**ASP.NET MVC 4 Hands On Lab**

**Part 4: Adding View(s)**

1. Modify the **Index()** action method of the **HomeController** as follows, so that it now returns a view instead of a simple string:

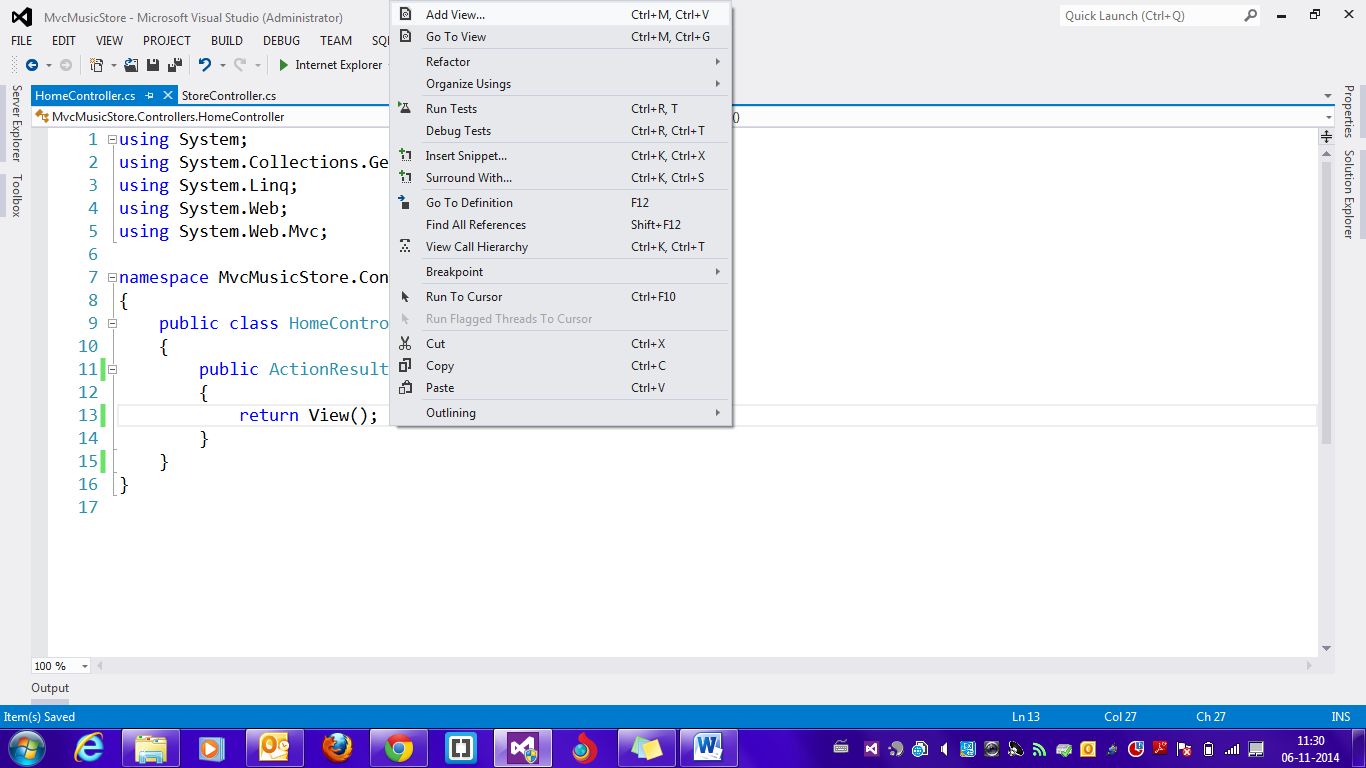
public ActionResult Index()

{

return View();

}

1. Right-click anywhere inside the action method & choose **Add View** as shown below:



1. Just click on **Add** in the add view dialog which is displayed. This will add an **Index.cshtml** page inside the **Home** folder within the **Views** folder of your project.
2. Inside **Index.cshtml,** update the contents of the **<h2>** element as follows:

<h2>This is the Home Page</h2>

1. Run the application by pressing **F5** & observe the results.

**Part 5: Using a layout for common site elements**

Most websites have content which is shared between many pages: navigation, footers, logo images, stylesheet references, etc. The Razor view engine makes this easy to manage using a page called **\_Layout.cshtml** which has automatically been created for us inside the **/Views/Shared** folder.

