1. What is MVC (Model View Controller)?

Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representation of information from the way that information is presented to or accepted from the user.

MVC is a framework for building web applications using an MVC (Model View Controller) design:
The Model represents the application core (for instance a list of database records).
The View displays the data (the database records).
The Controller handles the input (to the database records).
The MVC model also provides full control over HTML, CSS, and JavaScript.
The MVC model defines web applications with 3 logic layers,
The business layer (Model logic)
The display layer (View logic)
The input control (Controller logic)
The Model is the part of the application that handles the logic for the application data.

Often model objects retrieve data (and store data) from a database.

The View is the part of the application that handles the display of the data.

Most often the views are created from the model data.

The Controller is the part of the application that handles user interaction.

Typically controllers read data from a view, control user input, and send input data to the model.

The MVC separation helps you manage complex applications because you can focus on one aspect a time. For example, you can focus on the view without depending on the business logic. It also makes it easier to test an application.

The MVC separation also simplifies group development. Different developers can work on the view, the controller logic, and the business logic in parallel.

Learn more about ASP.NET MVC here: Overview Of ASP.NET MVC

2 What are the advantages of MVC?

Multiple view support

Due to the separation of the model from the view, the user interface can display multiple views of the same data at the same time.

Change Accommodation

User interfaces tend to change more frequently than business rules (different colors, fonts, screen layouts, and levels of support for new devices such as cell phones or PDAs) because the model does not depend on the views, adding new types of views to the system generally does not affect the model. As a result, the scope of change is confined to the view.
SoC – Separation of Concerns

Separation of Concerns is one of the core advantages of ASP.NET MVC. The MVC framework provides a clean separation of the UI, Business Logic, Model or Data.

More Control

The ASP.NET MVC framework provides more control over HTML, JavaScript, and CSS than the traditional Web Forms.

Testability

ASP.NET MVC framework provides better testability of the Web Application and good support for test driven development too.

Lightweight

ASP.NET MVC framework doesn't use View State and thus reduces the bandwidth of the requests to an extent.

Full features of ASP.NET

One of the key advantages of using ASP.NET MVC is that it is built on top of the ASP.NET framework and hence most of the features of the ASP.NET like membership providers, roles, etc can still be used.

3. Explain MVC application life cycle?

Any web application has two main execution steps, first understanding the request and depending on the type of the request sending out an appropriate response. MVC application life cycle is not different it has two main phases, first creating the request object and second sending our response to the browser.

Creating the request object,

The request object creation has four major steps. The following is a detailed explanation of the same.

Step 1 - Fill route

MVC requests are mapped to route tables which in turn specify which controller and action to be invoked. So if the request is the first request the first thing is to fill the rout table with routes collection. This filling of the route table happens the global.asax file.

Step 2 - Fetch route

Depending on the URL sent "UrlRoutingModule" searches the route table to create "RouteData" object which has the details of which controller and action to invoke.

Step 3 - Request context created

The "RouteData" object is used to create the "RequestContext" object.

Step 4 - Controller instance created

This request object is sent to “MvcHandler” instance to create the controller class instance. Once the controller class object is created it calls the “Execute” method of the controller class.

Creating a Response object

This phase has two steps executing the action and finally sending the response as a result to the view.

4 List out different return types of a controller action method?

There are total of nine return types we can use to return results from the controller to view.

The base type of all these result types is ActionResult.

ViewResult (View)

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

PartialviewResult (Partialview)

This return type is used to send a part of a view that will be rendered in another view.

RedirectResult (Redirect)

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

RedirectToRouteResult (RedirectToAction, RedirectToRoute)

This return type is used when we want to redirect to any other action method.

ContentResult (Content)

This return type is used to return HTTP content type like text/plain as the result of the action.

jsonResult (json)

This return type is used when we want to return a JSON message.

javascriptResult (javascript)

This return type is used to return JavaScript code that will run in the browser.

FileResult (File)

This return type is used to send binary output in response.

EmptyResult

This return type is used to return nothing (void) in the result.

5 What are the Filters in MVC?

In MVC, controllers define action methods and these action methods generally have a one-to-one relationship with UI controls such as clicking a button or a link, etc. For example, in one of our previous examples, the UserController class contained methods UserAdd, UserDelete, etc.

