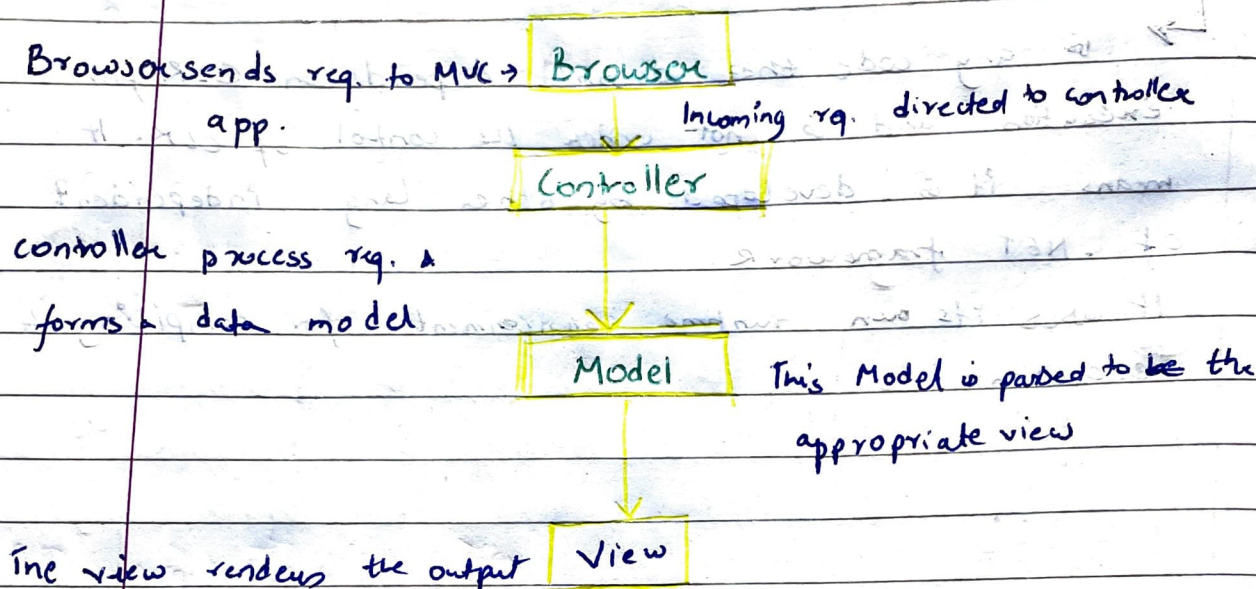


MVC Arch.

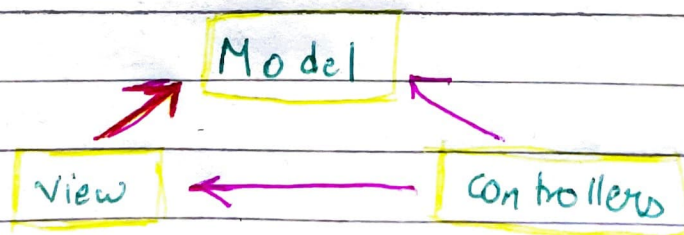
Flow Dig.



Model - View - Controller (MVC)

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

↳ Each comp. handles diff. aspects.



ASP.NET MVC

↳ Supports 3 major models

1. Web pages

2. Web forms.

3. MVC

↳ lightweight, highly testable presentable framework.

↳ defined within .Net in System.Web.Mvc 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

Separation of concern

No, all web forms are tightly coupled

Very clean separation of concerns.

Automated testing

Really difficult

Quite 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, ViewState is used

Stateless

Performance

Slow due to Large view state.

Fast due to no view state & no 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

```

graph TD
    A[App-Start] --> B[AuthConfig]
    A --> C[Route]
    A --> D[Bundle]
    A --> E[WebApi]
    A --> F[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 ~~by~~ & after controller's ~~one~~.

2. Authorization filters

↳ implement auth & authorization for controller actions

3. Result filters

↳ contains logic that is executed before *

after a view result is executed.

4. Exception filter

↳ Last type of filter to run

↳ handle errors.

- Action filters

↳ 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 encounters an exception.

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 unobtrusive Ajax.

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

< add Key = "UnobtrusiveJavaScriptEnabled" 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 techs

- 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 controller action, filters & views.