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

To get a good practical understanding ASP.NET MVC, this hands on lab will teach you how to build an application using ASP.NET MVC.

We will build an **online Music Store** which is a lightweight sample store implementation which sells music albums online, & implements basic site administration, user sign-in, & shopping cart functionality.

There are 3 main parts to the application:

1. Shopping
2. Checkout
3. Administration

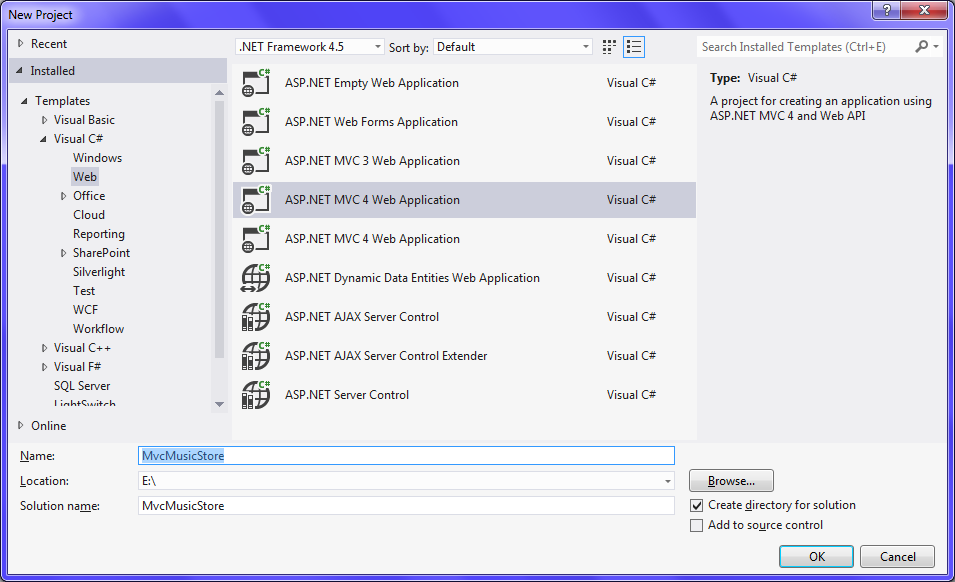
Here is a short summary of what the application will do:

1. Visitors can browse music albums by **genere.**
2. They can view a single album and add it to their shopping cart.
3. They can review their shopping cart, removing any items they no longer want.
4. Proceeding to Checkout will prompt them to login or register for a user account.
5. After creating an account, they can complete the order by filling out shipping and payment information. To keep things simple, we’re running an amazing promotion: everything’s free if they enter promotion code **FREE.**
6. After ordering, they see a simple confirmation screen.
7. In addition to customer-based pages, there is also an administrator section that shows a list of albums from which administrators can Create, Edit, & Delete albums.

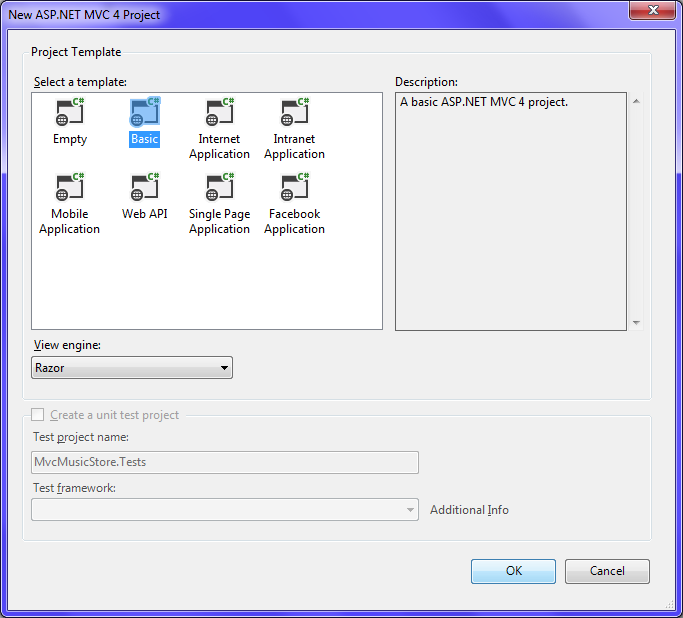
So, let’s get started…

**Part 1: Creating the ASP.NET MVC 4 Music Store Application**

1. Start Visual Studio 2010/2012.
2. Choose **File🡪New🡪Project.**
3. On the left hand side, under **Visual C#**, choose **Web** & select **ASP.NET MVC 4 Web Application** on the right side.
4. Type the application name as **MvcMusicStore**.
5. Save the application anywhere on your local drive.
6. Click on **OK.**

