ASP.NET MVC Online Book Store Tutorial

Abhishek

This document is an unofficial printable version of ASP.NET MVC 3 tutorial written by Abhishek Goenka. This document is free to be shared with the developer community. It's provided "as is" without warranty of any kind. Enjoy it!

Prepared by Abhishek Goenka

Version- 1.0

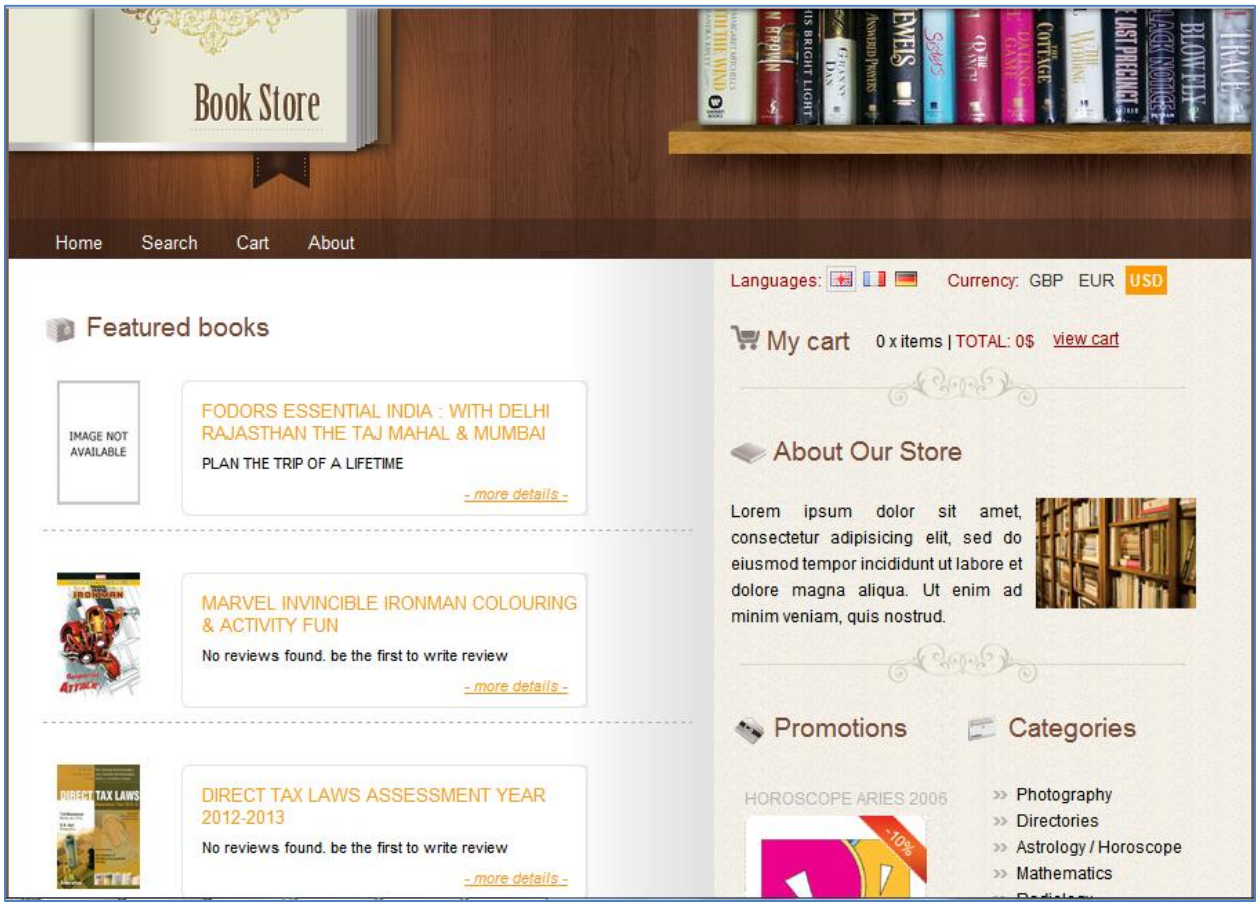
+91-9008719714

Overview

This tutorial explains step by step process of creating basic ASP.NET MVC application. This tutorial contains screenshot and videos whenever – wherever needed.

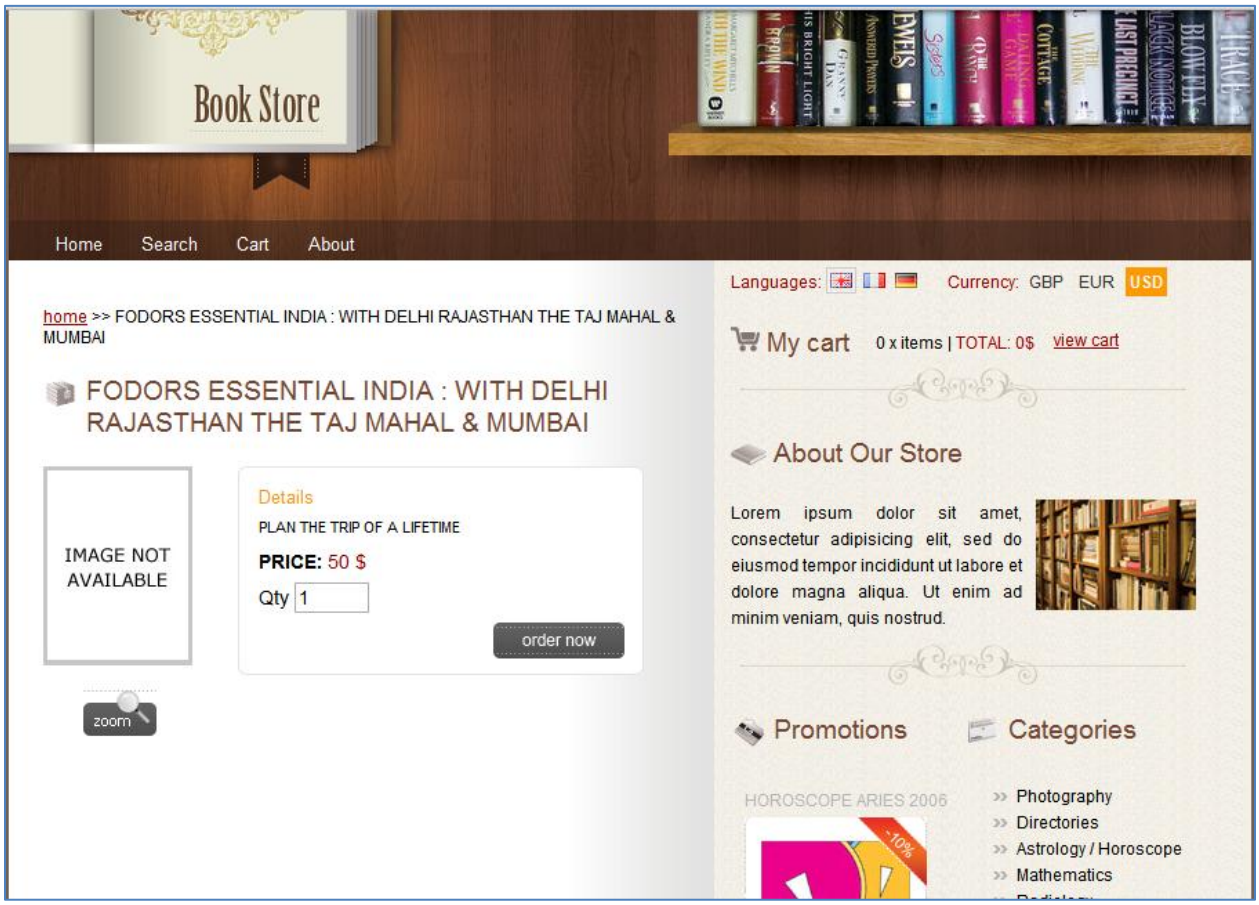
This application mainly focuses on product searching and shopping.

