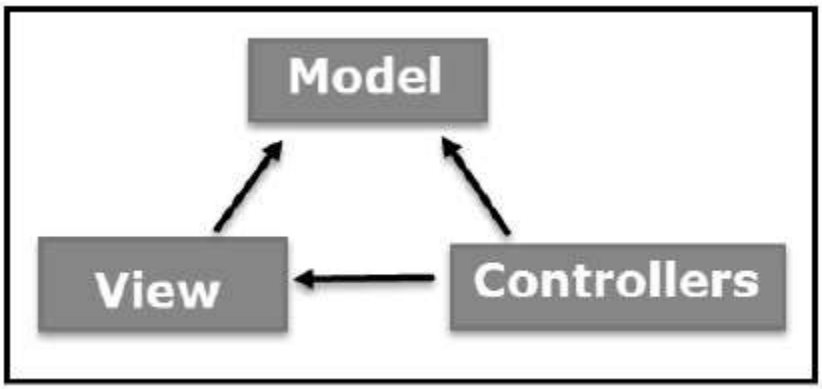
The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the **model**, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

MVC Components

Following are the components of MVC −



Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

View

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

ASP.NET MVC

ASP.NET supports three major development models: Web Pages, Web Forms and MVC (Model View Controller). ASP.NET MVC framework is a lightweight, highly testable presentation framework that is integrated with the existing ASP.NET features, such as master pages, authentication, etc. Within .NET, this framework is defined in the System.Web.Mvc assembly. The latest version of the MVC Framework is 5.0. We use Visual Studio to create ASP.NET MVC applications which can be added as a template in Visual Studio.

ASP.NET MVC Features

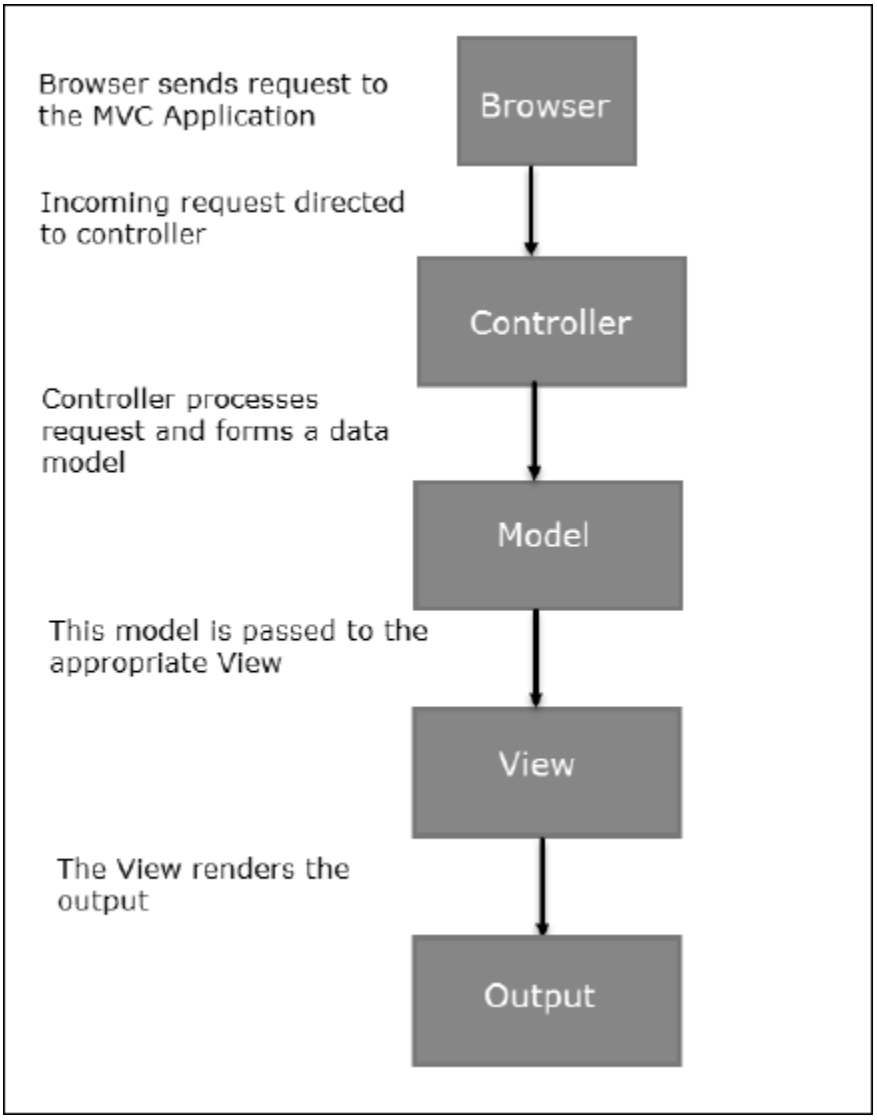
ASP.NET MVC provides the following features −

