. AS
5. BEGIN
6.     SELECT ROW_NUMBER() OVER ( ORDER BY ID DESC ) [RowNo] ,
7.             *,
8.             COUNT(ID) OVER ( ) [TotalCount]
9.     FROM dbo.CricketerProfile (NOLOCK)
10.    ORDER BY ID
11.    OFFSET ( @PageNumber - 1 ) * @PageSize ROWS
12.    FETCH NEXT @PageSize ROWS ONLY;
13. END;
```

Step 2: Modify the following methods as per the following classes:

CricketerProfile.cs

```

1. public class CricketerProfile : Result
2. {
3.     public int Id { get; set; }
4.     public string Name { get; set; }
5.     public int ODI { get; set; }
6.     public int Tests { get; set; }
7.     public int OdiRuns { get; set; }
8.     public int TestRuns { get; set; }
9.     public int Type { get; set; }
10.    public int TotalCount { get; set; }
11. }
```

CricketerDAL.cs

```

1. public List<CricketerProfile> GetCricketerList(int PageNumber, int PageSize)
2. {
3.     List<CricketerProfile> objGetCricketers = null;
4.     objDB = new SqlDatabase(ConnectionString);
5.     using (DbCommand objcmd = objDB.GetStoredProcedure("CC_GetCricketerList"))
6.     {
7.         try
```

```

8.    {
9.        objDB.AddInParameter(objcmd, "@PageNumber", DbType.Int32, PageNumber);
10.       objDB.AddInParameter(objcmd, "@PageSize", DbType.Int32, PageSize);
11.
12.      using (DataTable dataTable = objDB.ExecuteDataSet(objcmd).Tables[0])
13.      {
14.          objGetCricketers = ConvertTo<CricketerProfile>(dataTable);
15.      }
16.    }
17.    catch (Exception ex)
18.    {
19.        throw ex;
20.        return null;
21.    }
22. }
23. return objGetCricketers;
24. }
```

CricketerBL.cs

```

1.  public List<CricketerProfile> GetCricketerList(int PageNumber, int PageSize)
2.  {
3.      List<CricketerProfile> ObjCricketers = null;
4.      try
5.      {
6.          ObjCricketers = new CricketerDAL().GetCricketerList(PageNumber, PageSize);
7.      }
8.      catch (Exception)
9.      {
10.
11.         throw;
12.     }
13.     return ObjCricketers;
14. }
```

Method in API:

```

1.  [HttpGet, ActionName("GetCricketerList")]
2.  public HttpResponseMessage GetCricketerList(int PageNumber, int PageSize)
3.  {
4.      Result result;
5.      cricketerBL = new CricketerBL();
6.      try
7.      {
8.          var cricketerList = cricketerBL.GetCricketerList(PageNumber, PageSize);
9.
10.         if (!object.Equals(cricketerList, null))
11.         {
12.             response = Request.CreateResponse<List<CricketerProfile>>(HttpStatusCode.OK, cricketerList);
13.         }
14.     }
15.     catch (Exception ex)
```

```

16. {
17.     result = new Result();
18.     result.Status = 0;
19.     result.Message = ex.Message;
20.     response = Request.CreateResponse(HttpStatusCode.InternalServerError, result);
21. }
22. return response;
23. }

```

Step 3: Now first add a reference of Models project in the Web Project.

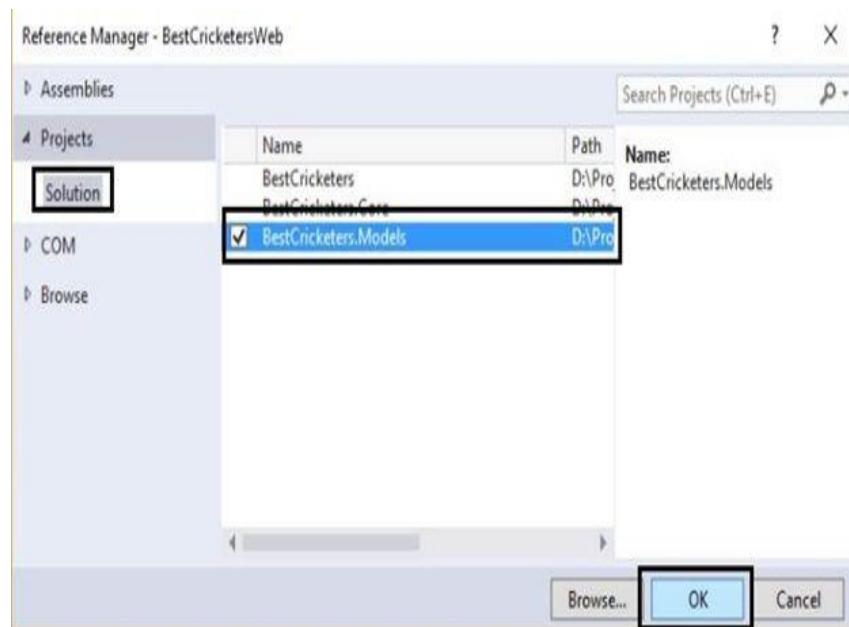


Figure 5: Adding Reference Wizard

Step 4: Add a class named "**CricketerInfo**" in the Models folder of the Web Project and modify the code from the following code in that class:

```

1. public class CricketerInfo
2. {
3.     public List<BestCricketers.Models.CricketerProfile> cricketerProfile { get; set; }
4. }

```

Step 5: Build the solution.

Step 6: Right click on the **Controllers** folder and go to Add, then click Controller

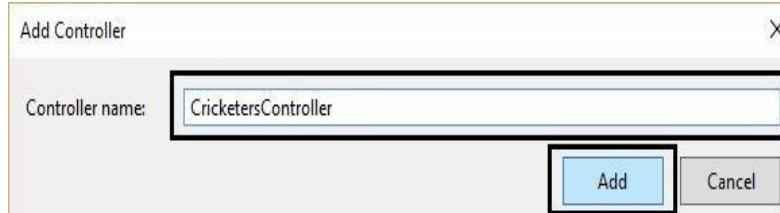


Figure 6: Adding Controller

Step 7: Now in the "Add Scaffold" wizard, select the "MVC 5 Controller - Empty":

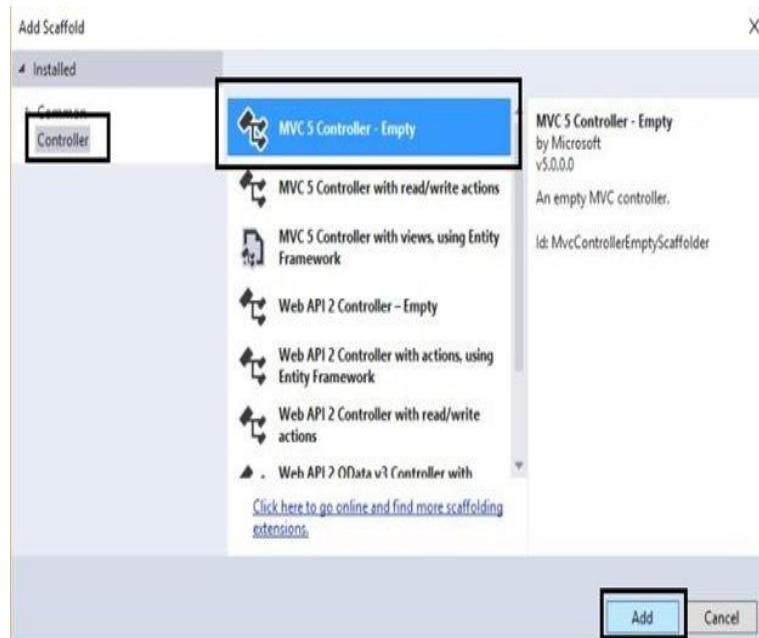


Figure 7: Add Scaffold Wizard

Step 8: In the next wizard, enter the name "**CricketersController**"

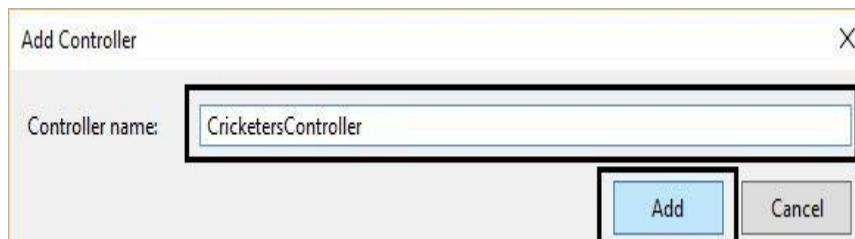


Figure 8: Add Controller

Step 9: Add the following keys in "Web.Config" file:

```

1. <add key="DefaultPageSize" value="5"/>
2. <add key="ApiUrl" value="http://localhost:6416/"></add>
```

Note: Please change the API Url as per your project.

Step 10: Add the following method in the "CricketersController"

```

1. /// <summary>
2. /// This method is used to get cricketers list.
3. /// </summary>
4. /// <param name="PageNumber"></param>
5. /// <param name="PageSize"></param>
6. /// <returns></returns>
7. [HttpGet, ActionName("GetBestCricketers")]
8. public async Task<ActionResult> GetBestCricketers(int? pageNumber, int? pageSize)
9. {
10.     List<CricketerProfile> cricketersList = new List<CricketerProfile>();
11.     var httpClient = new HttpClient();
12.     httpClient.BaseAddress = new Uri(ConfigurationManager.AppSettings["ApiUrl"]);
13.     httpClient.DefaultRequestHeaders.Clear();
14.     httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
15.     HttpResponseMessage response;
16.
17.     if (object.Equals(pageNumber, null))
18.     {
19.         pageNumber = 1;
20.     }
21.     if (object.Equals(pageSize, null))
22.     {
23.         pageSize = Convert.ToInt32(ConfigurationManager.AppSettings["DefaultPageSize"]);
24.     }
25. }
26. ViewBag.PageNumber = pageNumber;
27. ViewBag.PageSize = pageSize;
28.
29. response = httpClient.GetAsync(string.Format("api/Cricketers/GetCricketerList?PageNumber={0}&PageSize={1}", pageNumber, pageSize)).Result;
30.
31. response.EnsureSuccessStatusCode();
32. var responseAsString = await response.Content.ReadAsStringAsync();
33. cricketersList = JsonConvert.DeserializeObject<List<BestCricketers.Models.CricketerProfile>>(responseAsString);
34. return View("~/Views/Cricketers/BestCricketers.cshtml", new CricketerInfo() { cricketerProfile = cricketersList });
35. }
```

Step 11: Go to Solution Explorer, right click on the web project, then click **References**

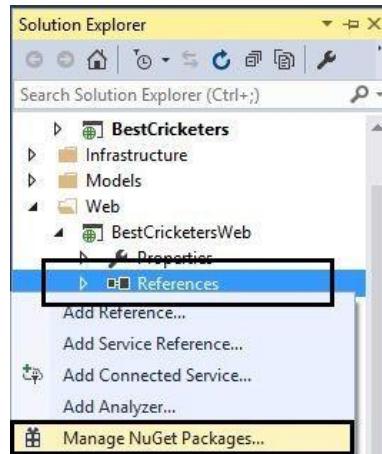


Figure 9: Adding NuGet Package

Step 12: Search out **PagedList.Mvc** and install this package:

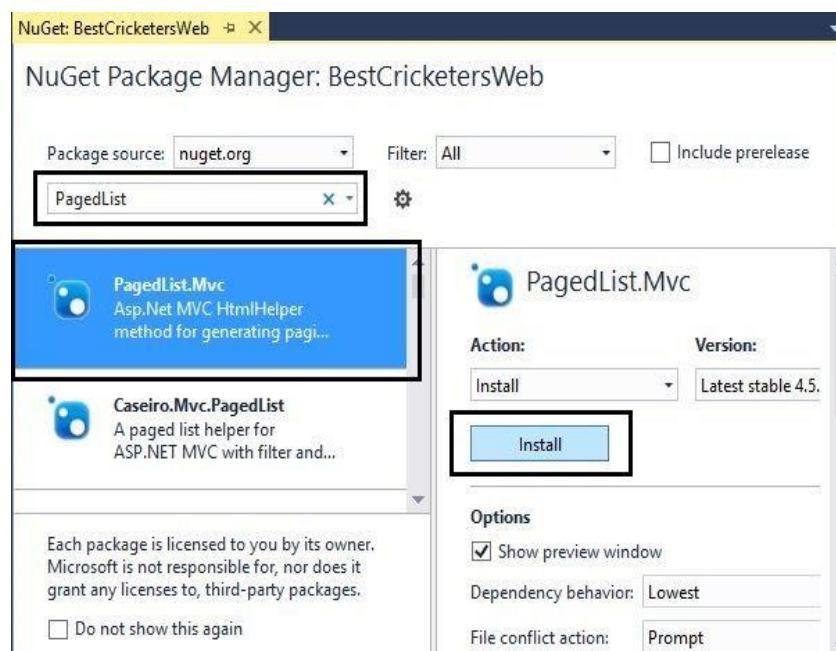


Figure 10: Adding PagedList in MVC 5

Step 13: Now go to **View**, then **Cricketers** and add a new view in this folder:

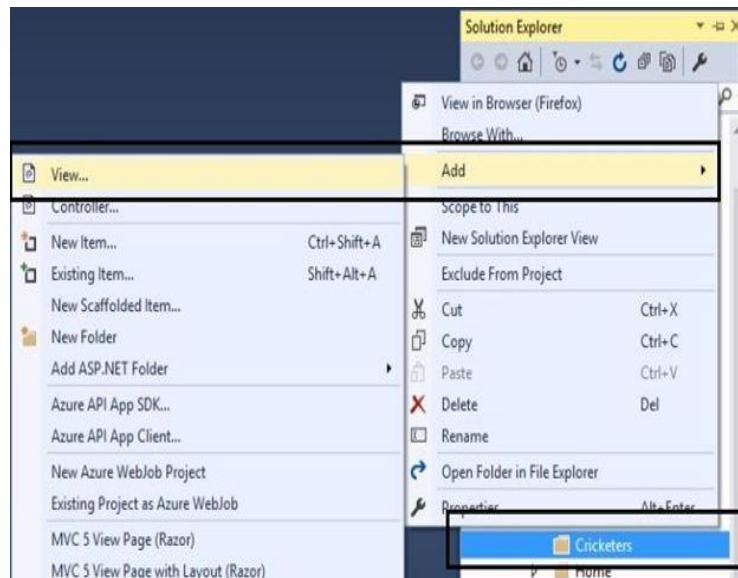


Figure 11: Adding View in MVC 5

Step 14: Add the jQuery Unobtrusive ajax from the NuGet Package as shown below:

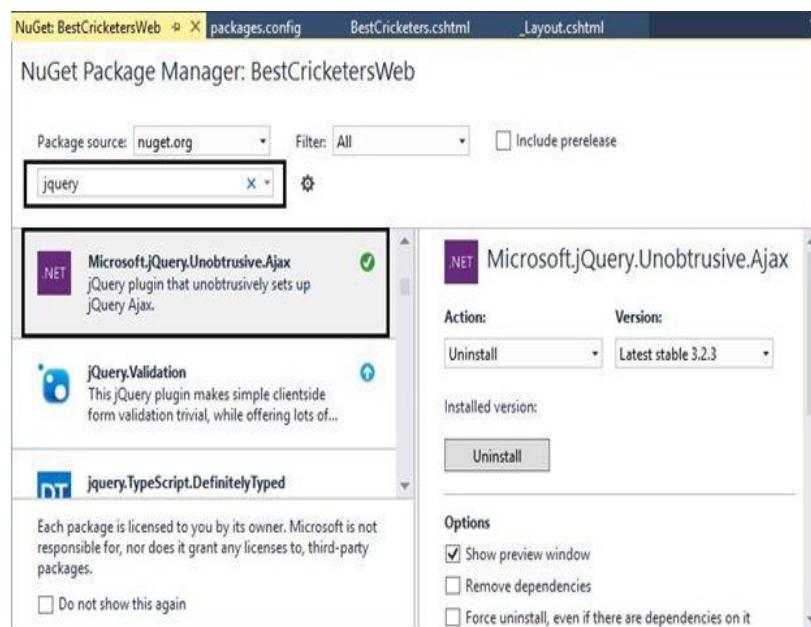


Figure 12: JQuery Unobtrusive Ajax

Step 15: Add the following code in the View:

```

1. @model BestCricketersWeb.Models.CricketerInfo
2. @using PagedList;
3. @{
4.     ViewBag.Title = "BestCricketers";
5. }
6. <script src="~/Scripts/jquery-1.10.2.js"></script>
7. <script src="~/Scripts/jquery.unobtrusive-ajax.js"></script>
8. <h2>Best Cricketers</h2>
9.
10. @using (Ajax.BeginForm("GetBestCricketersInfo", "Cricketers", null, new AjaxOptions() { LoadingElementId = "", HttpMethod = "Get", UpdateTargetId = "CricketersGrid" }))
11. {
12.     <div class="CricketersList">
13.         <div id="CricketersGrid">
14.             @Html.Partial("~/Views/Cricketers/_BestCricketersPartial.cshtml", new StaticPagedList<BestCricketers.Models.CricketerProfile>(Model.cricketerProfile, Convert.ToInt32(ViewBag.PageNumber), Convert.ToInt32(ViewBag.PageSize), Model.cricketerProfile.Count > 0 ? Model.cricketerProfile.FirstOrDefault().TotalCount : 0))
15.         </div>
16.     </div>
17. }

```

Step 16: Add the following code in the CricketersController.