But many times we would like to perform some action before or after a particular operation. For achieving this functionality, ASP.NET MVC provides a feature to add pre and post-action behaviors on the controller's action methods.

6 Explain types of filters in MVC?

Types of Filters

ASP.NET MVC framework supports the following action filters,
Action Filters

Action filters are used to implement logic that gets executed before and after a controller action executes. We will look at Action Filters in detail in this chapter.

Authorization Filters

Authorization filters are used to implement authentication and authorization for controller actions.

Result Filters

Result filters contain logic that is executed before and after a view result is executed. For example, you might want to modify a view result right before the view is rendered to the browser.

Exception Filters

Exception filters are the last type of filter to run. You can use an exception filter to handle errors raised by either your controller actions or controller action results. You can also use exception filters to log errors.
Action filters are one of the most commonly used filters to perform additional data processing, or manipulating the return values or canceling the execution of an action or modifying the view structure at run time.

7 What are Action Filters in MVC?

Action Filters

Action Filters are additional attributes that can be applied to either a controller section or the entire controller to modify the way in which action is executed. These attributes are special .NET classes derived from System.Attribute which can be attached to classes, methods, properties, and fields.

ASP.NET MVC provides the following action filters,
Output Cache

This action filter caches the output of a controller action for a specified amount of time.

Handle Error

This action filter handles errors raised when a controller action executes.

Authorize

This action filter enables you to restrict access to a particular user or role.
Now we will see the code example to apply these filters on an example controller ActionFilterDemoController. (ActionFilterDemoController is just used as an example. You can use these filters on any of your controllers.)

Output Cache

Code Example

Specifies the return value to be cached for 10 seconds.

```
publicclassActionFilterDemoController: Controller
{
   [HttpGet]
   OutputCache(Duration = 10)]
publicstringIndex()
{
   returnDateTime.Now.ToString("T");

}
}
```

8 Explain what is routing in MVC? What are the three segments for routing important?

Routing is a mechanism to process the incoming URL that is more descriptive and gives the desired response. In this case, URL is not mapped to specific files or folder as was the case of earlier days web sites.

There are two types of routing (after the introduction of ASP.NET MVC 5).
Convention-based routing - to define this type of routing, we call MapRoute method and set its unique name, URL pattern and specify some default values.

Attribute-based routing - to define this type of routing, we specify the Route attribute in the action method of the controller.
Routing is the URL pattern that is mapped together to a handler,routing is responsible for incoming browser request for particular MVC controller. In other ways let us say routing help you to define a URL structure and map the URL with controller. There are three segments for routing that are important,
ControllerName
ActionMethodName

Parammeter
Code Example

ControllerName/ActionMethodName/{ParamerName} and also route map coding written in a Global.asax file.

9 What is Route in MVC? What is Default Route in MVC?

A route is a URL pattern that is mapped to a handler. The handler can be a physical file, such as a .aspx file in a Web Forms application. A handler can also be a class that processes the request, such as a controller in an MVC application. To define a route, you create an instance of the Route class by specifying the URL pattern, the handler, and optionally a name for the route.

You add the route to the application by adding the Route object to the static Routes property of the RouteTable class. The Routesproperty is a RouteCollection object that stores all the routes for the application.

You typically do not have to write code to add routes in an MVC application. Visual Studio project templates for MVC include preconfigured URL routes. These are defined in the MVC Application class, which is defined in the Global.asax file.

Route definition Example of matching URL
{controller}/{action}/{id} /Products/show/beverages
{table}/Details.aspx /Products/Details.aspx
blog/{action}/{entry} /blog/show/123
{reporttype}/{year}/{month}/{day} /sales/2008/1/5
{locale}/{action} /US/show
{language}-{country}/{action} /en-US/show

Default Route

The default ASP.NET MVC project templates add a generic route that uses the following URL convention to break the URL for a given request into three named segments.

URL: "{controller}/{action}/{id}"

This route pattern is registered via a call to the MapRoute() extension method of RouteCollection.

10 Mention what is the difference between Temp data, View, and View Bag?

In ASP.NET MVC there are three ways to pass/store data between the controllers and views.