- Ideal for developing complex but lightweight applications.
- Provides an extensible and pluggable framework, which can be easily replaced and customized. For example, if you do not wish to use the in-built Razor or ASPX View Engine, then you can use any other third-party view engines or even customize the existing ones.
- Utilizes the component-based design of the application by logically dividing it into Model, View, and Controller components. This enables the developers to manage the complexity of large-scale projects and work on individual components.
- MVC structure enhances the test-driven development and testability of the application, since all the components can be designed interface-based and tested using mock objects. Hence, ASP.NET MVC Framework is ideal for projects with large team of web developers.
- Supports all the existing vast ASP.NET functionalities, such as Authorization and Authentication, Master Pages, Data Binding, User Controls, Memberships, ASP.NET Routing, etc.
- Does not use the concept of View State (which is present in ASP.NET). This helps in building applications, which are lightweight and gives full control to the developers.

Thus, you can consider MVC Framework as a major framework built on top of ASP.NET providing a large set of added functionality focusing on component-based development and testing.

Now let us take a look at how the execution of an MVC application takes place when there is a certain request from the client. The following diagram illustrates the flow.

MVC Flow Diagram



Flow Steps

Step 1 − The client browser sends request to the MVC Application.

Step 2 − Global.ascx receives this request and performs routing based on the URL of the incoming request using the RouteTable, RouteData, UrlRoutingModule and MvcRouteHandler objects.

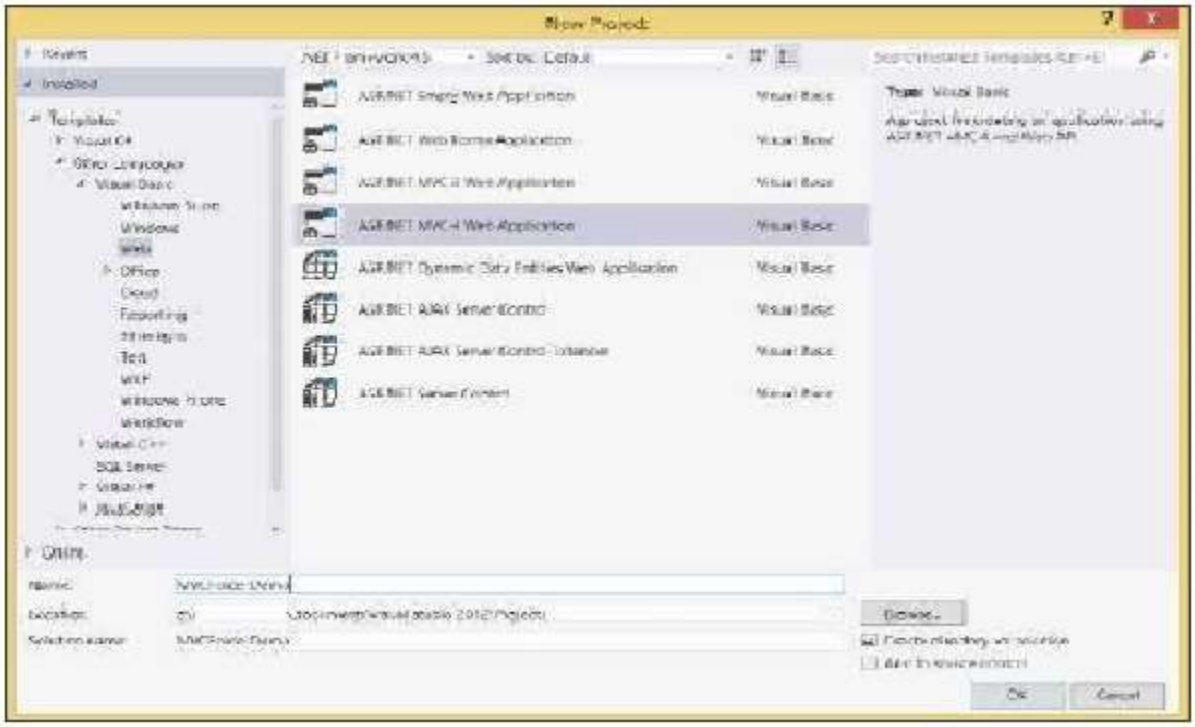
Step 3 − This routing operation calls the appropriate controller and executes it using the IControllerFactory object and MvcHandler object's Execute method.