```

1. /// <summary>
2. /// This method is used to get cricketers list.
3. /// </summary>
4. /// <param name="page"></param>
5. /// <param name="pageSize"></param>
6. /// <returns></returns>
7. [HttpGet, ActionName("GetBestCricketersInfo")]
8. public async Task<ActionResult> GetBestCricketersInfo (int? page, int? pageSize)
9. {
10.     List<CricketerProfile> cricketersList = new List<CricketerProfile>();
11.     var httpClient = new HttpClient();
12.     httpClient.BaseAddress = new Uri(ConfigurationManager.AppSettings["ApiUrl"]);
13.     httpClient.DefaultRequestHeaders.Clear();
14.     httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
15.     HttpResponseMessage response;
16.
17.     if (object.Equals(page, null))
18.     {
19.         page = 1;
20.     }
21.     if (object.Equals(pageSize, null))
22.     {
23.         pageSize = Convert.ToInt32(ConfigurationManager.AppSettings["DefaultPageSize"]);
24.     }
25. }
26. ViewBag.PageNumber = page;

```

```

27.     ViewBag.PageSize = pageSize;
28.
29.     response = httpClient.GetAsync(string.Format("api/Cricketers/GetCricketerList?PageNumber={0}&PageSize={1}", pa
ge, pageSize)).Result;
30.
31.     response.EnsureSuccessStatusCode();
32.     var responseAsString = await response.Content.ReadAsStringAsync();
33.     cricketersList = JsonConvert.DeserializeObject<List<CricketerProfile>>(responseAsString);
34.     return PartialView("~/Views/Cricketers/_BestCricketersPartial.cshtml", new StaticPagedList<CricketerProfile>(crick
etersList, Convert.ToInt32(page), Convert.ToInt32(pageSize), cricketersList.Count > 0 ? cricketersList.FirstOrDefault().T
otalCount : 0));
35. }

```

Step 17: Add the Partial View as in the following screenshot:

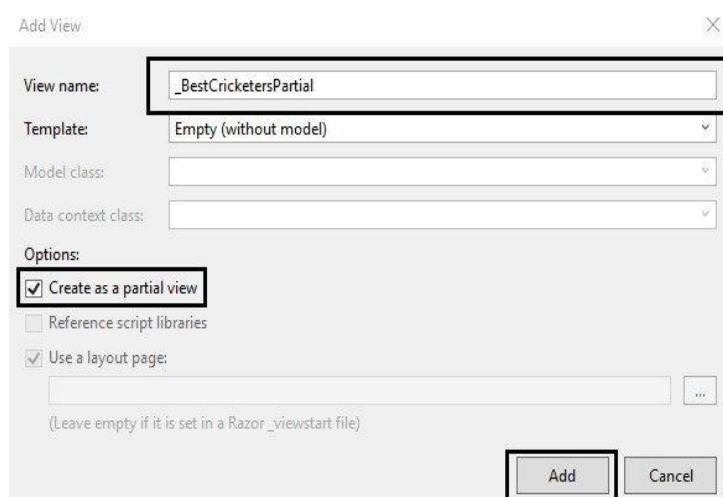


Figure 13: Adding Partial View

Step 18: Add the following code in the partial view page:

```

1.  @model PagedList.IPagedList<BestCricketers.Models.CricketerProfile>
2.  @using PagedList.Mvc
3.
4.  <table class="table-responsive table">
5.      <thead>
6.          <tr>
7.              <th>Name</th>
8.              <th>ODI</th>
9.              <th>Tests</th>
10.             <th>ODI Runs</th>
11.             <th>Test Runs</th>
12.         </tr>
13.     </thead>

```

```

14.    <tbody>
15.        @if (Model.Count > 0)
16.    {
17.        foreach (var cricketer in Model)
18.        {
19.            <tr>
20.                <td>@cricketer.Name</td>
21.                <td>@cricketer.ODI</td>
22.                <td>@cricketer.Tests</td>
23.                <td>@cricketer.OdiRuns</td>
24.                <td>@cricketer.TestRuns</td>
25.            </tr>
26.        }
27.    }
28.    else
29.    {
30.        <tr>
31.            <td>
32.                No Data Found
33.            </td>
34.        </tr>
35.    }
36.    </tbody>
37. </table>
38. @if (Model.TotalItemCount > Convert.ToInt32(System.Configuration.ConfigurationManager.AppSettings["DefaultPageSize"]))
39. {
40.     <div class="pagingBox">
41.         <input id="HiddenPageSize" name="PageSize" type="hidden" />
42.         <input id="HiddenPage" name="Page" type="hidden" />
43.         <span class="selectBoxes display_none_mobile">
44.             @Html.DropDownList("PageSize", new SelectList(new Dictionary<string, int>{ { "10", 10 }, { "20", 20 } }, "Key", "Value", Convert.ToString(ViewBag.PageSize)), new { id = "pagesizelist" })
45.         </span>
46.         <div class="pagerecord display_none_mobile">
47.             Records
48.
49.             Page @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) of @Model.PageCount
50.         </div>
51.
52.         @Html.PagedListPager(Model, page => Url.Action("GetBestCricketersInfo", "Cricketers",
53.             new
54.             {
55.                 page,
56.                 currentFilter = ViewBag.CurrentFilter,
57.                 pageSize = ViewBag.PageSize
58.             }),
59.             PagedListRenderOptions.EnableUnobtrusiveAjaxReplacing(new PagedListRenderOptions
60.                 {
61.                     Display = PagedListDisplayMode.IfNeeded,
62.                     MaximumPageNumbersToDisplay = 5
63.                 },
64.                 new AjaxOptions
65.                 {

```

```

66.     InsertionMode = InsertionMode.Replace,
67.     HttpMethod = "Get",
68.     UpdateTargetId = "CricketersGrid",
69.     LoadingElementId = "divProcessing"
70.   })
71. </div>
72.
73. <div id="divProcessing" class="processingButton" style="display: none;">
74.   
75. </div>
76. }

```

Step 19: Add the following link to the Views/Shared/_Layout Page.

```
1. <li>@Html.ActionLink("Cricketers", "GetBestCricketers", "Cricketers")</li>
```

Step 20: Build the solution and start both your projects as in the following screenshot:

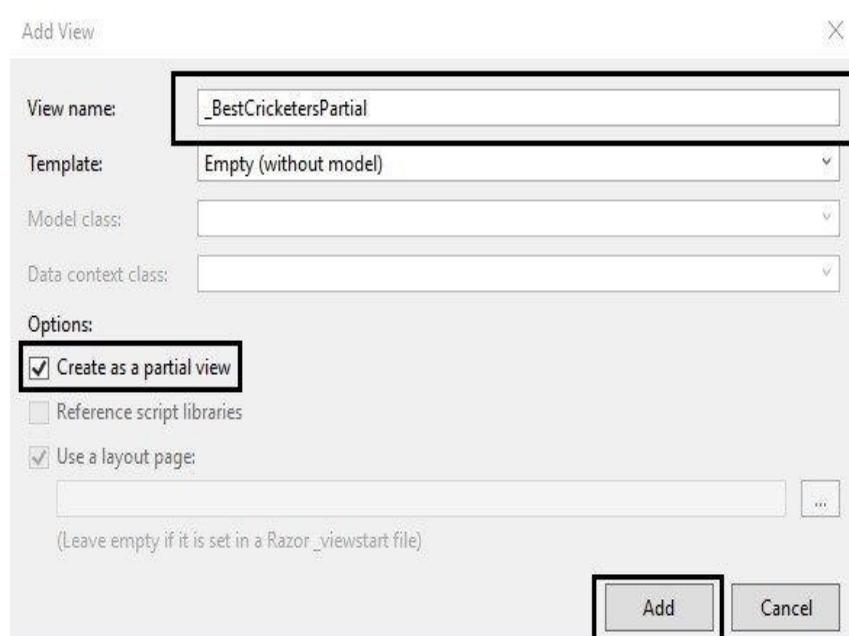
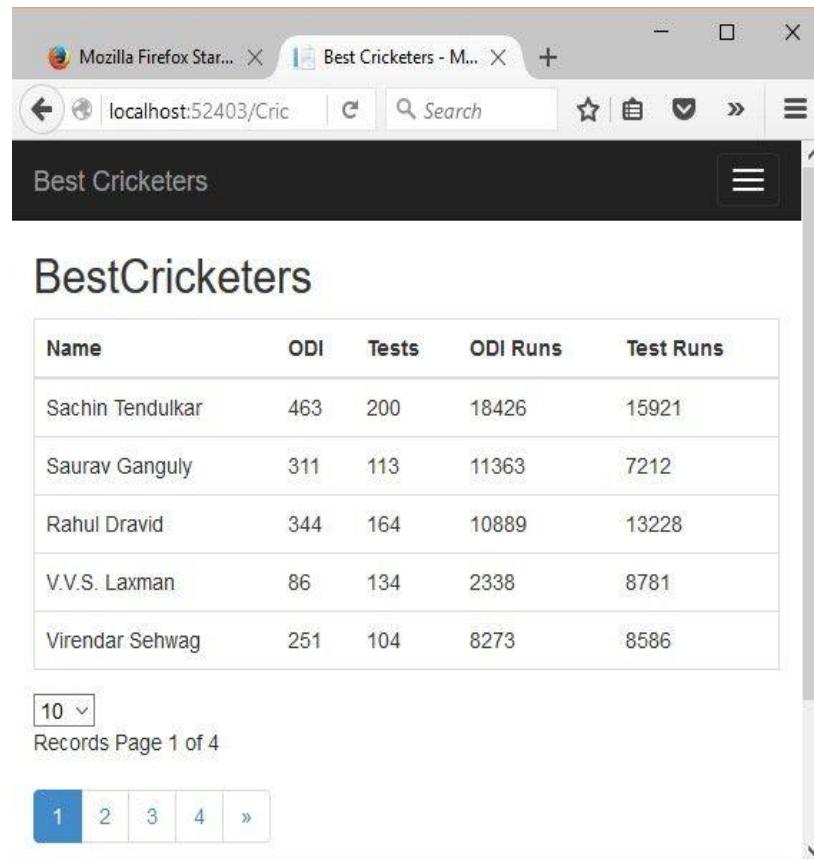


Figure 14: Setting Multiple Startup Projects

Step 21: Now run the application and click on the Cricketers:



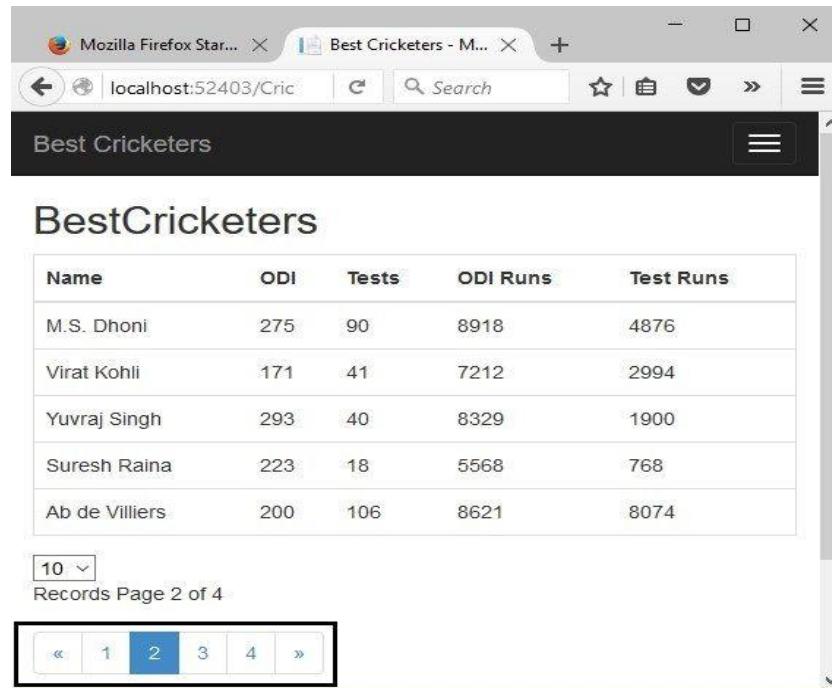
Name	ODI	Tests	ODI Runs	Test Runs
Sachin Tendulkar	463	200	18426	15921
Saurav Ganguly	311	113	11363	7212
Rahul Dravid	344	164	10889	13228
V.V.S. Laxman	86	134	2338	8781
Virender Sehwag	251	104	8273	8586

10 ▾
Records Page 1 of 4
1 2 3 4 >>

Figure 15: Cricketers View

As you can see in the above screenshot that the records are displayed with the paging. Paging Numbers are based upon the total count of records and your page size defined in the "Web.Config."

Step 22: Now click on Page No. 2 from below and you can see that the second page records will display.



Name	ODI	Tests	ODI Runs	Test Runs
M.S. Dhoni	275	90	8918	4876
Virat Kohli	171	41	7212	2994
Yuvraj Singh	293	40	8329	1900
Suresh Raina	223	18	5568	768
Ab de Villiers	200	106	8621	8074

10 ▾
Records Page 2 of 4

« 1 2 3 4 »

Figure 16: Cricketers View with Paging

Note: Here every time the request is going to the database and records are fetched instead of getting whole data at first time and then perform paging on that grid. Getting whole data at one time may slow your data performance.”

That's it for the paging. Now we will proceed in the next section.

27.4 Sorting

In this section we will sort the data over ODI, Tests, ODI Runs, Tests Runs of player. We have to change the database and methods to do that. So, follow the steps below to perform that.

Step 1: Change the stored procedure from the following script:

```
1. ALTER PROC[dbo].[CC_GetCricketerList]
```

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

```

2.    @PageNumber INT,
3.    @PageSize INT,
4.    @SortExp VARCHAR(20)
5.    AS
6.    BEGIN;
7.    WITH CteCricketer
8.    AS(SELECT ROW_NUMBER() OVER(ORDER BY CASE WHEN @SortExp = 'ODI_Asc'
9.        THEN ODI END ASC, CASE WHEN @SortExp = 'ODI_Desc'
10.       THEN ODI END DESC, CASE WHEN @SortExp = 'Tests_Asc'
11.       THEN Tests END ASC, CASE WHEN @SortExp = 'Tests_Desc'
12.       THEN Tests END DESC, CASE WHEN @SortExp = 'OdiRuns_Asc'
13.       THEN OdiRuns END ASC, CASE WHEN @SortExp = 'OdiRuns_Desc'
14.       THEN OdiRuns END DESC, CASE WHEN @SortExp = 'TestRuns_Asc'
15.       THEN TestRuns END ASC, CASE WHEN @SortExp = 'TestRuns_Desc'
16.       THEN TestRuns END DESC, CASE WHEN @SortExp = ""
17.       THEN ID END ASC)[RowNo], * FROM dbo.CricketerProfile(NOLOCK))
18.    SELECT CteCricketer.RowNo,
19.           CteCricketer.ID,
20.           CteCricketer.Name,
21.           CteCricketer.ODI,
22.           CteCricketer.Tests,
23.           CteCricketer.OdiRuns,
24.           CteCricketer.TestRuns, (SELECT COUNT(CteCricketer.ID) FROM CteCricketer) AS TotalCount
25.    FROM CteCricketer
26.    ORDER BY CteCricketer.RowNo
27.    OFFSET(@PageNumber - 1) * @PageSize ROWS
28.    FETCH NEXT @PageSize ROWS ONLY;
29.    END;

```

Step 2: Add the parameters in the *CricketerDAL* and *CricketerBL* as we have added in the previous section here and change the method of API from the highlighted code below:

```

1.  [HttpGet, ActionName("GetCricketerList")]
2.  public HttpResponseMessage GetCricketerList(int PageNumber, int PageSize, string SortExpression)
3.  {
4.      Result result;
5.      cricketerBL = new CricketerBL();
6.      try
7.      {
8.          var cricketerList = cricketerBL.GetCricketerList(PageNumber, PageSize, SortExpression);
9.
10.         if (!object.Equals(cricketerList, null))
11.         {
12.             response = Request.CreateResponse<List<CricketerProfile>>(HttpStatusCode.OK, cricketerList);
13.         }
14.     }
15.     catch (Exception ex)
16.     {
17.         result = new Result();
18.         result.Status = 0;
19.         result.Message = ex.Message;
20.         response = Request.CreateResponse(HttpStatusCode.InternalServerError, result);

```

```

21.    }
22.    return response;
23. }
```

Step 3: Now change the *CricketersController* in web project from the highlighted code below:

