MVC Arch.

- Flow Dig.

Browser sends req. to MVC → app.

Incoming rq. directed to controller

controller process rq. & forms a data model

This Model is parsed to ~~be~~ the appropriate view

The view renders the output

$$\begin{tikzcd}
\boxed{\text{Browser}} \arrow[d] \\
\boxed{\text{Controller}} \arrow[d] \\
\boxed{\text{Model}} \arrow[d] \\
\boxed{\text{View}}
\end{tikzcd}$$

- Model - View - Controller (MVC)

  ↳ is an arch. pattern that sep. an app in 3 parts : M, V, C.

  ↳ Each comp. handles diff. aspects.

$$\begin{tikzcd}
 & \boxed{\text{Model}} & \\
\boxed{\text{View}} \arrow[ur] & & \boxed{\text{Controllers}} \arrow[ul] \arrow[ll]
\end{tikzcd}$$

Page No.
Date

ASP.NET MVC

↳ Supports 3 major models

1. Web pages

2. Webforms.

3. MVC

↳ lightweight, highly testable presentable framework.

↳ defined within .Net in System.WebMvc assembly.

ASP. NET form vs MVC

↳ Inter related yet diff.

↳ At high level, MVC is adv. A sophisticated web app. framework. with testability.

Comparison factors	Webforms	MVC
Rendering approach	Follows Page control pattern approach for rendering layouts	front controller approach is used
Sepration of concern	No, all web forms are tightly coupled	Very clean separation of concern.
Automated testing	Really difficult	Quiet simple

Control over Layout ★	The above abstraction was good but provides limited control over HTML, JS & CSS which is necessary in many cases	Full control over HTML, JS & CSS
State	Yes, View State is used	Stateless
Performance	Slow due to Large View state.	Fast due to no View state & too clean approach
Lifecycle	Page	No Page
Controls	Lot of server side	3rd Party.
★		
RAD Support	Yes	NO
Scalability	It's good for small scale app.	Large app.

- Folders

1. Models

   ↳ Classes which are used to work on app. data.

2. Views

↳ HTML related files.

↳ One folder for each controller

3. App_Start

↳ Contains file needed during app. load

↓ Auth config, ↓ Route, ↓ Bundle, ↓ WebApi, ↓ Filter

4. Content

↳ Static files like css, images, icons etc

↳ site.css → default styling

5. Script

↳ Js files.

- Routing Engine

↳ enables use of URLs

↳ can be used to hide data.

- format

http: // servername / {controller} / {action}

• Action ~~filters~~ / Methods

↳ to add pre & post - action behaviours on the controller's action methods

- Types :-

1. Action filters

↳ implement logic that gets executed b4 & after controller's exe.

2. Authorization filters

↳ Implement auth & authorization for controller actions

3. Result filters

↳ contains logic that is executes before &

Date

after a view result is executed.

4. Exception filter

↳ Last type of filter to run

↳ handle errors.

- Action filers

↳ mostly used.

↳ MVC provides 3 action filters

a. Output Cache

↓

caches the output of a controller action for a specified amount of time.

b. Handle Error

↓

handles errors raised ~~by~~ when a controller action executes

c. Authorize

↓

enables you to restrict access to a particular user or role.

Ajax Support

↳ A synchronous Java Script and XML.

↳ MVC built-in support for un obtrusive Ajax.

↳ To enable , open Web.config & inside appsettings add:-

<add key = "UnobtrusiveJavaSriptEnabled" value = "true" />

After this open _Layout.cshtml.

<script src = "~/Scripts/jquery-ui-1.8.24.min.js" type = "text/javascript">

</script>

<script src = "~/scripts/jquery.unobtrusive-ajax.min.js" type = "text/javascript">

</script>

Bundling & Minification

↳ performance improvement techqs

- Bundling : bundle multiple files into one, fewer HTTP requests.

- Minification : it optimizes Js, css code by shortening variable names, remove not need white spaces, line etc. ↓ file size.

• Exception Handling

1. Override OnException Method

↳ used when we want to handle all exceptions across the Action methods at the controller level.

2. HandleError Attribute

↳ action filter

↳ default implementation of IExceptionFilter.

↳ handles all exp. raised by contoller action, filters & views.