MVC

Global.asax contains :   
AreaRegistration.RegisterAllAreas();

WebApiConfig.Register(GlobalConfiguration.Configuration);

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

ViewData is property of controller that exposes an instance of the ViewDataDictionary class

ViewBag is a dynamic property (dynamic keyword which is introduced in .net framework 4.0).

In ASP.NET MVC, we have many built-in *ActionResults* type:

* ViewResult
* PartialViewResult
* RedirectResult
* RedirectToRouteResult
* ContentResult
* JsonResult
* EmptyResult
* JavascriptResult
* FileResult
* HttpStatusCodeResult
* ViewResultBase

public ActionResult Index()

{

      List<string> Student = new List<string>();

      Student.Add("Jignesh");

      Student.Add("Tejas");

      Student.Add("Rakesh");

      ViewData["Student"] = Student;

      return View();

}

//page code

<ul>

    <% foreach (var student in ViewData["Student"] as List<string>)

        { %>

    <li><%: student%></li>

    <% } %>

</ul>

//Controller Code

public ActionResult Index()

{

      List<string> Student = new List<string>();

      Student.Add("Jignesh");

      Student.Add("Tejas");

      Student.Add("Rakesh");

      ViewBag.Student = Student;

      return View();

}

//page code

<ul>

    <% foreach (var student in ViewBag.Student)

        { %>

    <li><%: student%></li>

    <% } %>

</ul>

ViewData is a ViewDataDictionary, not just a generic Dictionary. One reason for this is that it has an additional Model property to allow for a speciﬁ c model object to be available to the view. Since there can be only one model object in ViewData, it’s convenient to use this to pass a speciﬁ c class to the view. This allows your view to specify the class it is expecting the model object to be, which allows you take advantage of strong typing.

if you had a shopping cart summary page that needed to display a list of products, the total cost for the cart, and a message to the user, you could create the ShoppingCartViewModel class, shown as follows: public class ShoppingCartViewModel {

public IEnumerable<Product> Products { get; set; }

public decimal CartTotal { get; set; }

public string Message { get; set; }

}

Now you can make a view strongly typed to this model, using the following @model directive: @model ShoppingCartViewModel This gives you the beneﬁ ts of a strongly typed view (including type checking, IntelliSense, and freedom from having to cast untyped ViewDataDictionary objects) without requiring any changes to the Model classes.

|  |  |
| --- | --- |
| ASP.NET Web Forms | ASP.NET MVC |
| ASP.NET Web Forms uses Page controller pattern approach for rendering layout. In this approach, every page has it’s own controller i.e. code-behind file that processes the request. | ASP.NET MVC uses Front Controller approach. That approach means ,a common controller for all pages, processes the requests. |
| No separation of concerns. As we discussed that every page (.aspx) has it’s own controller (code behind i.e. aspx.cs/.vb file), so both are tightly coupled. | Very clean separation of concerns. View and Controller are neatly separate. |
| Because of this coupled behavior, automated testing is really difficult. | Testability is key feature in ASP.NET MVC. Test driven development is quite simple using this approach. Please [follow here](http://msdn.microsoft.com/en-us/magazine/dd942838.aspx) for demo on building testable applications. |
| In order to achieve stateful behavior, viewstate is used. Purpose was to give developers, the same experience of a typical WinForms application. | ASP.NET MVC approach is stateless as that of the web. So here no concept of viewstate. |
| Statefulness has a lots of problem for web environment in case of excessively large viewstate. Large viewstate means increase in page size. | As controller and view are not dependent and also no viewstate concept in ASP.NET MVC, so output is very clean. |
| ASP.NET WebForms model follows a Page Life cycle. | No Page Life cycle like WebForms. Request cycle is simple in ASP.NET MVC model. |
| Along with statefulness, microsoft tries to introduce server-side controls as in Windows applications. Purpose was to provide  somehow an abstraction to the details of HTML. In ASP.NET Web Forms, minimal knowledge of HTML, JavaScript and CSS is required. | In MVC, detailed knowledge of HTML, JavaScript and CSS is required. |
| Above abstraction was good but provides limited control over HTML, JavaScript and CSS which is necessary in many cases. | Full control over HTML, JavaScript and CSS. |
| With a lots of control libraries availability and limited knowledge of other related technologies, ASP.NET WebForms is RAD(Rapid Application Development) approach. | It’s a step back. For developers decrease in productivity. |
| It’s good for small scale applications with limited team size. | It’s better as well as recommended approach for large-scale applications where different teams are working together. |

TempData vs ViewBag vs ViewData

Action Helper in MVC

@Url.Action(“Login”,”Account”); to generate url = /Account/Login

@Html.ActionLink : To generate anchor tag with href and anchor text in MVC

@Html.DispalyNameFor :

@Html.DisplayFor :

@Html.Partial() : for partial html pages

@RenderBody()

@RenderSection()

Data Annotation in ASP.Net MVC – For validations

System.ComponentModel.DataAnnotations

[Required(ErrorMessage = "Please Enter Email Address")]

[RegularExpression(".+@.+\\..+", ErrorMessage = "Please Enter Correct Email Address")]

[RoutePrefix("MyHome")]

Controller based routing in ASP.net MVC5

[Route("{action=index}")] //default action

public class HomeController : Controller

{

//new route: /MyHome/Index

public ActionResult Index()

{

return View();

}

//new route: /MyHome/About

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View();

}

//new route: /MyHome/Contact

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

}

Action Level Routing

[Route("users/about")] //route" /users/about

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View();

}

*routes.MapMvcAttributeRoutes()* method with in *RegisterRoutes()* method of RouteConfig.cs file.