**ASP .NET MVC**

In mvc, separates an application into three main components:

1. Model
2. View
3. Controller

Model :

Implements the logic for the data domain of the application.

Used to retrieve and store model state in a database such as SQL server database.

Used for business logic separation from the data in the application

View:

Component that forms the application’s user interface.

Example : edit view of products table that displays text boxes, drop down lists, check boxes based on the current state of a product object.

Controller:

Handles user interaction.

Works with the model and selects the view to render the web page.

View -> displays information

Controller -> handles and responds to the user inputs and requests

**Advantages:**

Manages application complexity by dividing application into the model, view and controller.

Doesnot use the view state or server based forms. So provides more control over how data is handled and presented. This flexibility makes MVC ideal for developers who prefer to have full control over the architecture and implementation details of their applications.

Better support for test driven development

Suitable for large scale development

**Controller MVC**

Handles user requests

Retrieves data from model, renders view as response

Maps requested URLs to the classes that are referred to as controllers.

Processes incoming requests, handle user input and interactions and executes appropriate business logic

Tasks of controller:

1. Locates for appropriate action method to call and validate
2. Handles errors that might occur during execution of the action
3. Uses WebFormViewEngine class to render Asp.net page
4. Gets the values to use as th action method’s arguments

Controller defines the action methods that are used to handle user requests and render view as the response.

User request : Entering URL, clicking a link, submitting a form

MVC uses the global.asaz.cs file to determine the path of controller. Then controller executes the appropriate action to handle the user request

ActionResult return type:

1. Return View()
2. Return PartialView()
3. Return Redirect() -> to new url
4. Return redirectToAction() -> redirect to another action method

public string Welcome()

{

return "Hello, this is welcome action message";

}

ActionMethod Parameters

Retrieve user requested values from URL

public string ShowMusic(string MusicTitle)

{

return "You selected " + MusicTitle + " Music";

}

So, we are doing it by this URL localhost:port-no/MusicStore/ShowMusic?MusicTitle=Classic.

ActionName

public class MusicStoreController : Controller

{

[ActionName("store")]

public ActionResult Index()

{

return View();

}

}

we need to create a view in the MusicStore folder as same as the ActionName

ActionVerbs

HttpPost

HttpGet

HttpPut

HttpDelete

HttpOptions

HttpPatch

**ActionFilters**

Provides filter attribute so that we can filter the user requests

Modifies the way an action is called, though applied on an individual action or an entire controller.

1. OutputCache

Makes the controllers output cacheable for the specified time

1. HandleError

To handle the error raised when a controller executes

1. Authorize

Allows only authorized user to access resources

**Model**

Contains the business logic of application

Also used to access data from database

Doesnot handle input directly from browser also don’t have html code

controller interacts with model, access data and perform the logic and pass the data to the view.

**Model Binding**

Binding a model to controller and view

Map the posted form values to .net framework type and pass the type to an action method as a parameter

Converts http requests to objects that are passed to an action method

**View data**

Dictionary of objects

Derived from ViewDataDictionary class.

We can access value by using string as a key.

Type-safe. Send data from controller to view.

Requires type casting for data type.

Accessible only during current request

In controller

public ActionResult Index()

{

List<string> Courses = new List<string>();

Courses.Add("J2SE");

Courses.Add("J2EE");

Courses.Add("Spring");

Courses.Add("Hibernates");

ViewData["Courses"] = Courses;

return View();

}

In view:

<body>

<h2>List of Courses</h2>

<ul>

@{

foreach (var Courses in ViewData["Courses"] as List<string>)

{

<li> @Courses</li>

}

}

</ul>

ViewBag

Similar to ViewData

Sends data from controller to view

Can get and set values dynamically.

Doesnot require type checking

Converts type dynamically.