ViewData
ViewData is used to pass data from controller to view.
It is derived from ViewDataDictionary class.
It is available for the current request only.
Requires typecasting for complex data types and checks for null values to avoid an error.
If redirection occurs, then its value becomes null.
ViewBag
ViewBag is also used to pass data from the controller to the respective view.

ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0
It is also available for the current request only.
If redirection occurs, then its value becomes null.
It doesn't require typecasting for the complex data type.
TempData
TempData is derived from TempDataDictionary class
TempData is used to pass data from the current request to the next request
It keeps the information for the time of an HTTP Request. This means only from one page to another. It helps to maintain the data when we move from one controller to another controller or from one action to another action
It requires typecasting for complex data types and checks for null values to avoid an error.
Generally, it is used to store only one time messages like the error messages and validation messages

11 What is Partial View in MVC?

A partial view is a chunk of HTML that can be safely inserted into an existing DOM. Most commonly, partial views are used to componentize Razor views and make them easier to build and update. Partial views can also be returned directly from controller methods. In this case, the browser still receives text/html content but not necessarily HTML content that makes up an entire page. As a result, if a URL that returns a partial view is directly invoked from the address bar of a browser, an incomplete page may be displayed. This may be something like a page that misses title, script and style sheets. However, when the same URL is invoked via a script, and the response is used to insert HTML within the existing DOM, then the net effect for the end-user may be much better and nicer.

Partial view is a reusable view (like a user control) which can be embedded inside another view. For example, let's say all the pages of your site have a standard structure with left menu, header, and footer as in the following image,

12 Explain what is the difference between View and Partial View?

View
It contains the layout page.
Before any view is rendered, viewstart page is rendered.
A view might have markup tags like body, HTML, head, title, meta etc.
The view is not lightweight as compare to Partial View.
Partial View
It does not contain the layout page.
Partial view does not verify for a viewstart.cshtml.We cannot put common code for a partial view within the viewStart.cshtml.page.
Partial view is designed specially to render within the view and just because of that it does not consist any mark up.
We can pass a regular view to the RenderPartial method.

13 What are HTML helpers in MVC?

With MVC, HTML helpers are much like traditional ASP.NET Web Form controls.

Just like web form controls in ASP.NET, HTML helpers are used to modify HTML. But HTML helpers are more lightweight. Unlike Web Form controls, an HTML helper does not have an event model

and a view state.

In most cases, an HTML helper is just a method that returns a string.

With MVC, you can create your own helpers, or use the built in HTML helpers
Standard HTML Helpers

HTML Links

The easiest way to render an HTML link in is to use the HTML.ActionLink() helper.With MVC, the Html.ActionLink() does not link to a view. It creates a link to a controller action.

ASP Syntax
<%=Html.ActionLink("About this Website", "About")%>
The first parameter is the link text, and the second parameter is the name of the controller action.

The Html.ActionLink() helper above, outputs the following HTML:
About this Website
The Html.ActionLink() helper has several properties:
Property Description.
.linkText The link text (label).
.actionName The target action.
.routeValues The values passed to the action.
.controllerName The target controller.
.htmlAttributes The set of attributes to the link.
.protocol The link protocol.
.hostname The host name for the link.
.fragment The anchor target for the link.
HTML Form Elements

There following HTML helpers can be used to render (modify and output) HTML form elements:
BeginForm()
EndForm()
TextArea()
TextBox()
CheckBox()
RadioButton()
ListBox()
DropDownList()
Hidden()
Password()

14 Explain attribute based routing in MVC?

In ASP.NET MVC 5.0 we have a new attribute route,cBy using the "Route" attribute we can define the URL structure. For example in the below code we have decorated the "GotoAbout" action with the route attribute. The route attribute says that the "GotoAbout" can be invoked using the URL structure "Users/about".

Hide Copy Code
public class HomeController: Controller

```
{
   [Route("Users/about")]
   publicActionResultGotoAbout()
   {
      return View();
   }
}
```

15 What is TempData in MVC?

TempData is a dictionary object to store data temporarily. It is a TempDataDictionary class type and instance property of the Controller base class.

TempData is able to keep data for the duration of a HTP request, in other words it can keep live data between two consecutive HTTP requests. It will help us to pass the state between action methods. TempData only works with the current and subsequent request. TempData uses a session variable to store the data. TempData Requires type casting when used to retrieve data.