```

1.  [HttpGet, ActionName("GetBestCricketers")]
2.  public async Task<ActionResult> GetBestCricketers(int? pageNumber, int? pageSize, string sortExpression)
3.  {
4.      List<CricketerProfile> cricketersList = new List<CricketerProfile>();
5.      var httpClient = new HttpClient();
6.      httpClient.BaseAddress = new Uri(ConfigurationManager.AppSettings["ApiUrl"]);
7.      httpClient.DefaultRequestHeaders.Clear();
8.      httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
9.      HttpResponseMessage response;
10.
11.     if (object.Equals(pageNumber, null))
12.     {
13.         pageNumber = 1;
14.     }
15.     if (object.Equals(pageSize, null))
16.     {
17.         pageSize = Convert.ToInt32(ConfigurationManager.AppSettings["DefaultPageSize"]);
18.     }
19. }
20. if (string.IsNullOrEmpty(sortExpression))
21. {
22.     sortExpression = string.Empty;
23. }
24. ViewBag.PageNumber = pageNumber;
25. ViewBag.PageSize = pageSize;
26. ViewBag.CurrentSort = sortExpression;
27.
28. response = httpClient.GetAsync(string.Format("api/Cricketers/GetCricketerList?PageNumber={0}&PageSize={1}&SortExpression={2}", pageNumber, pageSize, sortExpression)).Result;
29.
30. response.EnsureSuccessStatusCode();
31. var responseAsString = await response.Content.ReadAsStringAsync();
32. cricketersList = JsonConvert.DeserializeObject<List<BestCricketers.Models.CricketerProfile>>(responseAsString);
33. return View("~/Views/Cricketers/BestCricketers.cshtml", new CricketerInfo() { cricketerProfile = cricketersList });
34. }
35.
36. /// <summary>
37. /// This method is used to get cricketers list.
38. /// </summary>
39. /// <param name="page"></param>
40. /// <param name="pageSize"></param>
41. /// <param name="SortExpression"></param>
42. /// <returns></returns>
43. [HttpGet, ActionName("GetBestCricketersInfo")]
44. public async Task<ActionResult> GetBestCricketersInfo(int? page, int? pageSize, string sortExpression)
45. {
46.     List<CricketerProfile> cricketersList = new List<CricketerProfile>();
```

```

47. var httpClient = new HttpClient();
48. httpClient.BaseAddress = new Uri(ConfigurationManager.AppSettings["ApiUrl"]);
49. httpClient.DefaultRequestHeaders.Clear();
50. httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
51. HttpResponseMessage response;
52.
53. if (object.Equals(page, null))
54. {
55.     page = 1;
56. }
57. if (object.Equals(pageSize, null))
58. {
59.     pageSize = Convert.ToInt32(ConfigurationManager.AppSettings["DefaultPageSize"]);
60.
61. }
62. if (string.IsNullOrEmpty(SortExpression))
63. {
64.     SortExpression = string.Empty;
65. }
66. ViewBag.PageNumber = page;
67. ViewBag.PageSize = pageSize;
68. ViewBag.CurrentSort = SortExpression;
69.
70. response = httpClient.GetAsync(string.Format("api/Cricketers/GetCricketList?PageNumber={0}&PageSize={1}&So
rtExpression={2}", page, pageSize, SortExpression)).Result;
71.
72. response.EnsureSuccessStatusCode();
73. var responseAsString = await response.Content.ReadAsStringAsync();
74. cricketersList = JsonConvert.DeserializeObject<List<CricketerProfile>>(responseAsString);
75. return PartialView("~/Views/Cricketers/_BestCricketersPartial.cshtml", new StaticPagedList<CricketerProfile>(crick
etersList, Convert.ToInt32(page), Convert.ToInt32(pageSize), cricketersList.Count > 0 ? cricketersList.FirstOrDefault().T
otalCount : 0));
76. }

```

Step 4: Now change the partial view code from the highlighted code below:

```

1. @model PagedList.IPagedList<BestCricketers.Models.CricketerProfile>
2. @using PagedList.Mvc
3.
4. <table class="table-responsive table">
5.     <thead>
6.         <tr>
7.             <th>Name</th>
8.             <th>
9.                 @Ajax.ActionLink("ODI", "GetBestCricketersInfo", new
10.                {
11.                    SortExpression = ViewBag.CurrentSort == "ODI_Asc" ? "ODI_Desc" : "ODI_Asc",
12.                    currentFilter = ViewBag.CurrentFilter,
13.                    pageSize = ViewBag.PageSize
14.
15.                }, new AjaxOptions
16.                {
17.                    LoadingElementId = "divProcessing",

```

```

18.         InsertionMode = InsertionMode.Replace,
19.         HttpMethod = "Get",
20.         UpdateTargetId = "CricketersGrid"
21.     })
22.   </th>
23.   <th>
24.     @Ajax.ActionLink("Tests", "GetBestCricketersInfo", new
25.     {
26.       SortExpression = ViewBag.CurrentSort == "Tests_Asc" ? "Tests_Desc" : "Tests_Asc",
27.       currentFilter = ViewBag.CurrentFilter,
28.       pageSize = ViewBag.PageSize
29.
30.     }, new AjaxOptions
31.     {
32.       LoadingElementId = "divProcessing",
33.       InsertionMode = InsertionMode.Replace,
34.       HttpMethod = "Get",
35.       UpdateTargetId = "CricketersGrid"
36.     })
37.   </th>
38.   <th>
39.     @Ajax.ActionLink("ODI Runs", "GetBestCricketersInfo", new
40.     {
41.       SortExpression = ViewBag.CurrentSort == "OdiRuns_Asc" ? "OdiRuns_Desc" : "OdiRuns_Asc",
42.       currentFilter = ViewBag.CurrentFilter,
43.       pageSize = ViewBag.PageSize
44.
45.     }, new AjaxOptions
46.     {
47.       LoadingElementId = "divProcessing",
48.       InsertionMode = InsertionMode.Replace,
49.       HttpMethod = "Get",
50.       UpdateTargetId = "CricketersGrid"
51.     })
52.   </th>
53.   <th>
54.     @Ajax.ActionLink("Test Runs", "GetBestCricketersInfo", new
55.     {
56.       SortExpression = ViewBag.CurrentSort == "TestRuns_Asc" ? "TestRuns_Desc" : "TestRuns_Asc",
57.       currentFilter = ViewBag.CurrentFilter,
58.       pageSize = ViewBag.PageSize
59.
60.     }, new AjaxOptions
61.     {
62.       LoadingElementId = "divProcessing",
63.       InsertionMode = InsertionMode.Replace,
64.       HttpMethod = "Get",
65.       UpdateTargetId = "CricketersGrid"
66.     })
67.   </th>
68. </tr>
69. </thead>
70. <tbody>
71.   @if (Model.Count > 0)

```

```

72.    {
73.        foreach (var cricketer in Model)
74.        {
75.            <tr>
76.                <td>@cricketer.Name</td>
77.                <td>@cricketer.ODI</td>
78.                <td>@cricketer.Tests</td>
79.                <td>@cricketer.OdiRuns</td>
80.                <td>@cricketer.TestRuns</td>
81.            </tr>
82.        }
83.    }
84.    else
85.    {
86.        <tr>
87.            <td>
88.                No Data Found
89.            </td>
90.        </tr>
91.    }
92.    </tbody>
93.    </table>
94.    @if (Model.TotalItemCount > Convert.ToInt32(System.Configuration.ConfigurationManager.AppSettings["DefaultPageSize"]))
95.    {
96.        <div class="pagingBox">
97.            <input id="HiddenPageSize" name="PageSize" type="hidden" />
98.            <input id="HiddenPage" name="Page" type="hidden" />
99.            <span class="selectBoxes display_none_mobile">
100.                @Html.DropDownList("PageSize", new SelectList(new Dictionary<string, int> { { "10", 10 }, { "20", 20 } }, "Key", "Value", Convert.ToString(ViewBag.PageSize)), new { id = "pagesizelist" })
101.            </span>
102.            <div class="pagerecord display_none_mobile">
103.                Records
104.
105.                Page @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) of @Model.PageCount
106.            </div>
107.
108.            @Html.PagedListPager(Model, page => Url.Action("GetBestCricketersInfo", "Cricketers",
109.                new
110.                {
111.                    page,
112.                    SortExpression = ViewBag.CurrentSort,
113.                    currentFilter = ViewBag.CurrentFilter,
114.                    pageSize = ViewBag.PageSize
115.                }),
116.                PagedListRenderOptions.EnableUnobtrusiveAjaxReplacing(new PagedListRenderOptions
117.                {
118.                    Display = PagedListDisplayMode.IfNeeded,
119.                    MaximumPageNumbersToDisplay = 5
120.                },
121.                new AjaxOptions
122.                {
123.                    InsertionMode = InsertionMode.Replace,

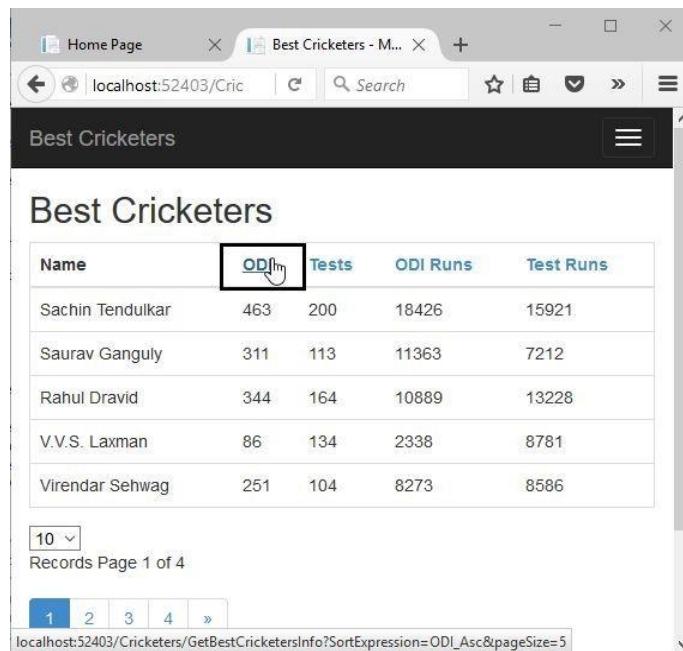
```

```

124.         HttpMethod = "Get",
125.         UpdateTargetId = "CricketersGrid",
126.         LoadingElementId = "divProcessing"
127.     }})
128. </div>
129.
130. <div id="divProcessing" class="processingButton" style="display: none;">
131.     
132. </div>
133. }

```

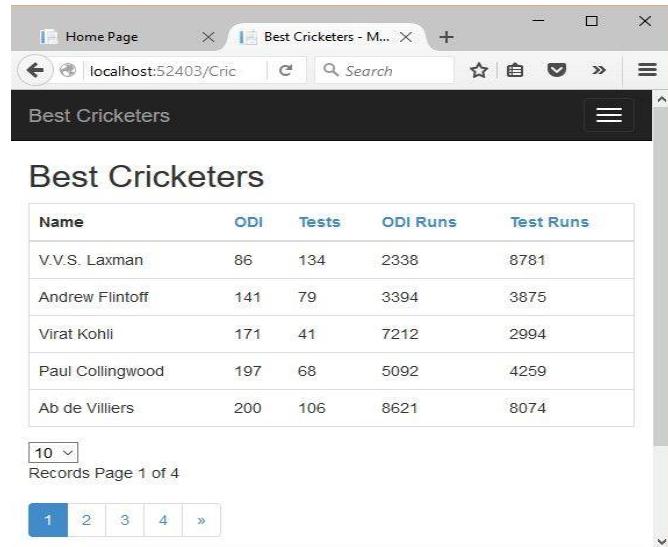
Step 5: Now run the application and after opening the Cricketers page, click on the ODI link as shown below:



Name	ODI	Tests	ODI Runs	Test Runs
Sachin Tendulkar	463	200	18426	15921
Saurav Ganguly	311	113	11363	7212
Rahul Dravid	344	164	10889	13228
V.V.S. Laxman	86	134	2338	8781
Virender Sehwag	251	104	8273	8586

Figure 17: Perform Sorting on MVC

When you click on the ODI link, the records will be sorted as ascending order of ODI matches



Name	ODI	Tests	ODI Runs	Test Runs
V.V.S. Laxman	86	134	2338	8781
Andrew Flintoff	141	79	3394	3875
Virat Kohli	171	41	7212	2994
Paul Collingwood	197	68	5092	4259
Ab de Villiers	200	106	8621	8074

10
Records Page 1 of 4

1 2 3 4 >>

Figure 18: Sorting in MVC

Note: I have created sorting on four fields. You can change out the condition based upon your situation. In this type of sorting, whole data is not sorted. The data is sorted on that page only but when you open the next page, the database will be called again and return the data as in the sorted order because we are sending the SortExpression also.

27.5 Filtering

In this section we will filter the data over Cricketer Name. We have to change the database and methods to do that. So, follow the steps below to perform that.

Step 1: Change the stored procedure from the following script:

```

1. ALTER PROC[dbo].[CC_GetCricketerList]
2. @PageNumber INT,
3. @PageSize INT,
4. @SortExp VARCHAR(20),
5. @SearchText NVARCHAR(150)
6. AS
7. BEGIN;
8. WITH CteCricketer
9. AS(SELECT ROW_NUMBER() OVER(ORDER BY CASE WHEN @SortExp = 'ODI_Asc'
10. THEN ODI END ASC, CASE WHEN @SortExp = 'ODI_Desc'
11. THEN ODI END DESC, CASE WHEN @SortExp = 'Tests_Asc'
12. THEN Tests END ASC, CASE WHEN @SortExp = 'Tests_Desc'
13. THEN Tests END DESC, CASE WHEN @SortExp = 'OdiRuns_Asc'
```

```

14. THEN ODIRuns END ASC, CASE WHEN @SortExp = 'OdiRuns_Desc'
15. THEN ODIRuns END DESC, CASE WHEN @SortExp = 'TestRuns_Asc'
16. THEN TestRuns END ASC, CASE WHEN @SortExp = 'TestRuns_Desc'
17. THEN TestRuns END DESC, CASE WHEN @SortExp = ''
18. THEN ID END ASC)[RowNo], * FROM dbo.CricketerProfile(NOLOCK) WHERE Name LIKE '%' + @SearchText + '%')
19. SELECT CteCricketer.RowNo,
20. CteCricketer.ID,
21. CteCricketer.Name,
22. CteCricketer.ODI,
23. CteCricketer.Tests,
24. CteCricketer.ODIRuns,
25. CteCricketer.TestRuns, (SELECT COUNT(CteCricketer.ID) FROM CteCricketer) AS TotalCount
26. FROM CteCricketer
27. ORDER BY CteCricketer.RowNo
28. OFFSET(@PageNumber - 1) * @PageSize ROWS
29. FETCH NEXT @PageSize ROWS ONLY;

```

Step 2: Add the parameters in the CricketerDAL and CricketerBL and change the method of API from the highlighted code below:

```

1. [HttpGet, ActionName("GetCricketerList")]
2. public HttpResponseMessage GetCricketerList(int PageNumber, int PageSize, string SortExpression, string SearchText)
3. {
4.     Result result;
5.     cricketerBL = new CricketerBL();
6.     try
7.     {
8.         var cricketerList = cricketerBL.GetCricketerList(PageNumber, PageSize, SortExpression, SearchText == "undefined"
9.             ? string.Empty : SearchText);
10.        if (!object.Equals(cricketerList, null))
11.        {
12.            response = Request.CreateResponse<List<CricketerProfile>>(HttpStatusCode.OK, cricketerList);
13.        }
14.    }
15.    catch (Exception ex)
16.    {
17.        result = new Result();
18.        result.Status = 0;
19.        result.Message = ex.Message;
20.        response = Request.CreateResponse(HttpStatusCode.InternalServerError, result);
21.    }
22.    return response;
23. }

```

Step 3: Now change the **CricketersController** in web project from the highlighted code below:

```

1. public async Task<ActionResult> GetBestCricketers(int? PageNumber, int? PageSize, string SortExpression, string SearchText)
2. {
3.     List<CricketerProfile> cricketersList = new List<CricketerProfile>();
4.     var httpClient = new HttpClient();
5.     httpClient.BaseAddress = new Uri(ConfigurationManager.AppSettings["ApiUrl"]);
6.     httpClient.DefaultRequestHeaders.Clear();
7.     httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
8.     HttpResponseMessage response;
9.
10.    if (object.Equals(PageNumber, null))
11.    {
12.        PageNumber = 1;
13.    }
14.    if (object.Equals(PageSize, null))
15.    {
16.        PageSize = Convert.ToInt32(ConfigurationManager.AppSettings["DefaultPageSize"]);
17.    }
18. }
19. if (string.IsNullOrEmpty(SortExpression))
20. {
21.     SortExpression = string.Empty;
22. }
23. if (string.IsNullOrEmpty(SearchText))
24. {
25.     SearchText = "undefined";
26. }
27. ViewBag.PageNumber = PageNumber;
28. ViewBag.PageSize = PageSize;
29. ViewBag.CurrentSort = SortExpression;
30.
31. response = httpClient.GetAsync(string.Format("api/Cricketers/GetCricketerList?PageNumber={paging-sorting-and-
filtering-with-partial-view-in-Asp-Net-
m}&PageSize={1}&SortExpression={2}&SearchText={3}", PageNumber, PageSize, SortExpression, SearchText)).Result;
32.
33. response.EnsureSuccessStatusCode();
34. var responseAsString = await response.Content.ReadAsStringAsync();
35. cricketersList = JsonConvert.DeserializeObject<List<BestCricketers.Models.CricketerProfile>>(responseAsString);
36. return View("~/Views/Cricketers/BestCricketers.cshtml", new CricketerInfo() { cricketerProfile = cricketersList });
37. }
38.
39. /// <summary>
40. /// This method is used to get cricketers list.
41. /// </summary>
42. /// <param name="page"></param>
43. /// <param name="pageSize"></param>
44. /// <param name="SortExpression"></param>
45. /// <returns></returns>
46. [HttpGet, ActionName("GetBestCricketersInfo")]
47. public async Task<ActionResult> GetBestCricketersInfo(int? page, int? pageSize, string SortExpression, string SearchText)

```