Step 4 − The Controller processes the data using Model and invokes the appropriate method using ControllerActionInvoker object

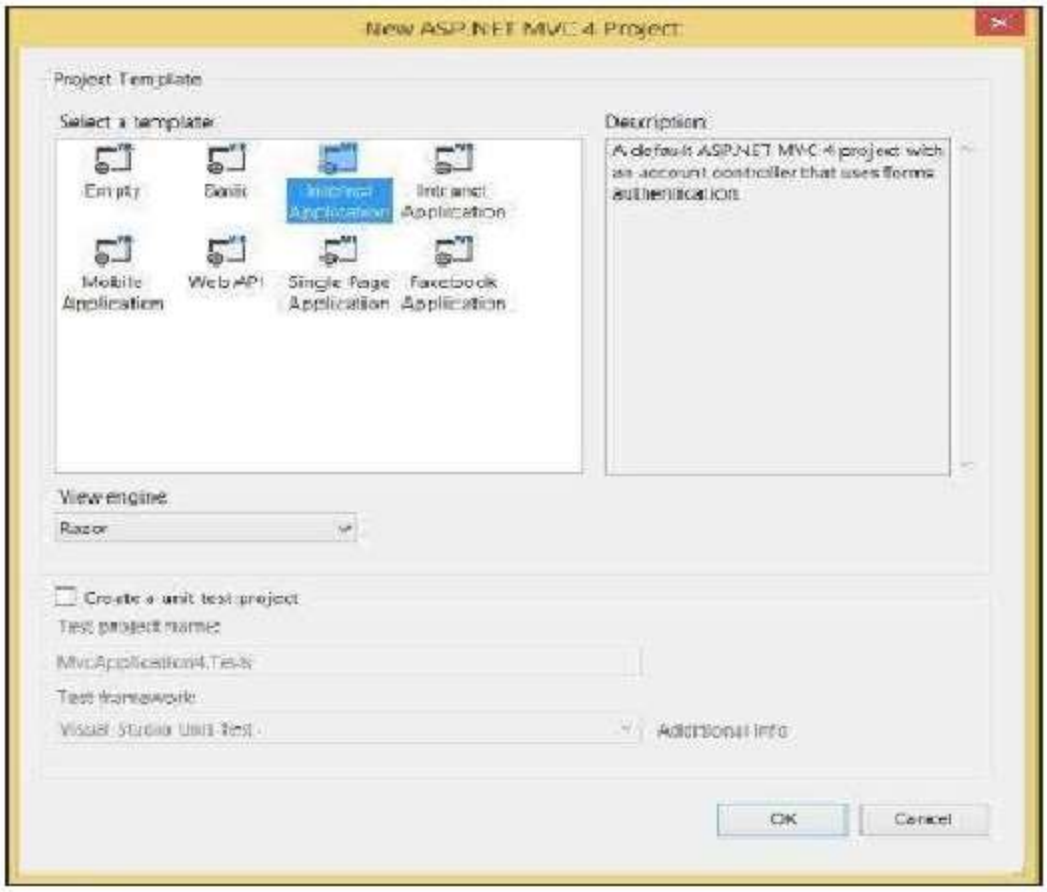
Step 5 − The processed Model is then passed to the View, which in turn renders the final output.

Now that we have already created a sample MVC application, let us understand the folder structure of an MVC project. We will create new a MVC project to learn this.