By default application loads with HOME page



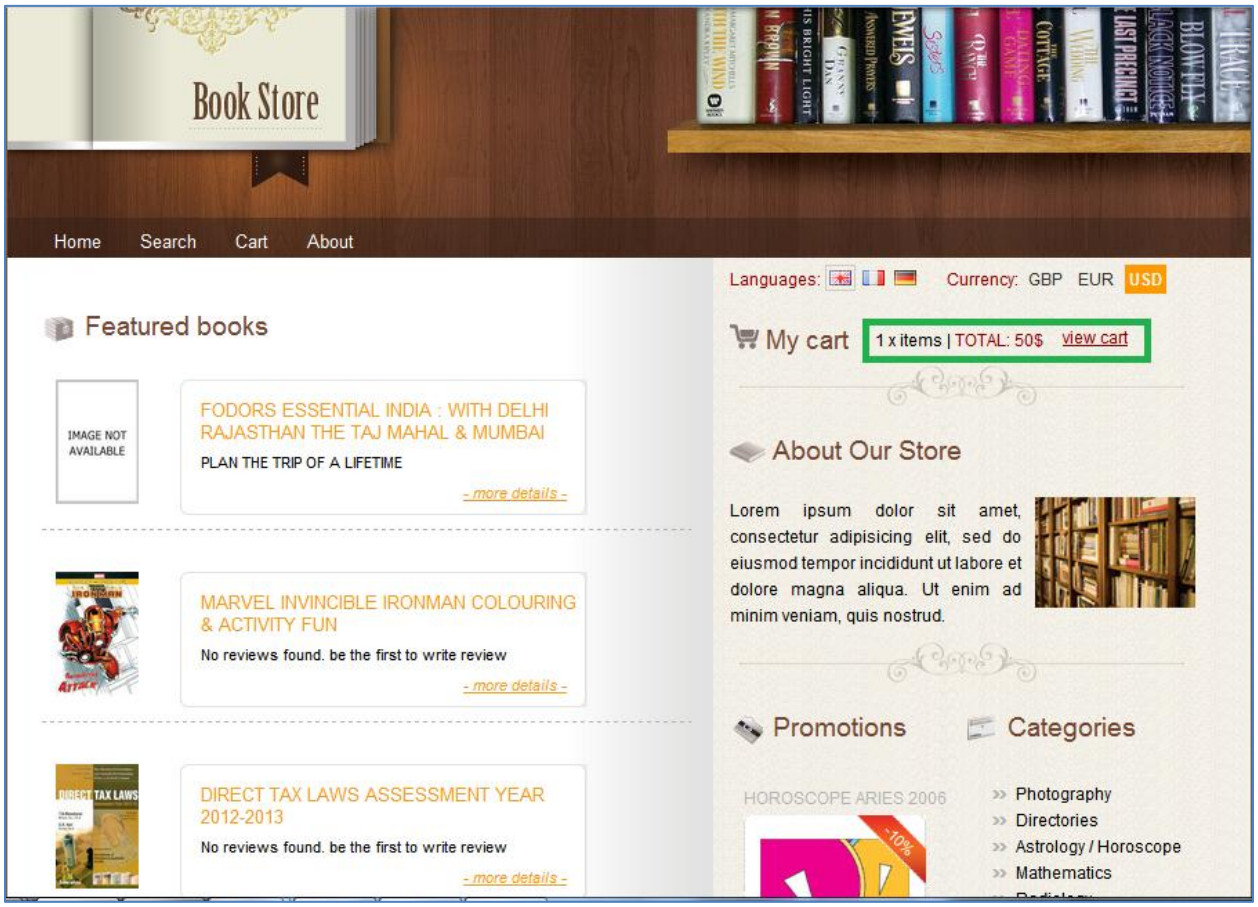
Email – mailtoagoenka@gmail.com

To see more details click on IMAGE or ‘more details’ link



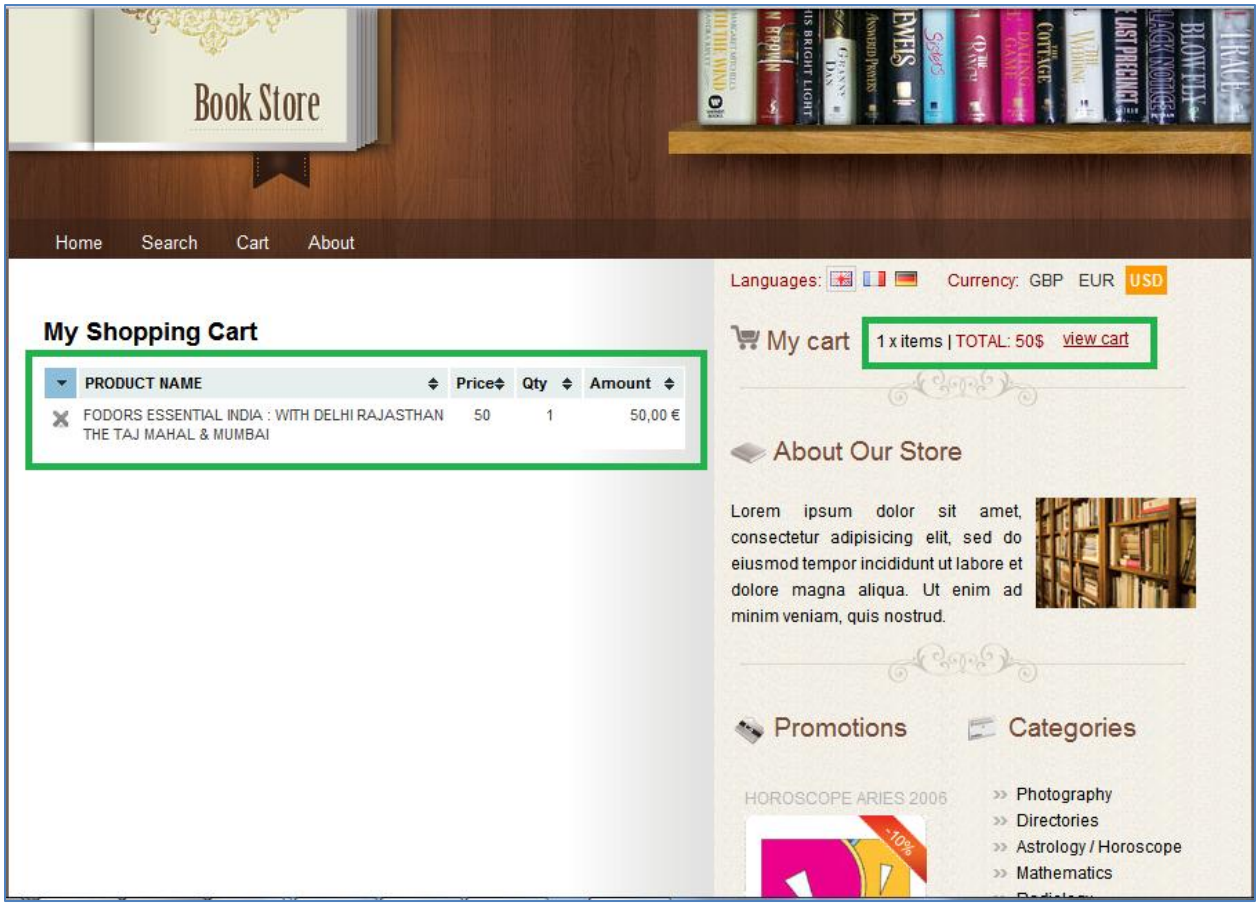
Email – mailtoagoenka@gmail.com

Click Order Now to place order



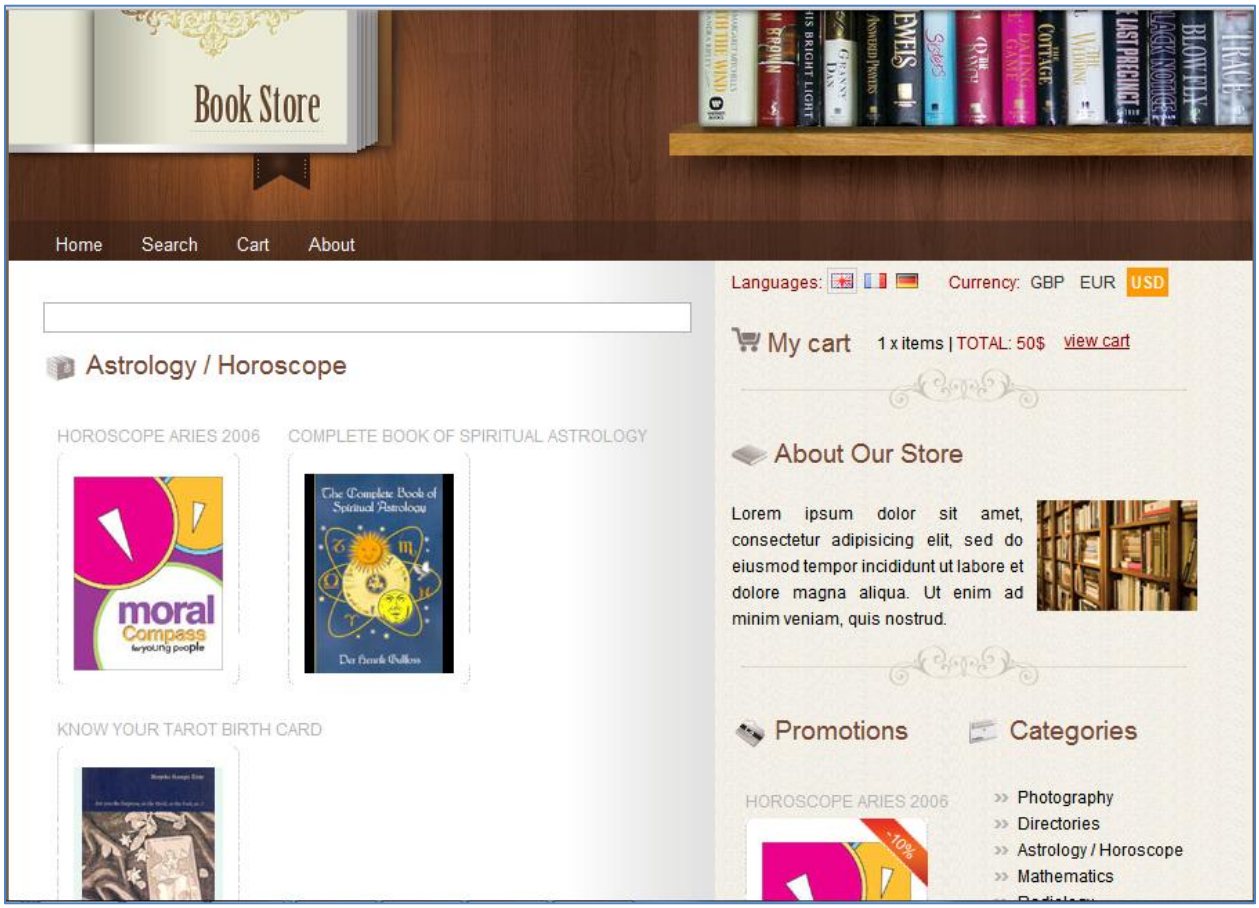
Email – mailtoagoenka@gmail.com

Cart info is updated on MAIN page and CART can be opened by “View Cart” (from mail screen) or “Cart” link



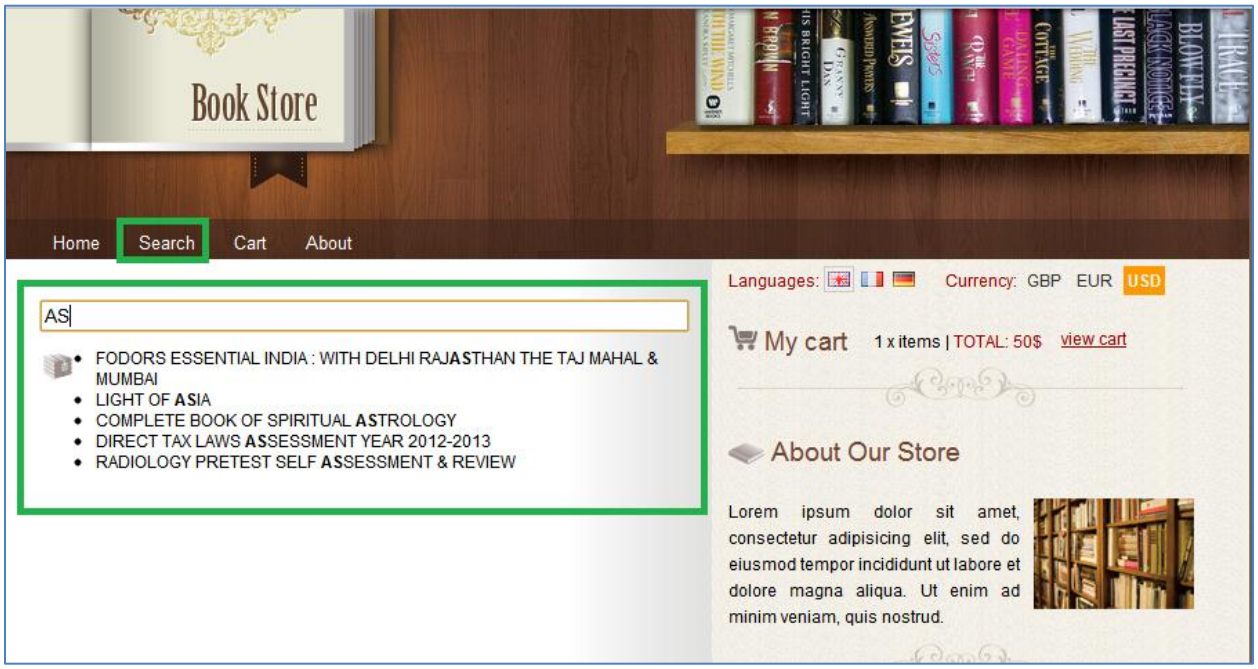
Email – mailtoagoenka@gmail.com

Select Category from Categories. This will show all the books of selected Category



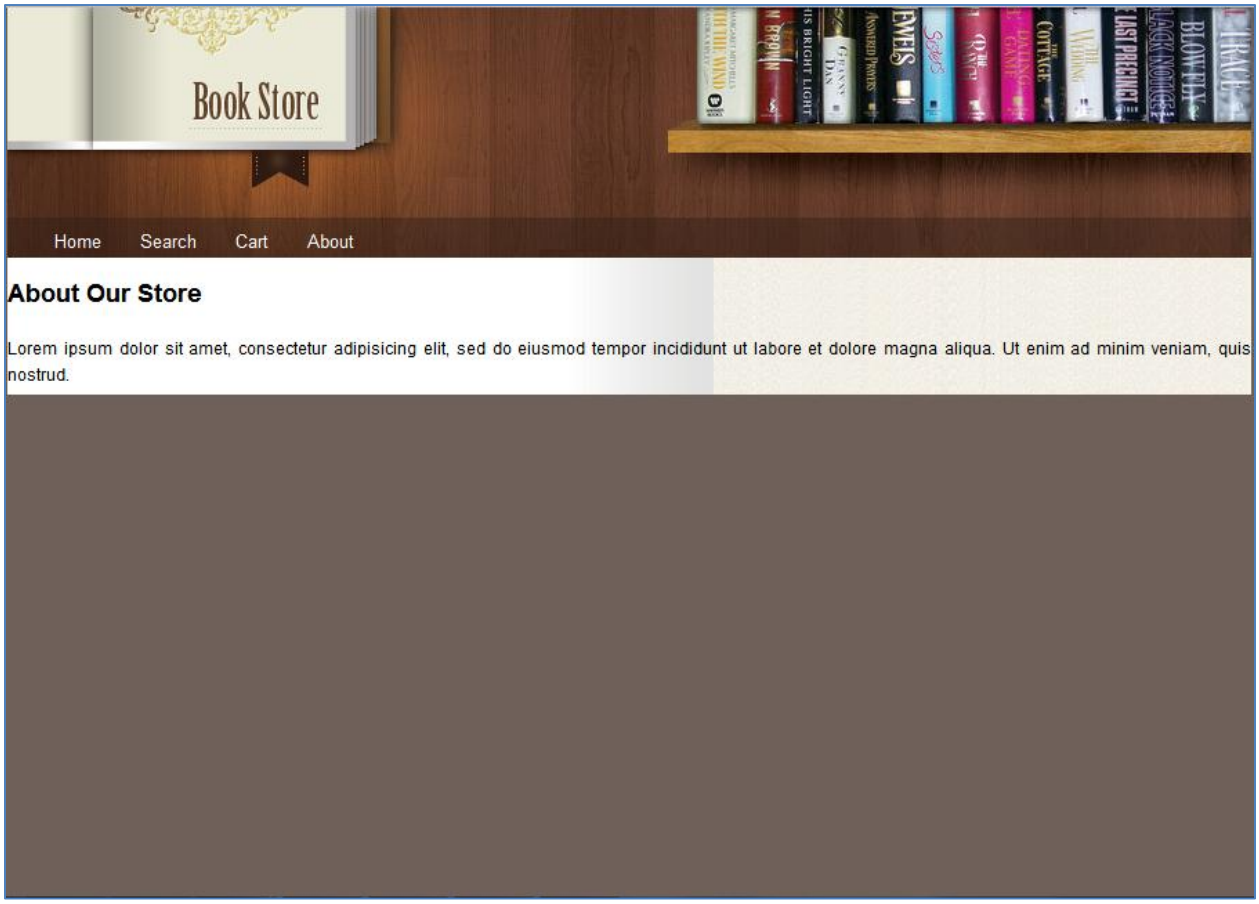
Email – mailtoagoenka@gmail.com

Click Search to search any BOOK



Email – mailtoagoenka@gmail.com

Select ABOUT to get more information about site – although this doesn't contain any meaningful text. Also you will notice that ABOUT doesn't contain right panel