```

48. {
49.     List<CricketerProfile> cricketersList = new List<CricketerProfile>();
50.     var httpClient = new HttpClient();
51.     httpClient.BaseAddress = new Uri(ConfigurationManager.AppSettings["ApiUrl"]);
52.     httpClient.DefaultRequestHeaders.Clear();
53.     httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
54.     HttpResponseMessage response;
55.
56.     if (object.Equals(page, null))
57.     {
58.         page = 1;
59.     }
60.     if (object.Equals(pageSize, null))
61.     {
62.         pageSize = Convert.ToInt32(ConfigurationManager.AppSettings["DefaultPageSize"]);
63.     }
64. }
65. if (string.IsNullOrEmpty(SortExpression))
66. {
67.     SortExpression = string.Empty;
68. }
69. if (string.IsNullOrEmpty(SearchText))
70. {
71.     SearchText = "undefined";
72. }
73. ViewBag.PageNumber = page;
74. ViewBag.PageSize = pageSize;
75. ViewBag.CurrentSort = SortExpression;
76.
77. response = httpClient.GetAsync(string.Format("api/Cricketers/GetCricketerList?PageNumber={paging-sorting-and-
filtering-with-partial-view-in-Asp-Net-
m}&PageSize={1}&SortExpression={2}&SearchText={3}", page, pageSize, SortExpression, SearchText)).Result;
78.
79. response.EnsureSuccessStatusCode();
80. var responseAsString = await response.Content.ReadAsStringAsync();
81. cricketersList = JsonConvert.DeserializeObject<List<CricketerProfile>>(responseAsString);
82. return PartialView("~/Views/Cricketers/_BestCricketersPartial.cshtml", new StaticPagedList<CricketerProfile>(crick-
etersList, Convert.ToInt32(page), Convert.ToInt32(pageSize), cricketersList.Count > paging-sorting-and-filtering-with-
partial-view-in-Asp-Net-m ? cricketersList.FirstOrDefault().TotalCount : paging-sorting-and-filtering-with-partial-view-
in-Asp-Net-m));
83. }

```

Step 4: Now change the Main view (BestCricketers.cshtml) code from the highlighted code below:

```

1. @using Ajax.BeginForm("GetBestCricketersInfo", "Cricketers", null, new AjaxOptions() { LoadingElementId = "divProc-
essing", HttpMethod = "Get", UpdateTargetId = "CricketersGrid" }, htmlAttributes: new { @id = "cricketerInfo" })
2. {
3.     <div id="SearchCricketer" style="float:right;">
4.         <input type="text" class="form-control" id="TxtSearchCricketer" placeholder="Search" />
5.     </div>
6.     <div class="clearfix"></div>
7.     <div id="CricketersGrid">

```

```

8.     <div id="divProcessing" class="processingButton" style="display: none;">
9.         
10.    </div>
11.    @Html.Partial("~/Views/Cricketers/_BestCricketersPartial.cshtml", new StaticPagedList<BestCricketers.Models.CricketerProfile>(Model.cricketerProfile, Convert.ToInt32(ViewBag.PageNumber), Convert.ToInt32(ViewBag.PageSize),
12.        Model.cricketerProfile.Count > paging-sorting-and-filtering-with-partial-view-in-Asp-Net-
13.        m ? Model.cricketerProfile.FirstOrDefault().TotalCount : paging-sorting-and-filtering-with-partial-view-in-Asp-Net-
14.        m))
15.    </div>
16. }
17. <script type="text/javascript">
18.     var timer;
19.     $("#TxtSearchCrickete").on('keyup', function (event) {
20.         clearTimeout(timer); //clear any running timeout on key up
21.         timer = setTimeout(function () { //then give it a second to see if the user is finished
22.             $.ajax({
23.                 url:'@Url.Action("GetBestCricketersInfo", "Cricketers")',
24.                 dataType:"html",
25.                 data:{page:@ViewBag.PageNumber,pageSize:@ViewBag.PageSize, SortExpression:'@ViewBag.CurrentSort',
26.                     SearchText:$("#TxtSearchCrickete").val()},
27.                 type:"Get",
28.                 success: function (data) {
29.                     $("#divProcessing").hide();
30.                     $("#CricketersGrid").html(data);
31.                 },
32.                 beforeSend: function () {
33.                     $("#divProcessing").show();
34.                 }
35.             });
36.         }, Spaging-sorting-and-filtering-with-partial-view-in-Asp-Net-m);
37.     });
38. </script>

```

Note: I have created an Ajax call to perform the filtering in MVC grid. You can apply any other procedure to call the Action in Controller.

Step 5: Now run the application and you can see the **Search** Textbox.



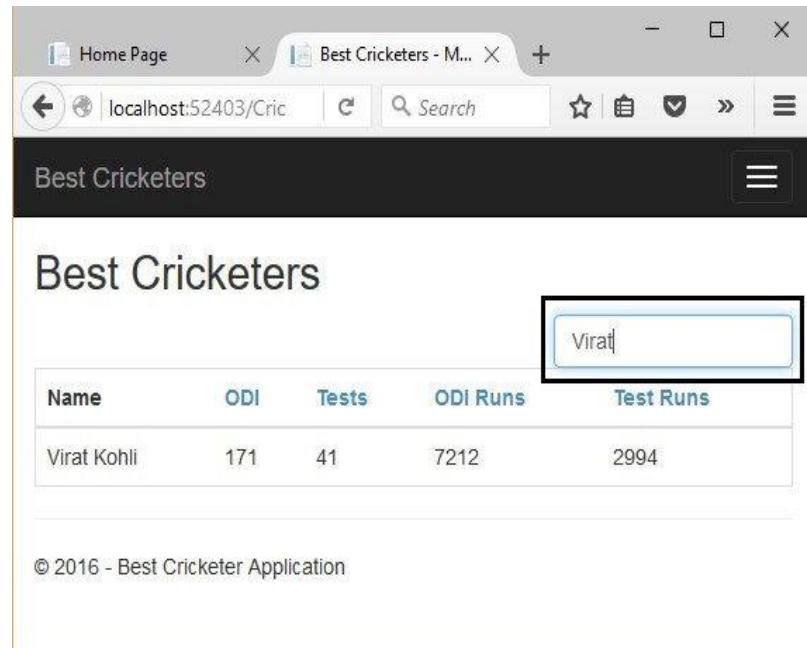
Name	ODI	Tests	ODI Runs	Test Runs
Sachin Tendulkar	463	200	18426	15921
Saurav Ganguly	311	113	11363	7212
Rahul Dravid	344	164	10889	13228
V.V.S. Laxman	86	134	2338	8781
Virender Sehwag	251	104	8273	8586

10
Records Page 1 of 4

1 2 3 4 >>

Figure 19: Filtering in MVC

Enter the name to filter the records.



Name	ODI	Tests	ODI Runs	Test Runs
Virat Kohli	171	41	7212	2994

© 2016 - Best Cricketer Application

Figure 20: Filtered Records in MVC

Chapter 28: Benefits of Partial View in MVC 5

28.1 Introduction Partial View

In this chapter you will learn to use Partial View in a different manner in any MVC application. As you know, Partial View is a view that will be rendered on a parent view. So, I will show you with the help of this chapter that how we can render partial view in a different way on a parent view.

Partial View can be a parent view and we can use it as a parent view of any partial view. We can simply pass Model to show data in the partial view or we can send the data in the partial view with the help of AJAX call.

We will use Partial View to show the data as in jQuery UI dialogue and we will put Partial View in a div to show the data. **In my Previous chapter on Paging, Searching, Filtering in MVC 5 with Partial View , we saw that how we can use Partial View to show the data in MVC Grid.**

So, let's get started and learn to use Partial View in MVC Application with the help of following structure:

- Creating ASP.NET MVC Application
- Performing Database Operation
- Working with Web Application

28.2 Creating ASP.NET MVC Application

In this section, we will create the MVC application. I am creating MVC 5 application and you can use it on MVC 3 or MVC 4. MVC 5 application can be created through Visual Studio 2013 or Visual Studio 2015. Let's begin with the following steps:

Step 1: In the Visual Studio 2013, click on “New Project”,

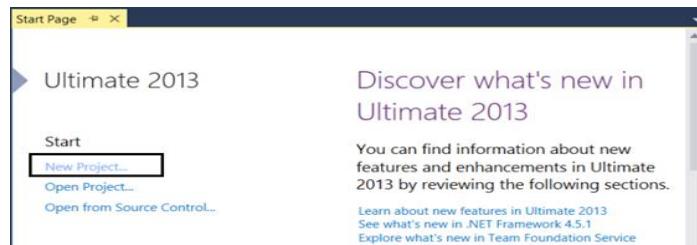


Figure 1: Creating New Project in VS 2013

Step 2: Select the Web from the left pane and click on “*ASP.NET Web Application*” and enter the app name as “BestMovies”,

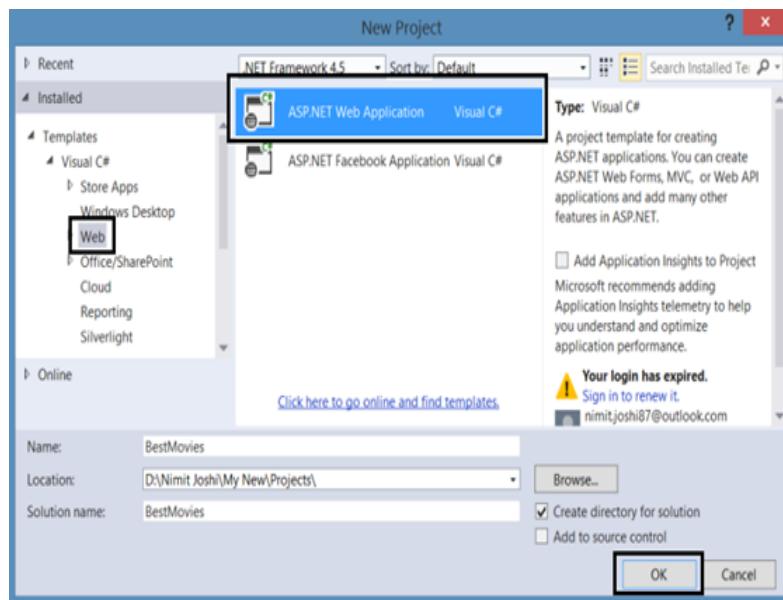


Figure 2: Creating Web App in VS 2013

Step 3: Select the MVC Project Template in the next “One ASP.Net” Wizard,

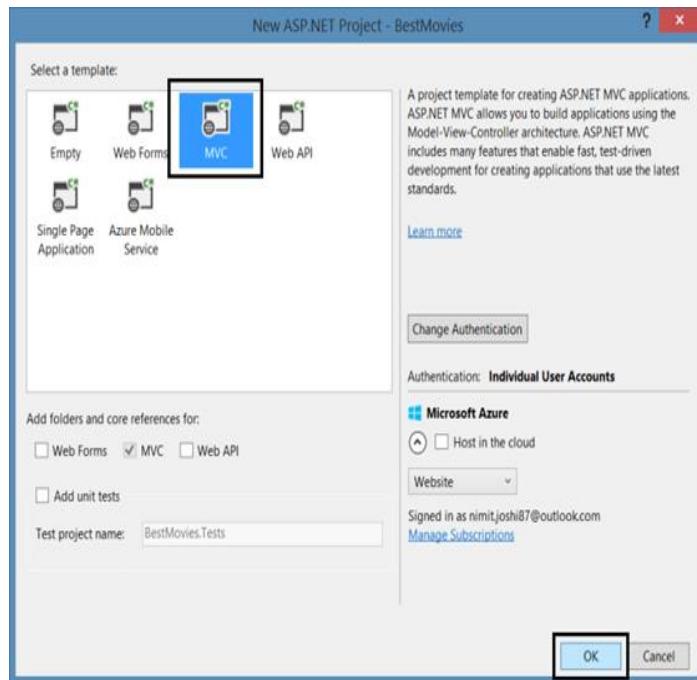


Figure 3: MVC Template in VS 2013

28.3 Performing Database Operation

In this section we will create the database and table for performing data operation so that web application can fetch the data from the database. Start with the following steps:

Step 1: Just Create the database from the following code:

```
1. CREATE DATABASE BestMovies
```

Step 2: Now run the following script to perform the database operations:

```
1. USE [BestMovies]
2. GO
3. /****** Object: Table [dbo].[Actor]  Script Date: 4/29/2016 2:47:20 PM *****/
4. SET ANSI_NULLS ON
5. GO
6. SET QUOTED_IDENTIFIER ON
7. GO
8. SET ANSI_PADDING ON
9. GO
10. CREATE TABLE [dbo].[Actor](
11.     [ActorInfoId] [int] IDENTITY(1,1) NOT NULL,
12.     [Name] [varchar](50) NULL,
```

```

13.    [Age] [int] NULL,
14.    [DOB] [datetime] NULL,
15.    CONSTRAINT [PK_Actor] PRIMARY KEY CLUSTERED
16.    (
17.        [ActorInfoId] ASC
18.    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
19. ) ON [PRIMARY]
20.
21. GO
22. SET ANSI_PADDING OFF
23. GO
24. /****** Object: Table [dbo].[Movie] Script Date: 4/29/2016 2:47:20 PM *****/
25. SET ANSI_NULLS ON
26. GO
27. SET QUOTED_IDENTIFIER ON
28. GO
29. SET ANSI_PADDING ON
30. GO
31. CREATE TABLE [dbo].[Movie](
32.     [ID] [int] IDENTITY(1,1) NOT NULL,
33.     [Name] [varchar](50) NULL,
34.     [Genre] [varchar](50) NULL,
35.     [ReleasedDate] [datetime] NULL,
36.     [Actor] [varchar](50) NULL,
37.     [Actress] [varchar](50) NULL,
38.     CONSTRAINT [PK_Movie] PRIMARY KEY CLUSTERED
39.    (
40.        [ID] ASC
41.    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
42. ) ON [PRIMARY]
43.
44. GO
45. SET ANSI_PADDING OFF
46. GO
47. /****** Object: Table [dbo].[MovieInfo] Script Date: 4/29/2016 2:47:20 PM *****/
48. SET ANSI_NULLS ON
49. GO
50. SET QUOTED_IDENTIFIER ON
51. GO
52. SET ANSI_PADDING ON
53. GO
54. CREATE TABLE [dbo].[MovieInfo](
55.     [MovieInfoId] [int] IDENTITY(1,1) NOT NULL,
56.     [MovieId] [int] NULL,
57.     [Director] [varchar](150) NULL,
58.     [Production] [nvarchar](150) NULL,
59.     [ImdbRating] [decimal](2, 1) NULL,
60.     [FilmfareAward] [int] NULL,
61.     [LeadRole] [varchar](50) NULL,
62.     CONSTRAINT [PK_MovieInfo] PRIMARY KEY CLUSTERED
63.    (
64.        [MovieInfoId] ASC

```

```

65. )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
66. ) ON [PRIMARY]
67.
68. GO
69. SET ANSI_PADDING OFF
70. GO
71. SET IDENTITY_INSERT [dbo].[Actor] ON
72.
73. GO
74. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (1, N'Amitabh Bachhan', 73, CAST(N'1942-11-10 00:00:00.000' AS DateTime))
75. GO
76. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (2, N'Rajesh Khanna', 73, CAST(N'1942-12-29 00:00:00.000' AS DateTime))
77. GO
78. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (3, N'Shahrukh Khan', 50, CAST(N'1965-02-12 00:00:00.000' AS DateTime))
79. GO
80. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (4, N'Anil Kapoor', 59, CAST(N'1956-12-24 00:00:00.000' AS DateTime))
81. GO
82. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (5, N'Aishwarya', 42, CAST(N'1973-12-01 00:00:00.000' AS DateTime))
83. GO
84. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (6, N'Akshay Kumar', 48, CAST(N'1967-09-09 00:00:00.000' AS DateTime))
85. GO
86. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (7, N'Dilip Kumar', 93, CAST(N'1922-12-11 00:00:00.000' AS DateTime))
87. GO
88. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (8, N'Amir Khan', 51, CAST(N'1965-03-14 00:00:00.000' AS DateTime))
89. GO
90. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (9, N'Farhan Akhtar', 42, CAST(N'1942-01-09 00:00:00.000' AS DateTime))
91. GO
92. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (10, N'Saif Ali Khan', 45, CAST(N'1970-08-16 00:00:00.000' AS DateTime))
93. GO
94. INSERT [dbo].[Actor] ([ActorInfoId], [Name], [Age], [DOB]) VALUES (11, N'Prabhas', 36, CAST(N'1979-10-23 00:00:00.000' AS DateTime))
95. GO
96. SET IDENTITY_INSERT [dbo].[Actor] OFF
97. GO
98. SET IDENTITY_INSERT [dbo].[Movie] ON
99.
100. GO
101. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (1, N'Sholay', N'Action', CAST(N'1975-08-15 00:00:00.000' AS DateTime), N'Amitabh, Dharmendar, N'Hema Malini')
102. GO
103. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (2, N'Deewar', N'Action', CAST(N'1979-05-14 00:00:00.000' AS DateTime), N'Amitabh, Shashi, N'Parveen Babu')
104. GO

```