In controller:  
public ActionResult Index()

{

List<string> Courses = new List<string>();

Courses.Add("J2SE");

Courses.Add("J2EE");

Courses.Add("Spring");

Courses.Add("Hibernates");

ViewBag.Courses = Courses;

return View();

}

In view:

<body>

<h2>List of Courses</h2>

<ul>

@{

foreach (var Courses in ViewBag.Courses)

{

<li> @Courses</li>

}

}

</ul>

</body>

TempData

TempData persists only from one request to the next. Retention is used to mark key to persist data so that it can retain for the next request.

Can use in one controller to another controller

**RAZOR**

Standard markup syntax

To embed server code into web pages

If there us server code in web page, server executes that code first then send the response to the browser.

Allows us to perform logical tasks in the view page

Can create expressions, loops and variables in view page

Simplified syntax, easy to learn and code.

File extension : .cshtml

@ -> transits into razor specific markup

**HTML helpers**

To create html controls programmatically.

Provides built-in methods to generate controls on view page

Partial view :

A view that can be plugged into parent view

Same extension as others-> .cshtml

Use : a large view file that contains several logical sections, we can break it into smaller components that further can be rendered as partial view.

Partial(string) : renders the specified partial view as an HTML- encoded string

RenderPartial(String) : renders the specified partial view but return type is void. Has better performance than Partial()

**Routing**

Two types of routing

1. Convention based routing

Define more general routes

Config.Routes.MapHttpRoute(

Name: “default”,

url : “api/{controller}/{Action}/{id}”,

defaults : new { controller=”Home”, Action=”index”,id=RouteParameter.Optional});

Custom routing :

Config.Routes.MapHttpRoute(

Name: “default”,

url : “/Home”,

defaults : new { controller=”Home”, Action=”index”,id=RouteParameter.Optional});

1. Attribute Routing

Define to specify more specific routes

To enable routing in the route config //in routeconfig.cs

Config.MapHttpAttributeRoutes();

[Route("products/{id?}")] -> remders for /product or /product/1

public ActionResult Details(string id)

{

if (string.IsNullOrEmpty(id))

{

return View("List", GetProductList());

}

return View("Details", GetProductDetails());

}

1. **public** **class** HomeController : Controller
2. {
3. //URL: /Mvctest
4. [Route(“Mvctest”)]
5. **public** ActionResult Index()
6. ViewBag.Message = "Welcome to ASP.NET MVC!";
7. **return** View();
8. }}
9. **public** **class** HomeController : Controller
10. {
12. // Optional URI Parameter
13. // URL: /Mvctest/
14. // URL: /Mvctest/0023654
16. [Route(“Mvctest /{ customerName ?}”)]
17. **public** ActionResult OtherTest(**string** customerName)
18. ViewBag.Message = "Welcome to ASP.NET MVC!";
19. **return** View();
20. }
22. // Optional URI Parameter with default value
23. // URL: /Mvctest/
24. // URL: /Mvctest/0023654
26. [Route(“Mvctest /{ customerName =0036952}”)]
27. **public** ActionResult OtherTest(**string** customerName)
28. {
29. ViewBag.Message = "Welcome to ASP.NET MVC!";
30. **return** View();
31. }
32. }
33. [RoutePrefix(“Mvctest”)]
34. **public** **class** HomeController : Controller
35. {
36. // URL: /Mvctest/
37. [Route]
38. **public** ActionResult Index()
39. {
40. ViewBag.Message = "Welcome to ASP.NET MVC!";
41. **return** View();
42. }
44. // Optional URI Parameter
45. // URL: /Mvctest/
46. // URL: /Mvctest/0023654
47. [Route(“{ customerName }”)]
48. **public** ActionResult OtherTest(**string** customerName)
49. {
50. ViewBag.Message = "Welcome to ASP.NET MVC!";
51. **return** View();
52. }
53. }
54. //route gets called as /products/productname
55. [Route("products/{id:alpha}")]
56. **public** ActionResult GetProduct(**string** name)
57. {
58. **return** View();
59. }
61. //route gets called as /products/2
62. [Route("products/{id:int}")]
63. **public** ActionResult GetProduct(**int** id)
64. {
65. **return** View();
66. }