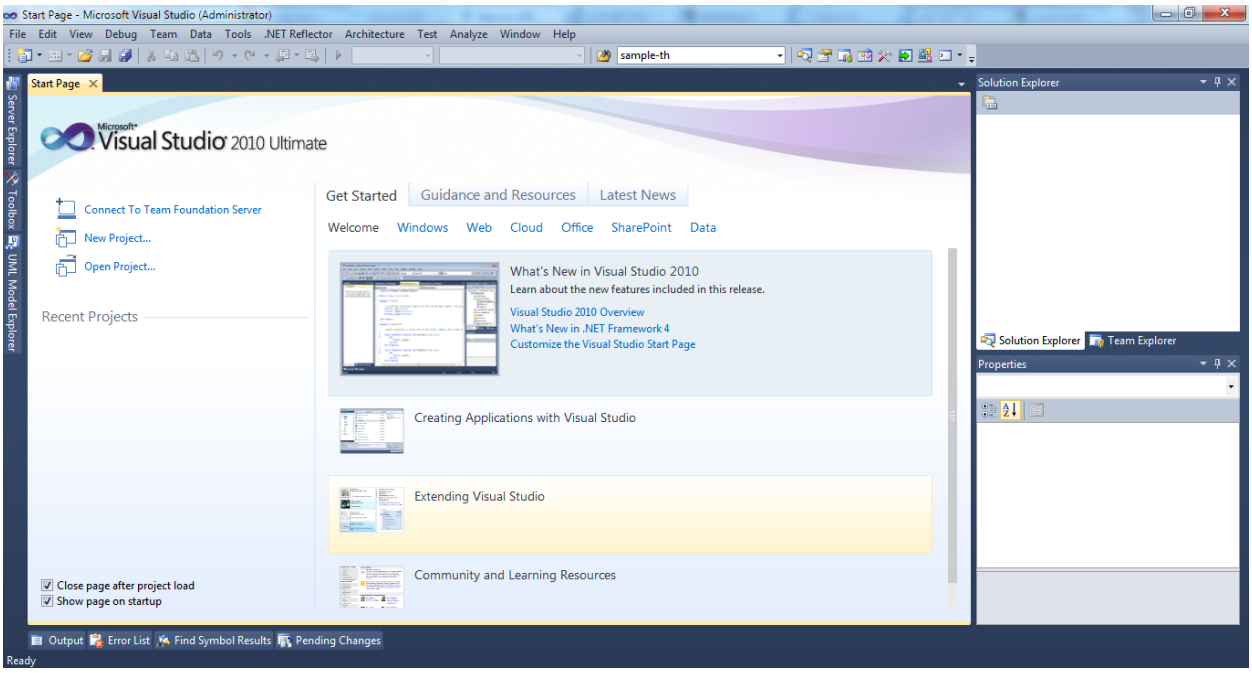
Getting started with MVC3

By default Visual Studio 2010 comes with MVC2. Download MVC3 framework if not already installered. ASP.NET MVC 3 RTM can be downloaded from –
http://www.microsoft.com/download/en/details.aspx?id=4211

Email – mailtoagoenka@gmail.com

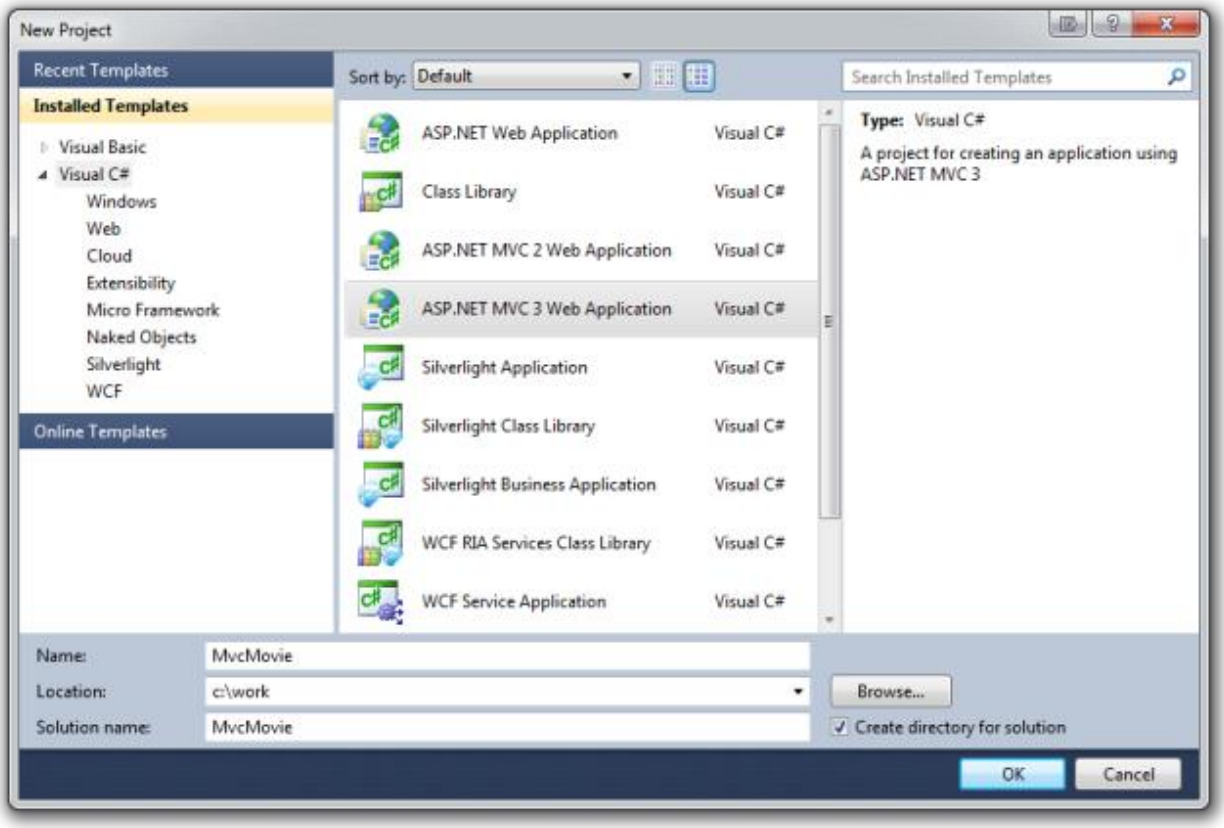
Getting Started

Start by running Visual Web Developer 2010 Express ("Visual Web Developer" for short) and select New Project from the Start page.



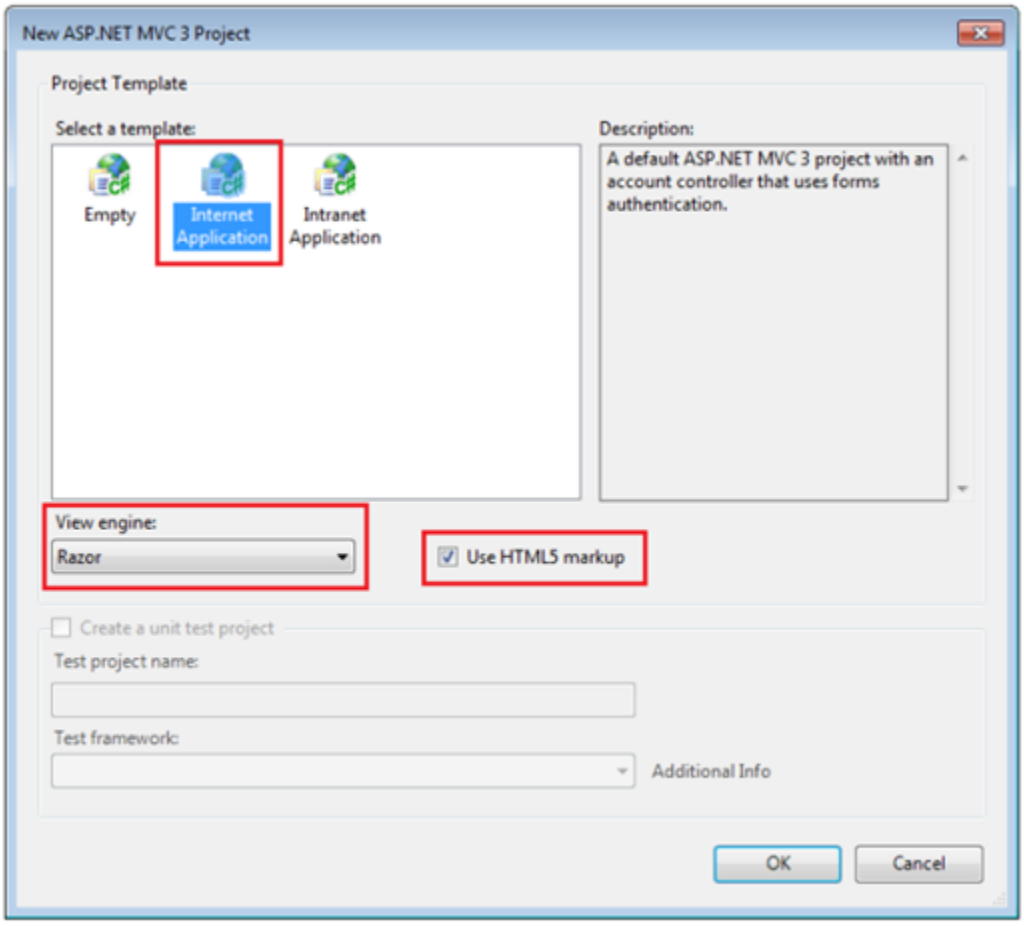
Email – mailtoagoenka@gmail.com

You can create applications using either Visual Basic or Visual C# as the programming language. Select Visual C# on the left and then select ASP.NET MVC 3 Web Application. Name your project "BookStore" and then click OK



Email – mailtoagoenka@gmail.com

In the New ASP.NET MVC 3 Project dialog box, select Internet Application. Check Use HTML5 markup and leave Razor as the default view engine.



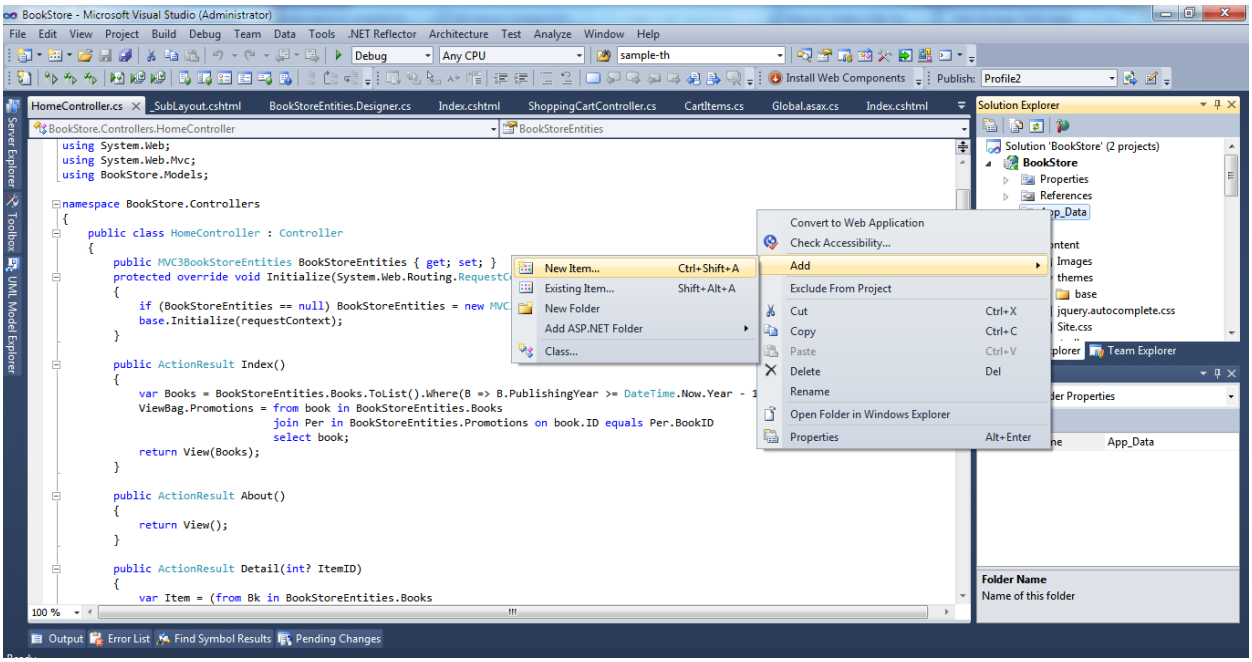
Click **OK**. Visual Studio used a default template for the ASP.NET MVC project you just created, so you have a working application right now without doing anything! This is a simple "Hello World!" project, and it's a good place to start your application.

