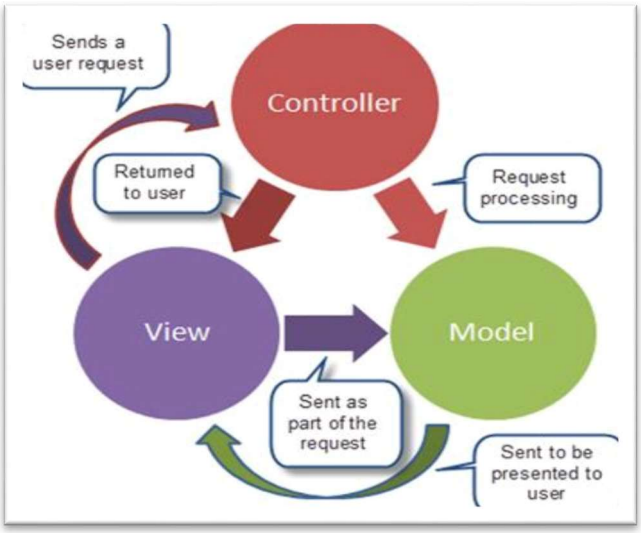
What is MVC?

- Model-View-Controller (MVC)
- Standard Architectural Pattern
- Separation of concerns: model, view, controller



ASP .NET MVC Framework

- An alternative to ASP .NET Web Forms
- Presentation framework
  - Lightweight
  - Highly testable
  - Integrated with the existing ASP .NET features:
    - Master pages
    - Membership-Based Authentication

ASP .NET Web Forms vs ASP.Net MVC

ASP.NET Web Forms	ASP.NET MVC
ASP.NET Web Forms use Page controller pattern approach for rendering layout. In this approach, every page has its own controller, i.e., code-behind file that processes the request.	ASP.NET MVC uses Front Controller approach. That approach means a common controller for all pages processes the requests.
No separation of concerns. As we discussed that every page (.aspx) has its own controller (code behind i.e. aspx.cs/.vb file), so both are tightly coupled.	Very clean separation of concerns. View and Controller are neatly separate
Because of this coupled behavior, automated testing is really difficult	Testability is a key feature in ASP.NET MVC. Test driven development is quite simple using this approach.
In order to achieve stateful behavior, viewstate is used. Purpose was to give developers the same experience of a typical WinForms application.	ASP.NET MVC uses Front Controller approach. That approach means a common controller for all pages processes the requests.
Statefulness has a lots of problem for web environment in case of excessively large viewstate. Large viewstate means increase in page size.	As controller and view are not dependent and also no viewstate concept in ASP.NET MVC, so output is very clean.

ASP .NET MVC Framework Components

- Models
  - Business/domain logic
  - Model objects, retrieve and store model state in a persistent storage (database).
- Views
  - Display application's UI
  - UI created from the model data

- Controllers
  - Handle user input and interaction
  - Work with model
  - Select a view for rendering UI

ASP.NET MVC 5.0

- Scaffolding
- ASP.NET Identity
- One ASP.NET
- Bootstrap
- Attribute Routing
- Filter Overrides

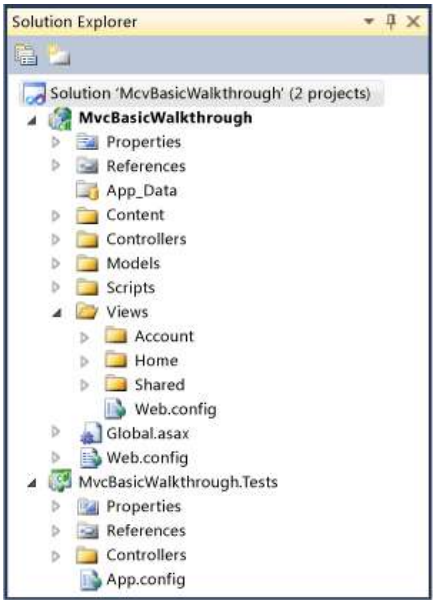
Advantages of using MVC approach

- Easier to manage complexity (divide and conquer)
- It does not use server forms and view state
- Front Controller pattern (rich routing)
- Better support for test-driven development
- Ideal for distributed and large teams
- High degree of control over the application behavior