TempDataDictionary is inherited from the IDictionary<string, object>, ICollection<KeyValuePair<string, object>>, IEnumerable<KeyValuePair<string, object>> and IEnumerable interfaces.

Example

```
public ActionResult FirstRequest()
{
   List < string > TempDataTest = new List < string > ();
   TempDataTest.Add("Tejas");
   TempDataTest.Add("Jignesh");
   TempDataTest.Add("Rakesh");
   TempData["EmpName"] = TempDataTest;
   return View();
}
public ActionResult ConsecutiveRequest()
{
   List < string > modelData = TempData["EmpName"] as List < string > ;
   TempData.Keep();
   return View(modelData);
}
```

16 What is Razor in MVC?

ASP.NET MVC has always supported the concept of "view engines" - which are the pluggable modules that implement different template syntax options. The "default" view engine for ASP.NET MVC uses the same .aspx/.ascx/. master file templates as ASP.NET Web Forms. Other popular ASP.NET MVC view engines are Spart&Nhaml.

MVC 3 has introduced a new view engine called Razor.

Why is Razor?
Compact & Expressive.

Razor minimizes the number of characters and keystrokes required in a file, and enables a fast coding workflow. Unlike most template syntaxes, you do not need to interrupt your coding to explicitly denote server blocks within your HTML. The parser is smart enough to infer this from your code. This enables a really compact and expressive syntax which is clean, fast and fun to type.
Easy to Learn: Razor is easy to learn and enables you to quickly be productive with a minimum of effort. We can use all your existing language and HTML skills.
Works with any Text Editor: Razor doesn't require a specific tool and enables you to be productive in any plain old text editor (notepad works great).
Has great Intellisense:
Unit Testable: The new view engine implementation will support the ability to unit test views (without requiring a controller or web-server, and can be hosted in any unit test project - no special app-domain required).

17 Differences between Razor and ASPX View Engine in MVC?

Razor View Engine ASPX View Engine (Web form view engine)
The namespace used by the Razor View Engine is System.Web.Razor The namespace used by the ASPX View Engine is System.Web.Mvc.WebFormViewEngine
The file extensions used by the Razor View Engine are different from a web form view engine. It uses cshtml with C# and vbhtml with vb for views, partial view, templates and layout pages. The file extensions used by the Web Form View Engines are like ASP.Net web forms. It uses the ASPX extension to view the aspc extension for partial views or User Controls or templates and master extensions for layout/master pages.
The Razor View Engine is an advanced view engine that was introduced with MVC 3.0. This is not a new language but it is markup. A web form view engine is the default view engine and available from the beginning of MVC
Razor has a syntax that is very compact and helps us to reduce typing. The web form view engine has syntax that is the same as an ASP.Net forms application.
The Razor View Engine uses @ to render server-side content. The ASPX/web form view engine uses "<%= %>" or "<%: %>" to render server-side content.
By default all text from an @ expression is HTML encoded. There is a different syntax ("<%: %>") to make text HTML encoded.
Razor does not require the code block to be closed, the Razor View Engine parses itself and it is able to decide at runtime which is a content element and which is a code element. A web form view engine requires the code block to be closed properly otherwise it throws a runtime exception.
The Razor View Engine prevents Cross-Site Scripting (XSS) attacks by encoding the script or HTML tags before rendering to the view. A web form View engine does not prevent Cross-Site Scripting (XSS) attacks.
The Razor Engine supports Test Driven Development (TDD). Web Form view engine does not support Test Driven Development (TDD) because it depends on the System.Web.UI.Page class to make the testing complex.
Razor uses "@* â€¦ *@" for multiline comments. The ASPX View Engine uses "<!--...-->" for markup and "/* â€¦ */" for C# code.
There are only three transition characters with the Razor View Engine. There are only three transition characters with the Razor View Engine.

