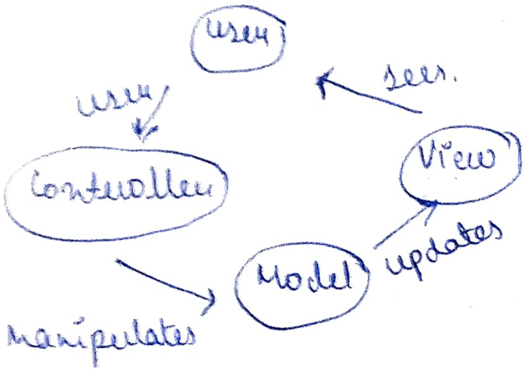
Lecture:-01 (what is Asp.net MVC)

- MVC ≠ Asp.net MVC
- M → Model, V → View, C → Controller.



what is MVC?

- Model View Controller.
- MVC is design Pattern or a way to do things.
- MVC Pattern is used for mobile, desktop & web applications.
- All major technologies are using MVC to design web applications.
- MVC is used by all technologies like Java, .net, Php, Angular etc.

what is Asp.net MVC?

- Asp.net MVC is a .net framework which follows MVC design Pattern.
- It is used for web, ~~mobile & desktop~~ development.
- This framework provides everything required to build a web app using MVC design Pattern.

Lecture:-04 (understanding MVC structure)

•) App-Data:- All data which is required for read & write into ~~a~~ application.

•) App-Start:- Here you put all the configuration. Ex→ route configuration, bundle configuration.

•) content:- It Stores images, css etc.

•) controllers:- here you put all controllers.

•) fonts:- These are custom fonts used by bootstrap.

•) Models:- These are related to database, all data comes through these Models.

•) Scripts:- all .js files comes under script folder.

•) Views:- all HTML files comes in this folder.

•) favicon.ico:- when project runs there is icon above side that is favicon icon.

•) global.asax:- when our applications starts, this files calls first.

•) Packages.config:- all ~~f~~ installed Packages with version.

•) web.config:- all configurations. like connection string of database.

Lecture:-05 (what is controller in MVC)

•) It is .cs file.

•) HomeController.cs :- It is a $\overset{child}{\wedge}$ class of controller (Parent class).

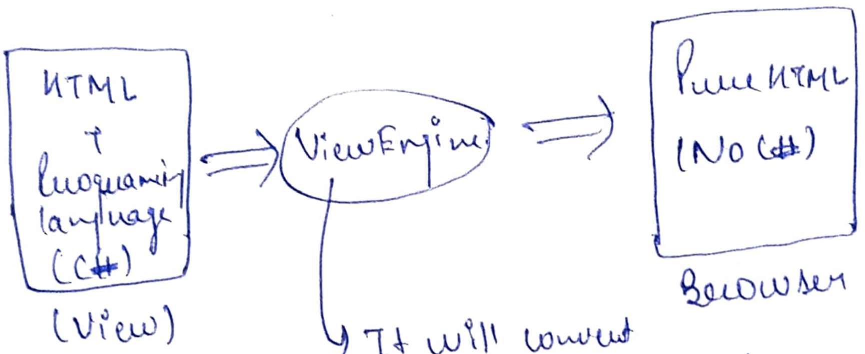
•) To make a controller, always class name ends with Controller like HomeController.cs, DataController.cs etc.

Role of Controller :-

•) work with user input.
•) Play an important role

Lecture :-08 (Razor View Engine)

what is View Engine?



It will convert (HTML & C#) code into Pure HTML code when shows on Browser.

why ViewEngine?

•) ViewEngine is responsible for creating HTML for Views.
•) To find corresponding view for action method.
•) To find view from shared folder.
•) used to write C# code on View
•) used for HTML helpers

Razor :-

•) It is a View Engine.
•) It is not a programming language.
•) Before MVC3 there is a view Engine aspx & now it is cshtml.

•) Razor is a way to write C# code in View.

Razor Syntax:-

It all begins with @

•) Single line syntax → @ expression

•) Multi line Syntax. → @(expression).

Comments in View:-

•) Single line ⇒ //comment

•) Multiline ⇒ /**/

Lecture:-09 (Loops, conditional statements & directives)

If Statement:-

```
@if()
{
}
```

For loop:-

```
@for(int i=0; i<10; i++)
{
   // statement
}
```

Ternary operator

@(condition ? Statement1 : st 2).