ASP .NET MVC Features

- Separation of application tasks
  - Input logic, business logic, UI logic
- Support for test-driven development
  - Unit testing
  - No need to start app server
- Extensible and pluggable framework
  - Components easily replaceable or customized (view engine, URL routing, data serialization)
- Support for Dependency Injection (DI)
  - Injecting objects into a class
  - Class doesn't need to create objects itself
- Support for Inversion of Control (IOC)
  - If an object requires another object, the first should get the second from an outside source (configuration file)
- Extensive support for ASP .NET routing
- Building apps with comprehensible and searchable URLs
- Customizable URLs
  - Adapted to work well with search engines
  - Adapted to REST addressing
  - Decoupled from resource files
- Use of existing ASP .NET features (backward compatibility)

ASP .NET App Structure



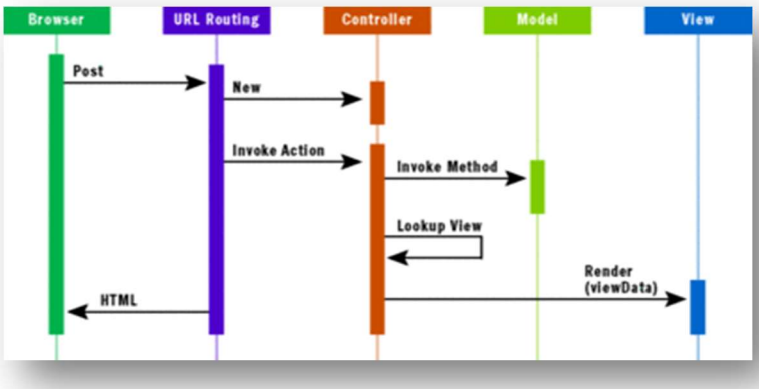
- URLs mapped to controller classes
- Controller
  - handles requests,
  - executes appropriate logic and
  - calls a View to generate HTML response
- URL routing
  - ASP .NET routing engine (flexible mapping)
  - Support for defining customized routing rules
  - Automatic passing/parsing of parameters
- No Postback interaction!
- All user interactions routed to a controller
- No view state and page lifecycle events

MVC App Execution

- Entry points to MVC:
  - UrlRoutingModule and MvcRouteHandler
- Request handling:
  - Select appropriate controller
  - Obtain a specific controller instance
  - Call the controller's Execute method

MVC App Execution - Stages

- Receive first request for the application
  - Populating RouteTable
- Perform routing
- Create MVC Request handler
- Create controller
- Execute controller
- Invoke action
- Execute result
  - ViewResult, RedirectToRouteResult, ContentResult, FileResult, JsonResult, RedirectResult



Razor Engine

- A new view-engine
- Optimized around HTML generation
- Code-focused templating approach

Razor Engine – Design goals

- Compact, Expressive and Fluid
- Easy to learn
- It is not a new language
- Works with any text editor
- Great Intellisense
- Unit-testable
  - Testing views without server, controllers…

Working with Layout

Methods to render the content between the master pages and content pages…

@RenderBody() : Used to render the portion of a content page inside the master page.
@RenderPage() : Used to render the content of one page with in another page(layout or normal page)
@RenderSection() : Used to render the content of a named section of content page inside master page

HTML Helpers

- Methods that can be invoked within code-blocks
- Encapsulate generating HTML
- Implemented using pure code
- Work with Razor engine

Views

- Views created using Razor view engine
- Controller method returns View object
- Controller method return type is ActionResult
- Common pattern: all view pages share the same master layout page