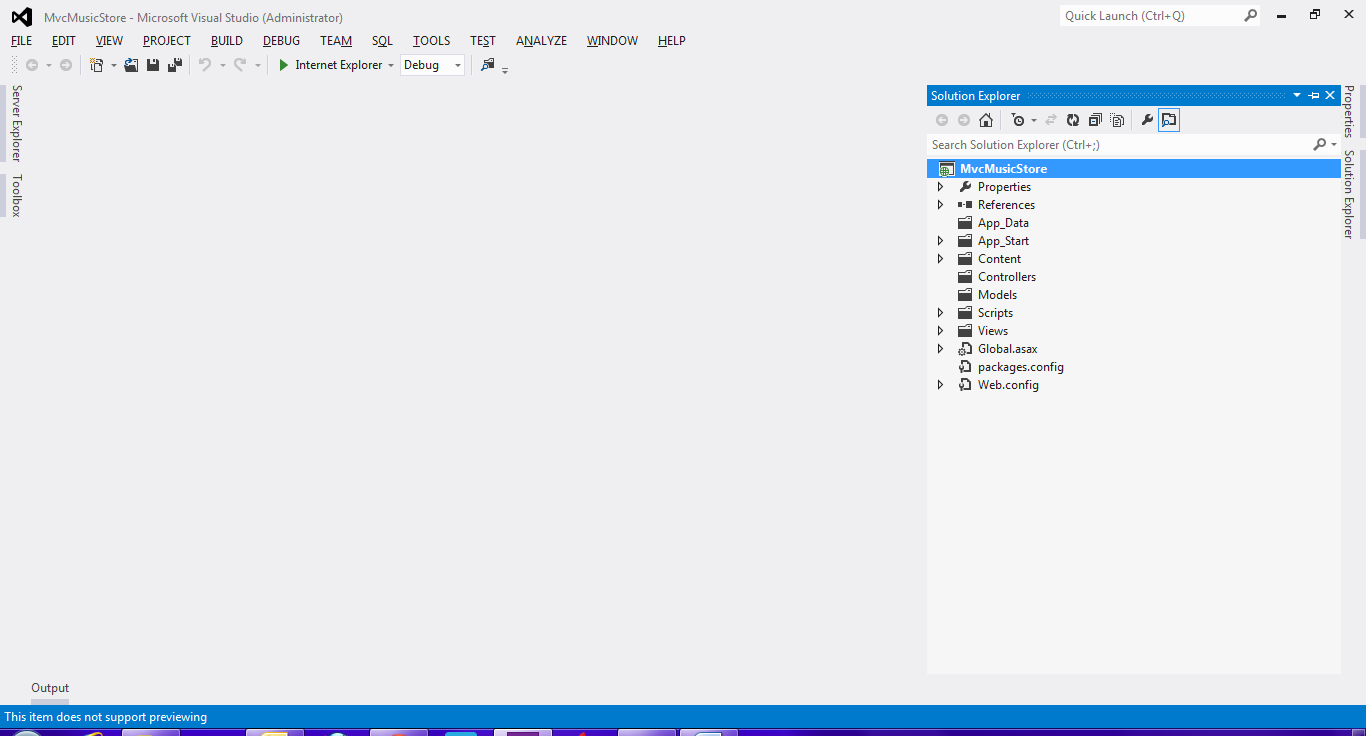


1. From the project template dialog displayed, choose **Basic** & click on **OK.**



**Part 2: Examining the application structure in Visual Studio**

Your application should look as shown below. Examine it carefully from the **Solution Explorer** window.



ASP.NET MVC framework by default uses a **convention over configuration** approach & makes some default assumptions based on folder naming conventions. For instance, controllers look for views in the **Views** folder by default without you having to explicitly specify this in your code. Sticking with the default conventions reduces the amount of code you need to write, & can also make it easier for other developers to understand your project.