18 What are the Main Razor Syntax Rules?

Answer
Razor code blocks are enclosed in @{ ... }
Inline expressions (variables and functions) start with @

Code statements end with semicolon
Variables are declared with the var keyword
Strings are enclosed with quotation marks
C# code is case sensitive
C# files have the extension .cshtml
C# Example

```
<!-- Single statement block -->
@ {
   varmyMessage = "Hello World";
}
<!-- Inline expression or variable -->
< p > The value of myMessage is: @myMessage < /p>
   <!-- Multi-statement block -->
@ {
   var greeting = "Welcome to our site!";
   varweekDay = DateTime.Now.DayOfWeek;
   vargreetingMessage = greeting + " Here in Huston it is: " + weekDay;
} < p > The greeting is: @greetingMessage < /p>
```

19 How do you implement Forms authentication in MVC?

Authentication is giving access to the user for a specific service by verifying his/her identity using his/her credentials like username and password or email and password. It assures that the correct user is authenticated or logged in for a specific service and the right service has been provided to the specific user based on their role that is nothing but authorization.

ASP.NET forms authentication occurs after IIS authentication is completed. You can configure forms authentication by using forms element with in web.config file of your application. The default attribute values for forms authentication are shown below,

```
<system.web>
   <authenticationmode="Forms">
      <formsloginUrl="Login.aspx" protection="All" timeout="30" name=".ASPXAUTH" path="/"
requireSSL="false" slidingExpiration="true" defaultUrl="default.aspx"
cookieless="UseDeviceProfile" enableCrossAppRedirects="false" />
      </authentication>
</system.web>
```

20 Explain Areas in MVC?

From ASP.Net MVC 2.0 Microsoft provided a new feature in MVC applications, Areas. Areas are just a way to divide or “isolate” the modules of large applications in multiple or separated MVC. like,When you add an area to a project, a route for the area is defined in an AreaRegistration file. The route sends requests to the area based on the request URL. To register routes for areas, you add code to theGlobal.asax file that can automatically find the area routes in the AreaRegistration file.

```
AreaRegistration.RegisterAllAreas();
```

Benefits of Area in MVC
Allows us to organize models, views and controllers into separate functional sections of the application, such as administration, billing, customer support and much more.

Easy to integrate with other Areas created by another.
Easy for unit testing.

21 Explain the need of display mode in MVC?

DisplayModes give you another level of flexibility on top of the default capabilities we saw in the last section. DisplayModes can also be used along with the previous feature so we will simply build off of the site we just created.

Using display modes involves in 2 steps
We should register Display Mode with a suffix for particular browser using "DefaultDisplayMode"e class inApplication_Start() method in the Global.asax file.
View name for particular browser should be appended with suffix mentioned in first step. Desktop browsers (without any suffix. e.g.: Index.cshtml, _Layout.cshtml).
Mobile browsers (with a suffix "Mobile". e.g.: Index.Mobile.cshtml,Layout.Mobile.cshtml)

If you want design different pages for different mobile device browsers (any different browsers) and render them depending on the browser requesting. To handle these requests you can register custom display modes. We can do that using DisplayModeProvider.Instance.Modes.Insert(int index, IDisplayMode item) method.

22 Explain the concept of MVC Scaffolding?

ASP.NET Scaffolding is a code generation framework for ASP.NET Web applications. Visual Studio 2013 includes pre-installed code generators for MVC and Web API projects. You add scaffolding to your project when you want to quickly add code that interacts with data models. Using scaffolding can reduce the amount of time to develop standard data operations in your project.

Scaffolding consists of page templates, entity page templates, field page templates, and filter templates. These templates are called Scaffold templates and allow you to quickly build a functional data-driven Website.

23 What is Route Constraints in MVC?

Routing is a great feature of MVC, it provides a REST based URL that is very easy to remember and improves page ranking in search engines.

This article is not an introduction to Routing in MVC, but we will learn a few features of routing and by implementing them we can develop a very flexible and user-friendly application. So, let's start without wasting valuable time.

Add constraint to URL

This is very necessary for when we want to add a specific constraint to our URL. Say, for example we want a URL.

So, we want to set some constraint string after our host name. Fine, let's see how to implement it.

It's very simple to implement, just open the RouteConfig.cs file and you will find the routing

definition in that. And modify the routing entry as in the following. We will see that we have added “abc” before.