Partial View

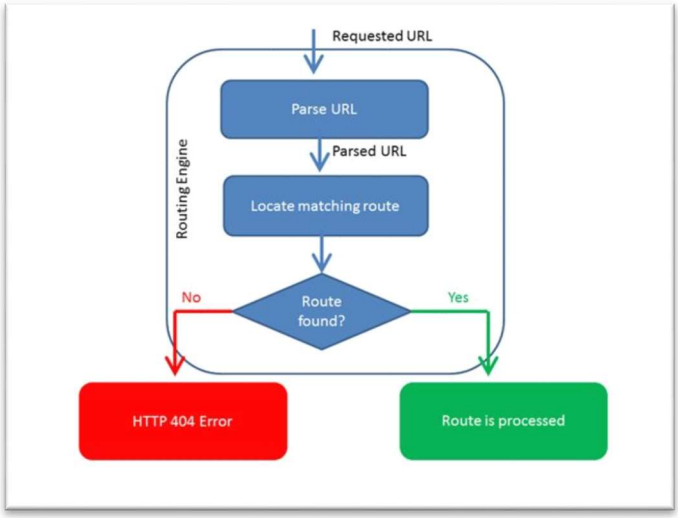
- Used to define a view that will be rendered inside a parent view.
- Using partial view we can render a view inside a parental view and to create reusable content in the project.
- Are Implemented as ASP.NET user controls (.ascx)
- Can access the parent view data
- Plays a vital role in AJAX implementation(Partial page updation)

Razor – Summary

- A good new view engine

- Code-focused templating
- Fast and expressive
- Compact syntax
- Integrated with C# and VB

ASP.NET MVC Routing



Overview - ASP.NET MVC routing is a pattern matching system that is responsible for mapping incoming browser requests to specified MVC controller actions.

We can see how the routing engine processes a request and what response it sends. It gives a response according to URL match or not in the route table.

1) When the request's URL matches any of the registered route patterns in the route table then the routing engine forwards the request to the appropriate handler for that request.
2) When the request's URL does not match any of the registered route patterns then the routing engine indicates that it could not determine how to handle the request by returning a 404 HTTP status code.

Properties of Route

ASP.NET MVC routes are responsible for determining which controller method to execute for a given URL. A URL consists of the following properties:

Route Name: A route is a URL pattern that is mapped to a handler. A handler can be a controller in the MVC application that processes the request. A route name may be used as a specific reference to a given route.

URL Pattern: A URL pattern can contain literal values and variable placeholders (referred to as URL parameters). The literals and placeholders are located in segments of the URL that are delimited by the slash (/) character.

Defaults: When you define a route, you can assign a default value for a parameter. The defaults is an object that contains default route values.

Constraints: A set of constraints to apply against the URL pattern to more narrowly define the URL that it matches.

Understand the Default Route

The default ASP.NET MVC project templates add a generic route that uses the following URL convention to break the URL for a given request into three named segments.

url: "{controller}/{action}/{id}"

This route pattern is registered via call to the MapRoute() extension method of RouteCollection.

