Asp.net mvc based on the mvc design patterns allows you to develop software solution

Enables separate data access business and presentation logic from each other

Asp.net mvc is loosly coupled the logic separate in 3 parts

Model

Models are basically class in C#

Data base operation such as fetch data and update data

Represent information about domain that can be the application data of web application

View

View is end user gui which user can interact with the system

Represent the presentation logic provides the data of the model

Controller

Controller are class C#

Contain the logic and provide the link between view and model

Handle the request to the application by the user

View data

|  |  |
| --- | --- |
| public ActionResult Index()  {  ViewData["Message"] = "hello";  return View();  } | @{  ViewBag.Title = "Index";  }  <h2>@ViewData["Message"];  </h2> |

|  |  |
| --- | --- |
| Foreach  public ActionResult Index()  {  string[] fruits = { "apple", "banana", "strawberry" };  ViewData["fru"] = fruits;  return View();  } | <ul>  @{  foreach (var i in (string[])ViewData["fru"])  {  <li>@i</li>  }  }  </ul> |

|  |  |
| --- | --- |
| public ActionResult Index()  {    ViewData["sport"] = new List<string>()  {  "cricket",  "hockey",  "football"  };  return View();  } | @{  foreach (var i in (List<string>)ViewData["sport"])  {  <li>@i</li>  }  } |

|  |  |
| --- | --- |
| Employee ali = new Employee();  ali.Id = 1;  ali.Name = "mangal";  ali.disignation = "manager";  ViewData["emp"] = ali; | @{  var a = (WebApplication10.Models.Employee)ViewData["emp"];  }  <p>  @a.Name  </p>  <p>  @a.Id  </p>  <p>  @a.disignation  </p> |

Viewbag same as view data

Syntext is

Viewbag.message = “Hello”;

Use in view like

Viewbag.message

|  |  |
| --- | --- |
| Employee ali = new Employee();  ali.Id = 1;  ali.Name = "mangal";  ali.disignation = "manager";    ViewBag.emp = ali; | <div>  @{  var b = ViewBag.emp;  }  </div>  <p>@b.Id</p>  <p>@b.Name</p> |

Viewbag data can be access by viewdata also like this

|  |  |
| --- | --- |
| Employee ali = new Employee();  ali.Id = 1;  ali.Name = "mangal";  ali.disignation = "manager";    ViewBag.emp = ali; | viewbag  <div>  @{  var b = ViewBag.emp;  }  </div>  <p>@b.Id</p>  <p>@b.Name</p>  Viewdata  var a = (WebApplication10.Models.Employee)ViewData["emp"]; |

TempData

Tempdata syntax same as viewdata

Tempdata[“name”] = “message or object or array”

|  |  |
| --- | --- |
| TempData["Message"] = "hello"; | <h2>@TempData[“Message”]</h2> |

Tempdata can access from one controller to anther controller

Session

|  |  |
| --- | --- |
| Session[“key”] = “message” | @session[“key”] |

Html Helper

ActionLink

Html.BeginForm() and Html.EndForm()

Html.Label()

Html.TextBox()

Html.TextArea()

Html.Password()

Html.CheckBox()

Html.RadioButton()

Html.DropDownList()

Html.Hidden();

<div>@Html.ActionLink(“name”,”Controllername”)</div>

@html.TextBox(“name”);

@html.TextBox(“name”,””,new{style=”background:red;”})

@Html.TextBox(“name,””,new{@class=”m-10 container”})

<https://github.com/frankiefab100/Blockchain-Development-Resources>

@{

Html.BeginForm(“Action”,”Controller”);

}

@Html.Label(“username”) &nbsp; @Html.TextBox(“username”)

<br />

<br />

@Html.Label(“Password”) &nbsp; @Html.Password(“Password”)

<br />

<br />

@Html.Label(“Male”) &nbsp; @Html.RadioButton(“Gender”,”Male”)

@Html.Label(“Female”) &nbsp; @Html.RadioButton(“Gender”,”Female”)

<br />

<br />

@Html.Label(“Select Country”) &nbsp; @Html.DropDownList(“CountryList”,new SelectList(new[] {“india”,”pakistan”,usa}),”select”)

@Html.Label(“Comment”)&nbsp; @Html.TextArea(‘CommentArea’ ,””,rows Number,column Number,null)

@html.CheckBox(“AcceptTerm”) &nbsp; html.lable(“I Accept Terms”)

@{

Html.EndForm();

}

<a href=”@ Url.Action(“Action”,”Controller”)” > got to about page</a>

Strongly typed view

Return view(object)

On top

@model project.models.Employee

<h1>@Model.id</h1>

<h1>@Model.age</h1>

<h1>@Model.name</h1>

Using(html.Beginform()){

}