```

105. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (3, N'Zanjeer', N'Action', CAST(T(N'1973-05-11 00:00:00.000' AS DateTime), N'Amitabh', N'Jaya Bhaduri')
106. GO
107. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (4, N'Don', N'Action', CAST(N'1978-04-20 00:00:00.000' AS DateTime), N'Amitabh', N'Zeenat Aman')
108. GO
109. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (5, N'Anand', N'Drama', CAST(N'1971-04-03 00:00:00.000' AS DateTime), N'Rajesh Khanna', N'Sumita Snyal')
110. GO
111. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (6, N'Bawarchi', N'Drama', CAST(N'1972-06-10 00:00:00.000' AS DateTime), N'Rajesh Khanna', N'Jaya Bhaduri')
112. GO
113. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (7, N'D D L J', N'Romantic', CAST(N'1995-08-19 00:00:00.000' AS DateTime), N'Shahrukh Khan', N'Kajol')
114. GO
115. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (8, N'Kuch Kuch Hota Hai', N'Romantic', CAST(N'1998-08-16 00:00:00.000' AS DateTime), N'Shahrukh Khan', N'Kajol')
116. GO
117. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (9, N'Nayak', N'Action', CAST(N'2001-07-07 00:00:00.000' AS DateTime), N'Anil Kapoor', N'Rani Mukharjee')
118. GO
119. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (10, N'Taal', N'Drama', CAST(N'1999-08-13 00:00:00.000' AS DateTime), N'Anil Kapoor', N'Aishwarya Rai')
120. GO
121. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (11, N'Sainik', N'Action', CAST(N'1993-07-10 00:00:00.000' AS DateTime), N'Akshay Kumar', N'Ashwini Bhave')
122. GO
123. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (12, N'Karma', N'Action', CAST(T(N'1986-08-08 00:00:00.000' AS DateTime), N'Dilip Kumar', N'Nutan')
124. GO
125. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (13, N'Sarfarosh', N'Action', CAST(N'1995-08-08 00:00:00.000' AS DateTime), N'Amir Khan', N'Sonali Bendre')
126. GO
127. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (14, N'Saudagar', N'Action', CAST(N'1999-09-12 00:00:00.000' AS DateTime), N'Dilip Kumar, Raj Kumar', N'Manish Koirala')
128. GO
129. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (15, N'Three Idiots', N'Drama', CAST(N'2012-09-09 00:00:00.000' AS DateTime), N'Amir Khan', N'Kareena Kapoor')
130. GO
131. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (16, N'Rowdy Rathore', N'Action', CAST(N'2013-09-10 00:00:00.000' AS DateTime), N'Akshay Kumar', N'Sonakshi Sinha')
132. GO
133. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (17, N'Baby', N'Action', CAST(N'2015-12-12 00:00:00.000' AS DateTime), N'Akshay Kumar', N'Tapsi')
134. GO
135. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (18, N'Bhaag Milka Bhaag', N'Biographic', CAST(N'2014-10-10 00:00:00.000' AS DateTime), N'Farhan Akhtar', N'Sonam Kapoor')
136. GO
137. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (19, N'Phantom', N'Action', CAST(N'2015-08-12 00:00:00.000' AS DateTime), N'Saif Ali Khan', N'Kareena Kapoor')
138. GO
139. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (20, N'Airlift', N'Action', CAST(N'2016-05-06 00:00:00.000' AS DateTime), N'Akshay Kumar', N'Nimrat Kaur')
140. GO

```

```

141. INSERT [dbo].[Movie] ([ID], [Name], [Genre], [ReleasedDate], [Actor], [Actress]) VALUES (21, N'Bahubali', N'Action', C
    AST(N'2015-08-12 00:00:00.000' AS DateTime), N'Prabhas', N'Tamannah Bhatiya')
142. GO
143. SET IDENTITY_INSERT [dbo].[Movie] OFF
144. GO
145. SET IDENTITY_INSERT [dbo].[MovieInfo] ON
146.
147. GO
148. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (1, 1, N'Ramesh Sippy', N'United Producers
149. Sippy Films', CAST(8.5 AS Decimal(2, 1)), 5, N'Amitabh Bachhan')
150. GO
151. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (2, 2, N'Yash Chopra', N'Trimurti Films', CAST(8.2 AS Decimal(2, 1)), 3, N'Amitabh Bachhan')
152. GO
153. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (3, 3, N'Prakash Meshra', N'Asha Studios', CAST(7.0 AS Decimal(2, 1)), 2, N'Amitabh Bachhan')
154. GO
155. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (4, 4, N'Chandra Barot', N'Nariman Films', CAST(7.9 AS Decimal(2, 1)), 3, N'Amitabh Bachhan')
156. GO
157. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (5, 5, N'Hrishikesh Mukherjee', N'Digital Entertainment', CAST(8.9 AS Decimal(2, 1)), 5, N'Rajesh Khanna')
158. GO
159. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (6, 6, N'Hrishikesh Mukherjee', N'Digital Entertainment', CAST(8.0 AS Decimal(2, 1)), 1, N'Rajesh Khanna')
160. GO
161. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (7, 7, N'Aditya Chopra', N'Yash Raj Films', CAST(8.3 AS Decimal(2, 1)), 3, N'Shahrukh Khan')
162. GO
163. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (8, 8, N'Karan Johar', N'Dharma Productions', CAST(7.8 AS Decimal(2, 1)), 1, N'Shahrukh Khan')
164. GO
165. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (9, 9, N'S. Shankar', N'Sri Surya Movies', CAST(7.8 AS Decimal(2, 1)), NULL, N'Anil Kapoor')
166. GO
167. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (10, 10, N'Subhash Ghai', N'Mukta Arts', CAST(6.8 AS Decimal(2, 1)), NULL, N'Aishwarya')
168. GO
169. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (11, 11, N'Sikander Bharti', N'Manish Arts', CAST(6.6 AS Decimal(2, 1)), NULL, N'Akshay Kumar')
170. GO
171. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (12, 12, N'Sikander Bharti', N'Mukta Arts', CAST(7.4 AS Decimal(2, 1)), NULL, N'Dilip Kumar')
172. GO
173. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (13, 13, N'John Matthew Matthan', N'Cinematt Pictures', CAST(8.2 AS Decimal(2, 1)), 2, N'Amir Khan')
174. GO
175. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (14, 14, N'Subhash Ghai', N'Mukta Arts', CAST(6.7 AS Decimal(2, 1)), NULL, N'Dilip Kumar')
176. GO
177. INSERT [dbo].[MovieInfo] ([MovieInfoId], [MovieId], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole])
    VALUES (15, 15, N'Rajkumar Hirani', N'Vinod Chopra Films', CAST(8.4 AS Decimal(2, 1)), 3, N'Amir Khan')
178. GO

```

```

179. INSERT [dbo].[MovieInfo] ([MovieInfoID], [MovieID], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole]) VALUES (16, 16, N'Prabhu Deva', N'SLB Films', CAST(5.8 AS Decimal(2, 1)), NULL, N'Akshay Kumar')
180. GO
181. INSERT [dbo].[MovieInfo] ([MovieInfoID], [MovieID], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole]) VALUES (17, 17, N'Neeraj Pandey', N'T-Series', CAST(8.2 AS Decimal(2, 1)), 2, N'Akshay Kumar')
182. GO
183. INSERT [dbo].[MovieInfo] ([MovieInfoID], [MovieID], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole]) VALUES (18, 18, N'Rakeysh Omprakash Mehra', N'ROMP Pictures', CAST(8.3 AS Decimal(2, 1)), 3, N'Farhan Akhtar')
184. GO
185. INSERT [dbo].[MovieInfo] ([MovieInfoID], [MovieID], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole]) VALUES (19, 19, N'Kabir Khan', N'Nadiadwala Grandson Entertainment', CAST(5.6 AS Decimal(2, 1)), NULL, N'Saif Ali Khan')
186. GO
187. INSERT [dbo].[MovieInfo] ([MovieInfoID], [MovieID], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole]) VALUES (20, 20, N'Raja Krishna Menon', N'Abundantia Entertainment', CAST(9.1 AS Decimal(2, 1)), 2, N'Akshay Kumar')
188. GO
189. INSERT [dbo].[MovieInfo] ([MovieInfoID], [MovieID], [Director], [Production], [ImdbRating], [FilmfareAward], [LeadRole]) VALUES (21, 21, N'S. S. Rajamouli', N'
190. Arka Media Works', CAST(8.6 AS Decimal(2, 1)), NULL, N'Prabhas')
191. GO
192. SET IDENTITY_INSERT [dbo].[MovieInfo] OFF
193. GO
194. ALTER TABLE [dbo].[MovieInfo] WITH CHECK ADD CONSTRAINT [FK_MovieInfo_Movie] FOREIGN KEY([MovieID])
195. REFERENCES [dbo].[Movie] ([ID])
196. GO
197. ALTER TABLE [dbo].[MovieInfo] CHECK CONSTRAINT [FK_MovieInfo_Movie]
198. GO
199. /****** Object: StoredProcedure [dbo].[BM_GetActorDetails] Script Date: 4/29/2016 2:47:20 PM *****/
200. SET ANSI_NULLS ON
201. GO
202. SET QUOTED_IDENTIFIER ON
203. GO
204. -- =====
205. -- Author: Nimit
206. -- Create date: 04/29/2016
207. -- Description: get actor info
208. -- =====
209. CREATE PROCEDURE [dbo].[BM_GetActorDetails]
210.   -- Add the parameters for the stored procedure here
211.   @Name varchar(50)
212. AS
213. BEGIN
214.   -- SET NOCOUNT ON added to prevent extra result sets from
215.   -- interfering with SELECT statements.
216.   SET NOCOUNT ON;
217.
218.   -- Insert statements for procedure here
219.   SELECT * FROM Actor WHERE Name = @Name
220. END
221.
222. GO
223. /****** Object: StoredProcedure [dbo].[BM_GetMovies] Script Date: 4/29/2016 2:47:20 PM *****/

```

```

224. SET ANSI_NULLS ON
225. GO
226. SET QUOTED_IDENTIFIER ON
227. GO
228. -- =====
229. -- Author: Nimit Joshi
230. -- Create date: 02/19/2016
231. -- Description: This sp is used to get data
232. -- =====
233. CREATE PROCEDURE [dbo].[BM_GetMovies]
234.   -- Add the parameters for the stored procedure here
235.   @PageNumber INT ,
236.   @PageSize INT
237. AS
238. BEGIN
239.   -- SET NOCOUNT ON added to prevent extra result sets from
240.   -- interfering with SELECT statements.
241.   SET NOCOUNT ON;
242.
243.   -- Insert statements for procedure here
244.   SELECT * FROM dbo.Movie (NOLOCK) ORDER BY ID
245.
246. END
247.
248.
249. GO
250. ***** Object: StoredProcedure [dbo].[BM_GetMoviesInfo]  Script Date: 4/29/2016 2:47:20 PM *****/
251. SET ANSI_NULLS ON
252. GO
253. SET QUOTED_IDENTIFIER ON
254. GO
255. -- =====
256. -- Author: Nimit Joshi
257. -- Create date: 04/29/2016
258. -- Description: This sp is used to get movie info
259. -- =====
260. CREATE PROCEDURE [dbo].[BM_GetMoviesInfo]
261.   -- Add the parameters for the stored procedure here
262.   @MovieID INT
263. AS
264. BEGIN
265.   -- SET NOCOUNT ON added to prevent extra result sets from
266.   -- interfering with SELECT statements.
267.   SET NOCOUNT ON;
268.
269.   -- Insert statements for procedure here
270.   SELECT m.Name ,
271.         m.Genre ,
272.         mi.Director ,
273.         mi.Production ,
274.         mi.ImdbRating ,
275.         mi.FilmfareAward ,
276.         mi.LeadRole
277.   FROM Movie m ( NOLOCK )

```

```

278.      INNER JOIN dbo.MovieInfo mi ON m.ID = mi.MovieID
279.      WHERE m.ID = @MovieID
280.      END
281. GO

```

That's it for the database section.

28.4 Working with Web Application

In this section, we will create the architecture of web application and fetch the data from the database and view the data in Razor View and in the Partial View. We will display the data in the Partial view with the help of jQuery UI dialogue or in any simple div.

So, let's begin with the following procedure:

Adding Model:

Step 1: In the Solution Explorer, right click on the solution and click on “Add New Project”,

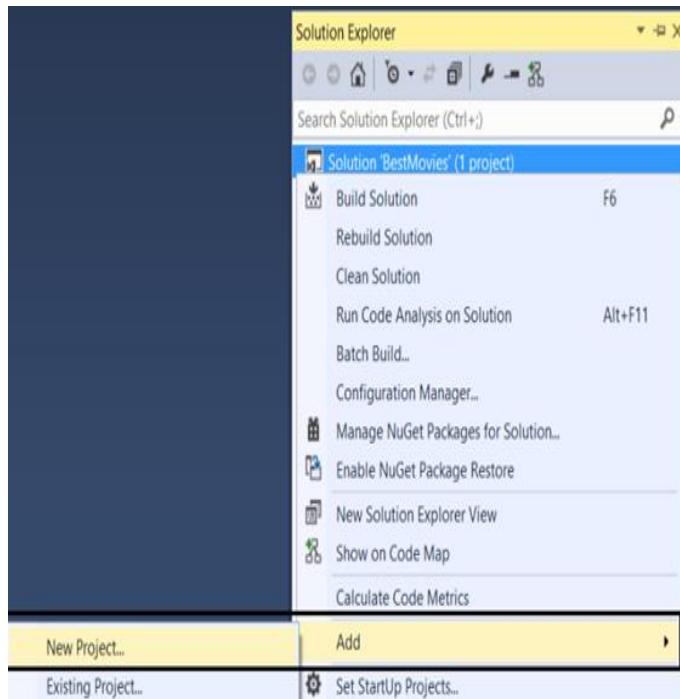


Figure 4: Adding New Project

Step 2: Select the “Class Library” and enter the name as “BestMoviesModel”,

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

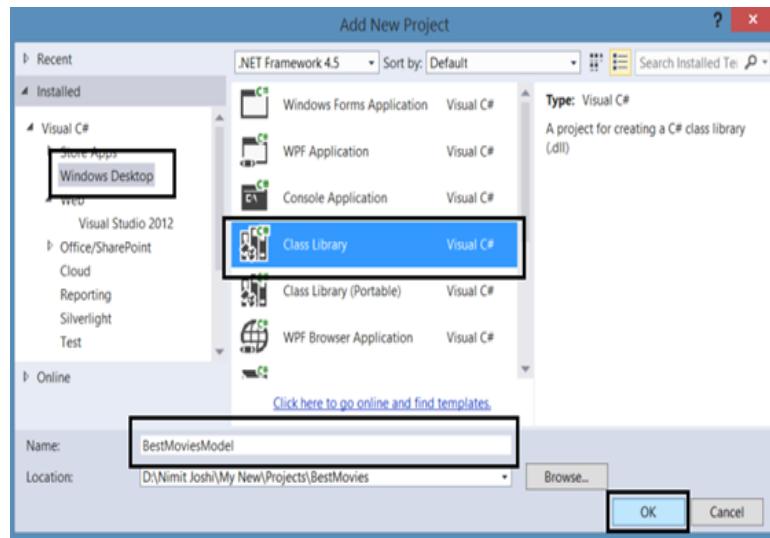


Figure 5: Adding Class Library Project

Step 3: Now add a class in this project as named “Movie”,

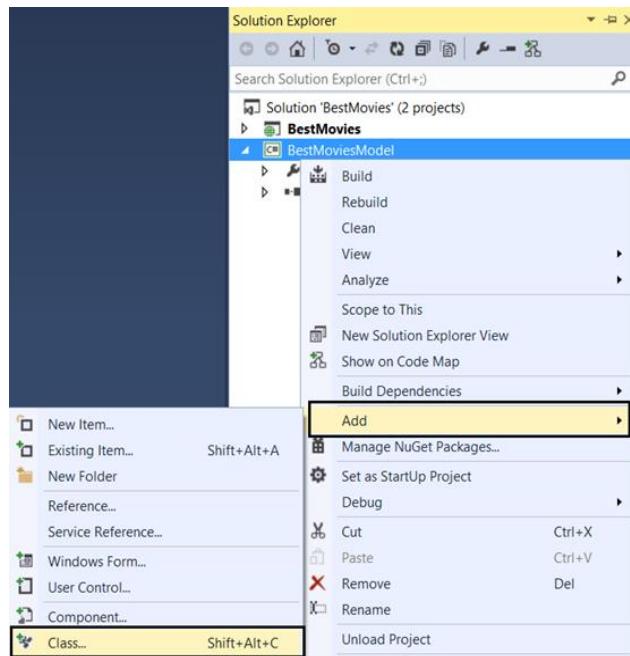


Figure 6: Adding Class

Step 4: Update the class code with the help of following code:

```

1.  namespace BestMoviesModel
2.  {
3.      public class Movie
4.      {
5.          #region Properties
6.          public int Id { get; set; }
7.          public string Name { get; set; }
8.          public string Genre { get; set; }
9.          public DateTime ReleasedDate { get; set; }
10.         public string Actor { get; set; }
11.         public string Actress { get; set; }
12.         #endregion
13.     }
14.
15.    public class MovieInfo
16.    {
17.        #region Properties
18.        public int MovieInfoId { get; set; }
19.        public string Director { get; set; }
20.        public string Production { get; set; }
21.        public decimal ImdbRating { get; set; }
22.        public int FilmfareAward { get; set; }
23.        public string LeadRole { get; set; }
24.        #endregion
25.    }
26.
27.    public class Actor
28.    {
29.        #region Properties
30.        public int ActorInfoId { get; set; }
31.        public string Name { get; set; }
32.        public int Age { get; set; }
33.        public DateTime DOB { get; set; }
34.        #endregion
35.    }
36. }
```

Step 5: Just build the solution.

Adding Core

In this section, we will add the project which will handle the database. Follow the steps below:

Step 1: In the Solution Explorer, right click on the solution and click on “Add New Project” and select “Class Library” as named “BestMoviesCore”

Step 2: Now add a reference of “BestMoviesModel” solution in this project,

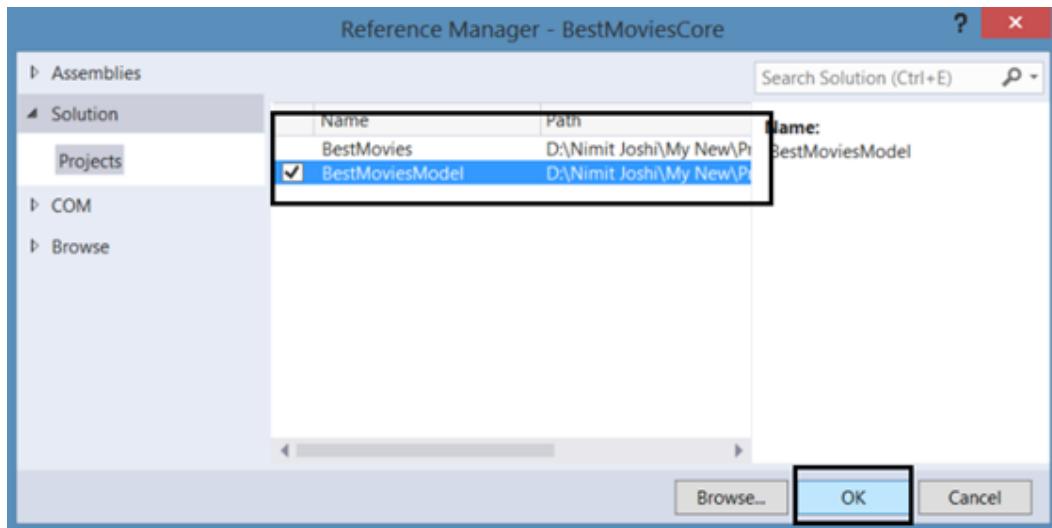


Figure 7: Adding Reference

Step 3: Now in the “BestMoviesCore” project, right click on the References and click on “Manage NuGet Packages” and search for “Enterprise Library” and install it in the project,

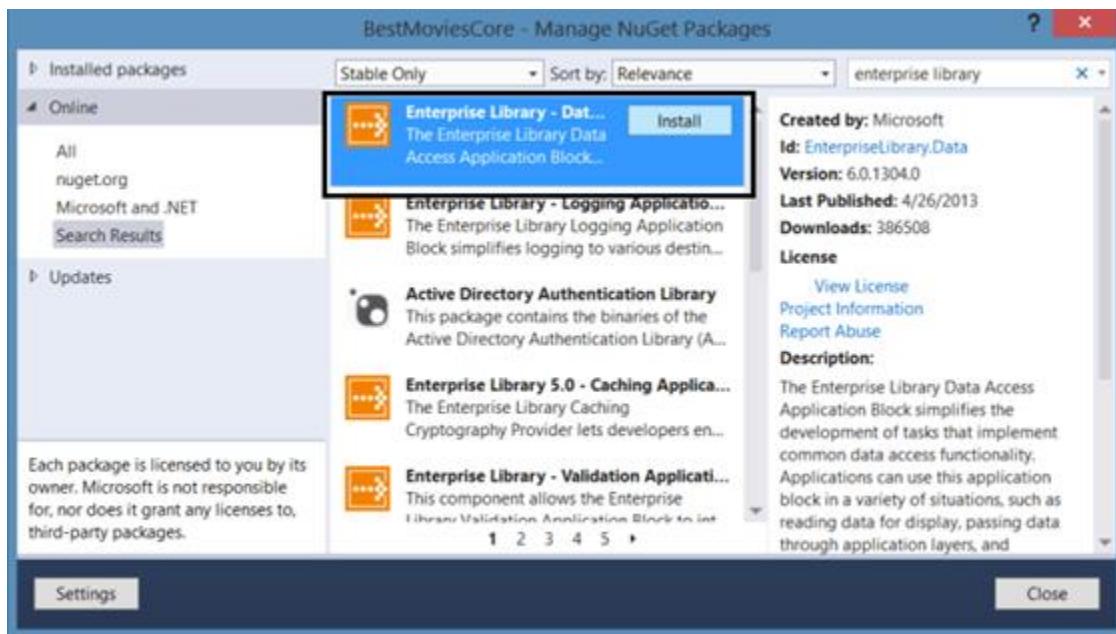


Figure 8: Adding Enterprise Library

Step 4: Now just create two folders as named “BL” and “DAL”.

Step 5: Now add a class in the DAL folder as named “MovieDAL” and replace the code with the following code:

```

1.  namespace BestMoviesCore.DAL
2.  {
3.      public class MovieDAL
4.      {
5.          #region Variable
6.          ///<summary>
7.          /// Specify the Database variable
8.          ///</summary>
9.          Database objDB;
10.         ///<summary>
11.         /// Specify the static variable
12.         ///</summary>
13.         static string ConnectionString;
14.         #endregion
15.
16.         #region Constructor
17.         ///<summary>
18.         /// This constructor is used to get the connectionstring from the config file
19.         ///</summary>
20.         public MovieDAL()
21.         {
22.             ConnectionString = ConfigurationManager.ConnectionStrings["BestMovieConnectionString"].ToString();
23.         }
24.         #endregion
25.
26.         #region Database Method
27.         public List<T> ConvertTo<T>(DataTable datatable) where T : newnew List<T>();
30.             try
31.             {
32.                 List<string> columnsNames = new List<string>();
33.                 foreach (DataColumn DataColumn in datatable.Columns)
34.                     columnsNames.Add(DataColumn.ColumnName);
35.                 Temp = datatable.AsEnumerable().ToList().ConvertAll<T>(row => getObject<T>(row, columnsNames));
36.                 return Temp;
37.             }
38.             catch
39.             {
40.                 return Temp;
41.             }
42.         }
43.         public T getObject<T>(DataRow row, List<string> columnName) where T : newnew T();
46.             try
47.             {

```

```

48.     string columnname = "";
49.     string value = "";
50.     PropertyInfo[] Properties;
51.     Properties = typeof(T).GetProperties();
52.     foreach (PropertyInfo objProperty in Properties)
53.     {
54.         columnname = columnsName.Find(name => name.ToLower() == objProperty.Name.ToLower());
55.         if (!string.IsNullOrEmpty(columnname))
56.         {
57.             value = row[columnname].ToString();
58.             if (!string.IsNullOrEmpty(value))
59.             {
60.                 if (Nullable.GetUnderlyingType(objProperty.PropertyType) != null)
61.                 {
62.                     value = row[columnname].ToString().Replace("$", "").Replace(",", "");
63.                     objProperty.SetValue(obj, Convert.ChangeType(value, Type.GetType(Nullable.GetUnderlyingType(o
bjProperty.PropertyType).ToString()))), null);
64.                 }
65.                 else
66.                 {
67.                     value = row[columnname].ToString();
68.                     objProperty.SetValue(obj, Convert.ChangeType(value, Type.GetType(objProperty.PropertyType.ToS
tring()))), null);
69.                 }
70.             }
71.         }
72.     }
73.     return obj;
74. }
75. catch (Exception ex)
76. {
77.     return obj;
78. }
79. }
80. #endregion
81.
82. #region Movie Details
83. ///<summary>
84. /// This method is used to get the movie data
85. ///</summary>
86. ///<param name="PageNumber"></param>
87. ///<param name="PageSize"></param>
88. ///<returns></returns>
89. public List<Movie> GetMovieList(int? PageNumber, int? PageSize)
90. {
91.     List<Movie> objGetMovie = null;
92.     objDB = new SqlDatabase(ConnectionString);
93.     using (DbCommand objcmd = objDB.GetStoredProcCommand("BM_GetMovies"))
94.     {
95.         try
96.         {
97.             objDB.AddInParameter(objcmd, "@PageNumber", DbType.Int32, PageNumber);
98.             objDB.AddInParameter(objcmd, "@PageSize", DbType.Int32, PageSize);
99.         }

```

```

100.     using (DataTable dataTable = objDB.ExecuteDataSet(objcmd).Tables[0])
101.    {
102.        objGetMovie = ConvertTo<Movie>(dataTable);
103.    }
104. }
105. catch (Exception ex)
106. {
107.     throw ex;
108.     return null;
109. }
110. }
111. return objGetMovie;
112. }
113.
114. ///<summary>
115. /// This method is used to get movie details by movie id
116. ///</summary>
117. ///<returns></returns>
118. public List<MovieInfo> GetMovieInfoById(int Id)
119. {
120.     List<MovieInfo> objMovieDetails = null;
121.     objDB = new SqlDatabase(ConnectionString);
122.     using (DbCommand objcmd = objDB.GetStoredProcedure("BM_GetMoviesInfo"))
123.     {
124.         try
125.         {
126.             objDB.AddInParameter(objcmd, "@MovieId", DbType.Int32, Id);
127.             using (DataTable dataTable = objDB.ExecuteDataSet(objcmd).Tables[0])
128.             {
129.                 objMovieDetails = ConvertTo<MovieInfo>(dataTable);
130.             }
131.         }
132.         catch (Exception ex)
133.         {
134.             throw ex;
135.             return null;
136.         }
137.     }
138.     return objMovieDetails;
139. }
140. #endregion
141.
142. #region LeadRole Details
143. ///<summary>
144. /// This method is used to get the movie data
145. ///</summary>
146. ///<returns></returns>
147. public List<Actor> GetLeadRoleDetails(string Name)
148. {
149.     List<Actor> objGetLeadRoleActor = null;
150.     objDB = new SqlDatabase(ConnectionString);
151.     using (DbCommand objcmd = objDB.GetStoredProcedure("BM_GetActorDetails"))
152.     {
153.         try

```

```

154.    {
155.        objDB.AddInParameter(objcmd, "@Name", DbType.String, Name);
156.        using (DataTable dataTable = objDB.ExecuteDataSet(objcmd).Tables[0])
157.        {
158.            objGetLeadRoleActor = ConvertTo<Actor>(dataTable);
159.        }
160.    }
161.    catch (Exception ex)
162.    {
163.        throw ex;
164.        return null;
165.    }
166. }
167. return objGetLeadRoleActor;
168. }
169. #endregion
170. }
171. }
```

Step 6: Now add a class in the “BL” folder as named “MovieBL” and replace the code with the following code:

```

1. namespace BestMoviesCore.BL
2. {
3.     public class MovieBL
4.     {
5.         /// <summary>
6.         /// This method is used to get the movie data
7.         /// </summary>
8.         /// <param name="PageNumber"></param>
9.         /// <param name="PageSize"></param>
10.        /// <returns></returns>
11.        public List<Movie> GetMovieList(int? PageNumber, int? PageSize)
12.        {
13.            List<Movie> objGetMovie = null;
14.            try
15.            {
16.                objGetMovie = new MovieDAL().GetMovieList(PageNumber, PageSize);
17.            }
18.            catch (Exception)
19.            {
20.                throw;
21.            }
22.            return objGetMovie;
23.        }
24.
25.        /// <summary>
26.        /// This method is used to get movie details by movie id
27.        /// </summary>
28.        /// <returns></returns>
29.        public List<MovieInfo> GetMovieInfoById(int Id)
30.        {
```

```
31.     List<MovieInfo> objMovieDetails = null;
32.     try
33.     {
34.         objMovieDetails = new MovieDAL().GetMovieInfoById(Id);
35.     }
36.     catch (Exception)
37.     {
38.         throw;
39.     }
40.     return objMovieDetails;
41. }
42.
43. ///<summary>
44. /// This method is used to get the movie data
45. ///</summary>
46. ///<returns></returns>
47. public List<Actor> GetLeadRoleDetails(string Name)
48. {
49.     List<Actor> objGetLeadRoleActor = null;
50.     try
51.     {
52.         objGetLeadRoleActor = new MovieDAL().GetLeadRoleDetails(Name);
53.     }
54.     catch (Exception)
55.     {
56.         throw;
57.     }
58.     return objGetLeadRoleActor;
59. }
60. }
61. }
```

Step 7: That's it with this section. Now build the solution.

28.5 Working with Web Application

Now in this section, we will work with the MVC web application and view the data. Start with the following steps:

Step 1: At first, we will add the reference of “BestMoviesModel” and “BestMoviesCore” projects reference in this project.

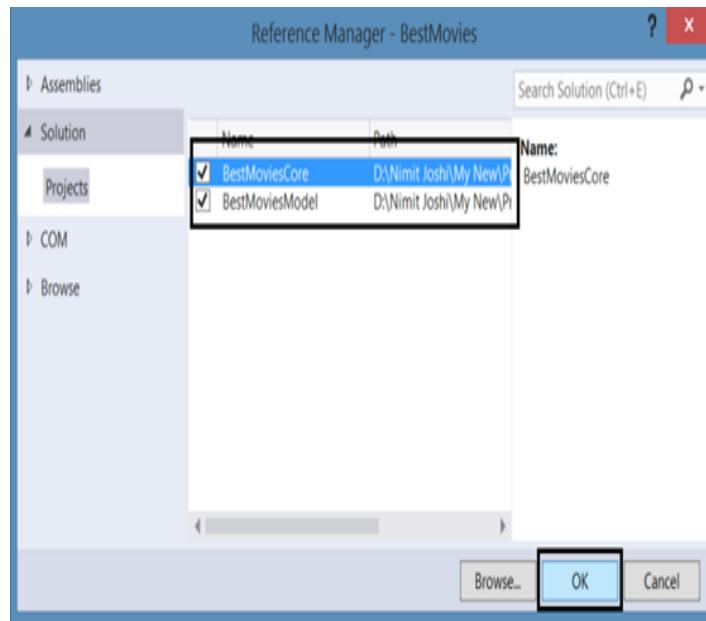


Figure 9: Adding Reference in Web

Step 2: Right click on the Models folder and add a class as named “MovieDetails”,

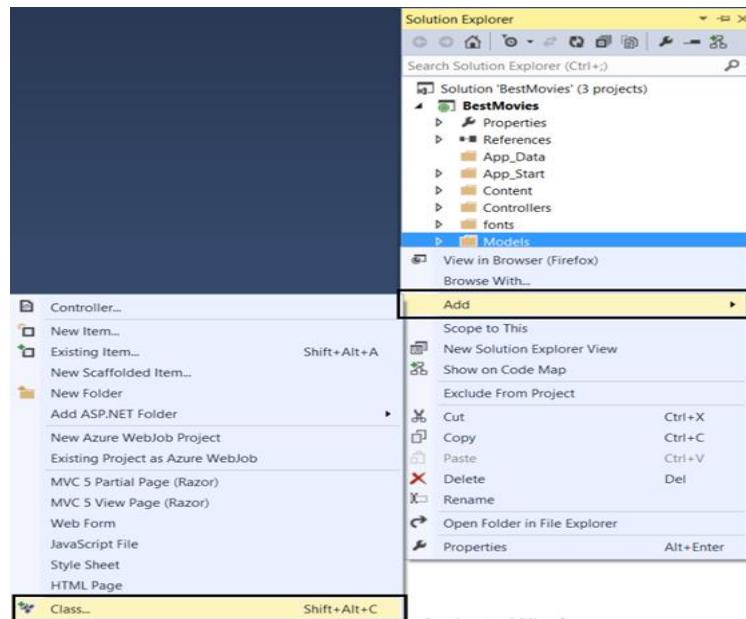


Figure 10: Adding Class in Models

Step 3: Replace the code with the following code:

```

1. using BestMoviesModel;
2. using System.Collections.Generic;
3.
4. namespace BestMovies.Models
5. {
6.     public class MovieDetails
7.     {
8.         ///<summary>
9.         /// get and set the Movies
10.        ///</summary>
11.        public List<Movie> Movies { get; set; }
12.    }
13. }
```

Step 4: Now right click on the Controllers folder and click on Add-> New -> Controller.

Step 5: Now in the wizard select the “MVC 5 Empty Controller”,

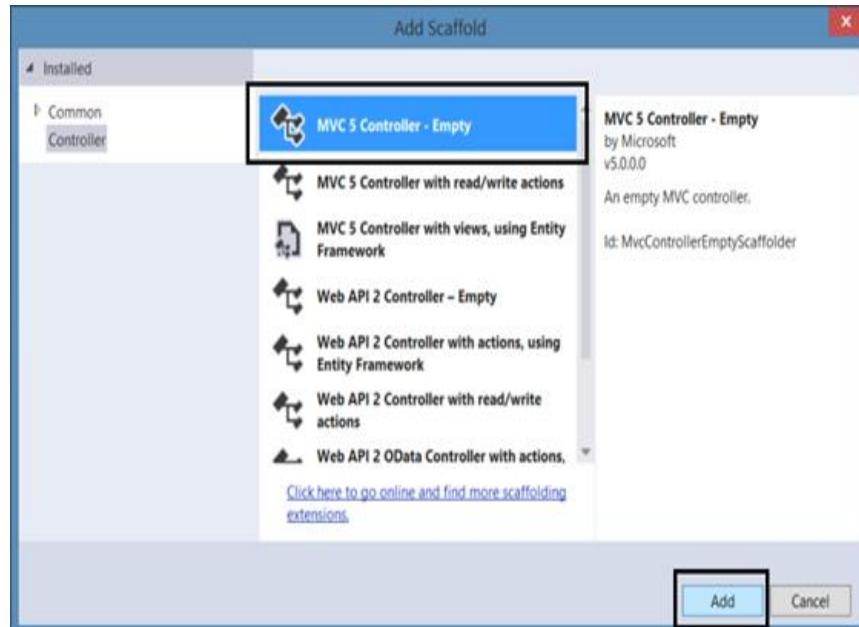


Figure 11: Add Scaffold in MVC 5

Step 6: Enter the controller name as “MovieController”,

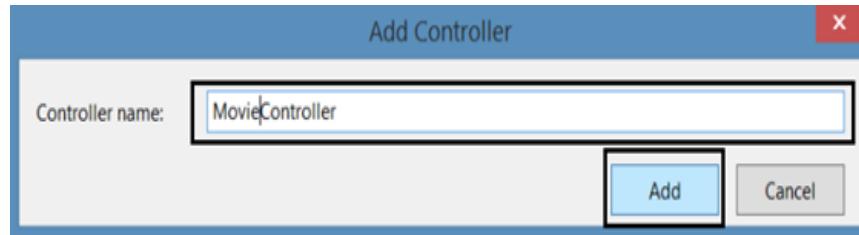


Figure 12: Add Controller in MVC 5

Step 7: Add the following method in the MovieController,

```

1. /// <summary>
2. /// This method is used to get all movies
3. /// </summary>
4. /// <param name="PageNumber"></param>
5. /// <param name="PageSize"></param>
6. /// <returns></returns>
7. [HttpGet, ActionName("GetAllMovies")]
8. public ActionResult GetAllMovies(int? PageNumber, int? PageSize)
9. {
10.     List<Movie> objMovie = new List<Movie>();
11.     MovieBL objMovieBL = new MovieBL();
12.
13.     if (object.Equals(PageNumber, null))
14.     {
15.         PageNumber = 1;
16.     }
17.     if (object.Equals(PageSize, null))
18.     {
19.         PageSize = Convert.ToInt32(ConfigurationManager.AppSettings["DefaultPageSize"]);
20.
21.     }
22.     objMovie = objMovieBL.GetMovieList(PageNumber, PageSize);
23.
24.     return View("~/Views/Movie/BestMovies.cshtml", new MovieDetails() { Movies = objMovie });
25. }
```

Step 8: Now goto Views-> Movie, right click on it and add view,

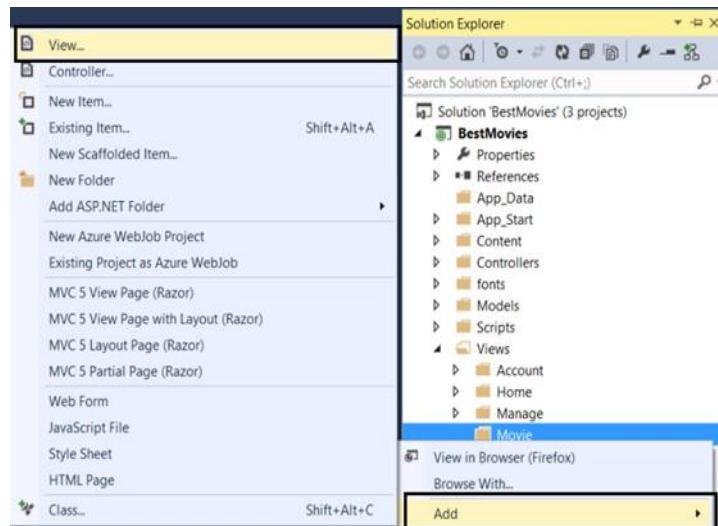


Figure 13: Adding View in MVC 5

Step 9: Enter the view name as “BestMovies”,

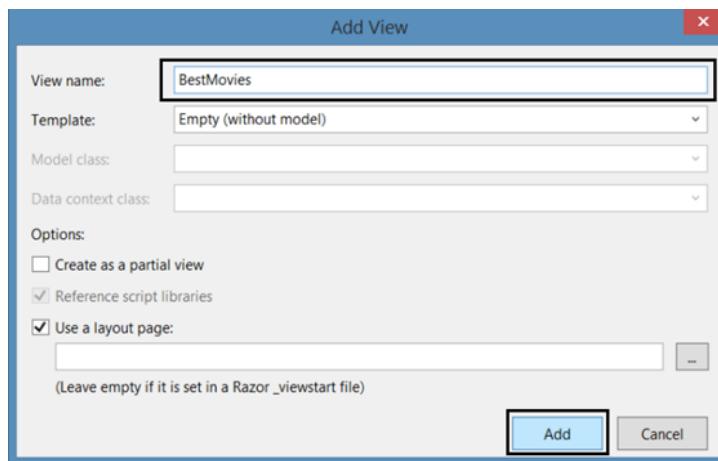


Figure 14: Add View Wizard

Step 10: Add the following code in the view,

```

1. @model BestMovies.Models.MovieDetails
2. @{
3.     ViewBag.Title = "BestMovies";
4. }
  
```

```

5.
6. <h2>Best Movies</h2>
7.
8. <div class="MovieList">
9.   <div id="MoviesGrid">
10.    @Html.Partial("~/Views/Movie/_BestMoviesPartial.cshtml", Model.Movies)
11.   </div>
12. </div>

```

Note: We are passing data to the Partial view to load the Movie data.

Step 11: Now add a Partial View as named “_BestMoviesPartial” in the View-> Movie folder,

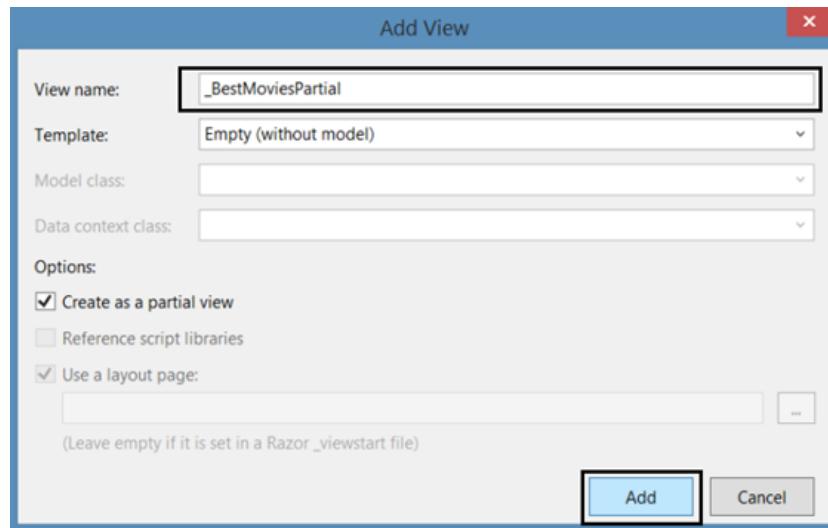


Figure 15: Adding Partial View in MVC 5

Step 12: Now add the following code in this view:

```

1. @model List<BestMoviesModel.Movie>
2.
3. <table class="table-responsive table">
4.   <thead>
5.     <tr>
6.       <th>Name</th>
7.       <th>Genre</th>
8.       <th>Released Date</th>
9.       <th>Actor</th>
10.      <th>Actress</th>
11.    </tr>
12.  </thead>
13.  <tbody>
14.    @if (Model.Count > 0)

```

```

15.    {
16.        foreach (var movieItem in Model)
17.        {
18.            <tr>
19.                <td>@movieItem.Name</td>
20.                <td>@movieItem.Genre</td>
21.                <td>@movieItem.ReleasedDate.ToShortDateString()</td>
22.                <td>@movieItem.Actor</td>
23.                <td>@movieItem.Actress</td>
24.            </tr>
25.        }
26.    }
27.    else
28.    {
29.        <tr>
30.            <td>
31.                No Data Found
32.            </td>
33.        </tr>
34.    }
35. </tbody>
36. </table>

```

Step 13: Now build the solution and open the Views-> Shared-> _Layout.cshtml and replace the code with the highlighted code below:

```

1.  <head>
2.    <meta charset="utf-8" />
3.    <meta name="viewport" content="width=device-width, initial-scale=1.0">
4.    <title>@ViewBag.Title - My Movies Application</title>
5.    @Styles.Render("~/Content/css")
6.    @Scripts.Render("~/bundles/modernizr")
7.
8.  </head>
9.  <body>
10. <div class="navbar navbar-inverse navbar-fixed-top">
11.   <div class="container">
12.     <div class="navbar-header">
13.       <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
14.         <span class="icon-bar"></span>
15.         <span class="icon-bar"></span>
16.         <span class="icon-bar"></span>
17.       </button>
18.       @Html.ActionLink("Best Movies", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
19.     </div>
20.     <div class="navbar-collapse collapse">
21.       <ul class="nav navbar-nav">
22.         <li>@Html.ActionLink("Home", "Index", "Home")</li>
23.         <li>@Html.ActionLink("About", "About", "Home")</li>
24.         <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
25.         <li>@Html.ActionLink("Movies", "GetAllMovies", "Movie")</li>
26.       </ul>

```

```

27.         @Html.Partial("_LoginPartial")
28.     </div>
29.   </div>
30. </div>
31. <div class="container body-content">
32.   @RenderBody()
33.   <hr />
34.   <footer>
35.     <p>&copy; @DateTime.Now.Year - Best Movies Application</p>
36.   </footer>
37. </div>
38.
39.   @Scripts.Render("~/bundles/jquery")
40.   @Scripts.Render("~/bundles/bootstrap")
41.   @RenderSection("scripts", required: false)
42. </body>
43. </html>

```

Step 14: Now open the “Web.config” file of Web application and add the following two lines in your file:

Adding Key:

1. <add key="DefaultPageSize" value="25" />

Adding ConnectionString:

1. <add name="BestMovieConnectionString" connectionString="Data Source=MCNDESKTOP07;Initial Catalog=BestMovies;User Id = "UserID";Password="UserPassword""
- 2.
3. providerName="System.Data.SqlClient" />

Step 15: Now run the application and click on the “Movies”,

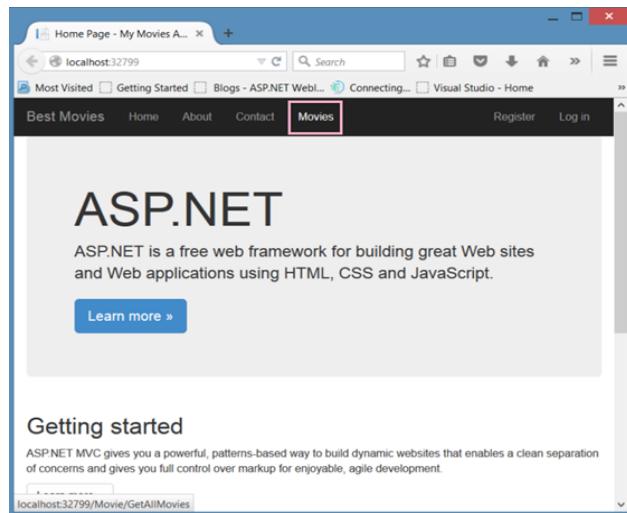
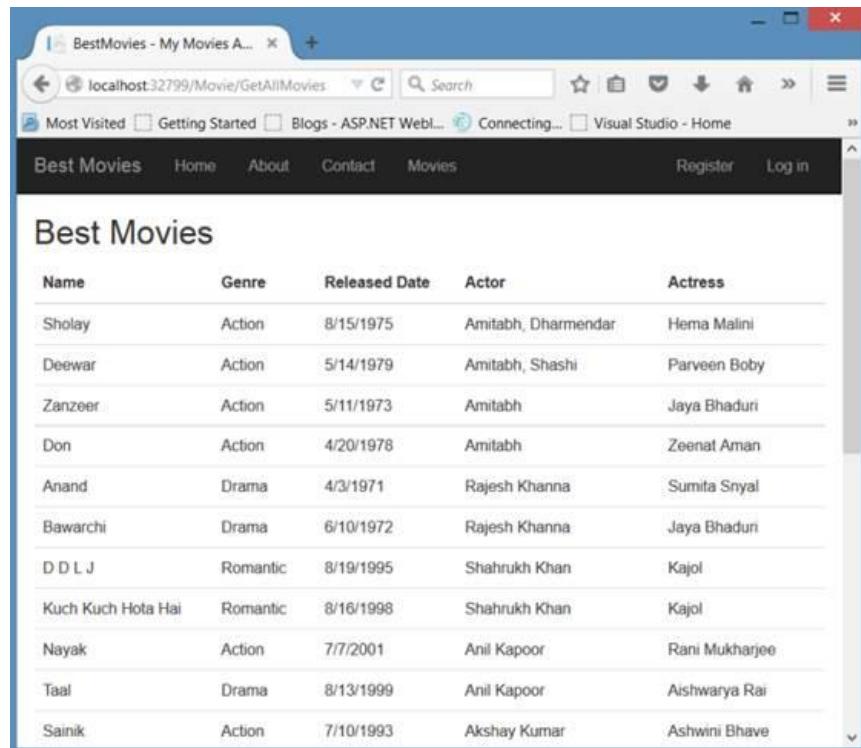


Figure 16: Main Page View of MVC

In the next page you will see Movies data,



The screenshot shows a Microsoft Edge browser window displaying the 'Best Movies' partial view. The URL in the address bar is 'localhost:32799/Movie/GetAllMovies'. The page has a dark header with navigation links: 'Best Movies', 'Home', 'About', 'Contact', 'Movies', 'Register', and 'Log in'. Below the header, the title 'Best Movies' is displayed in large letters. A table follows, showing a list of movies with columns: Name, Genre, Released Date, Actor, and Actress. The table contains 14 rows of movie data.

Name	Genre	Released Date	Actor	Actress
Sholay	Action	8/15/1975	Amitabh, Dharmendar	Hema Malini
Deewar	Action	5/14/1979	Amitabh, Shashi	Parveen Boby
Zanjeer	Action	5/11/1973	Amitabh	Jaya Bhaduri
Don	Action	4/20/1978	Amitabh	Zeenat Aman
Anand	Drama	4/3/1971	Rajesh Khanna	Sumita Snyal
Bawarchi	Drama	6/10/1972	Rajesh Khanna	Jaya Bhaduri
D D L J	Romantic	8/19/1995	Shahrukh Khan	Kajol
Kuch Kuch Hota Hai	Romantic	8/16/1998	Shahrukh Khan	Kajol
Nayak	Action	7/7/2001	Anil Kapoor	Rani Mukharjee
Taal	Drama	8/13/1999	Anil Kapoor	Aishwarya Rai
Sainik	Action	7/10/1993	Akshay Kumar	Ashwini Bhave

Figure 17: Partial View in MVC 5

28.6 Partial View with jQuery UI Dialogue

In this section, we will pass the Partial View in the jQuery UI Dialogue. For this please follow the steps below:

Step 1: Add the following method in the “MovieController”

```

1. /// <summary>
2. /// This method is used to get movie information
3. /// </summary>
4. /// <param name="Movield"></param>
5. /// <returns></returns>
6. [HttpGet, ActionName("GetMovieInfo")]
7. public ActionResult GetMovieInfo(string Movield)
8. {
9.     List<MovieInfo> ObjMovieInfo = new List<MovieInfo>();
10.    MovieBL objMovieBL = new MovieBL();
11.
12.    ObjMovieInfo = objMovieBL.GetMovieInfoById(Convert.ToInt32(Movield));
13.
14.    return PartialView("~/Views/Movie/_MovieDetailsPartial.cshtml", new List<MovieInfo>(ObjMovieInfo));
15. }
```

Step 2: Add another Partial View as named “_MovieDetailsPartial” in the Views->Movie folder and add the following code in it:

```

1. @model List<BestMoviesModel.MovieInfo>
2.
3. <ul class="responsive-MovieDetails">
4.     @if (Model.Count > 0)
5.     {
6.         foreach (var item in Model)
7.         {
8.             <li class="UserDetailList"><span class="UserDetailHeader">Director</span><span>@item.Director</span></li>
9.
10.            <li class="UserDetailList"><span class="UserDetailHeader">IMDB Rating</span><span>@item.ImdbRating</span></li>
11.
12.            <li class="UserDetailList"><span class="UserDetailHeader">Production</span><span>@item.Production</span></li>
13.
14.            <li class="UserDetailList"><span class="UserDetailHeader">Filmfare Awards</span><span>@item.FilmfareAwards</span></li>
15.
16.            <li class="UserDetailList"><span class="UserDetailHeader">Lead Role</span><span>@item.LeadRole</span></li>
17.
18.        }
19.    }</ul>
```

Step 3: Add the reference of jQueryUI from the NuGet Package Manager

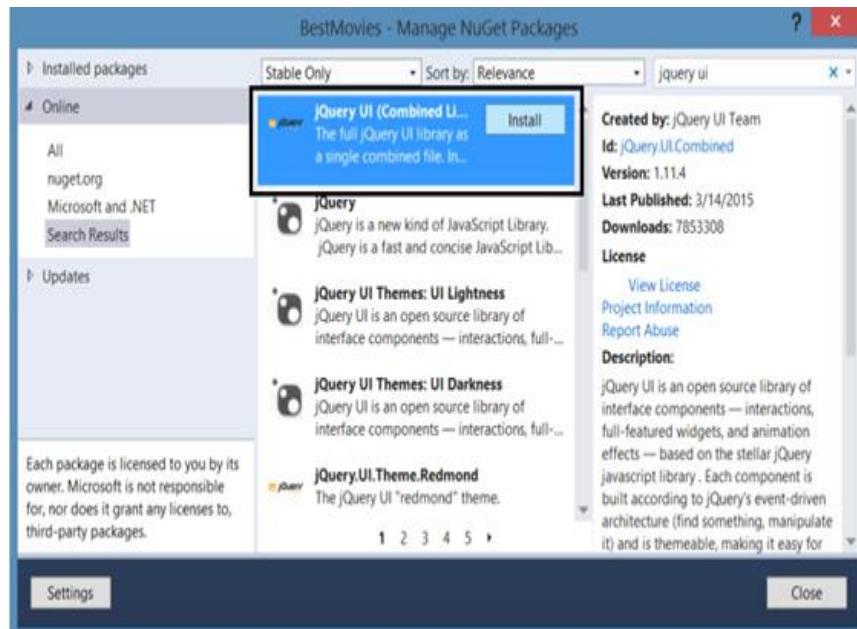


Figure 18: Adding jQuery UI NuGet Package

Step 4: Now open “BestMovies.cshtml” and add the following code at the end:

```

1. <div class="popupcntr" id="movieInfo_content" style="display: none;" title="Movie Information">
2.   <div class="innerBox">
3.     <div id="MovieDetails"></div>
4.   </div>
5. </div>
```

Step 5: Now open “_BestMoviesPartial.cshtml” and update the code with the highlighted code below:

```

1. @model List<BestMoviesModel.Movie>
2.
3.
4. <table class="table-responsive table">
5.   <thead>
6.     <tr>
7.       <th>Name</th>
8.       <th>Genre</th>
9.       <th>Released Date</th>
10.      <th>Actor</th>
11.      <th>Actress</th>
12.    </tr>
```

```

13. </thead>
14. <tbody>
15. @if (Model.Count > 0)
16. {
17.     foreach (var movieItem in Model)
18.     {
19.         <tr>
20.             <td><a href="javascript:void(0)" onclick="GetMovieDetails('@movieItem.Id')">@movieItem.Name</a></td>
d>
21.             <td>@movieItem.Genre</td>
22.             <td>@movieItem.ReleasedDate.ToShortDateString()</td>
23.             <td>@movieItem.Actor</td>
24.             <td>@movieItem.Actress</td>
25.         </tr>
26.     }
27. }
28. else
29. {
30.     <tr>
31.         <td>
32.             No Data Found
33.         </td>
34.     </tr>
35. }
36. </tbody>
37. </table>
38.
39. <script type="text/javascript">
40.     function GetMovieDetails(MovieId)
41.     {
42.         $('#movieInfo_content').dialog({
43.             dialogClass: 'moviedetail_dialog',
44.             modal: true,
45.             open: function (event, ui) {
46.                 $.ajax({
47.                     url: '@Url.Action("GetMovieInfo", "Movie")',
48.                     dataType: "html",
49.                     data: { MovieId: MovieId },
50.                     type: "GET",
51.                     error: function (xhr, status, error) {
52.                         var err = eval("(" + xhr.responseText + ")");
53.                         toastr.error(err.message);
54.                     },
55.                     success: function (data) {
56.                         $("#divProcessing").hide();
57.                         $('#MovieDetails').html(data);
58.                     },
59.                     beforeSend: function () {
60.                         $("#divProcessing").show();
61.                     }
62.                 });
63.             },
64.             close: function (event, ui) { $('#movieInfo_content').dialog("destroy"); $('#MovieDetails').html(""); },
65.         });
66.     }

```

```
66.    }
67. </script>
```

Step 6: Now add the following css code in the “Site.css”,

```
1. .moviedetail_dialog {
2.   padding: 20px;
3.   background-color: #fbfbfb;
4.   border: 1px solid rgba(0,0,0,0.2);
5.   box-shadow: 0 0 6px black;
6. }
7.
8. .ui-widget-header {
9.   display: inline-block;
10.  font-weight: bold;
11.  margin-right: 20px;
12. }
13.
14. .moviedetail_dialog button {
15.   display: inline-block;
16.   margin-left: 20px;
17. }
18.
19. .responsive-MovieDetails {
20.   list-style: none;
21.   margin-left: -41px;
22.   margin-top: 20px;
23. }
24.
25. .responsive-MovieDetails:after {
26.   content: "";
27.   display: table;
28.   clear: both;
29. }
30.
31. .UserDetailList {
32.   margin-bottom: 5px;
33.   margin: 0;
34.
35. }
36.
37. .UserDetailList:after {
38.   content: "";
39.   display: table;
40.   clear: both;
41. }
42.
43. .UserDetailHeader {
44.   width: 118px;
45.   float: left;
46.   font-weight: bold;
47. }
48.
```

```

49.
50. .UserDetailHeader + span:before {
51.   content: ":";
52.   margin-right: 15px;
53.   position: absolute;
54.   left: -5px;
55. }
56.
57. .UserDetailHeader + span {
58.   float: left;
59.   width: calc(100% - 118px);
60.   padding-left: 10px;
61.   position: relative;
62. }
63.
64. #ActorContent .responsive-MovieDetails{
65.   border: 1px solid black; padding:2px;
66.   margin-left:0;
67. }

```

Step 7: Now build the solution and run the application. Open the “Movies” page and just click on any Movie Name as shown below:

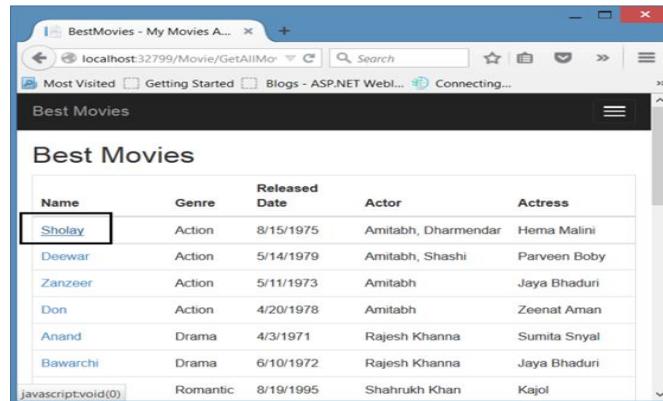


Figure 19: View in MVC 5

Step 8: You will show the popup in the main view as shown below:

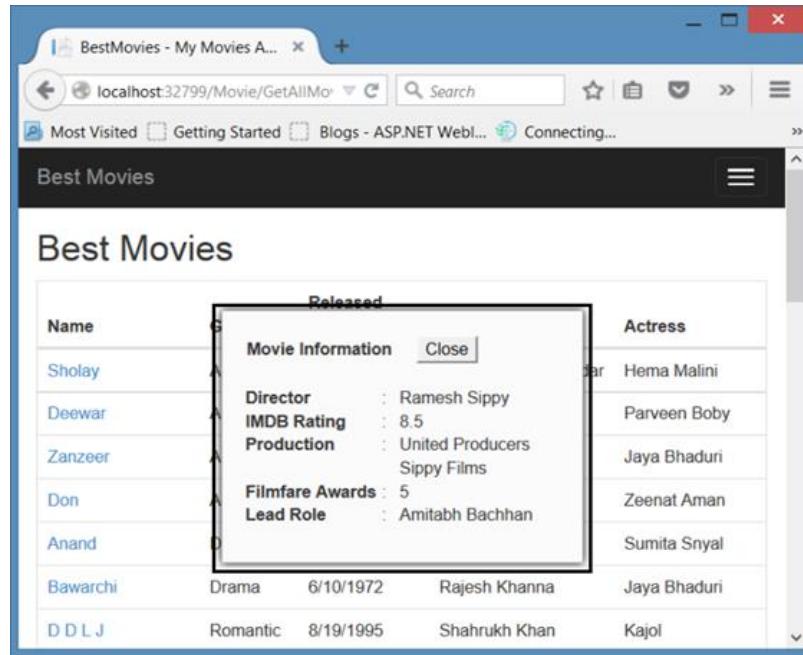


Figure 20: Partial View in UI dialogue

Now you can see that we have easily implemented and passed the Partial View in the jQuery UI dialogue.

Note: You have to add script tag of jQuery UI in the _Layout page.

28.7 Partial View with DIV Element

In this section we will load the Partial View in a div element. Start with the following steps:

Step 1: Add the following method in the “MovieController”,

```

1. /// <summary>
2. /// This method is used to get actor information
3. /// </summary>
4. /// <param name="Movield"></param>
5. /// <returns></returns>
6. [HttpGet, ActionName("GetLeadRoleDetails")]
7. public ActionResult GetLeadRoleDetails(string Name)
8. {
9.     List<Actor> ObjActorInfo = new List<Actor>();
10.    MovieBL objMovieBL = new MovieBL();
11.
12.    ObjActorInfo = objMovieBL.GetLeadRoleDetails(Name);

```

```

13.
14.     return PartialView("~/Views/Movie/_ActorDetailsPartial.cshtml", new List<Actor>(ObjActorInfo));
15. }
```

Step 2: Add another Partial View as named “_ActorDetailsPartial” and add the following code:

```

1.     @model List<BestMoviesModel.Actor>
2.
3.     <ul class="responsive-MovieDetails">
4.         @if (Model.Count > 0)
5.         {
6.             foreach (var item in Model)
7.             {
8.                 <li class="UserDetailList"><span class="UserDetailHeader">Name</span><span>@item.Name</span></li>
9.                 <li class="UserDetailList"><span class="UserDetailHeader">Age</span><span>@item.Age</span></li>
10.                <li class="UserDetailList"><span class="UserDetailHeader">DOB</span><span>@item.DOB.ToShortDateString()
    </span></li>
11.            }
12.        }
13.        else
14.        {
15.            <li class="NoData">No data found</li>
16.        }
17.    </ul>
```

Step 3: Now change the “_MovieDetailsPartial.cshtml” page code with the highlighted code below:

```

1.     @model List<BestMoviesModel.MovieInfo>
2.
3.     <ul class="responsive-MovieDetails">
4.         @if (Model.Count > 0)
5.         {
6.             foreach (var item in Model)
7.             {
8.                 <li class="UserDetailList"><span class="UserDetailHeader">Director</span><span>@item.Director</span></li>
9.                 <li class="UserDetailList"><span class="UserDetailHeader">IMDB Rating</span><span>@item.ImdbRating</span></li>
10.                <li class="UserDetailList"><span class="UserDetailHeader">Production</span><span>@item.Production</span></li>
11.                <li class="UserDetailList"><span class="UserDetailHeader">Filmfare Awards</span><span>@item.FilmfareAwards</span></li>
12.                <li class="UserDetailList">
13.                    <span class="UserDetailHeader">Lead Role</span><span id="MovieLeadRole">@item.LeadRole <a href="jav
    aScript:void(0)" onclick="GetActorDetails('@item.LeadRole')"><img src "~/Images/movie_info.png" /></a></span>
14.                    <div id="ActorContent"></div>
15.                </li>
16.            }
17.        }
18.        else
```

```

19.  {
20.      <li class="NoData">No data found</li>
21.  }
22. </ul>
23. <script>
24. //This method is used to edit user location
25. function GetActorDetails(name) {
26.     $.ajax({
27.         url: '@Url.Action("GetLeadRoleDetails", "Movie")',
28.         dataType: "html",
29.         data: { Name: name },
30.         type: "GET",
31.         error: function (xhr, status, error) {
32.             var err = eval("(" + xhr.responseText + ")");
33.             toastr.error(err.message);
34.         },
35.         success: function (data) {
36.             $("#MovieLeadRole").css("visibility", "hidden");
37.             $('#ActorContent').html("");
38.             $('#ActorContent').html(data);
39.             $('#ActorContent').show();
40.         }
41.     });
42. }
43. </script>

```

Step 4: Now run the application and click on the Movie Name to show the information,

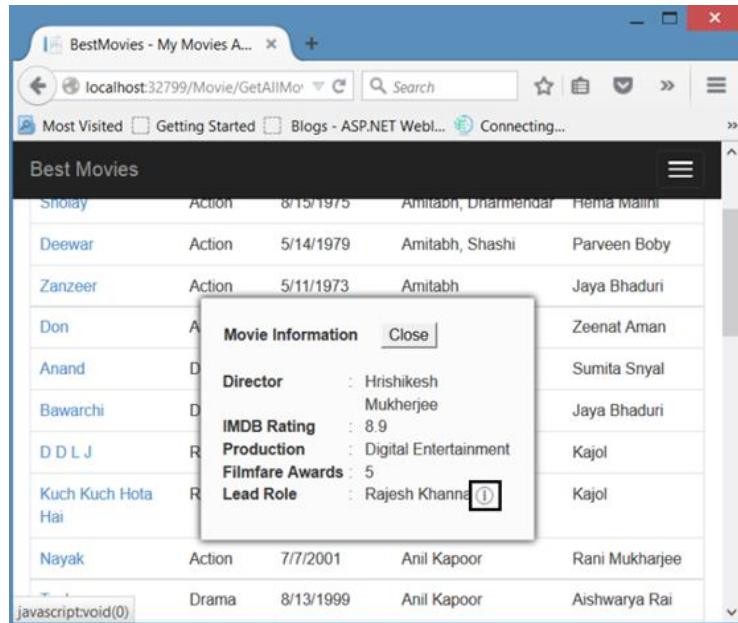


Figure 21: Loading Partial View in Div Element

Step 5: Now click on the info icon and the newly added Partial View will load in the Div element as shown below:

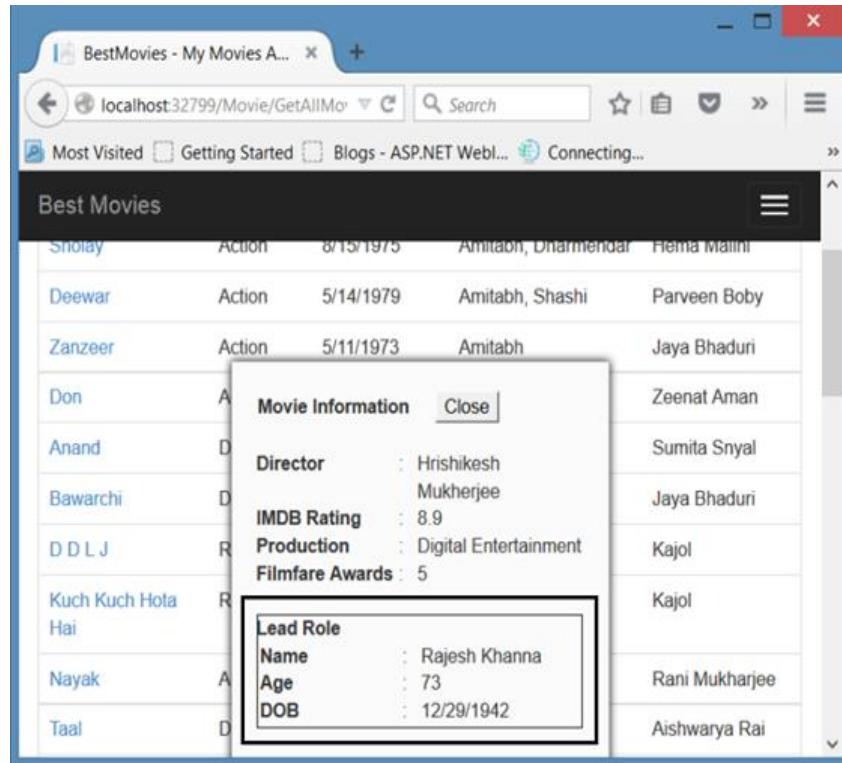


Figure 22: Partial View in Div Element