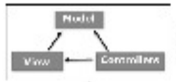
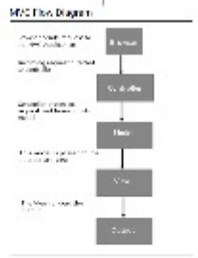


The Model View Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard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



MVC Flow Diagram

Flow Steps

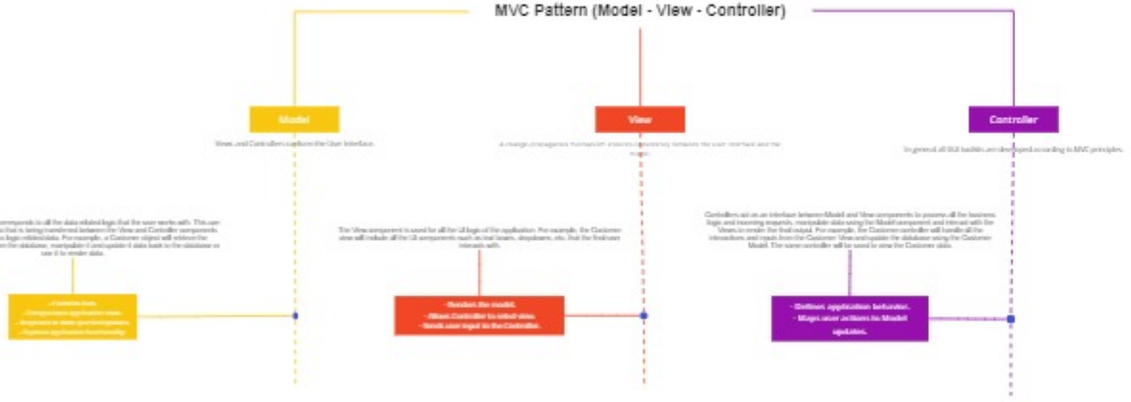
Step 1 – The client browser sends request to the MVC Application.

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.



Model, 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.

The component 'Model' is responsible for managing the data of the application. It responds to the request from the view and it also responds to instructions from the controller to update itself. Model classes can either be created manually or generated from database entities.

View, Folder
This folder stores the HTML files related to application display and user interface. It contains one folder for each controller.
Under Views, namely Account, Home and Shared which contains .html files specific to that view area.

As seen in the initial introductory chapters, Views is the component involved with the application's User Interface. These Views are generally bind from the model data and from controllers such as .html, .aspx, .ascx, .vbhtml, etc. In our First MVC Application we had used Views with Controller to display data to the final user. For rendering these static and dynamic content to the browser, MVC Framework utilizes View Engines. View Engines are basically markup syntax implementation, which are responsible for rendering the final HTML to the browser.
MVC Framework comes with two built in view engines –
Razor Engine – Razor is a markup syntax that enables the server side C# or VB code into web pages. This server side code can be used to render dynamic content within the web page in a long hooked. Razor is an advanced engine as compared to ASPX engine and was launched in the later versions of MVC.
ASPX Engine – ASPX or the Web Forms engine is the default view engine that is included in the MVC framework since the beginning. Writing a code with this engine is similar to writing a code in ASP.NET Web Forms.
Following are small code snippets comparing both Razor and ASPX engine.

Controller, 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.

Controllers are responsible for controlling the flow of the application execution. When you make a request (means request a page) to MVC application, a controller is responsible for returning the response to that request. The controller can perform one or more actions. The controller action can return different types of action results to a particular request.
The Controller is responsible for controlling the application logic and acts as the coordinator between the View and the Model. The Controller receives as input from the users via the View, then processes the user's data with the help of Model and passes the results back to the View.