24 What is Razor View Engine in MVC?

ASP.NET MVC has always supported the concept of "view engines" that are the pluggable modules that implement various template syntax options. The "default" view engine for ASP.NET MVC uses the same .aspx/.ascx/.master file templates as ASP.NET Web Forms. In this article I go through the Razor View Engine to create a view of an application. "Razor" was in development beginning in June 2010 and was released for Microsoft Visual Studio in January 2011.

Razor is not a new programming language itself, but uses C# syntax for embedding code in a page without the ASP.NET delimiters: <%= %>. It is a simple-syntax view engine and was released as part of ASP.NET MVC 3. The Razor file extension is "cshtml" for the C# language. It supports TDD (Test Driven Development) because it does not depend on the System.Web.UI.Page class.

25 What is Output Caching in MVC?

The main purpose of using Output Caching is to dramatically improve the performance of an ASP.NET MVC Application. It enables us to cache the content returned by any controller method so that the same content does not need to be generated each time the same controller method is invoked. Output Caching has huge advantages, such as it reduces server round trips, reduces database server round trips, reduces network traffic etc.

Keep the following in mind,
Avoid caching contents that are unique per user.
Avoid caching contents that are accessed rarely.
Use caching for contents that are accessed frequently.
Let's take an example. My MVC application displays a list of database records on the view page so by default each time the user invokes the controller method to see records, the application loops through the entire process and executes the database query. And this can actually decrease the application performance. So, we can advantage of the "Output Caching" that avoids executing database queries each time the user invokes the controller method. Here the view page is retrieved from the cache instead of invoking the controller method and doing redundant work.

Cached Content Locations

In the above paragraph I said, in Output Caching the view page is retrieved from the cache, so where is the content cached/stored?

Please note, there is no guarantee that content will be cached for the amount of time that we specify. When memory resources become low, the cache starts evicting content automatically.

OutputCache label has a "Location" attribute and it is fully controllable. Its default value is "Any", however there are the following locations available; as of now, we can use any one.
Any
Client
Downstream
Server
None
ServerAndClient

With "Any", the output cache is stored on the server where the request was processed. The recommended store cache is always on the server very carefully. You will learn about some security related tips in the following "Don't use Output Cache".

26 What is Bundling and Minification in MVC?

Answer

Bundling and minification are two new techniques introduced to improve request load time. It improves load time by reducing the number of requests to the server and reducing the size of requested assets (such as CSS and JavaScript).

Bundling

It lets us combine multiple JavaScript (.js) files or multiple cascading style sheet (.css) files so that they can be downloaded as a unit, rather than making individual HTTP requests.

Minification

It squeezes out whitespace and performs other types of compression to make the downloaded files as small as possible. At runtime, the process identifies the user agent, for example IE, Mozilla, etc. and then removes whatever is specific to Mozilla when the request comes from IE.

27 What is Validation Summary in MVC?

The ValidationSummary helper method generates an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.

The ValidationSummary can be used to display all the error messages for all the fields. It can also be used to display custom error messages. The following figure shows how ValidationSummary displays the error messages.

28 What is Database First Approach in MVC using Entity Framework?

Database First Approach is an alternative to the Code First and Model First approaches to the Entity Data Model which creates model codes (classes,properties, DbContextetc) from the database in the project and that classes behaves as the link between database and controller.

There are the following approach which is used to connect with database to application.
Database First
Model First
Code First Database first is nothing but only a approach to create web application where database is available first and can interact with the database. In this database, database is created first and after that we manage the code. The Entity Framework is able to generate a business model based on the tables and columns in a relational database.

29 What are the Folders in MVC application solutions?

Understanding the folders

When you create a project a folder structure gets created by default under the name of your project which can be seen in solution explorer. Below i will give you a brief explanation of what these folders are for.

Model

This folder contains classes that is used to provide data. These classes can contain data that is retrived from the database or data inserted in the form by the user to update the database.

Controllers

These are the classes which will perform the action invoked by the user. These classes contains methods known as "Actions" which responds to the user action accordingly.

Views

These are simple pages which uses the model class data to populate the HTML controls and renders it to the client browser.

App_Start

Contains Classes such as FilterConfig, RoutesConfig, WebApiConfig. As of now we need to understand the RouteConfig class. This class contains the default format of the url that should be supplied in the browser to navigate to a specified page.

