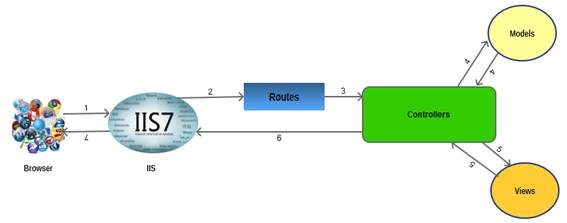
**What is MVC?**

ASP.NET MVC helps us to develop powerful and pattern-based dynamic websites that enables a clean separation of concerns and also gives you a full control on a markup. First time it was implemented by Trygve Reenskaug at 1979 and it was implemented on Smalltalk at Xerox labs. Also it includes many features that help us to develop a sophisticated and modern web application.

Here **M**stands for Model, **V**stands for View and **C** stands for controller.



**Controller: -** The controller is like a traffic cop.Whenever user requests any resource it first goes to the controller. Controller than interact with both the Model and View.

**Model: -** The model is responsible for the data for the application and also it creates data for the view. It handles data processing and database works part.

**View: -** It is a presentation layer i.e. it shows the data to the users.

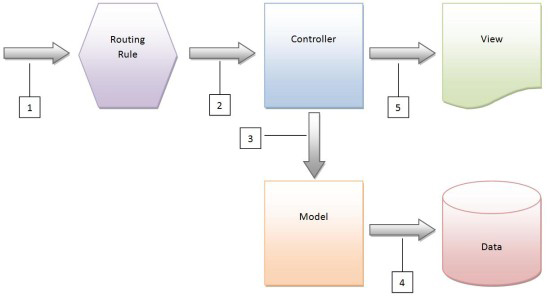
**Advantages of MVC**   In 3-tier architecture, Separation of concern i.e. independent of UI and Business layer. So Business logic is independent and can be used from different presentation layers.

**Disadvantages of MVC**   In a View, managing a state (i.e. View state) is a painful process. But it can be achieved by using JavaScript/Ajax in the view page.

**Request Flow:**

Request Flow handles the request from the user ( client ) and send it to the server. Lets look into details on the function flow internally when client raises a request. This is what happens when u type an asp .net mvc application URL into the address bar of a web browser & clicks on enter. 

Request -->Routing --> Handler -->Controller --> Action --> View --> Response



Detailed Look at the Request Flow

1. Request comes into ASP.NET

2. ASP.NET Routing finds the route match by calling RouteCollection.GetRouteData

3. This in turn calls RouteBase.GetRouteData on each route until it finds a match

4. The IRouteHandler for the matching route has its GetHttpHandler method called

5. The MvcHandler runs (ProcessRequest is called)

6. The MVC controller factory locates and creates the controller in CreateController

7. The ControllerActionInvoker determines which action to run in InvokeAction

8. The AuthorizationFilter stage executes (this includes the authorization method on the controller itself)

9. The ActionExecuting stage executes

10. The requested action method is executed

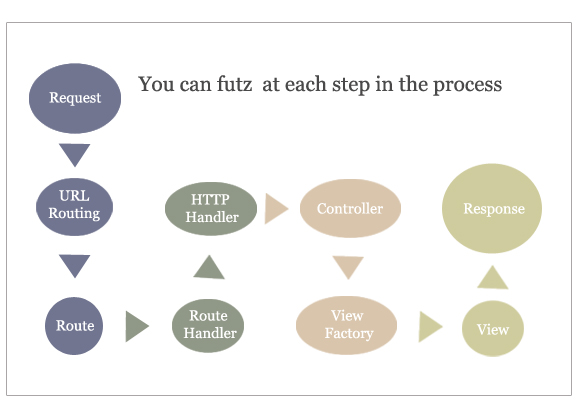
11. The ActionExecuted stage executes

12. If there is a result object then the ResultExecuting stage executes

13. If the result wasn't cancelled then the ActionResult's ExecuteResult method is executed

14. The ResultExecuted stage executes

15. If an error occured then the Exception stage executes



Its important that you understand how the request is handled. Now we understand the importance of the following basic elements in an asp .net application.

## What is a View Engine?

View Engines are responsible for rendering the HTML from your views to the browser. The view engine template will have different syntax for implementation. Currently there are few number of view engines available for MVC and the top four view engines are Razor, traditional ASPX, Spark and NHaml.

If you really want then you can use multiple view engines in parallel in ASP.NET MVC.

In this article we will do a basic go through with some of the view engines.

**Note**: All view engines may not support all versions of ASP.NET MVC. Please check the corresponding sites to get the latest updates.