Watch VIDEO - http://www.youtube.com/watch?v=8h-dXm-wsJE

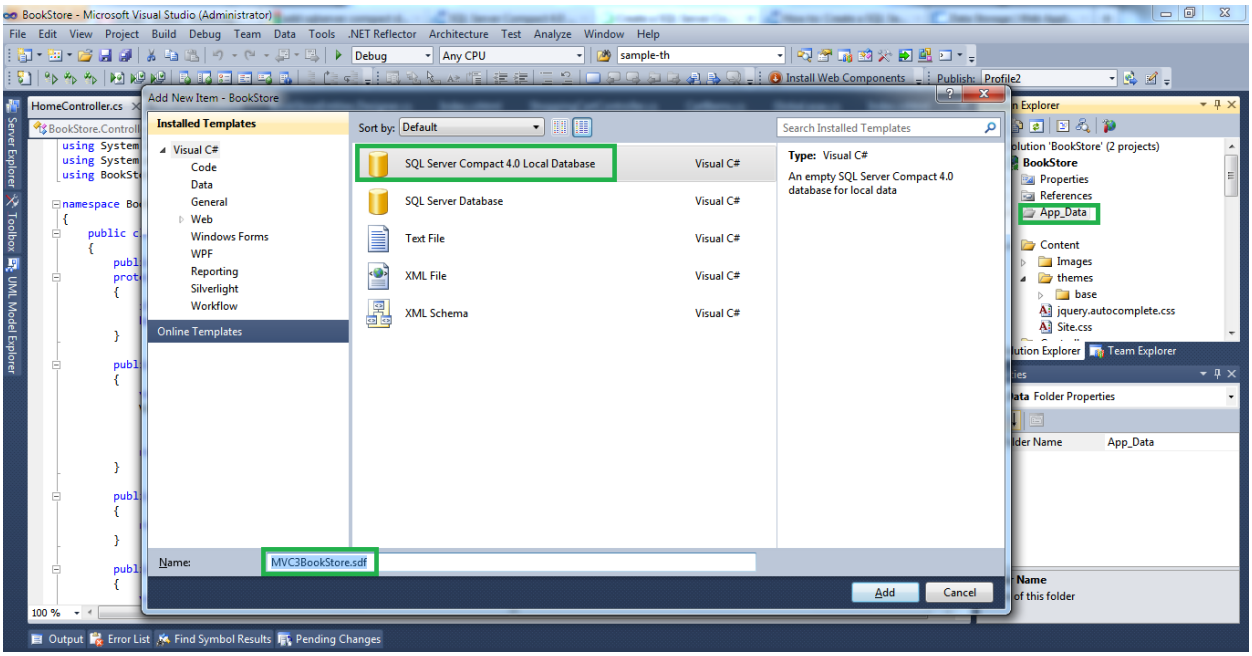
Email – mailtoagoenka@gmail.com

Creating DATABASE

Right click “**App_Data**” and select “**New Item**”



Select “SQL Server Compact 4.0 Local Database” and give some meaningful name.

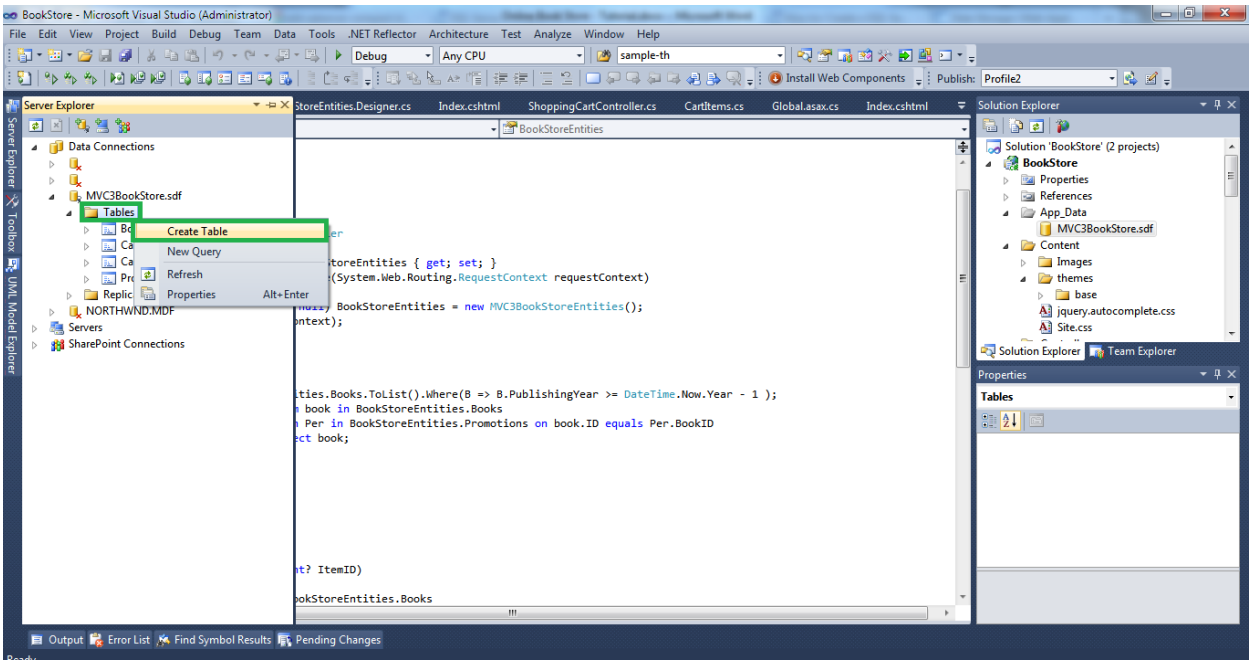


[Download “SQL Server Compact 4.0”] if not exists - [http://www.microsoft.com/web/gallery/install.aspx?appid=SQLCE;SQLCEVSTools_4_0]

Email – mailtoagoenka@gmail.com

Creating DATABASE TABLES

GOTO solution explorer and select Table. Right click and select "**Create Table**"



Database Structure

Table – BOOK

Column Name	Data Type	Length	Allow Nulls	Unique	Primary Ke
BookCode	nvarchar	100	No	Yes	No
BookName	nvarchar	100	No	No	No
ISBN	nchar	13	No	No	No
Binding	nvarchar	100	Yes	No	No
PublishingYear	int	4	Yes	No	No
Language	nvarchar	50	Yes	No	No
CategoryID	int	4	No	No	No
Description	nvarchar	500	Yes	No	No
ID	int	4	No	Yes	Yes
Publisher	nvarchar	100	Yes	No	No
Price	int	4	Yes	No	No

Email – mailtoagoenka@gmail.com

Category

Column Name	Data Type	Length	Allow Nulls	Unique	Primary Key
ID	int	4	No	No	Yes
CategoryName	nvarchar	100	No	Yes	No

Promotion

Column Name	Data Type	Length	Allow Nulls	Unique	Primary Key
ID	int	4	No	No	Yes
BookID	int	4	No	No	No

CART

Column Name	Data Type	Length	Allow Nulls	Unique	Primary Key
ID	int	4	No	No	Yes
CartID	nvarchar	4000	Yes	No	No
BookID	int	4	No	No	No
Qty	int	4	No	No	No
DateCreated	datetime	8	Yes	No	No
Amount	int	4	Yes	No	No

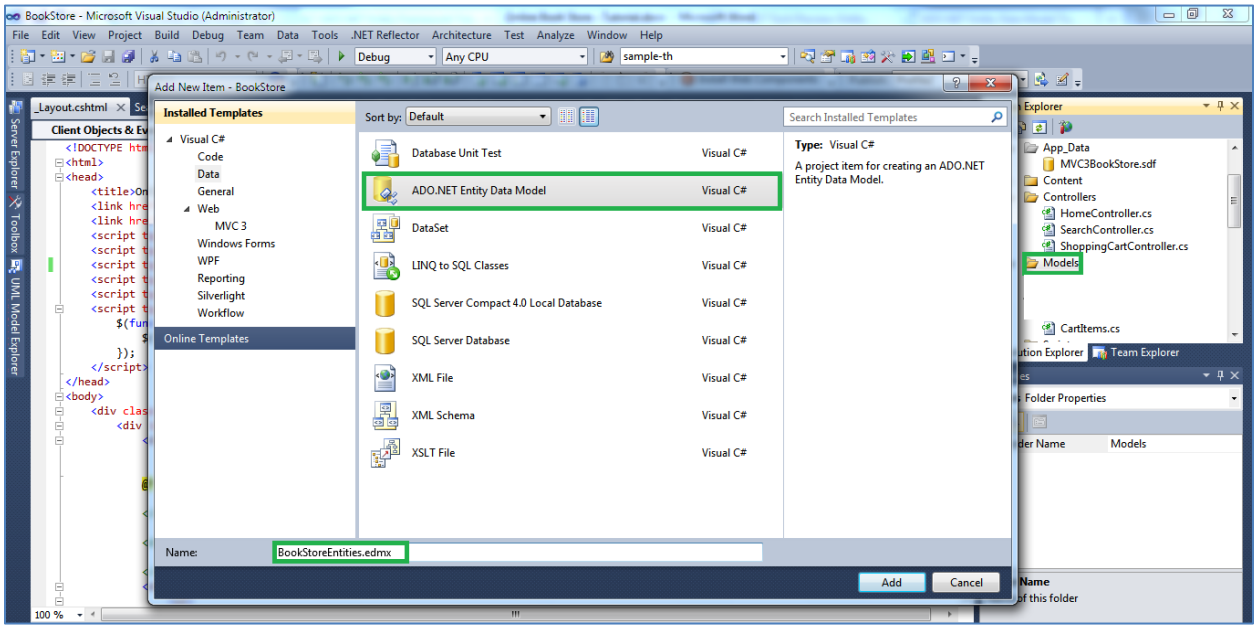
Email – mailtoagoenka@gmail.com

ADO.NET Entity Framework

The ADO.NET Entity Framework enables developers to create data access applications by programming against a conceptual application model instead of programming directly against a relational storage schema. The goal is to decrease the amount of code and maintenance required for data-oriented applications.

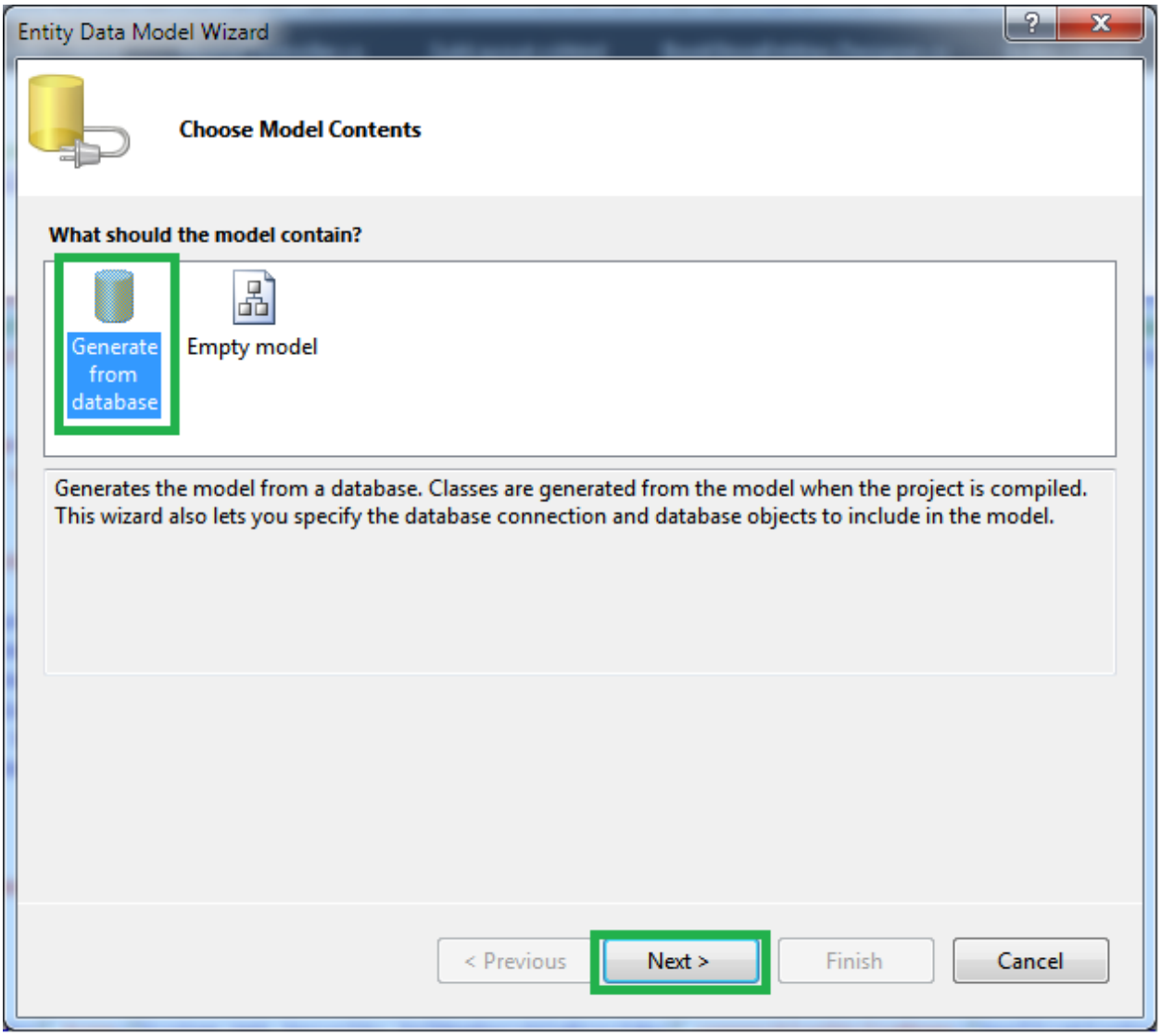
Adding ADO.NET Entity Data Model

Select Model folder. Right click and select "**New Item**" option. You will get below screen. Give some meaningful name and click **Add**



