**To create MVC project from CLI:**

* To check all the types of projects and there short name and default language used(c#, f#, vb)

Use Command **: dotnet new**

* Now to create Mvc Project with the name FirstMvcApp use the command: **dotnet new mvc --name FirstMvcApp**
* To run the app use the command : **dotnet run**

**Convert Console Application to Web Application:**

**Steps:**

* + Setup host builder for Application
  + By default every web application is a console application in dotnet core
  + create basic console base app using (.net core)
  + Update the SDF in .csproj file to open it double click on project and change following

<Project Sdk="Microsoft.NET.Sdk">" to <Project Sdk="Microsoft.NET.Sdk.Web">

* + A host is an object that encapsulates the resources of an application

e.g. Dependency Injection, Logging, Configuration, IHostedService implementation

* + We need A host builder to run our application
    1. So Host.CreateDefaultBuilder() gives the instance of that class

It basically set a default path for that app, host the environment variables

All configuration data form application.json we can use here

It is also used for default logging and event source output

Enables scope validation on the dependency injection

**Middlewre:**

* + Asp.net core create an HTTP application pipeline that processes the request
  + Http pipeline is configures in Configure method of Startup.cs
  + All request to the aap goes through the Http pipeline
  + An middleware is a piece of code which is used in Http pipeline
  + Middleware has access to all the request and response.
  + App can use multiple middleware.

E.g. We can use middleware for logging for each request or we can use in authorization of login logout request

**launchSettings.json:**

* It is present in properties forder
* It contains info about in which server you are running the app like IIS Express or or the default server having the name as project name which runs the server Kestro created when we create the project
* It also contains the Environment Variable which contains the info about environment like production, development of your custom name of environment
* It contains info about launch the app into browser or not
* It contains the info of default port
* If you are running in http you can change port in applicationUrl
* If running in https then set port in sslPort key

**Environment variables:**

**ASP**.**NET Core** uses an **environment variable** called ASPNETCORE\_ENVIRONMENT to indicate the runtime **environment**. The value of this **variable** can be anything as per your need but typically it can be Development, Staging, or Production. The value is case insensitive in Windows and Mac OS but it is case sensitive on Linux.

To check the name of environment:

IWebHostEnvironment env;

env.EnvironmentName // it will show the name of environment which is in launchSettings.json

To check the environment use

If(env.IsDevelopment()){

//code

}

If(env.IsProduction ()){

//code

}

If(env.IsStaging ()){

//code

}

If(env.IsEnvironment (“Develop”)){ **//To check Custom environment Name**

//code

}

**How to setup MVC in dotnet core Application:**

MVC setup depends on template used while creating project

Using Asp.net core we can create different type of app

**Web Api:** In this we need only Models and controllers but not views

**Razor Pages:** we can create cshtml file and it is concept in this there is no concept of model view and controller

**MVC:** If it is MVC template then it will come automatically if it is empty project then we have to setup manually

**Empty templates :** Empty templates can be convert in any of the above type

Adding dependency in empty template for MVC

Steps:

* Go to startup.cs file
* Add dependencies in ConfigureServices(IServiceCollection services) method
* All the services and dependency for the project is added here
* To add the dependencies there are few ways

services.AddMvc();

or

services.AddControllersWithViews();

And to allow controller to show use

In Startup.cs -> Configure()

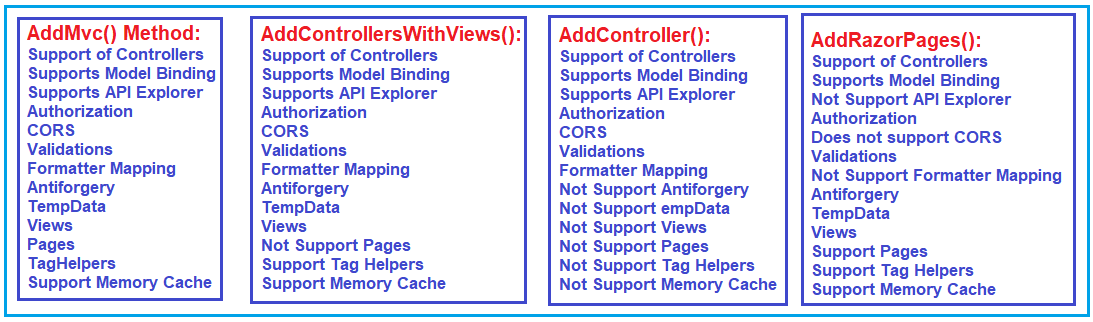
app.UseEndpoints(endpoints =>

{

endpoints.MapDefaultControllerRoute();

