MVC - Interview Questions

1) What is a MVC (Model View Controller)?

Model-View-Controller (MVC) is an architectural pattern. It separates an application into three main components. They are,

- Model => Business layer
- View => Display layer
- Controller => Input control

Model

It contains all application logic (business logic, validation logic, and data access logic).

It is responsible to

- ✓ Send data to database
- ✓ Retrieve data from database
- ✓ Store and manipulate data from database

View

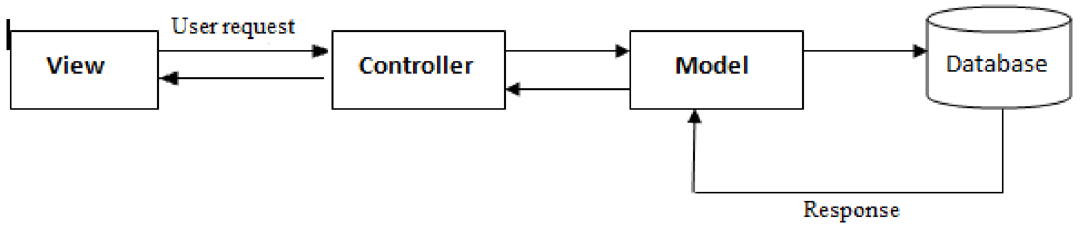
It is responsible for the look and feel. It handles the display of the data.

Controller

- ✓ **Controllers** are responsible for controlling the flow of the application execution.
- ✓ A **controller** will decide what to do and what to display in the view.
- ✓ When we make a request to MVC applications, a **controller** is responsible for returning the response to that request.
- ✓ It takes user request and based on that request it will load appropriate **Model** and **View**.

2) Explain the complete flow of MVC?

Below are the steps to control flows in MVC (Model, View, and controller) architecture:



In the above architecture, all end user requests are first sent to the **controller**. Depending on the request, the **controller** decides which **model** to load. The **controller** loads the **model** and attaches the **model** with the appropriate **view**.

The final **view** is then attached with the **model** data and sent as a **response** to the user on the browser.

3) Is MVC suitable for both Windows and Web applications?

The MVC architecture is suited for a web application. For Window applications, MVP, i.e., “Model View Presenter” is more applicable. If you are using WPF and Silverlight, MVVM is more suitable due to bindings.

4) What are HTML helpers in MVC?

HTML helpers help you to render HTML controls in the view.

For instance if you want to display a HTML textbox on the **view**, below is the HTML helper code.

<%= Html.TextBox("LastName") %>

5) What is routing in MVC?

Basically, Routing is a **pattern matching system** that monitors the **incoming request** and figure out what to do with that request.

6) How to define route in MVC?

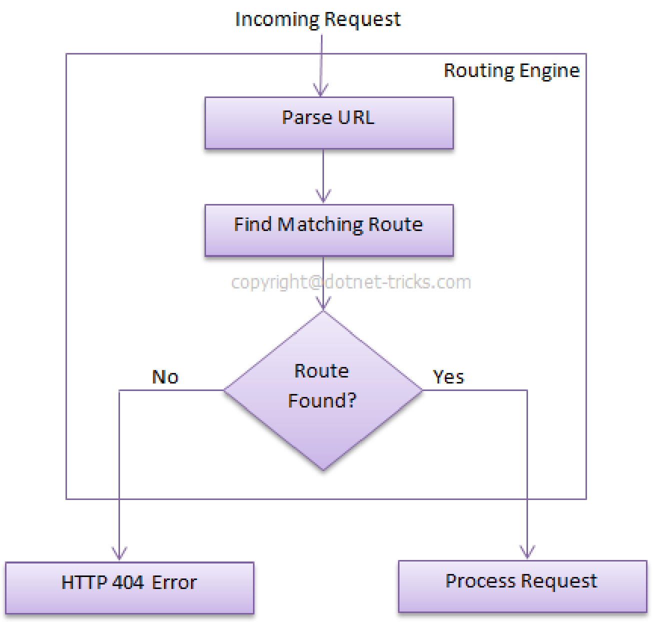
```
public static void RegisterRoutes(RouteCollection routes)
{
 routes.MapRoute(
 "Default", // Route name
 "{controller}/{action}/{id}", // Route Pattern
 new { controller = "Home", action = "Index", id = UrlParameter.Optional } // Default
values for above defined parameters
 );
}

protected void Application_Start()
{
 RegisterRoutes(RouteTable.Routes);
 //To:DO
}
```