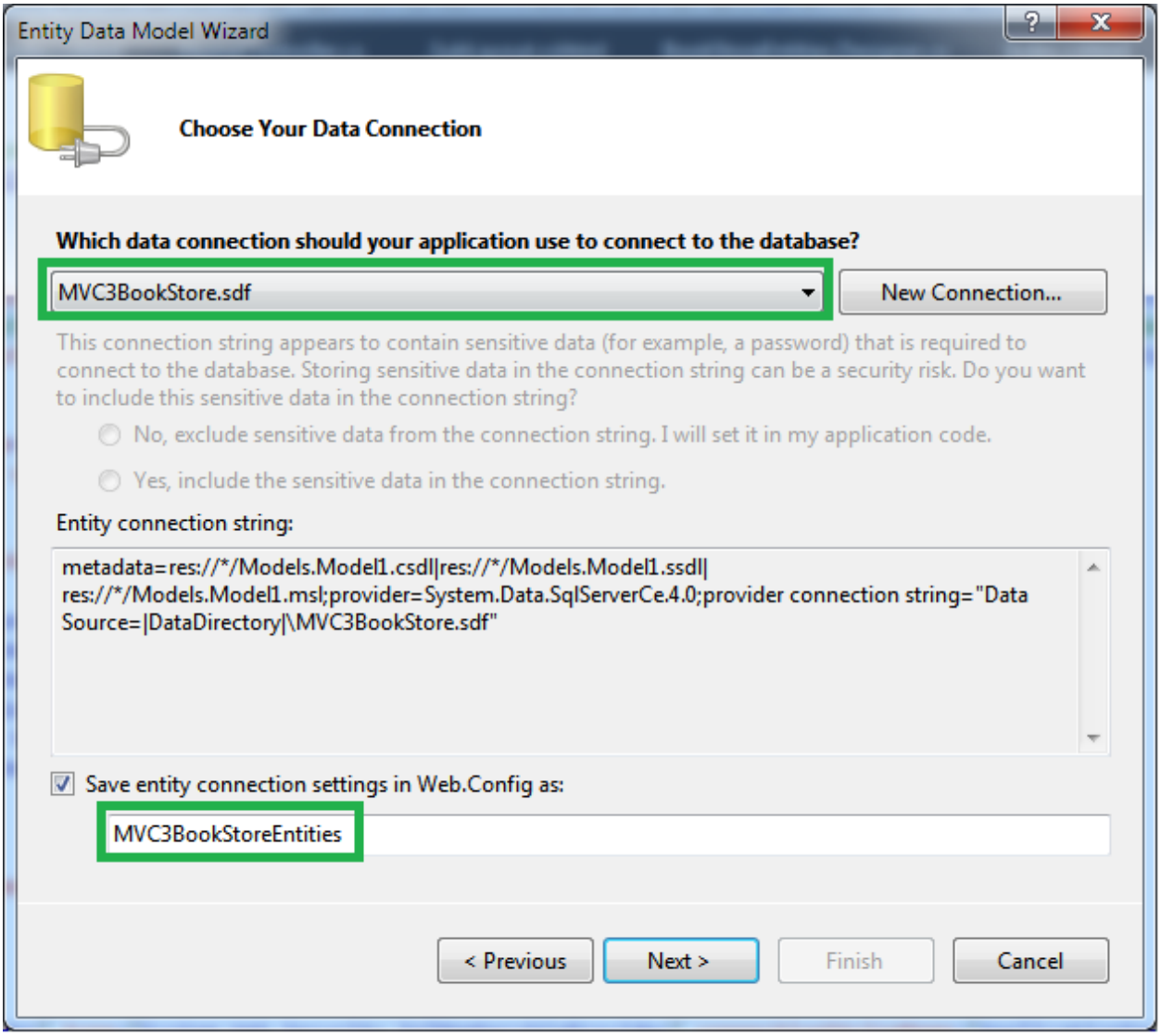
Select "**Generate from Database**" and click **Next**

Email – mailtoagoenka@gmail.com



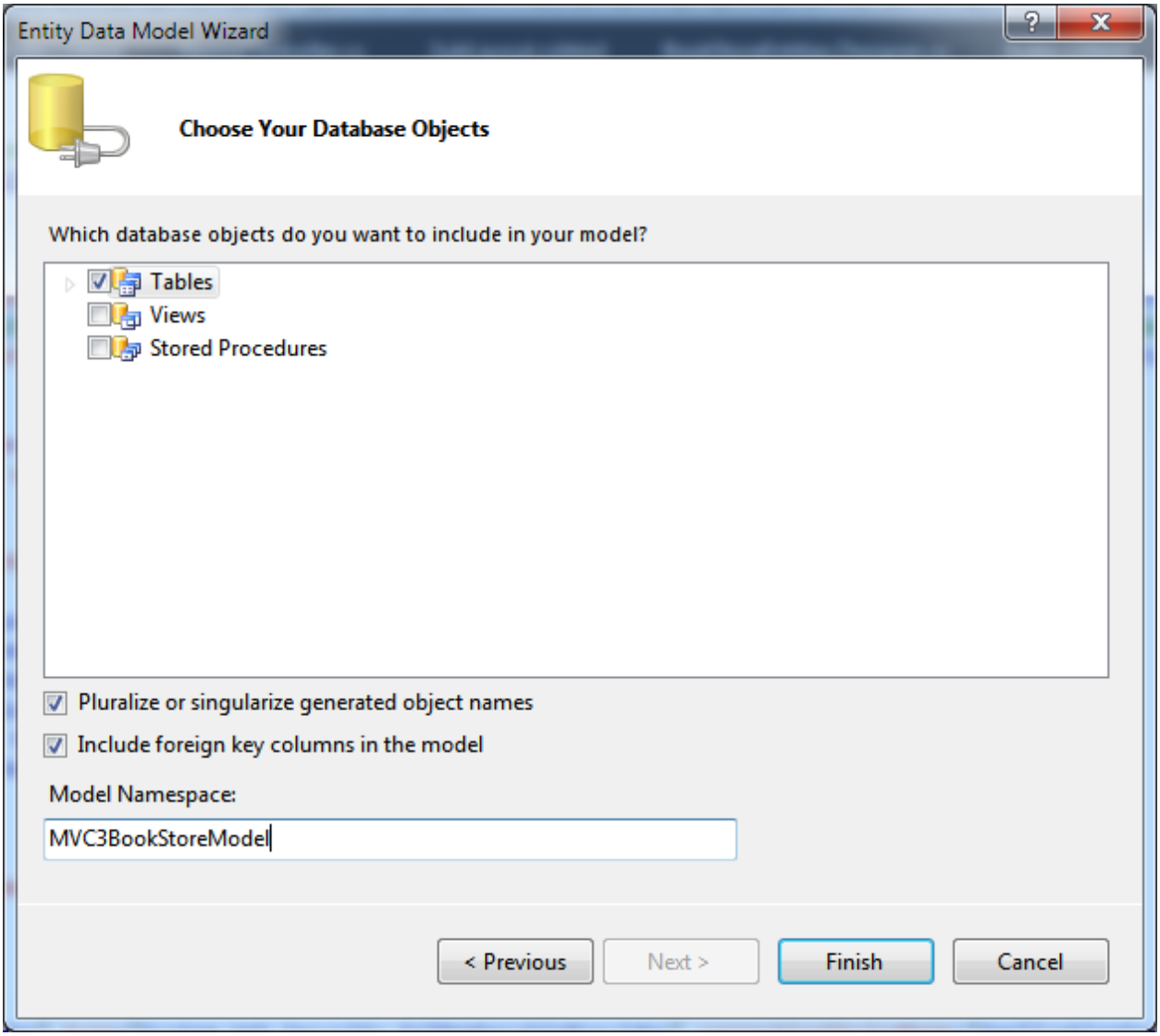
Email – mailtoagoenka@gmail.com

Select the database that we created in previous step. Give some meaningful name to "**Entity Connection String**"

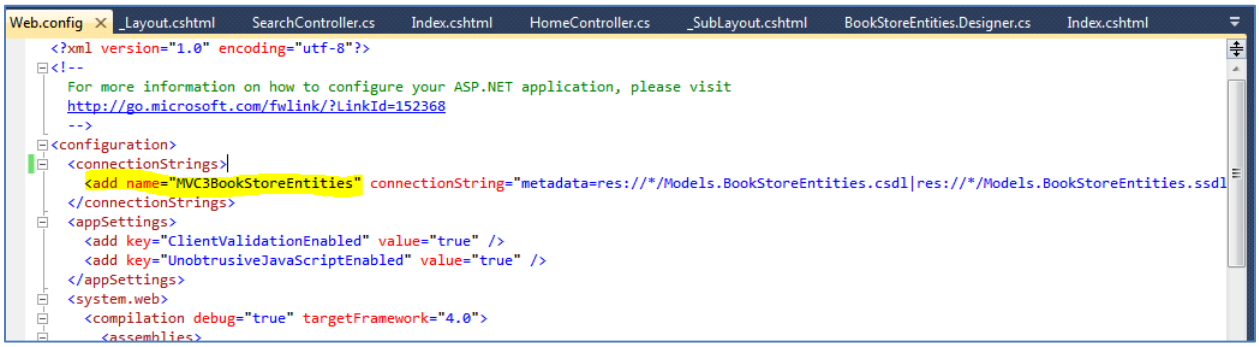


Email – mailtoagoenka@gmail.com

Select **Tables** checkbox and click finish



Now our entity framework is ready. New connection string is appended in Web.config file.



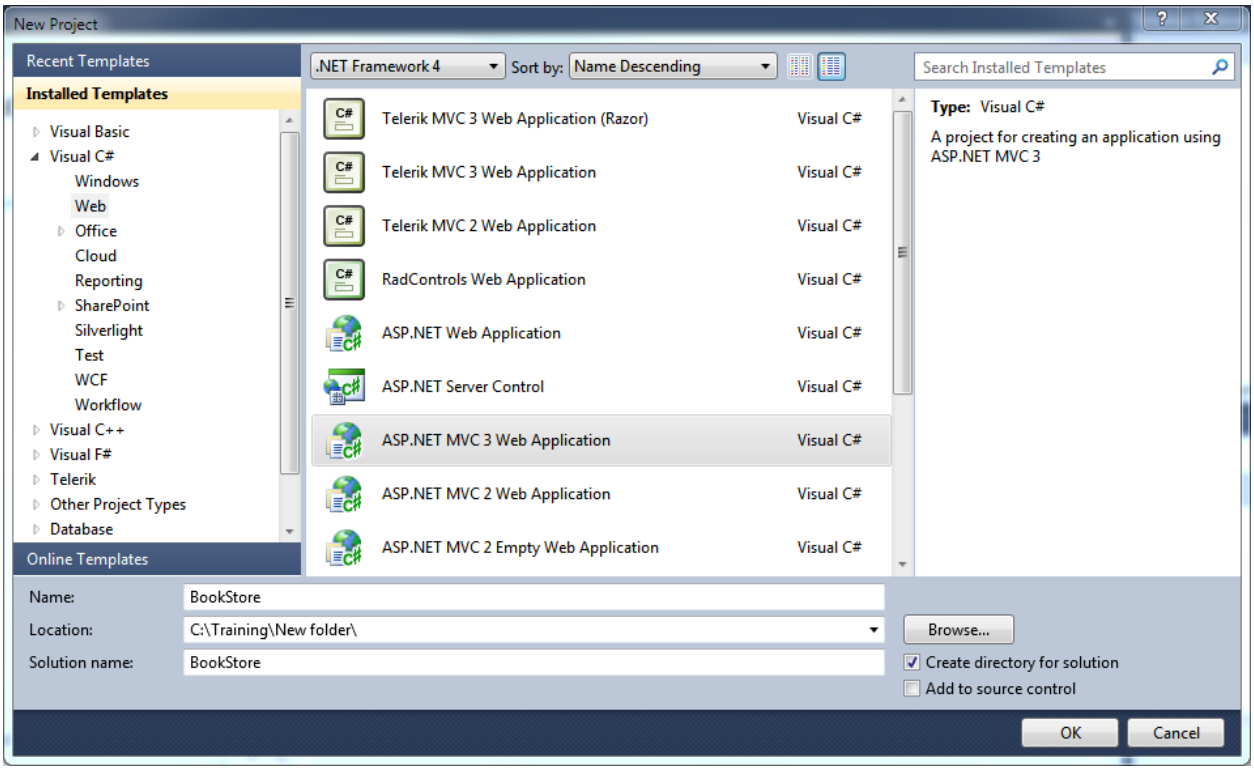
Email – mailtoagoenka@gmail.com

Design Template

Design Template is copied in the same folder where code exists. This template can be used to create new application from scratch.