**Since we are going to modify the contents of the file by adding some logo & navigation links, we also need to add some images & update the stylesheet which MVC by default uses.**

1. Create a new folder named **Images** inside the **Content** folder of your project.
2. Add the following image files inside the **Images** folder:

**home-showcase.png**

**logo.png**

**placeholder.gif**

1. Copy the **Site.css** file inside the **Content** folder of your project. This is a slightly updated version of the existing stylesheet. When prompted to overwrite the existing file, choose Yes.
2. Open the **\_Layout.cshtml** file.
3. Delete the existing code before the call to **@RenderBody()** method.
4. Add the following code inside the **<body>** element:

<table width="100%" cellpadding="2" cellspacing="2" align="center">

<tr>

<td align="left" width="1%">

<img src="~/Content/images/logo.png" width="90"

height="45"/>

</td>

<td>

<h1 style="color:orange;">

ASP.NET MVC Music Store App

</h1>

</td>

</tr>

<tr>

<td colspan="2">

<ul id="navlist">

<li class="first">

<a href="/" id="current">Home</a></li>

<li><a href="/Store/">Store</a></li>

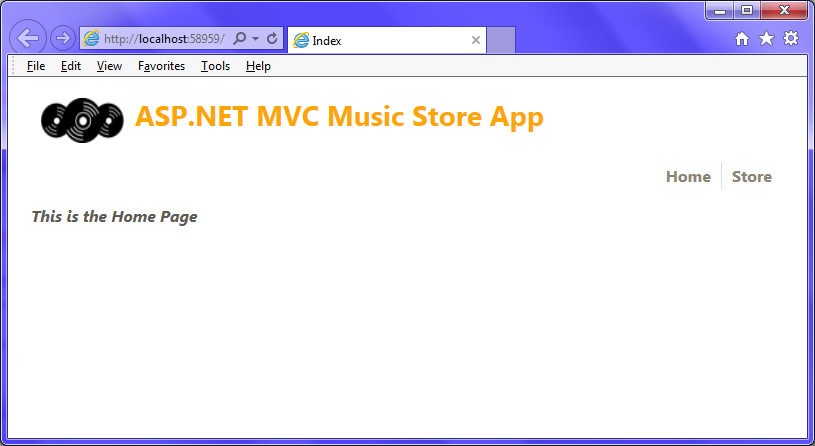
</ul>

</td>

</tr>

</table>

1. Run the application by pressing **F5.** Since the **Index.cshtml** page is using the layout file, the output must appear as follows:



**Part 6: Using a Model to pass information to a view**

**Let’s first create classes to represent an *Album* & a *Genre* respectively.**

1. Add two files named **Genre.cs** & **Album.cs** inside the **Models** folder of your project.
2. Add the following classes inside each file respectively:

public class Genre

{

public string GenreName { get; set; }

}

public class Album

{

public string Title { get; set; }

public Genre Genre { get; set; }

}

1. Build the project.

**Let’s first update the *Details()* action method of the *StoreController* so that it returns an *ActionResult* rather than a string, as we did with the *HomeController’s* *Index()* action method.**

1. Add the following **using** statement inside **StoreController.cs** file:

using MvcMusicStore.Models;

1. Modify the **Details()** action method of the **StoreController** as follows:

public ActionResult Details(string id)

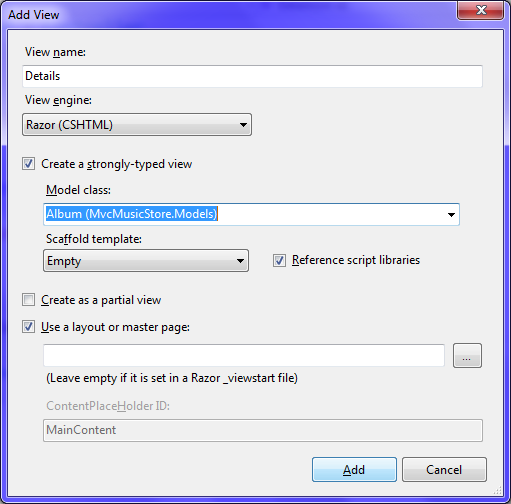
{

var album = new Album { Title = "Album " + id };

return View(album);

}

1. Add a view for the **Details()** action method. In the **Add View** dialog, check the **Create a strongly-typed view** option & choose the model class as **Album** from the dropdownlist as shown below:



1. Modify the contents of the **<h2>** element inside **Details.cshtml** as follows:

<h2>Album: @Model.Title</h2>

1. Run the application by pressing **F5.** Browse to the following URL & observe the results:

**/Store/Details/5**

1. Modify the **Browse()** action method of the **StoreController** as follows:

public ActionResult Browse(string genre)