Directives

@using ⇒ not need to mention Particular type.

@model ⇒ for Particular type of model.

for a single view we can only use single model only.

Lecture:-10 (Inline HTML Helpers)

HTML Helpers:-

•) These are methods

•) These return HTML String

•) These are used on View.

•) In simple terms these are C# methods which are used to return HTML.

•) using HTML helpers you can render textbox, area, image etc.

Types of HTML Helpers:-

•) Inline HTML Helpers

•) Built-in HTML Helpers:-
   ↳ Standard HTML Helpers
   ↳ Strongly Typed HTML Helpers
   ↳ Templated HTML Helpers

•) Custom HTML Helpers.

Inline HTML Helpers:-

•) These are created on single view & used on same Page.

•) These can be created using @helper tag.

```
@helper HelperName(Parameter
{
    // code
}
```

Lecture:-21 (Pass Data from View to Controller)

ways to Pass data from view to controller

1) using Parameters
2) using Request
3) using form collection
4) Strongly binding
5) using JS etc.

1) using Parameter
@Html.BeginForm ("Action method name", "controllername").

2) using Request.

String name = Request["yourname"] $\to$ which is in view.

3) using Form collection.

```
Public String M1 (FormCollection form)
{
}
```

$\to$ this need to Pass

Lecture:-22

(Validation in MVC in loosely Binding)

What is validation?

1) Validation is used to filter the user input
•) There are different attributes to validate the input

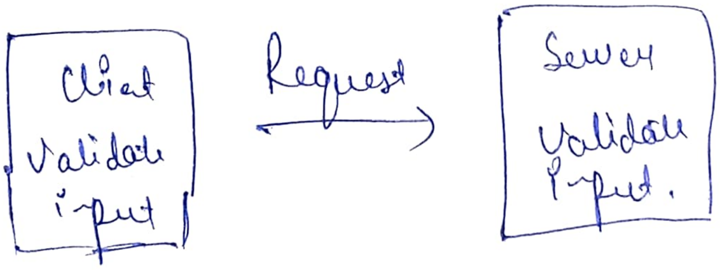
Ex →

•) To check if input is required
•) To check like min length.

Types of validation

1) Client Side Validation
2) Server Side Validation → for using datavalation we use modelstate on server side.

↳ It holds data also.



Validation Attribute → on client side

•) Required
•) Min - length.

etc.

Lecture :- 23

(Validation with strongly Binding)

•) ~~Just~~ use eager in view, ~~&~~ all remaining things are same.

Lecture :- 24

(Validation Summary)

•) It is only used for showing error message in view.

@Html.validationSummary(~~[illegible]~~)();

Lecture :- 25

(custom Validation)

•) It means we make some our validation attribute.

Creating custom validation :-

•) we have to implement validationAttribute class.

•) validation attribute is available in

System.ComponentModel.DataAnnotation namespace.

•) Then we have to override IsValid method.

Lecture :- 26

(layout in MVC)

- @RenderBody() will replace ~~the~~ another view for other controller.
- If ~~we need~~ did not want to add layout Path in every common layout, then add it into _ViewStart.cshtml file.

Lecture :- 27

(Multiple layout in MVC)

1) First way if have multiple layout

return view("Viewname", "layoutname").

2) Second is to manually write layout view Path in a Particular view of that controller.

Lecture :- 28

(Section in Layout)

To create space on layoutfile we use @RenderSection()

@RenderSection("Section Name", required: true).

↳ means compulsory.

```
@section Section Name
{
}
```

Lecture:-29 (@RenderPage)

•) @RenderPage can be used in any of view whether it is normal or shared layout.

@RenderPage("Path of the view"):

@RenderPage("Path", "Roy", "ankona") → this can be anything because it is object type.

↳ no of Parameters u can pass.

To ~~ka~~ use this add

var data = Page → use in view.

↳ already defined Property

Lecture:-30 (Entity Framework)

•) It connect our application with database.

•) class library:- which can not run individually, it runs with some Project.

•) To use this class library project in another project, first add reference of it.

•) Install Entity framework in a Project through nuget Package

Entity framework contains three approach:-

1) Database first approach. → After this edmx file will add in a project.
2) code first
3) Design first.

•) Entity framework use DB context.

•) context.cs ⇒ all tables are linked in this file.

Lecture:-31 (Save data in DB)

creating connection with database