**View Engine types:**

**For example, a code block in ASPX might look like this:** 

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/585873/Basic-Understanding-On-ASP-NET-MVC-4)

<% foreach(var item in Model) { %>

<tr>

<td><%: item.Name %></td>

</tr>

<% } %>

**Whereas the Razor equivalent will look like this:** 

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/585873/Basic-Understanding-On-ASP-NET-MVC-4)

@foreach(var item in Model) {

<tr>

<td>@item.Name</td>

</tr>

}

 A view engine is what **MVC**uses to find and render the views you are requesting from the controller. If you are satisfied with the default routing you probably won’t need to change anything, but let’s say you wanted to have your shared files usually located in root/views/shared to instead be located in root/common, a custom view engine is what you will need to create to be able to do that.

To Build a Custom View Engine: - [**http://coderjournal.com/2009/05/creating-your-first-mvc-viewengine/**](http://coderjournal.com/2009/05/creating-your-first-mvc-viewengine/)

**Different types of View Engine:**

**Razor :-**http://weblogs.asp.net/scottgu/archive/2010/07/02/introducing-razor.aspx

**ASPX :-** http://aspnet.codeplex.com/

**Spark :-** http://sparkviewengine.com/

**NHaml :-** http://code.google.com/p/nhaml/

**NDjango :-** http://ndjango.org/index.php?title=NDjango\_Home

**Hasic :-**http://www.codeproject.com/Articles/467850/ASP-NET-MVC-view-engines

**Brail :-** http://mvccontrib.codeplex.com/wikipage?title=Brail&ProjectName=mvccontrib

**Bellevue :-** http://www.ope.ag/Bellevue/Page/intro

**SharpTiles :-**http://www.sharptiles.org/

**String Template :-**  http://code.google.com/p/string-template-view-engine-mvc/

**Wing Beats :-**  http://wingbeats.codeplex.com/

**SharpDOM :-** http://sharpdom.codeplex.com/

**Project Templates in MVC 4:**

**Empty: -** The Empty template created the minimum references and resources required to run an Asp.net MVC application. As you can see in below image,  Models, Controllers, App\_Data are completely empty folders and View folder only contains web.config and a Global.asax file and web.config. App\_Start folder contains 3 configuration files ( FilterConfig.cs, RouteConfige.cs and WebApiConfig.cs ).

**Basic: -** It’s a new project type in MVC 4 and it was not available in MVC3. Basic project is much much closer structurally to Empty project in MVC3. It includes Contentand Scripts as well as few more references. Here is a short list of what’s included

jQuery

jQuery UI

jQuery Validation

modernizr

KnockoutJS

Antlr 3

Entity Framework

WebGrease

Bundling and minimization facilities have been prepared in this template. BundleConfig.cs file has been added to App\_Start folder. Additionally, what had been disappeared in Views folder came back to the board. \_Layout.cshtml includes the jQuery bundle as well as the default theme styling.

**Internet Application :-** Adds tow controllers( Account controller and Home controller)  to the Basic template with implemented actions and views.  Membership management functionality which allows you register, login, change password and so on is implemented in Account controller and Home controller gives you Index, About and Contact actions working with their own related views. Its a template used usually to start a normal web project in action.

**Intranet Application :-** In fact, It’s  the Internet Application except for Membership management.  the Account controller and the web.config has been configured  to use Windows as it’s authentication method.

**Mobile Application :-**Mobile website programming is one of  most important feature in MVC 4 so this template has everything that Internet Application template has, however it  is using  jQuery.mobile instead.

**Web API  Application :-**Its another new Template in VS2012 to make it easy to develop RESTful web services and applications to feed a vast variety rage of clients from desktop browsers to tablet applications. It allows you to expose your data and service to the web directly over  Http. This template includes everything from Basic Template expect Account controller and membership functionality.

# ViewData vs ViewBag vs TempData vs Session

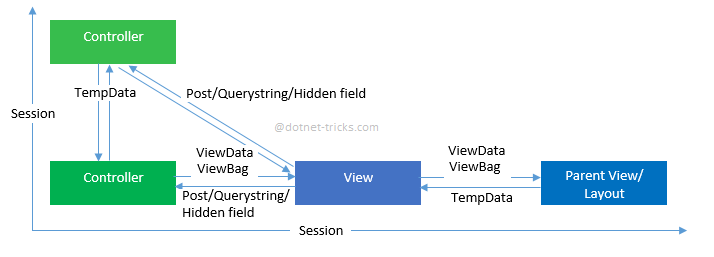
ASP.NET MVC there are three ways - ViewData, ViewBag and TempData to pass data from controller to view and in next request. Like WebForm, you can also use Session to persist data during a user session. Now question is that when to use ViewData, VieBag, TempData and Session. Each of them has its own importance. In this article, I am trying to explain the differences among these four.