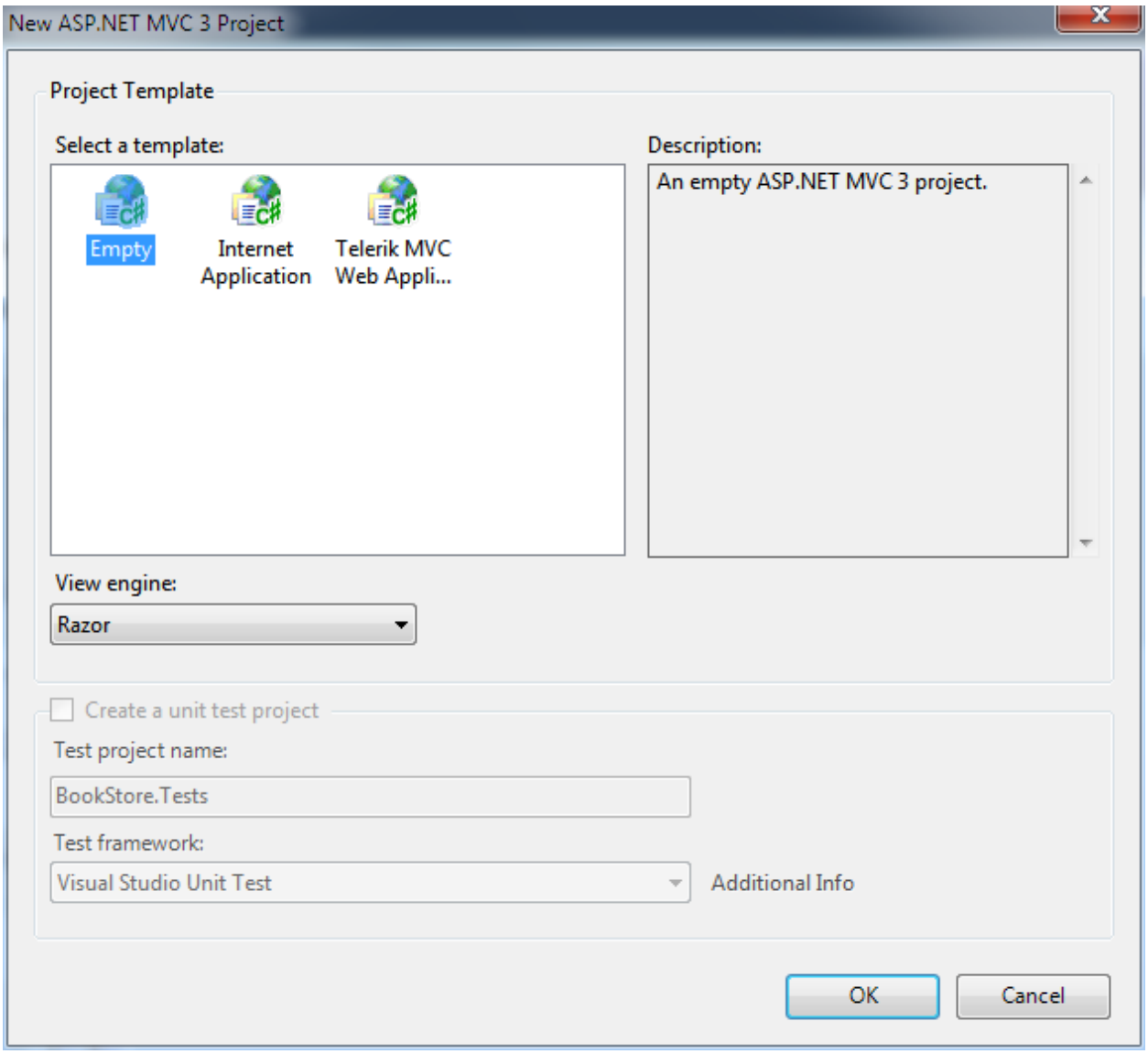
Creating a new ASP.NET MVC 3 project

Open Visual Studio 2010 and create a new project using the ASP.NET MVC 3 Web Application template. Name the application "BookStore".



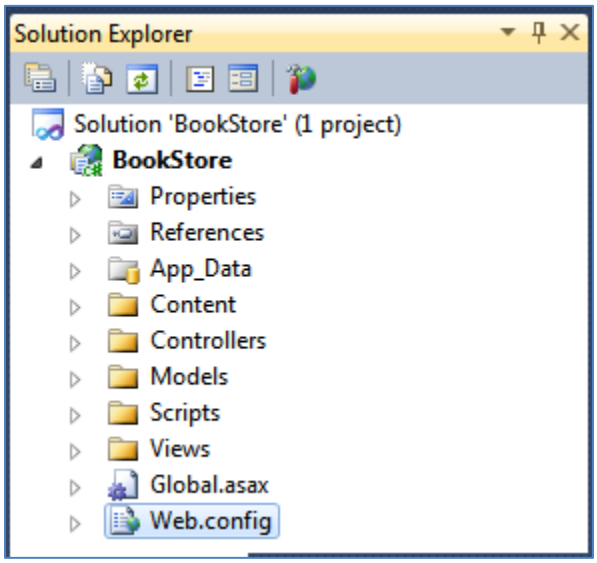
Email – mailtoagoenka@gmail.com

In the New ASP.NET MVC 3 Project dialog, select **Empty Application**, select the **Razor view engine**, and then click OK.



Email – mailtoagoenka@gmail.com

The Empty MVC 3 template isn't completely empty – it adds a basic folder structure:



Now look at the directory structures of the newly created application. You will find three main directories ('Models', 'Views' and 'Controllers') there, which contains their corresponding files. Besides, there is a directory named 'Scripts', which contains some already added JavaScript libraries (including jQuery) and a directory named 'Contents', which may contains style sheets and other static files. And sure, you can add other directories also. However, it is recommended that, if you are going to use something big to be integrated here, better to do those tasks on a separate project and add their references on this project.

Adding a Controller

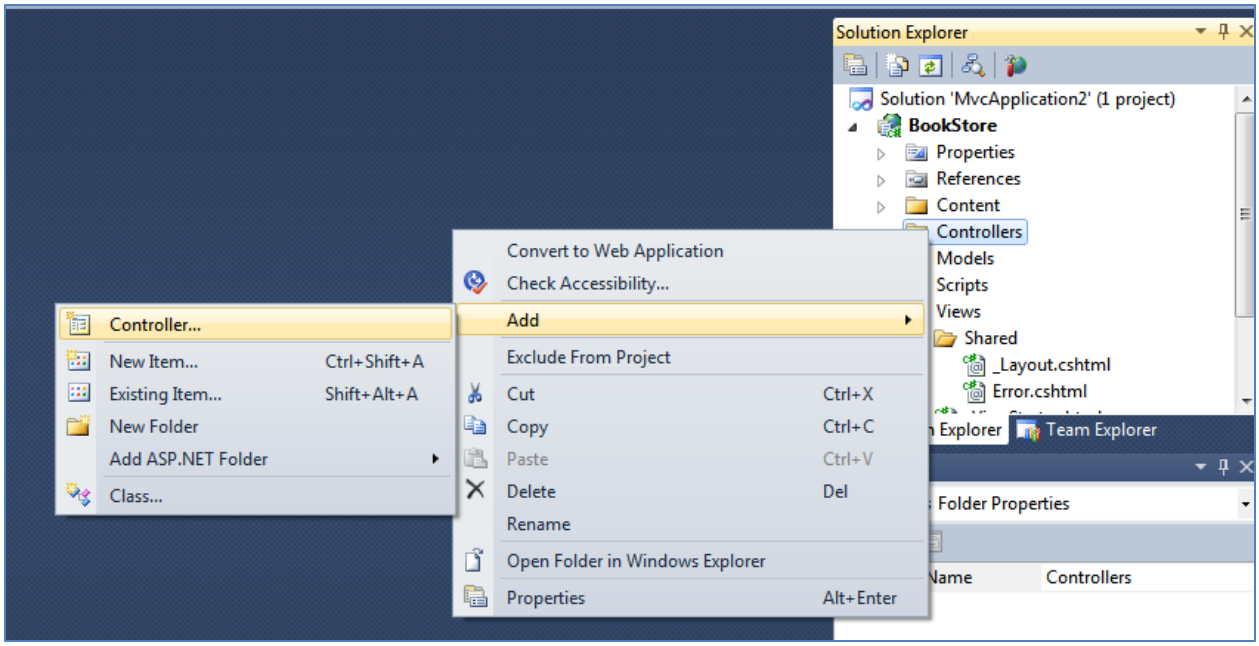
MVC stands for model-view-controller. MVC is a pattern for developing applications that are well architected and easy to maintain. MVC-based applications contain:

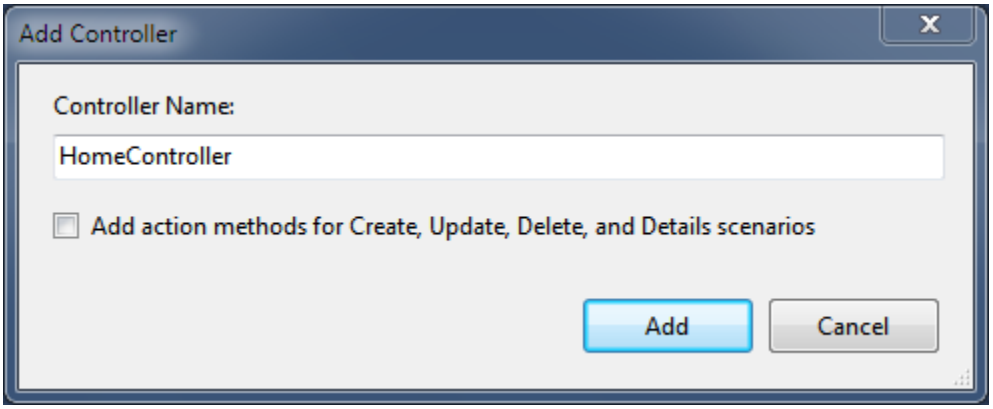
- Controllers: Classes that handle incoming requests to the application, retrieve model data, and then specify view templates that return a response to the client.
- Models: Classes that represent the data of the application and that use validation logic to enforce business rules for that data.
- Views: Template files that your application uses to dynamically generate HTML responses.

We'll be covering all these concepts in this tutorial series and show you how to use them to build an application.

Email – mailtoagoenka@gmail.com

Let's begin by creating a controller class. In **Solution Explorer**, right-click the Controllers folder and then **select Add Controller**.





Notice in Solution Explorer that a new file has been created named HomeController.cs. The file is open in the IDE.