});

**AddMvc(), AddControllers(), AddControllersWithViews(), AddRazorPages():**

****

**Uses of all above:**

* Web API- AddControllers()
* Razor Pages – AddRazorPAges()
* MVC - AddControllersWithViews()
* Else – AddMVC() //it contains all from above and some extra functionallity

**Controllers:**

* Is a special class with .cs(for C#) extension
* In MVC template Controllers class inherited from Controller class
* A controller is used to group actions(Action methods)
* A controller is used to handle an coming Http Request
* Mapping of Http Request is done using Routing
* Caching, Security etc. can also be applied on controllers

**Action Methods in Controller;**

* All Public methods of controllers class are known as Action methods
* Because they are created for specific actions in the app
* Action method returns several types
* When we get Http Call on controller, we actually are getting this on a particular action method.

**Note:** We should create new controller when we need to define a new group of operation into app

**Model:**

* Is a class with .cs(for C#) extension
* We generally have c# properties in one model e.g. id , name, etc
* Is responsible for data
* We get/set data from /to a source in form of model

**View:**

* A view is a file with .cshtml(for c#) extension
* .cshtml (c Sharp + html)
* View are generally returned from action method
* Return type for action method is ViewResult

**Location of View in app**

* Are available inside Views folder at root
* Usually views are grouped into folder name with app controller.

**View discovery:**

* When we return a view from action method of controller then the view with same name as action method will be returned and it is because of view discovery

**View Engine:**

* Responsible for differentiate c# and Html code in Html page and only show the Html code to browser
* It is a piece of code which is used to render server side code into view
* It works with Views

**Razor ViewEngine:**

* It is a View Engine
* It is responsible to look the view inside the View/folder same as Controller name or View/Shared folder
* If we have to define the view in some your location then ewe have to return the view with the path e.g.

return View("../../TempView/Temp"); //It is relative path

or return View("~/TempView/Temp.cshtml"); // It is absolute path

**Static Files:**

* Images, Css, Javascript, etc.
* By default all static files are place inside wwwroot folder this folder also know as **Content Root** **folder**.
* Note: we can not use static files direactly we have to tell our application and add in the

Startup.cs file a middleware

i.e. app.UseStaticFiles(); inside Configure() method

* Note: If we save static files some other folder(not in wwwroot) than it will not work directly
  + For this we have to tell our app again about it

app.UseStaticFiles(new StaticFileOptions()

{

FileProvider = new PhysicalFileProvider(Path.Combine(Directory.GetCurrentDirectory() , "MyStaticFiles")),

RequestPath = "/MyStaticFiles" //Note: here “MyStaticFiles” is new folder for static files

}) ;

**Library Manager:**

* To add External files like Jquery, Bootstrap etc;
* Also know as LIbMan

**Razor file compilation:**

* By default Razor file(.cshtml) is compiled 2 time
  + Build: when we build the project then entire code in view files converted into pure html
  + Publish: When we publish the solution

So, when we run the app and made any changes in view file it will not reflect in view as all view pages compiled at runtime. So, we have to stop our app again and run it again.

It is a tedious tast to stop and run again to the app

Solution:

The solution is add a new package:

Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation

Now add this service into startup.cs by adding and note it should not done in production means if any change made should not reflect directly in production only In development

So, we have to add this condition only for development purpose

#if DEBUG

services.AddRazorPages().AddRazorRuntimeCompilation();//in ConfigureServices() method

#endif

**What is Layout:**

* A Layout is a .cshtml file that is used to provide common structure to other views
* It is kind of Master page which contains all common code like footer static files etc.