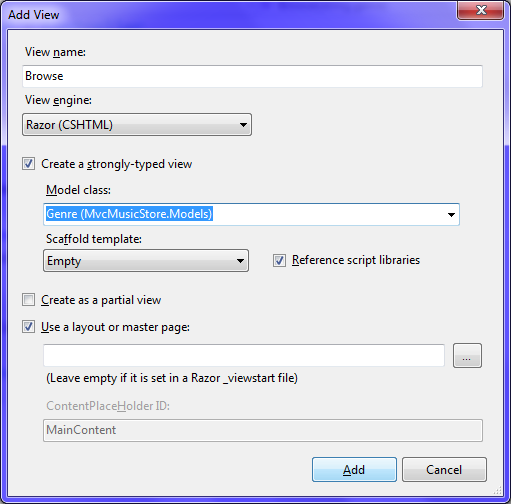
{

var genreModel = new Genre { GenreName = genre };

return View(genreModel);

}

1. Add a view for the **Browse()** action method. In the **Add View** dialog, check the **Create a strongly-typed view** option & choose the model class as **Genre** from the dropdownlist as shown below:



1. Modify the contents of the **<h2>** element inside **Browse.cshtml** as follows:

<h2>Browsing Genre: @Model.GenreName</h2>

1. Run the application by pressing **F5.** Browse to the following URL & observe the results:

**/Store/Browse?genre=Disco**

**When the user visits the Home page of the site, which in this case is the *Index()* action method, let’s modify the *Index()* action method so that it now returns a list of all genres in the store.**

1. Modify the **Index()** action method of the **StoreController** as follows:

public ActionResult Index()

{

var genres = new List<Genre>()

{

new Genre{GenreName="Disco"},

new Genre{GenreName="Rock"},

new Genre{GenreName="Jazz"},

};

return View(genres);

}

1. Add a view for the **Index()** action method. In the **Add View** dialog, check the **Create a strongly-typed view** option & choose the model class as **Genre** from the dropdownlist as mentioned in step 10.

1. Since this view is not using a single Genre, but a list of genres, specify the type accordingly by replacing the first line in the view with the following:

@model IEnumerable<MvcMusicStore.Models.Genre>

1. Delete the existing code below the razor block which contains **ViewBag.Title** statement.
2. Add the following code below the razor block to display the list of genres:

<h3>Browse Genres</h3>

<p>

Select from @Model.Count() genres:</p>

<ul>

@foreach (var genre in Model)

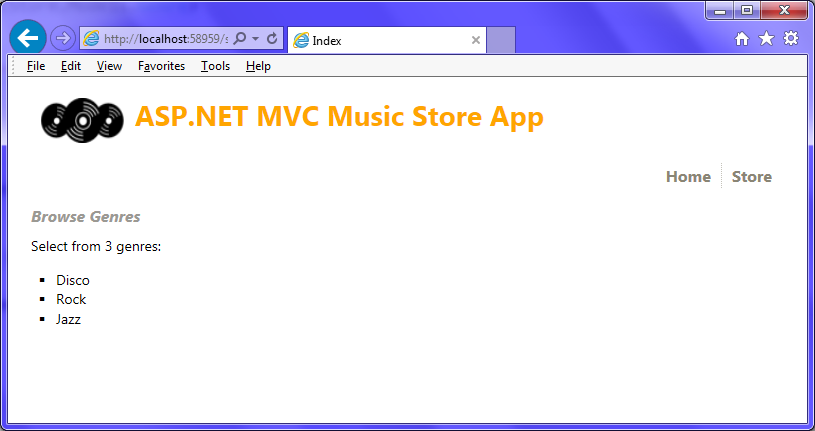
{

<li>@genre.GenreName</li>

}

</ul>

1. Run the application by pressing **F5.** A list of all genres must be displayed as shown below:



**Part 7: Adding links between views**

**In the above case, the genre names are displayed as plain text. Ideally, each genre name must be linked to the appropriate */Store/Browse* URL, so that clicking on a genre like *Disco* will navigate to */Store/Browse?genre=Disco*.**

1. Delete the existing code inside the **foreach** loop inside **Index.cshtml** view of the **StoreController.**
2. Add the following code inside the **foreach** loop:

<li>

@Html.ActionLink(genre.GenreName, "Browse", new { genre =

genre.GenreName })

</li>

1. Run the application by pressing **F5.** Observe that now the genre names are displayed as links. Also, clicking on a genre name navigates to the **Browse()** action with the genre name passed as a query string (see the address bar of the browser).