When the routing engine finds a match in the route table for the incoming request's URL, it forwards the request to the appropriate controller and action. If there is no match in the route table for the incoming request's URL, it returns a 404 HTTP status code.

Note:

Always remember route name should be unique across the entire application. Route name can’t be duplicate.

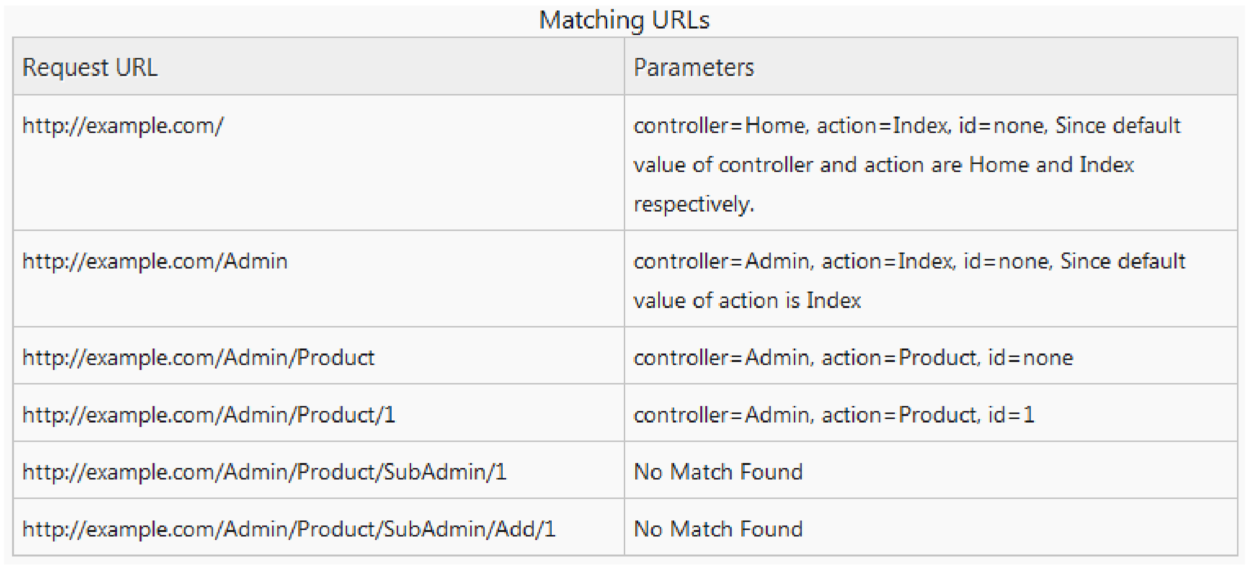
7) How routing will work in MVC?



Incoming Request Processing by Routing System

In above example we have defined the Route Pattern **{controller}/ {action}/ {id}** and also provide the default values for **controller**, **action** and **id** parameters. Default values means if you will not provide the values for **controller** (or) **action** (or) **id** defined in the pattern then these values will be serve by the routing system.

Suppose your web application is running on **www.example.com** then the **URL** pattern for you application will be **www.example.com/ {controller}/ {action}/ {id}**. Hence you need to provide the controller name followed by **action name** and **id** if it is required. If you will not provide any of the value then default values of these parameters will be provided by the routing system. I am sharing a list of URLs that match and don't match this route pattern.