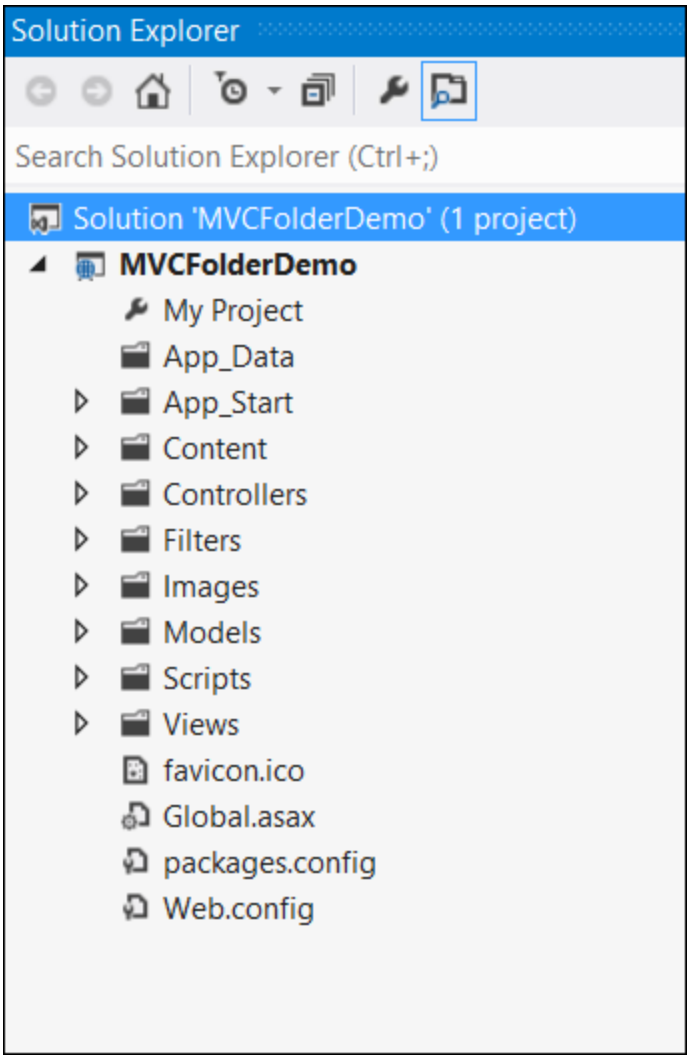
In your Visual Studio, open File → New → Project and select ASP.NET MVC Application. Name it as **MVCFolderDemo**.



Click OK. In the next window, select Internet Application as the Project Template and click OK.



This will create a sample MVC application as shown in the following screenshot.



Note − Files present in this project are coming out of the default template that we have selected. These may change slightly as per different versions.

Controllers Folder

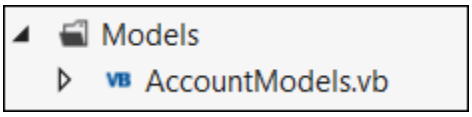
This folder will contain all the Controller classes. MVC requires the name of all the controller files to end with Controller.

In our example, the Controllers folder contains two class files: AccountController and HomeController.

Models Folder

This folder will contain all the Model classes, which are used to work on application data.

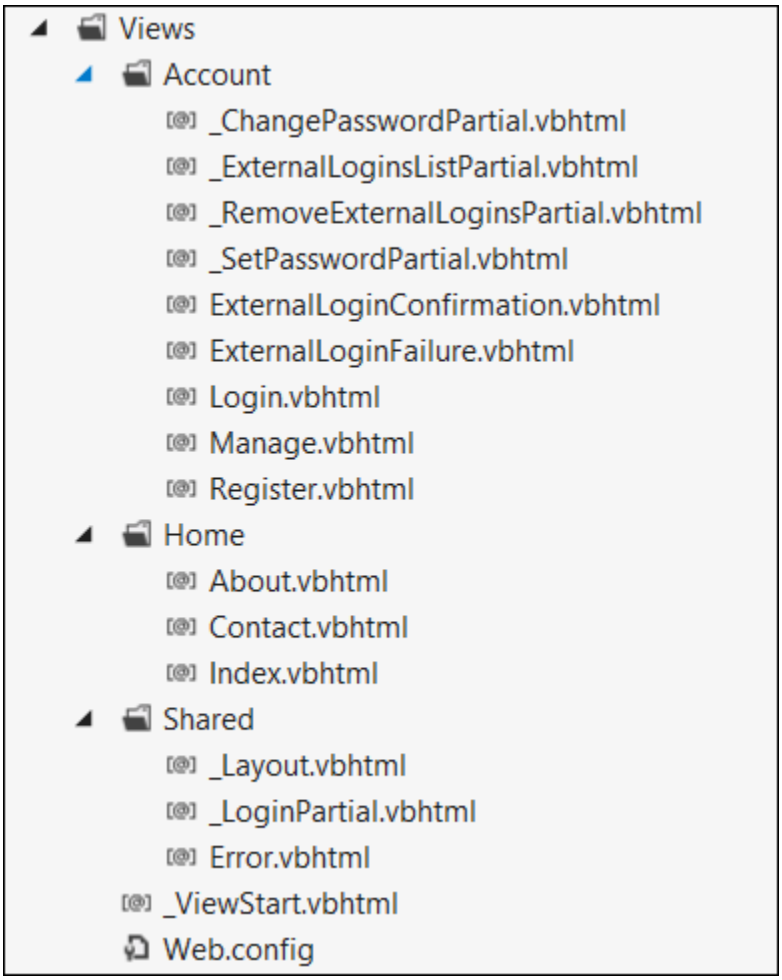
In our example, the Models folder contains AccountModels. You can open and look at the code in this file to see how the data model is created for managing accounts in our example.