Email – mailtoagoenka@gmail.com

HomeController.cs

BookStore.Controllers.HomeController | Index()

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace BookStore.Controllers
{
    public class HomeController : Controller
    {
        //
        // GET: /Home/

        public ActionResult Index()
        {
            return View();
        }

    }
}
```

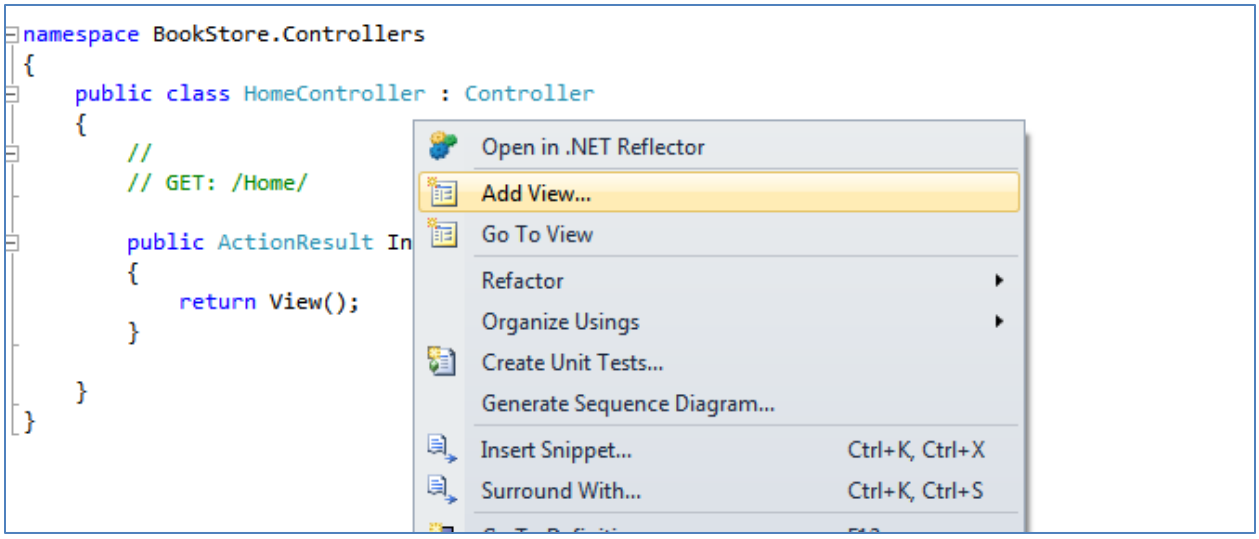
HomeController class is created with one method – **Index()**

Adding a View

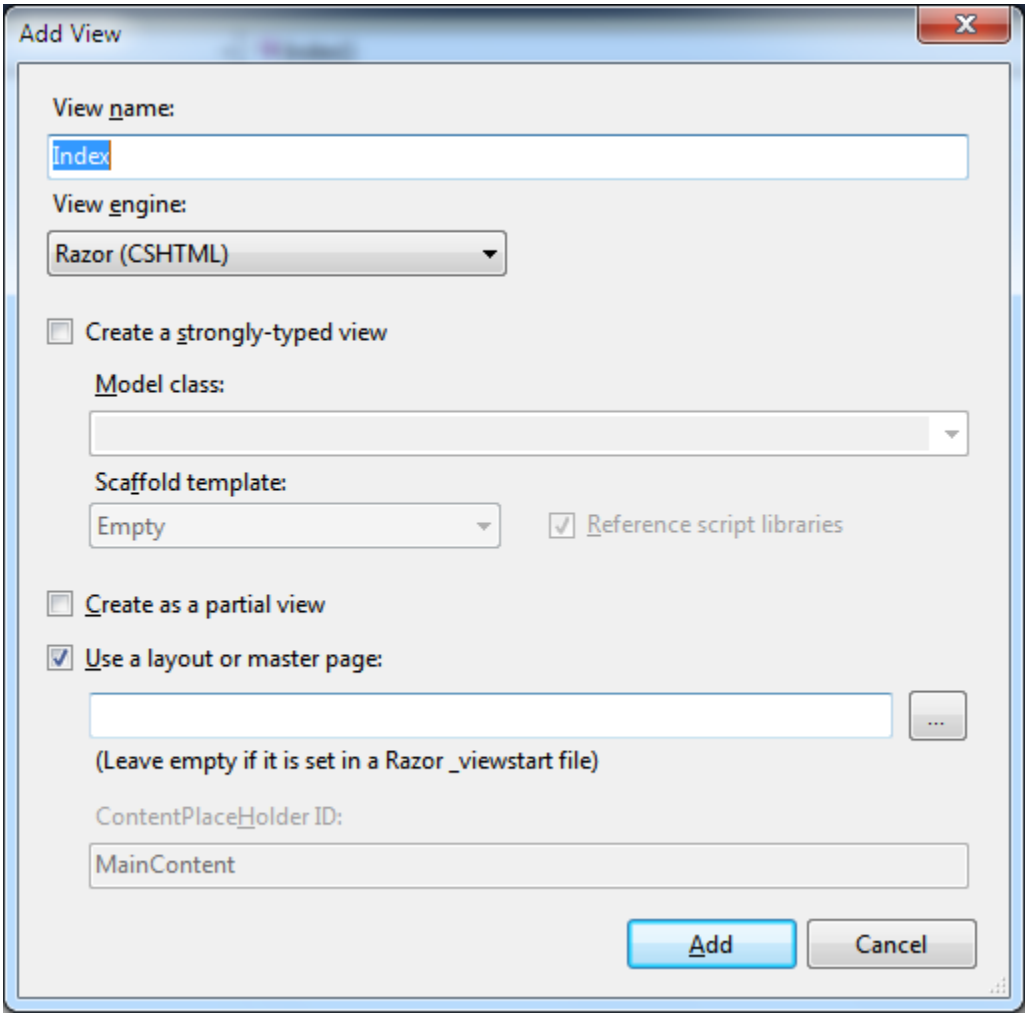
We will create our view template files using the new Razor view engine introduced with ASP.NET MVC 3. Razor-based view templates have a .cshtml file extension, and provide an elegant way to create HTML output using C#. Razor minimizes the number of characters and keystrokes required when writing a view template, and enable a fast, fluid coding workflow.

Let's start by using a view template with the Index method in the HomeController class. This code indicates that we want to use a view template to generate an HTML response to the browser. Let's add a view template to our project that we can we can use with the Index method. To do this, right-click inside the Index method and click Add View. The Add View dialog box appears.

Email – mailtoagoenka@gmail.com

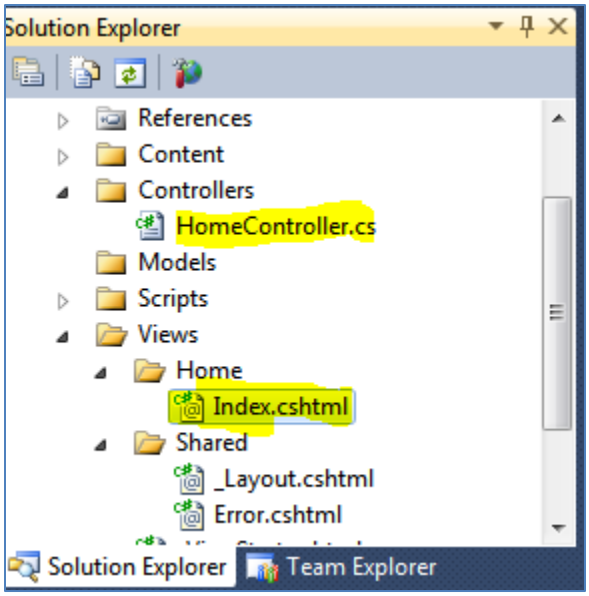


Leave the defaults the way they are and click the **Add** button.

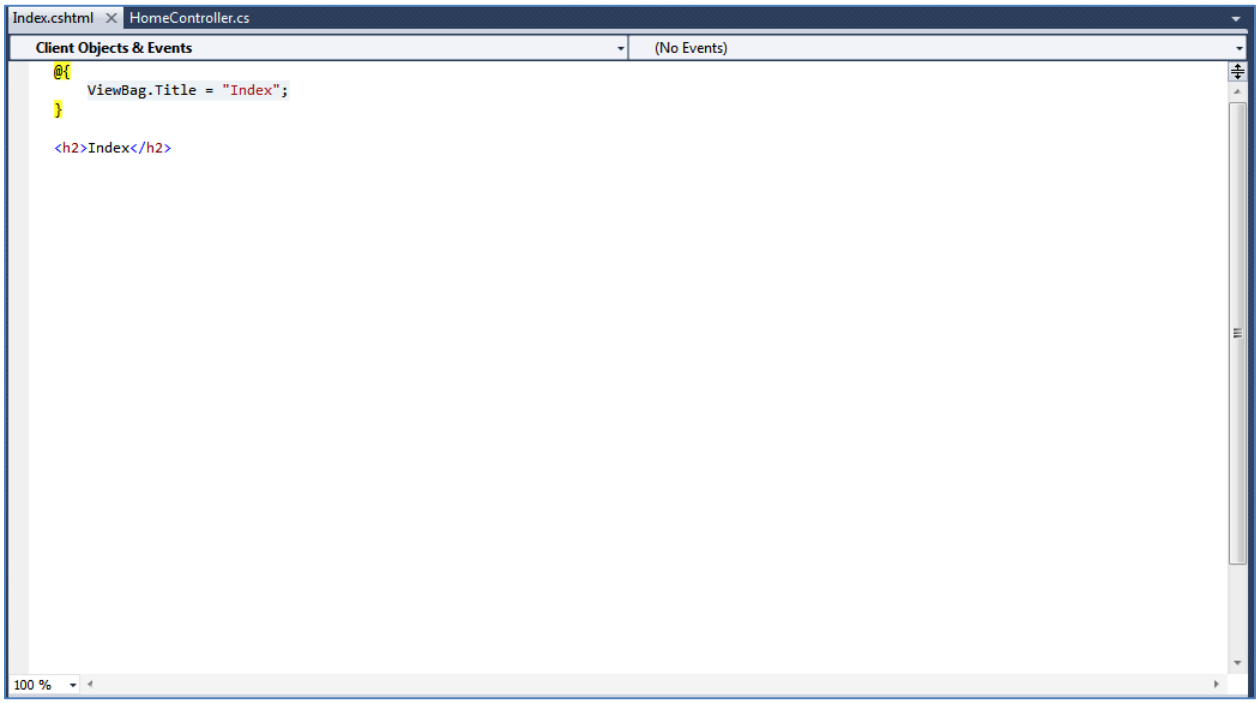


Email – mailtoagoenka@gmail.com

The BookStore\Views\Home folder and the BookStore\Views\Home\Index.cshtml file are created. You can see them in Solution Explorer:



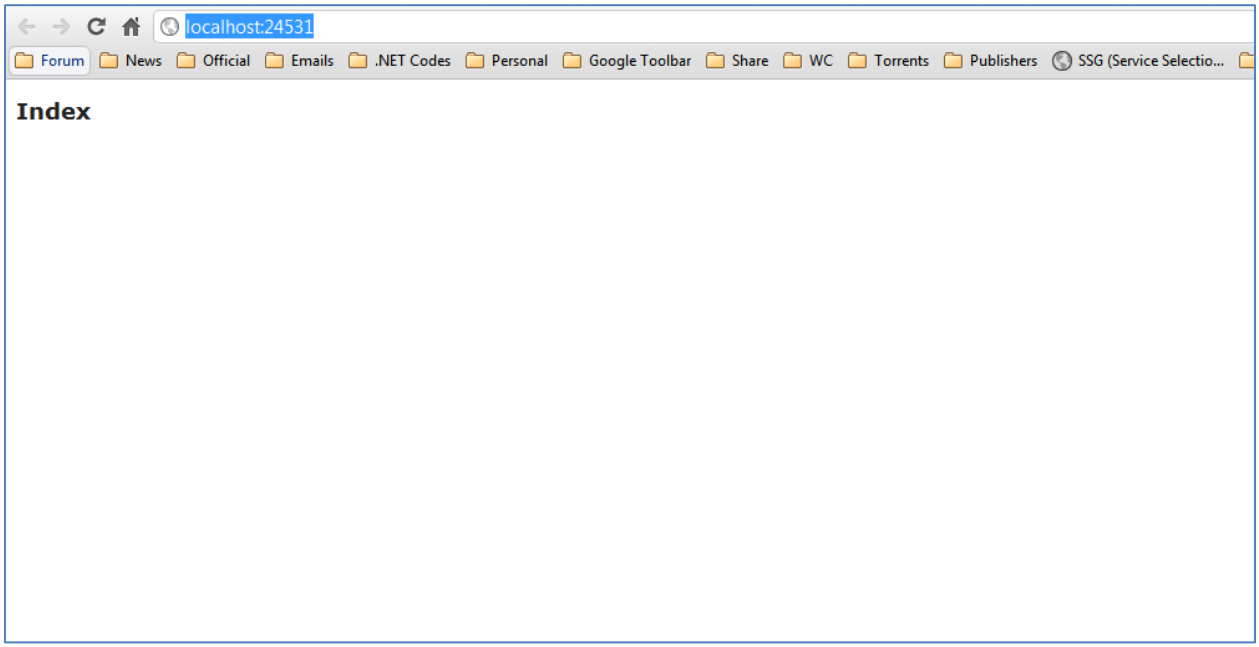
This shows the **Index.cshtml** file that was created:



Run the application. The Index method in your controller didn't do much work; it simply ran the statement return View();, which indicated that we wanted to use a view template file to render a

Email – mailtoagoenka@gmail.com

response to the browser. Because we did not explicitly specify the name of the view template file to use, ASP.NET MVC defaulted to using the Index.cshtml view file within the \Views\Home folder. The image below shows the string hard-coded in the view.



Looks pretty good. However, notice that application doesn’t contains any designs and theme Let's change those.

Changing Views and Layout Pages

First, let's change the title at the top of the page. That text is common to every page. It actually is implemented in only one place in the project, even though it appears on every page in the application. Go to the /Views/Shared folder in **Solution Explorer** and open the **_Layout.cshtml** file. This file is called a layout page and it's the shared "shell" that all other pages use.

Email – mailtoagoenka@gmail.com

_Layout.cshtml

Client Objects & Events | (No Events)