Matching URLs

Request URL	Parameters
http://example.com/	controller=Home, action=Index, id=none, Since default value of controller and action are Home and Index respectively.
http://example.com/Admin	controller=Admin, action=Index, id=none, Since default value of action is Index
http://example.com/Admin/Product	controller=Admin, action=Product, id=none
http://example.com/Admin/Product/1	controller=Admin, action=Product, id=1
http://example.com/Admin/Product/SubAdmin/1	No Match Found
http://example.com/Admin/Product/SubAdmin/Add/1	No Match Found

8) Difference between Routing and URL Rewriting?

- URL rewriting is focused on mapping one URL (new URL) to another URL (old URL) while routing is focused on mapping a URL to a resource.
- Actually, URL rewriting rewrites your old URL to new one while routing never rewrite your old url to new one but it map to the original route.

9) What are the things that are required to specify a route?

There are 3 things that are required to specify a route.

URL Pattern => We can pass the variable data to the request handler without using a query string. This is done by including placeholders in a URL pattern.

Handler => This handler can be a physical file of 2 types such as **.aspx** file (or) a **controller** class.

Name for the Route =>This name is an optional thing.

10) Where is the route mapping code written?

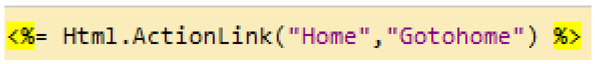
The route mapping code is written in the "**global.asax**" file.

11) Can we map multiple URL's to the same action?

Yes, we just need to make two entries with different key names and specify the same controller and action.

12) How can we navigate from one view to another using a hyperlink?

By using the **ActionLink** method as shown in the below code. The below code will create a simple **URL** which helps to navigate to the "**Home**" controller and invoke the "**GotoHome"** action.

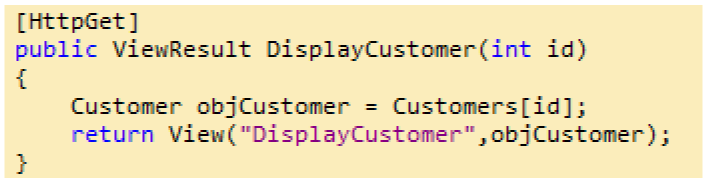


```
<%= Html.ActionLink("Home","Gotohome") %>
```

13) How can we restrict MVC actions to be invoked only by GET or POST?

Using **HttpGet** and **HttpPost** attributes to restrict the type of **HTTP** calls.

<u>Example:</u>



```
[HttpGet]
public ViewResult DisplayCustomer(int id)
{
    Customer objCustomer = Customers[id];
    return View("DisplayCustomer",objCustomer);
}
```

For instance we can see in the above code snippet the **DisplayCustomer** action can only be invoked by **HttpGet**. If we try to make **HTTP POST** on **DisplayCustomer**, it will throw an error.

14) How can we maintain sessions in MVC?

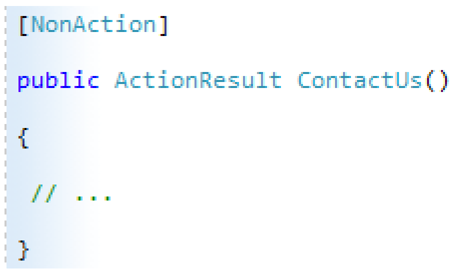
Sessions can be maintained in MVC by three ways:

- TempData
- ViewData
- ViewBag

15) To prevent a public controller method from being implicitly bound to an action name, which attribute is used?

NonAction attribute.

To prevent public controller method from being implicitly bound to action name we use **NonAction** attribute like below,



```
[NonAction]
public ActionResult ContactUs()
{
 // ...
}
```

In this case, user will not be able to call “**Contact Us”** URL to call above method.

16) Can we have multiple AcceptVerbs attribute for a public controller method?

No, multiple **AcceptVerbs** attribute are not allowed in the public controller methods.

17) What is Razor View Engine?

It is an apart of new rendering framework for ASP.NET web pages.

18) Which namespace is used for ASP.NET MVC?

System.Web.Mvc namespace contains all the **interfaces** and **classes** which supports ASP.NET MVC framework for creating web applications.