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index",
            id = UrlParameter.Optional }
    );
}
```

Ignore routes that end with axd extension

Route Name

URL Pattern

Default Values

Figure - Default Routes for MVC Application

When the MVC application launches the Application_Start() event handler of the global.asax execute that call the RegisterRoutes() method from RouteConfig class under App_Start directory (App_Start/RouteConfig.cs). The RegisterRoutes() route has a parameter that is a collection of routes called the RouteCollection that contains all the registered routes in the application. The Figure 1.2 represents the default method that adds routes to the route table.

Routing with an Example

URL	Controller	Action	Id
http://localhost:4736/	HomeController	Index	
http://localhost:4736/Book/	BookController	Index	
http://localhost:4736/Book/Create	BookController	Create	
http://localhost:4736/Book/Edit/2	BookController	Edit	2

The controller and action values in the route are not case-sensitive. URL route patterns are relative to the application root, so they don't need to start with a forward slash (/) or virtual path designator (~/).

Action Results

Name	Framework Behavior	Producing Method
ContentResult	Returns a string literal	Content
EmptyResult	No response	
FileContentResult / FilePathResult / FileStreamResult	Return the contents of a file	File
HttpUnauthorizedResult	Returns an HTTP 403 status	
JavaScriptResult	Returns a script to execute	JavaScript
JsonResult	Returns data in JSON format	Json
RedirectResult	Redirects the client to a new URL.	Redirect
RedirectToRouteResult	Redirect to another action, or another controller's action	RedirectToRoute / RedirectToAction
ViewResult PartialViewResult	Response is the responsibility of a view engine	View / PartialView

Action Filters

- ASP.NET MVC provides Action Filters for executing filtering logic either before or after an action method is called.
- Action Filters are custom attributes that provide declarative means to add pre-action and post-action behavior to the controller's action methods.
- Action Filters can be applied to method level, controller level and application level.
- In MVC 4 we can register filter in application level we need to add filter in Register Global Filters method under Filter Config.cs
- We can create our own Custom Action Filters.

ViewBag

- Pass data between view template and layout view file
- ViewBag is a dynamic object (has no defined properties)

Passing data from Controller to View

- View is used for data presentation
- Controller must provide a view with the data
- One approach: using ViewBag
  - Controller puts data to ViewBag,
  - View reads ViewBag and renders the data
  - No data binding!
- Alternative approach: the view model
  - Strongly typed approach

Model Components

- Entity framework - data access technology
- "Code first" development paradigm (first code classes, then generate DB schema)
- "Database first" development paradigm define db schema first, then generate models, controllers and views

ActionLink Helper

- Html.ActionLink – generates a link according to a given URL mapping policy
- Primer:

```
Html.ActionLink("Edit", "Edit", new{id=item.ID)}
```

HTTP Methods – Best Practices

- HttpGet and HttpPost method overloads
- All methods that modify data SHOULD use HttpPost method overload
- Modifying data in HttpGet method
  - security risk
  - Violates HTTP best practices
  - Violates REST architectural pattern
- GET method SHOULD NOT have any side effect and SHOULD NOT modify persistent data

Data Validation

- Keep Things DRY (Don't Repeat Yourself)
- Declarative validation rules in one place (Model class)
  - Regular expressions
  - Range validation
  - Length validation
  - NULL values validation
  - Data formatting
- Validation rules enforced before saving changes to the database!

DataType attributes

- Provide only hits for the view engine to format the data
- Date, Time, PhoneNumber, EmailAddress,…
- Automatic provision of type specific features
  - e.g. "mailto: ..." link for EmailAddress
- Do NOT provide any Validation (just presentation hints)

DisplayFormat attributes

- Used to explicitly specify format of the data
- Example: redefining the default date format

AJAX Helpers in MVC

The main purpose of AJAX Helpers are used to create AJAX enabled forms and links which performs request asynchronously.

@Ajax.BeginForm

- One of the problems in today's web applications is that they are full of page postbacks, so we will learn about Ajax.beginForm, which is the extension method of the Ajax helper class.
- By using it, we can post the request to the server without the whole page postback.
- Ajax.BeginForm is the extension method of the ASP.NET MVC Ajax helper class, which is used to submit form data to the server without whole page postback.

When to use **@Ajax.ActionLink** –

- When the user clicks on the link and it want to redirect to another page then we can use @HTML.ActionLink.
- When the user clicks on the link and don't want to redirect to different page, need to stay on the same page (without Post Back), then we can go for @AJAX.ActionLink.

Difference between Html and Ajax helper –

- HTML helper class performs request synchronously.
- Ajax helper class performs request asynchronously.

Unobtrusive AJAX –

- ASP.NET MVC supports unobtrusive Ajax which is based on jQuery. The unobtrusive Ajax is used to define the Ajax features through in our application.
- While creating project in ASP.NET MVC, the jquery-unobtrusive-ajax.js and jquery-unobtrusive-ajax.min.js files has added automatically to our application.

Below example code shows how to use AJAX action link in Asp.Net MVC.

```
@Ajax.ActionLink("Get User Details", "GetDetails", new AjaxOptions
{
UpdateTargetId = "divGetAllUsers",
HttpMethod = "GET",
OnSuccess = "fnSuccess",
})
```

OUTPUT: Run the application and we can see that the he ajax helper Action link has converted to the HTML anchor link element.