Views Folder

This folder stores the HTML files related to application display and user interface. It contains one folder for each controller.

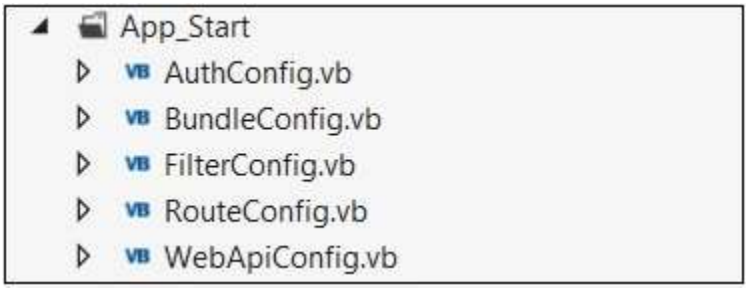
In our example, you will see three sub-folders under Views, namely Account, Home and Shared which contains html files specific to that view area.



App_Start Folder

This folder contains all the files which are needed during the application load.

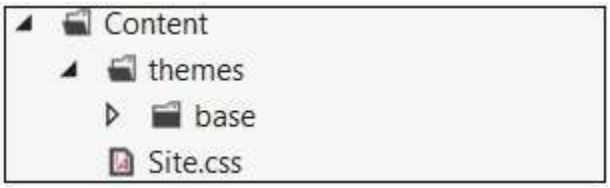
For e.g., the RouteConfig file is used to route the incoming URL to the correct Controller and Action.



Content Folder

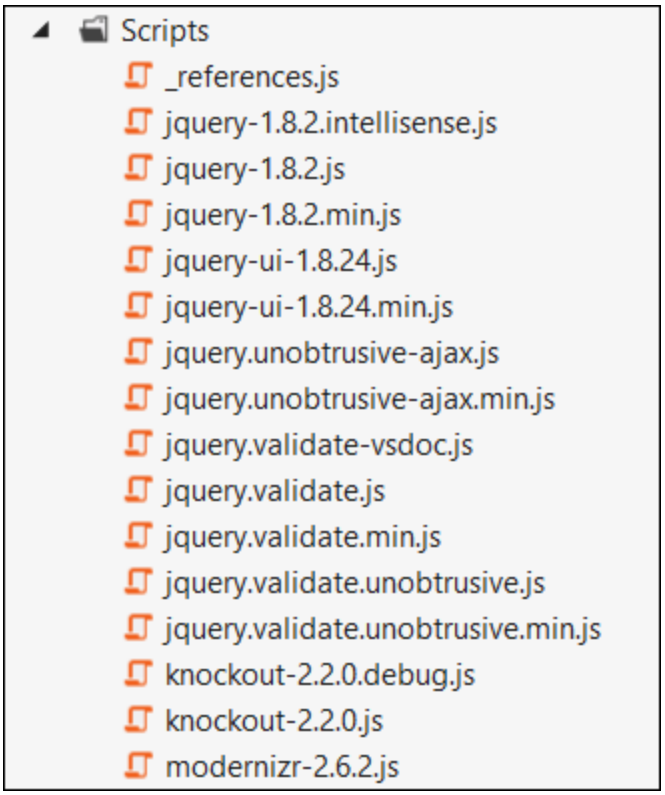
This folder contains all the static files, such as css, images, icons, etc.

The Site.css file inside this folder is the default styling that the application applies.



Scripts Folder

This folder stores all the JS files in the project. By default, Visual Studio adds MVC, jQuery and other standard JS libraries.


Scripts
_references.js
jquery-1.8.2.intellisense.js
jquery-1.8.2.js
jquery-1.8.2.min.js
jquery-ui-1.8.24.js
jquery-ui-1.8.24.min.js
jquery.unobtrusive-ajax.js
jquery.unobtrusive-ajax.min.js
jquery.validate-vsdoc.js
jquery.validate.js
jquery.validate.min.js
jquery.validate.unobtrusive.js
jquery.validate.unobtrusive.min.js
knockout-2.2.0.debug.js
knockout-2.2.0.js
modernizr-2.6.2.js