19) Can we share a view across multiple controllers?

Yes, it is possible to share a view across multiple controllers by putting a view into the shared folder. By doing like this, we can automatically make the view available across multiple controllers.

20) Explain about NonActionAttribute ?

All the public methods of a **controller** class are basically treated as **action methods**. If we don’t want this default behavior, then we can change the public method with **[NonAction]** Attribute.

21) What are the settings to be done for the Routing to work properly in an MVC application?

The settings must be done in 2 places for the routing to work properly. They are,

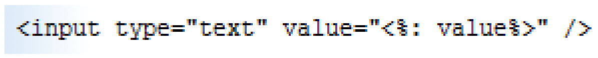
Web.Config => In the **web.config** file, the ASP.NET routing has to be enabled.

Global.asax => The Route table is created in the **application Start** event handler of the **Global.asax** file.

22) How to avoid XSS Vulnerabilities in ASP.NET MVC?

To avoid **XSS vulnerabilities**, we have to use the syntax as '**<%: %>**' in ASP.NET MVC instead of using the syntax as '**<%= %>**' in .net framework 4.0. This is because it does the HTML encoding.

Example:



```
<input type="text" value="<%: value%>" />
```

23) What are the file extensions for razor views?

There are two types of file extensions for **razor** views. They are,

.cshtml => This file extension is used, when the programming language is a C#.

.vbhtml => This file extension is used, when the programming language is a VB.

24) How can you specify comments using razor syntax?

In **razor** syntax, we use the below shown symbols for specifying comments.

- For indicating the beginning of a comment, we use **@*** syntax.
- For indicating the end of a comment, we use ***@** syntax.

25) What is the usage of validation helpers in ASP.NET MVC?

In ASP.NET MVC, **validation helpers** are used to show the validation error messages in a view.

The **Html.ValidationMessage()** and **Html.ValidationSummary()** helpers are used in the automatically generated Edit and Create views by the ASP.NET MVC.

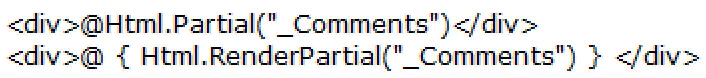
26) Partial View in Asp.net MVC3 Razor? What is the extension of partial view in Razor?

Partial view is like as **user control** in Asp.Net Web forms that is used for code re-usability.

It has **.csHtml** extension.

27) How to render partial view in MVC Razor?

We have to use **Partial** and **RenderPartial** Html methods to render partial view in MVC razor like below,



```
<div>@Html.Partial("_Comments")</div>
<div>@ { Html.RenderPartial("_Comments") } </div>
```

28) What is significance of 'using' in MVC?

```
@using (Html.Beginform ())
{
   Content Here.....
}
```

It ensures that an object is disposed when it goes out of scope.

29) How to add style sheet in view in Asp.Net MVC?

```
<link rel="StyleSheet" href="@Href(~Content/Site.css")" type="text/css"/>
```

30) How to call an action Method on the click of a link?

When we need to call an **ActionMethod** on the click of a link then the syntax should be as below,

@Html.ActionLink("LinkName", "ActionMethod", "Controller")

31) What is the use of ViewStart in ASP.NET MVC?

In the default template of ASP.NET MVC, we get **_ViewStart.cshtml** page that is used to almost similar to **MasterPage** in ASP.NET Web Form or like a layout template.

32) What is Caching and what are the Advantages of Caching in Asp.Net MVC?

Caching is a most important aspect of high-performance web application. Caching provides a way of storing frequently accessed data and reusing that data. Practically, this is an effective way for improving web application's performance.

<u>**Advantages of Caching**</u>

- ***Reduce hosting server round-trips***

  When content is cached at the client or in proxies, it cause minimum request to server.

- ***Reduce database server round-trips***

  When content is cached at the web server; it can eliminate the database request.

- ***Reduce network traffic***

  When content is cached at the client side, it also reduces the network traffic.