**@RanderBody()**

Is used inside layout file to provide space for the other view

We can have only one RenderBody() method inside one layout

**What is Section and RenderSection?**

It is used in scenario like we have added jQuery file in \_Layout file and in any view we used Jquery code and as if the jQuery in layout file is at bottom then it may load for the view page and we try to use jQuery before it loaded so we create a Render section in layout to tell view that use this section and what ever you write in that section in view file will be appear that section of layout file

* **RenderSection** is a spce with a specific name and it used on \_Layout file
* RenderSection tell the application that some other code (coming from view) will be placed here.
* **Section** is used in views
* To create a section we use @section directive
* Each section has a unique name and whatever we will write inside this section block that will replace RenderSection (defined in \_layout file) with same name.

**\_ViewStart.cshtml**

* As the extension (.cshtml) it is relate to view part
* Underscore denotes it is the common view to reusability of code
* The Main deference b/t this view file and other view is that when we run any app and we render any view before any view this \_viewStart.cshtml file get executed and whatever code is written in \_viewStart file will be used in rendering view file
* So, it contains common view for the view part
* E.g. Like we use

Same layout file so we have to define

@{

Layout = “\_LayoutName” //We have to use this in all view so we can separate this using viewStart file

}

If you want to overwrite the view or not want to use the common layout we can over write that by using same code in that particular view Like

@{

Layout = null; //Null or some other view

}

**Note:** We can also create \_viewStart for particular folder level just right click on folder -> add item -> search Razor -> Select view start file

**\_ViweImport.cshtml:**

* It is samething like \_viewStart.cshtml it is use to write the common Using Directive for the views
* It will run before all view render
* So, It is used to write common directives

E.g.

@addTagHelper

@removeTagHelper

@tagHelperPrefix

@using

@model

@inherits

@inject

**Note**: Scope of \_ViewImport applied to current folder( in which \_ViewImport file is placed) and to all the sub folders.

**Note**: We can not change \_ViewImport and \_ViewStart.cshtml file name

**ViewBag:**

* It is used to pass data from action method to view and we can display this data on view.
* This type of binding is know as loosely binding(means binding of data is loosely to the view)
* We can pass any type of data in viewBag
* It uses Dynamic property that’s why it is called loosely binding

**e.g**  ViewBag.PropertyName = Data;

* It is a server side code hence to use it on view we need to use razor syntax i.e. @

E.g. @ViewBAg.Property

Note: we can not pass anonymous type to ViewBag like

ViewBag.Data = new {Id=1, Name=”Nitish”}

If you want To do so we have to add

using System.Dynamic;

and use like as follow

dynamic data = new ExpandoObject();

data.Id = 1;

data.Name = "Avinash";

ViewBag.Data = data;

ViewBag.Type = new BookModel() {Id = 5, Author = "This is Author" };

Use it in view page like @ViewBag.Type.Id

* **Note:** It is used to send multiple data on view then ViewBag is the easiest option
* It is works on dynamic type it does not give compile time error(this might be a trouble sometime)

**ViewData:**

* It is used to pass data from action method to view and we can display this data on view
* This type of data binding is know as loosely binding
* It works in key Value pare
* We can pass any type of data
* It uses ViewDictionary
* It is same like ViewBag except it works as key value pare

E.g ViewData[“PrpertyNAme”]= Data;

* While sending object in viewData then before using it data on view we need to cast it to its data type
* It can also be used to pass data from view to its layout view

e.g First define the ViewData in Action methosd

ViewData["book"] = new BookModel() { Author = "Nitish", Title = "Home " };

//Use them into view as follow

@{

var bookInfo = ViewData["book"] as BookModel; //We need to cast data of specific type

}

<p>

@bookInfo.Author

@bookInfo.Title

</p>

**ViewData Attribute:**

* It is same as ViewData the only difference is in there use

e.g.