## ViewData

1. View Data is a dictionary object that is derived from ViewDataDictionary class.
2. View Data is used to pass data from controller to corresponding view.
3. It’s life lies only during the current request.
4. If redirection occurs then it’s value becomes null.
5. It’s required typecasting for getting data and check for null values to avoid error.

## ViewBag

1. ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0.
2. Basically it is a wrapper around the ViewData and also used to pass data from controller to corresponding view.
3. It’s life also lies only during the current request.
4. If redirection occurs then it’s value becomes null.
5. It doesn’t required typecasting for getting data.



## TempData

1. TempData is a dictionary object that is derived from TempDataDictionary class and stored in short lives session.
2. TempData is used to pass data from current request to subsequent request (means redirecting from one page to another).
3. It’s life is very short and lies only till the target view is fully loaded.
4. It’s required typecasting for getting data and check for null values to avoid error.
5. It is used to store only one time messages like error messages, validation messages.

## Session

1. Session is also used to pass data within the ASP.NET MVC application and Unlike TempData, it never expires.
2. Session is valid for all requests, not for a single redirect.
3. It’s also required typecasting for getting data and check for null values to avoid error.

ViewData vs ViewBag vs TempData

**Introduction**   
ViewData and ViewBag are used for the same purpose to transfer data from controller to view. Both life lies only in current request. ViewData is nothing but dictionary of object and it is accessible by string as key. ViewData is property of controller that exposes an instance of the ViewDataDictionary class. ViewBag is very similar to ViewData. ViewBag is a dynamic property (dynamic keyword which is introduced in .net framework 4.0). ViewBag is able to set and get value dynamically and able to add any number of additional fields without converts it to strongly typed. ViewBag is just a wrapper around the ViewData.  
   
**ViewData Example**

//Controller Code

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>) // Casting Performed Here

        { %>

    <li><%: student%></li>

    <% } %>

</ul>  
   
**ViewBag Example**

//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) //No Casting Required

        { %>

    <li><%: student%></li>

    <% } %>

</ul>  
   
TempData is a dictionary which is derived from TempDataDictionary class. TempData is stored data just like live session for short time. TempData Keep data for the time of HTTP Request it mean that it hold data between two consecutive requests. TempData help us to transfer data between controllers or between actions. TempData internally use Session variables. Note that TempData is only work during the current and subsequent request. It is generally used to store one time message. With the help of TempData.Keep() method we can keep value in TempData object after request completion.  
   
**TempData Example**

//Controller Code

public ActionResult Index()

{

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

    Student.Add("Jignesh");

    Student.Add("Tejas");

    Student.Add("Rakesh");

    TempData["Student"] = Student;

    return View();

}

//page code

<ul>

    <% foreach (var student in TempData["Student"] as List<string>) // Casting Performed Here

        { %>

    <li><%: student%></li>

    <% } %>

</ul>

**ViewData VS ViewBag VS TempData**

|  |  |  |
| --- | --- | --- |
| **ViewData** | **ViewBag** | **TempData** |
| It is Key-Value Dictionary collection | It is a type object | It is Key-Value Dictionary collection |
| ViewData is a dictionary object and it is property of ControllerBase class | ViewBag is Dynamic property of ControllerBase class. | TempData is a dictionary object and it is property of controllerBase class. |
| ViewData is Faster than ViewBag | ViewBag is slower than ViewData | NA |
| ViewData is introduced in MVC 1.0 and available in MVC 1.0 and above | ViewBag is introduced in MVC 3.0 and available in MVC 3.0 and above | TempData is also introduced in MVC1.0 and available in MVC 1.0 and above. |
| ViewData  is also work with .net framework 3.5 and above | ViewBag  is only  work with .net framework 4.0 and above | TempData  is also work with .net framework 3.5 and above |
| Type Conversion code is required while enumerating | In depth, ViewBag is used dynamic, so there is no need to type conversion while enumerating. | Type Conversion code is required while enumerating |
| It value become null if redirection is occurred. | Same as ViewData | TempData is used to pass data between two consecutive requests. |
| It lies only during the current request. | Same as ViewData | TempData is only work during the current and subsequent request |

**Conclusion**   
We have three options ViewData, ViewBag and TeampData for passing data from controller to view and in next request. ViewData and ViewBag are almost similar and it helps us to transfer the data from controller to view whereas TempData is also work during the current and subsequent request.