- **Avoid time-consumption for regenerating reusable content**

  When reusable content is cached, it avoids the time consumption for regenerating reusable content.

- **Improve performance**

  Since cached content reduce round-trips, network traffic and avoid time consumption for regenerating reusable content which cause a boost in the performance.

33) Difference between Asp.Net MVC and Web forms?

ASP.NET Web forms	ASP.NET MVC
Asp.Net Web Form follows a traditional **event driven** development model.	Asp.Net MVC is a lightweight and follow MVC (Model, View, Controller) pattern based development model.
There is no Fixed architecture for Asp.net Web Forms.	Project structure is predefined.
Asp.Net Web Form has built-in data controls and best for Rapid development with powerful data access.	Not best one for Rapid application development.
Asp.net Web form has **server controls** for creating UI of application	Asp.net MVC has **Html helpers** for creating UI of application
Asp.net Web forms has **state management** (like as view state, session)techniques	Asp.net MVC doesn't have automatic **state management** techniques
Asp.net Web form has **file- based** URLs means file name exist in the URL must have its physically existence.	Asp.net MVC has **route-based** URL means URLs are divided into controllers and actions and moreover it is based on controller not on physical file.
Asp.net Web form has **Master pages** for consistent look and feel	Asp.net MVC **layouts** is for consistent look and feel
Asp.net Web form has **User Controls** for code re-usability	Asp.net MVC has **Partial view** is for code re-usability
Viewstate increases the size of the page affecting performance.	No **Viewstate** so better performance.
The time when one developer will be working on ASPX, second developer may can't work on code behind (because he is unaware about the controls other developer is adding - specially id of the controls)	The time when one developer will be working on View, second developer may work on controller.
Asp.Net Web Form is not Open Source.	Asp.Net Web MVC is an Open Source.

34) How to remove web form (ASPX) engine in MVC?

We can remove all the view engines and add only **Razor** view engine by using **Application_Start** event of **Global.asax.cs** file like as,

```
 protected void Application_Start()
{
//Remove All Engine
 ViewEngines.Engines.Clear();
 //Add Razor Engine
 ViewEngines.Engines.Add(new RazorViewEngine());
 ...
}
```

35) Difference between Razor View Engine and ASPX View Engine?

View Engine is responsible for rendering the view into html form to the browser. By default, Asp.net MVC support **Web Form (ASPX)** and **Razor** View Engine.

Razor View Engine	Web Form (ASPX) View Engine
Razor Engine is an advanced view engine that was introduced with MVC3. This is not a new language but it is new markup syntax.	Web Form Engine is the default view engine for the Asp.net MVC that is included with Asp.net MVC from the beginning.
The namespace for Razor Engine is **`System.Web.Razor`**	The namespace for Web form Engine is **`System.Web.Mvc.WebFormViewEngine`**
The file extensions used with **Razor** Engine are different from Web Form Engine. It has **.cshtml** (Razor with C#) or **.vbhtml** (Razor with VB) extension for **views**, **partial views** and **editor templates** and for layout pages.	The file extensions used with **Web Form** Engine are also like Asp.net Web Forms. It has **.aspx** extension for views, **.ascx** extension for partial views & editor templates and **.master** extension for layout/master pages.
Razor has new and advance syntax that are compact, expressive and reduces typing.	Web Form Engine has the same syntax like Asp.net Web Forms uses for **.aspx** pages.
Razor uses **@** symbol to make the code like as, `@Html.ActionLink("SignUp", "SignUp")`	Web form uses **<%** and **%>** delimiters to make the code like as, `<%: Html.ActionLink("SignUp","SignUp") %>`
By default, Razor Engine prevents **XSS attacks (Cross-Site Scripting Attacks)** means it encodes the script or html tags like <,> before rendering to view.	Web Form Engine does not prevent XSS attacks means any script saved in the database will be fired while rendering the page
Razor Engine is little bit slowly as compared to Web form Engine.	Web Form Engine is faster than Razor Engine.
Razor Engine doesn't support design mode in visual studio means you cannot see your page look and feel.	Web Form engine support design mode in visual studio means you can see your page look and feel without running the application.
Razor Engine support TDD (Test Driven Development) since it is not depend on **System.Web.UI.Page** class.	Web Form Engine doesn't support TDD (Test Driven Development) since it depends on **`System.Web.UI.Page`** class which makes the testing complex.