```
<!DOCTYPE html>
<html>
<head>
    <title>Online Book Store : @ViewBag.Title</title>
    <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />
    <link href="@Url.Content("~/Content/jquery.autocomplete.css")" rel="stylesheet" type="text/css" />
    <script type="text/javascript" src="@Url.Content("~/Scripts/jquery-1.4.4.min.js")" ></script>
    <script type="text/javascript" src="@Url.Content("~/Scripts/jquery-latest.js")"></script>
    <script type="text/javascript" src="@Url.Content("~/Scripts/__jquery.tablesorter.min.js")"></script>
    <script type="text/javascript" src="@Url.Content("~/Scripts/jquery.validate.min.js")"></script>
    <script type="text/javascript" src="@Url.Content("~/Scripts/jquery.autocomplete.min.js")"></script>
    <script type="text/javascript">
        $(function () {
            $("#Cart").tablesorter();
        });
    </script>
</head>
<body>
    <div class="wrap">
        <div id="header">
            <div class="logo">
                <a href="">
                    <img src="@Url.Content("~/Content/Images/logo.gif")" alt="" title="" border="0" /></a></div>

            <div id="menu">
                <ul>
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("Search", "Index", "Search")</li>
                    <li>@Html.ActionLink("Cart", "Index", "ShoppingCart")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                </ul>
            </div>
```

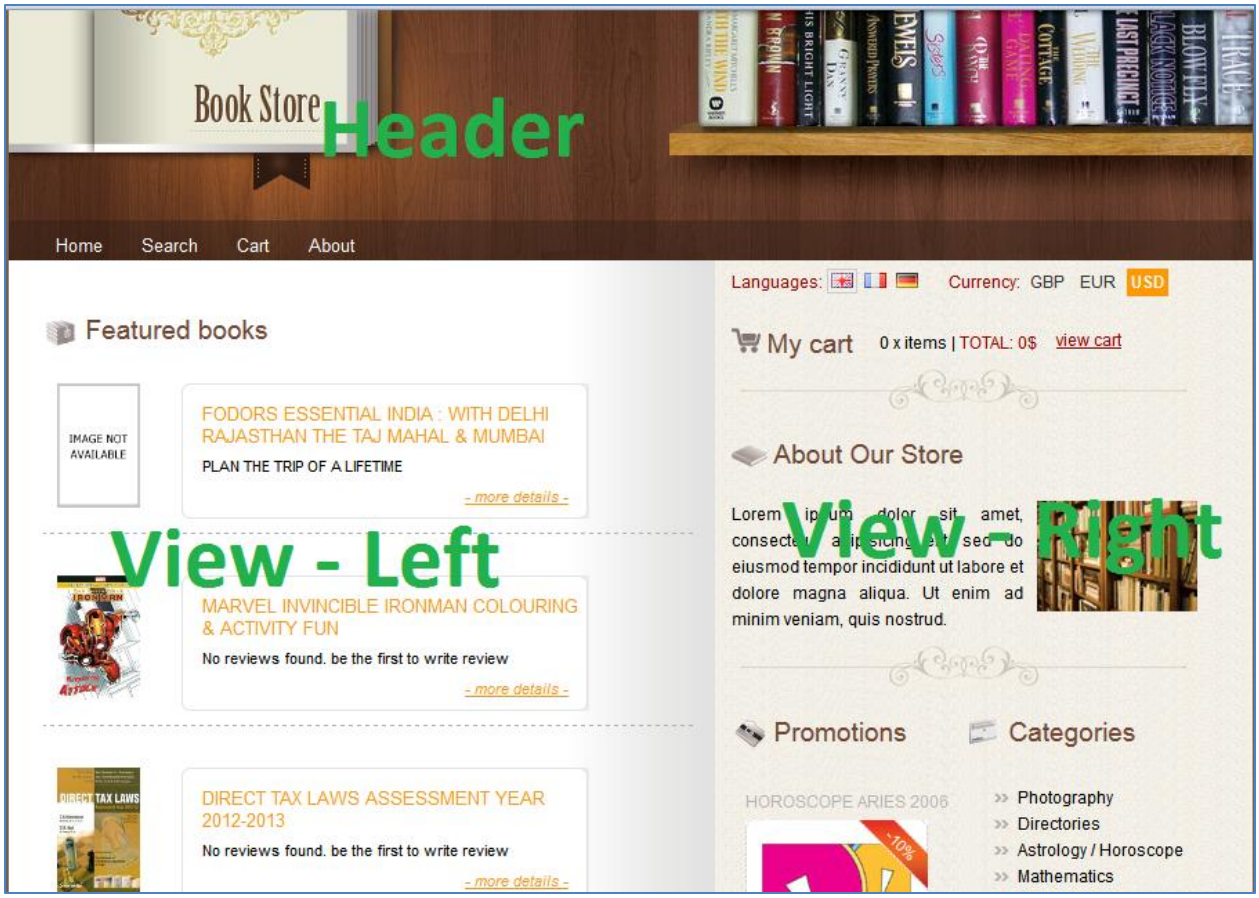
100 %

Layout templates allow you to specify the HTML container layout of your site in one place and then apply it across multiple pages in your site. Note the @RenderBody() line of code near the bottom of the file. RenderBody is a placeholder where all the view-specific pages you create show up, "wrapped" in the layout page.

BookStore application is divided into two layouts

- Right layout
- Left Layout

Email – mailtoagoenka@gmail.com



Header section is defined in **_Layout.cshtml** file and Left-Right view comes through Layout concept of MVC

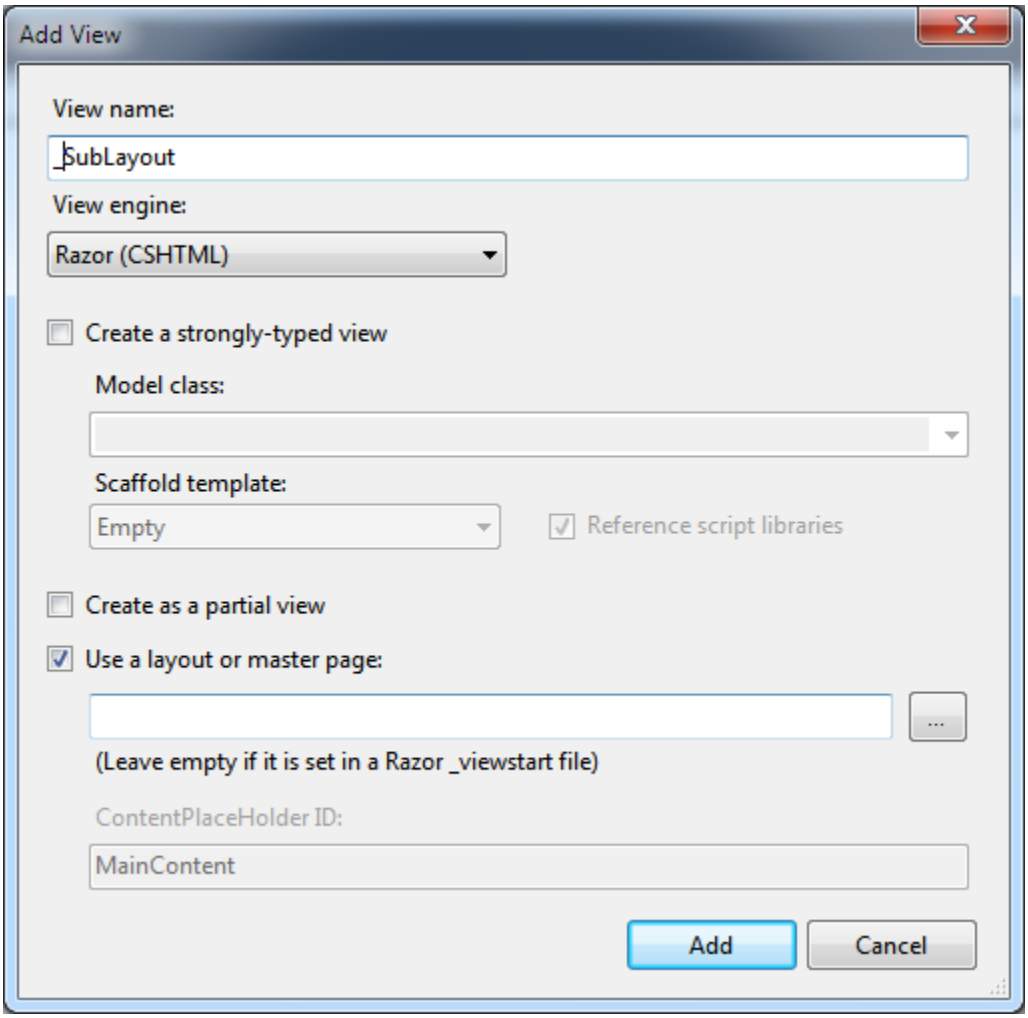
What are Layouts?

You typically want to maintain a consistent look and feel across all of the pages within your web-site/application. ASP.NET 2.0 introduced the concept of "**master pages**" which helps enable this when using .aspx based pages or templates. Razor also supports this concept with a feature called "layouts" – which allow you to define a common site template, and then inherit its look and feel across all the views/pages on your site.

Email – mailtoagoenka@gmail.com

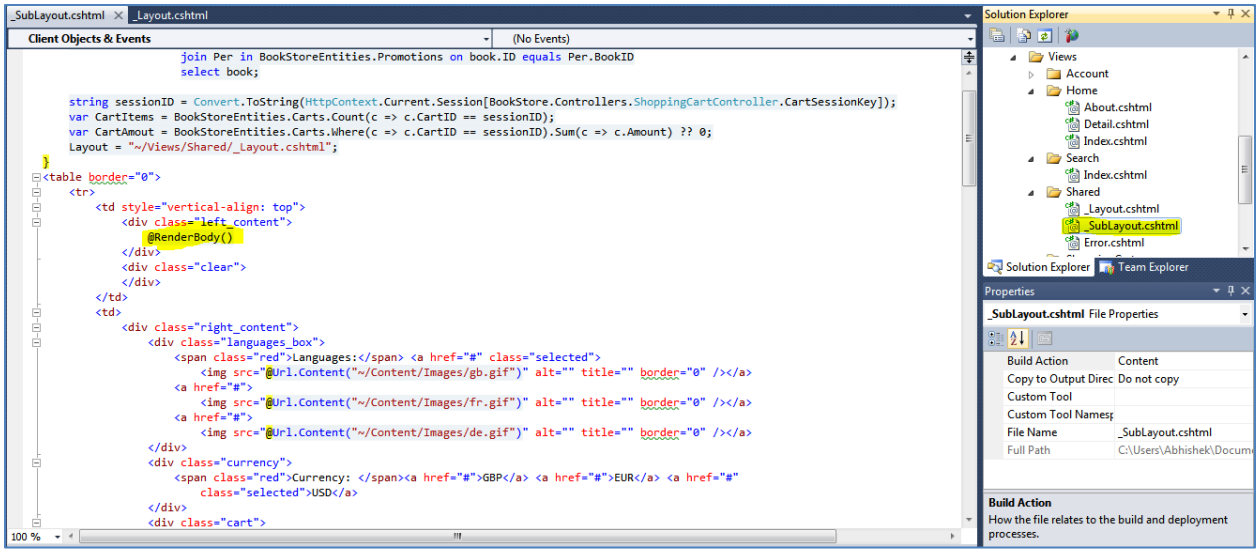
Using Layouts with Razor

Goto Shared Folder and create a shared view with name of _SubLayout.cshtml



_SubLayout.cshtml contains RenderBody method (Used to render Left View of Site) with some piece of code to display Right view of Site

Email – mailtoagoenka@gmail.com



Email – mailtoagoenka@gmail.com