MVC :

M > Model

V > View

C > Controller

Pattern in which all the different parts of an application are divided into different components

And all these are loosely coupled

There is a separation of concerns from the beginning

MVC

Restaurant >

Cook Manager Waiter can be some person

Controller (Manager) (Request Handler)

View (Waiter) (cshtml pages)

Model (Cook) (Domain / Data)

When request comes from browser , it is taken by controller , in controller, we have action methods, The request from browser goes to some action method in the controller. After that that action method will return some output, which could be view also

Controller is a class which inherits from Controller class

Controller contains public Action Methods

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

namespace WebApplication4.Controllers

{

public class FirstController : Controller

{

public IActionResult Index()

{

return View();

}

public IActionResult first()

{

return View();

}

public string second()

{

return "This is a string";

}

public IActionResult third()

{

return RedirectToAction("Index");

}

public IActionResult fourth()

{

return RedirectToAction("index", "home");

}

}

}

How to pass data from controller(action method) to view

1.ViewBag

2.ViewData

# TempData

ViewBag is dynamic object

ViewBag.dept =”HR”;

ViewData is a dictionary

ViewData[“name”]= “aaaa”;

-----------------------------------------

public IActionResult fifth()

{

ViewBag.data = "Batch Code is B001";

ViewData["name"] = "Ajay Sood";

return View();

}

@{

ViewData["Title"] = "fifth";

}

<h1>fifth</h1>

We use @@ where ever Server side code is needed <br />

<h1> @ViewBag.data </h1>

<br /> <br /><br /><br /><br /><br />

Name is @ViewData["name"]

Model > Data / Domain

**Model**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace WebApplication4.Models

{

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

public string Batch { get; set; }

public int Marks { get; set; }

}

}

**Controller Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using WebApplication4.Models;

namespace WebApplication4.Controllers

{

public class StudentController : Controller

{

public IActionResult Index()

{

Student student = new Student() { Id = 1, Name = "Deepak", Batch = "B001", Marks = 90 };

ViewBag.student = student;

return View();

}

}

}

**View**

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<h3> Studnet Details </h3>

<pre>

ID : @ViewBag.student.Id <br />

Name : @ViewBag.student.Name <br />

Batch Code : @ViewBag.student.Batch <br />

Marks : @ViewBag.student.Marks <br />

</pre>

**Practice**

Create Controllers User

Add action Methods

AboutUs , ContactUs

Add Views for them

Also create Employee Class and pass its object from Employee Controller to Index View of Employee Controller

**We can also pass data from controller’s action method to view as objects**

**Controller**

public IActionResult Display()

{

Student student = new Student() { Id = 1, Name = "Deepak", Batch = "B001", Marks = 90 };

**return View(student);**

}

----------------------------------------------------------------

**View**

@\*This "@model WebApplication4.Models.Student

" statement indicates that this view has received an object of following class\*@

**@model WebApplication4.Models.Student**

@{

ViewData["Title"] = "Display";

}

<h1>Display</h1>

ID : @Model.Id

Name : @Model.Name

Batch Code : @Model.Batch

Marks : @Model.Marks

Here @Model is a keyword @Model means current object

**Such a view in which we pass an object is known as Strongly typed View**

**Why? Because that view is based on some object**

**It provide Intellisense.**