36) What is Entity Framework? What are Advantages and Disadvantages of Entity Framework?

Entity framework is an **Object Relational Mapping (ORM)** data access framework.

It is an enhancement to **ADO.NET** that gives developers an automated mechanism for accessing and storing the data in the database and working with the results in addition to **DataReader** and **DataSet**.

Entity Framework returns the data in your database as an **object**.

Advantages of Entity Framework:

- ✓ It improves Database performance
- ✓ Easy to implement **CRUD** (Create, Read, Update, Delete) operations without writing SQL queries.
- ✓ It provides Data mapping between **result set** and **entity collection** (or) **entity.**
- ✓ Automatically translates **LINQ** queries to **database** queries.
- ✓ Easy maintenance because, we have the fewer lines of code to fetch data from database.
- ✓ Ability to have inheritance relationships between entities.

Disadvantages of Entity Framework:

- ✓ If there is any **schema** change in database it won't work, we have to update the **schema** in solution as well.
- ✓ Syntax is complicated.

37) Why is it suggested not to use controls such as GridView, Repeater and DataList with the ASP.NET MVC Framework?

GridView, Repeater and DataList need a **ViewState** to function properly. But, ASP.NET MVC Framework does not support **ViewState**. So that, it is not advised to use these controls in the ASP.NET MVC Framework.

38) What is an .edmx file?

Visual Studio saves the entire Entity Data Model configurations in a file with an **.edmx** extension.

This file has an XML format. The **.edmx** file mainly has three sections namely,

- ✓ **Storage schema**
- ✓ **Conceptual schema**
- ✓ **Mappings**

39) What is an ActionResult in MVC?

An **ActionResult** is a return type of a **controller** method in MVC**.**

40) What are different return types of a controller action method in MVC?

There are total **9** return types we can use to return results from **Controller** to **View**.

ViewResult

This return type is used to return a webpage from an action method.

```
public ViewResult ViewResultTest()
{
    return View("ViewResultTest");
}
```

PartialViewResult

This return type is used to send a part of a view (means a view without its layout) which will be rendered inside another view.

```
public PartialViewResult PartialViewResultTest()
{
    return PartialView("PartialViewResult");
}
```

RedirectResult

This return type is used to redirect to any other **controller** and **action** method depending on the **URL**.

```
public ActionResult RedirectResultTest()
{
    return Redirect("http://www.google.com/");
}
public RedirectResult RedirectResultTest()
{
    return Redirect("http://www.google.com/");
}
```

RedirectToRouteResult

This return type is used to redirect to any another action method by using the specified route values dictionary.

```
public ActionResult RedirectToRouteResultTest(int? Id)
{
    return new RedirectToRouteResult(new System.Web.Routing.RouteValueDictionary(new { controller = "Home",
action = "List", Id = new int?() }));
}
```

ContentResult

This return type is used to return user-defined content type like text/plain

```
public ActionResult ContentResultTest()
{
    return Content("Hello My Friend!");
}
public ContentResult ContentResultTest()
{
    return Content("Hello My Friend!");
}
```

.

JsonResult

This return type is used to return serialized JSON object.

```
public JsonResult JsonResultTest()
{
    return Json("Hello My Friend!");
}
```

JavaScriptResult

This return type is used to return JavaScript code like as "function hello () {alert (Hello, World!) ;}" that will run in browser. This is used only in **AJAX** scenarios.

FileResult

This return type is used to send binary output as response (Or) This return type is used to renders the content of a file like as PDF, DOC, and Excel etc.

EmptyResult

This return type is used to return nothing (returns **null** result).

41) What is the page lifecycle of MVC?