[ViewData]

public string CustumProp { get; set; }

public ViewResult Index()

{

CustumProp = "Demo Attr";

return View();

}

To use it in View syntax: @ViewData["CustumProp"]

**Dynamic Views Display Model Data without @model:**

* It is used to add some extra data with our Model data
* Example

public ViewResult GetBook(int id)

{

// Create dynamic type object

dynamic data = new ExpandoObject();

// Get data from database and add in key of dynamic data obj

data.book = \_bookRepository.GetBookById(id);

// Add some extra data with Our database data

data.Name = "Avinash";

return View(data) ;

}

**To use this data in view use**

@Model.Name

@Model.book.Title

@Model.book.Id

@Model.book.Author

**Disadvantage Dynamic Views**

* No compile time error

//Like if use @ Model.book1.Title then it will not give error at compile time we get error runtime

* Not a good architecture
* Generally developer avoid using dynamic view

**Tag helpers**

* It is used to render server side code o Razor(.cshtml) file to create ad render Html element
* **Examples**

Create Image Tag

Create form tag

Create link tag

* In Asp.Net MVC we were having HTML helpers and these HTML helpers were a nightmare for the developer

How to start with Tag Helpers

* First we need to set the scope for the Tag helper
* This scope can be set by using:
  + @addTagHelper directive
  + @removeTagHelper directive

Syntax: @addTagHelper tagHelper, Assebly

e.g.

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers // \* means include all tag helpers

* If you need heplers only on one page then you can use it n you view file directly
* If you need in entire app then the best to define it is \_ViewImport.cshtml file

**Anchor Tag helper:**

* It is used to create a link using <a></a> html element.

**Problem with simple <a>**

If we have something like this  
<a class="nav-link text-dark" href="/home/aboutus">About</a>

In this before /home/ default rout will come like <http://localhost:53768/home>

Automatically

But what happen if we change the default route like

endpoints.MapControllerRoute(

name: "Default",

pattern: "bookApp/{controller=Home}/{action=Index}/{id?}"

);

Now Default route contains bookApp Part It will become

[**http://localhost:53768/bookApp/home**](http://localhost:53768/bookApp/home)

but in simple <a> tag href still search for <http://localhost:53768/home> so it will show error

**Solution** (Is Anchor tag helper)

* The main feature of it is to navigate to specific page.
* Foe this we have to add following attribute in <a> tag
  + asp-controller : name of the controller for href(like Home as in above)
  + asp-action: name of action method(like aboutus as in above)
  + asp-area

e.g:

<a class="nav-link text-dark" **asp-controller**="Home" **asp-action**="Index">Home</a>

Now even if we change the default route it will automatically add in the url and try to find only controller and action name

**How to pass parameter using Anchor tag**

* Attribute
  + Asp-route-{paramName} //in the braces parameter name which you want to pass

<a href=”[/book/getbook/@book.Id](mailto:/book/getbook/@book.Id)”

class="btn btn-sm btn-outline-secondary">View details</a>

Same as below

//Note Here in asp-route-Id Id is the Name of Parameter which we pass

<a **asp-controller**="book" **asp-action**="getbook" **asp-route-Id**="@book.Id"

class="btn btn-sm btn-outline-secondary">View details</a>

If we want to pass multiple parameters we can use another asp-route-NewParameterName

But if we have too many parameter to pass the we have another attribute to

i.e. **asp-all-aoute-data**

it will pass data in url

e.g @{

var parameters = new Dictionary<string, string>()  
 {“Name”,”pqr”},

{“company”,”xyz”},

{“appname”,”lmp”},

}

<a **asp-controller**="book" **asp-action**="getbook" asp-all-route-data=”@parameters”></a>

To pass the id of particular id of a page to directly move to that part of page we can use pass fregment

**asp-fragment**="similarBooks" //where similar book is the id of div to focus of the url page

**Pass complete URl using Anchor tag**

Attribute : asp-protocol and asp-host

e.g. <a asp-protocol=”http” asp-host=”xyz.com”></a>

**Compatible with routing**

**Attribute**

Asp-route

e.g.

//Creted a route for action method and give it a name

[Route("book-details/{id}", Name ="bookDetailsRoute")]

public ViewResult GetBook(int id)

{

dynamic data = new ExpandoObject();

data.book = \_bookRepository.GetBookById(id);

data.Name = "Avinash";

return View(data) ;

}

And now want to use this rout in <a> tag then use as follow

And in this way we don’t need to give asp-controller and action attribute

e.g.

<a asp-route="bookDetailsRoute" asp-route-id="@book.Id"

class="btn btn-sm btn-outline-secondary">View details</a>