- ✓ App initialization
- ✓ Routing
- ✓ Instantiate and execute controller
- ✓ Locate and invoke controller action
- ✓ Instantiate and render view

42) What are the different types of filters in an asp.net MVC application?

- ✓ Authorization filters
- ✓ Action filters
- ✓ Result filters
- ✓ Exception filters

43) What are the new features of MVC4?

- ✓ ASP.NET Web API
- ✓ Enhancements to default project templates and Mobile project template
- ✓ ASP.NET MVC 4 complete focus on **Mobile application development**
- ✓ Working with different mobile and desktop web browsers
- ✓ Display Modes
- ✓ HTML5 support

44) What is Web API's in Asp.net MVC?

Web API is a new framework for consuming & building **HTTP** Services.

45) What are the various types of Application Templates used to create an MVC application?

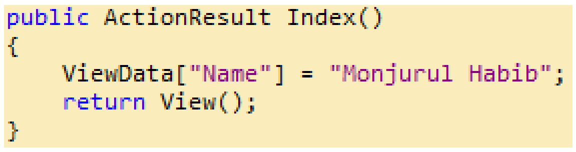
- ✓ Internet Application template
- ✓ Intranet Application template
- ✓ Basic template
- ✓ Empty template
- ✓ Mobile Application template
- ✓ Web API template

46) What is the difference between ViewData, ViewBag and TempData?

ViewData, **ViewBag** and **TempData** are used to pass data from **Controller** to **View.**

ViewData

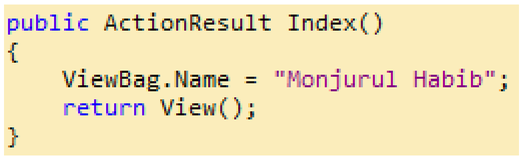
- ✓ ViewData is a dictionary of objects that is derived from **ViewDataDictionary** class.
- ✓ ViewData is used to pass data from controller to corresponding view.
- ✓ Its life span is only during the current request. If redirection occurs then its value becomes null.
- ✓ It's required typecasting for getting data and check for null values to avoid error.



```
public ActionResult Index()
{
    ViewData["Name"] = "Monjurul Habib";
    return View();
}
```

ViewBag

- ✓ ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0.
- ✓ ViewBag is used to pass data from controller to corresponding view.
- ✓ Its life span is only during the current request. If redirection occurs then its value becomes null.
- ✓ It doesn't required typecasting for getting data.



```
public ActionResult Index()
{
    ViewBag.Name = "Monjurul Habib";
    return View();
}
```

TempData

- ✓ TempData is a dictionary object that is derived from **TempDataDictionary** class.
- ✓ Using **TempData** we can maintain data one **Controller** to another **Controller**.
- ✓ It's required typecasting for getting data and check for null values to avoid error.

47) What is the use of Bundling and Minification in MVC?

Bundling => Bundle helps us to combine multiple **JavaScript** and **CSS** files into single entity. So, we can call all the **Scripts** and **CSS** files once.

Minification => Minification reduces the size of **Script** and **CSS** files by removing unnecessary whitespaces, line breaks and comments from code and improves load times.

```
// This is test
var x = 0;
x = x + 1;
x = x * 2;
```

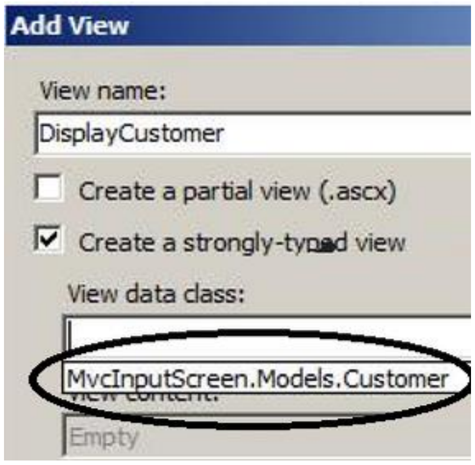
After implementing minification the JavaScript code looks like below. You can see how whitespaces and comments are removed to minimize file size, thus increasing performance.

```
var x=0;x=x+1;x=x*2;
```

48) What is View Model in MVC? How can we use two (multiple) models with a single View?

A **view model** is a simple class which represents data to be displayed on the view.

When we bind a **model** with a **view**, we use the model dropdown as shown in the below figure.


Add View

View name: DisplayCustomer

Create a partial view (.ascx)

Create a strongly-typed view

View data class: MvcInputScreen.Models.Customer

In the above figure we can only select one model. But what if we want to bind **Customer** as well as **Order** class to the view.

For that we need to create a view model which aggregates both the classes as shown in the below code. And then bind that view model with the view.

```
public class CustOrderVM
{
public  Customer cust = new Customer();
public Order Ord = new Order();
}
```

In the **view** we can refer both the model using the **view model** as shown in the below code.

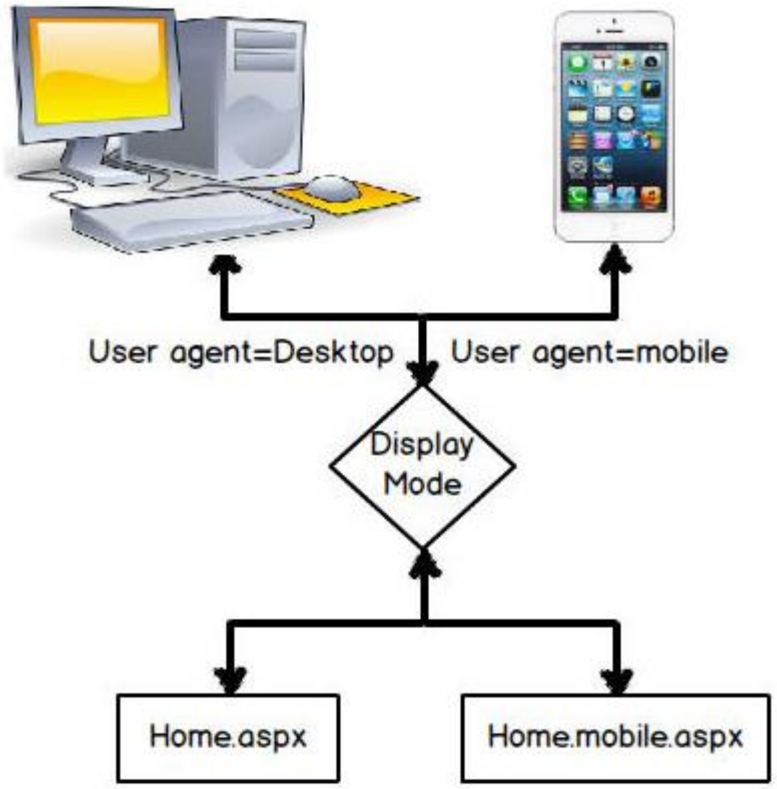
```
<%= model.cust.Name %>
<%= model.Ord.Number %>
```

49) What is the use of Display Modes in MVC?

Display Mode displays **Views** depending on the device (For **mobile** and **desktop browsers**).

Example:

We can create a view **Home.aspx** which will render for the desktop computers and **Home.Mobile.aspx** for mobile devices.



50) What are the advantages of using ASP.NET MVC?

- ✓ In MVC, Project structure is predefined. It separates an application into 3 main components **Model**, **View** and **Controller**. So, it will be easier to manage the complexity of application.
- ✓ No **View State, PostBack events** and **Server-based forms**. So, it improves page performance.
- ✓ MVC **Razor** Engine by default prevents **XSS attacks (Cross-Site Scripting Attacks)** means it encodes the script (or) html tags like **<,>** before rendering to view.
- ✓ It provides better support for test-driven development (TDD).

51) What is the use of action filters in an MVC application?

Action Filters allow us to add **pre-action** and **post-action** behavior to **controller** action **methods** means we can run some code before and after the action method executes.