30 What are the methods of handling an Error in MVC?

Exception handling may be required in any application, whether it is a web application or a Windows Forms application.

ASP.Net MVC has an attribute called "HandleError" that provides built-in exception filters. The HandleError attribute in ASP.NET MVC can be applied over the action method as well as Controller or at the global level. The HandleError attribute is the default implementation of IExceptionFilter. When we create a MVC application, the HandleError attribute is added within the Global.asax.cs file and registered in the Application_Start event.

```
public static void RegisterGlobalFilters(GlobalFilterCollection filters)
{
   filters.Add(new HandleErrorAttribute());
}
protected void Application_Start()
{
   AreaRegistration.RegisterAllAreas();
   RegisterGlobalFilters(GlobalFilters.Filters);
   RegisterRoutes(RouteTable.Routes);
}
```

Important properties of HandleError attribute

The HandleError Error attribute has a couple for properties that are very useful in handling the exception.

ExceptionType

Type of exception to be catch. If this property is not specified then the HandleError filter handles all exceptions.

View

Name of the view page for displaying the exception information.

Master

Master View for displaying the exception.

Order

Order in which the action filters are executed. The Order property has an integer value and it specifies the priority from 1 to any positive integer value. 1 means highest priority and the greater the value of the integer is, the lower is the priority of the filter.

AllowMultiple

It indicates whether more than one instance of the error filter attribute can be specified.

31 How can we pass the data From Controller To View In MVC?

There are three options in Model View Controller (MVC) for passing data from controller to view. This article attempts to explain the differences among ViewData, ViewBag and TempData with examples. ViewData and ViewBag are similar and TempData performs additional responsibility. The following are the key points on those three objects.

ViewData
The ViewData is used to move data from controller to view.
The ViewData is a dictionary of objects that are derived from the "ViewDataDictionary" class and it will be accessible using strings as keys.
ViewData contains a null value when redirection occurs.
ViewData requires typecasting for complex data types.
ViewBag
ViewBag is just a dynamic wrapper around ViewData and exists only in ASP.NET MVC 3. ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0.
ViewBag doesn't require typecasting for complex data types.
ViewBag also contain a null value when redirection occurs.
TempData
ViewData moves data from controller to view.
Use TempData when you need data to be available for the next request, only. In the next request, it will be there but will be gone after that.
TempData is used to pass data from the current request to the subsequent request, in other words in case of redirection. That means the value of TempData will not be null.

32 What is Scaffolding in MVC?

Scaffolding is a code generation framework for ASP.NET Web applications. Visual Studio 2013 includes pre-installed code generators for MVC and Web API projects. You add scaffolding to your

project when you want to quickly add code that interacts with data models. Using scaffolding can reduce the amount of time to develop standard data operations in your project.

Prerequisites

To use ASP.NET Scaffolding, you must have,
Microsoft Visual Studio 2013
Web Developer Tools (part of default Visual Studio 2013 installation)
ASP.NET Web Frameworks and Tools 2013 (part of default Visual Studio 2013 installation)
What are the Advantages of using Scaffolding ?
Minimal or no code to create a data-driven Web applications.
Quick development time.
Pages that are fully functional and include display, insert, edit, delete, sorting, and paging functionalities.
Built-in data validation that is based on the database schema.
Filters that are created for each foreign key or Boolean fields.

33 What is ViewStart?

Razor View Engine introduced a new layout named _ViewStart which is applied on all view automatically. Razor View Engine firstly executes the _ViewStart and then start rendering the other view and merges them.

Example of Viewstart

```
@ {
   Layout = "~/Views/Shared/_v1.cshtml";
} < !DOCTYPE html >
   < html >
   < head >
   < meta name = "viewport"
content = "width=device-width" / >
   < title > ViewStart < /title> < /head> < body >
   < /body> < /html>
```

34 What is JsonResultType in MVC?

Action methods on controllers return JsonResult (JavaScript Object Notation result) that can be used in an AJAX application. This class is inherited from the "ActionResult" abstract class. Here Json is provided one argument which must be serializable. The JSON result object that serializes the specified object to JSON format.

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

35 What is TempData?

TempData is a dictionary object derived from the TempDataDictionary class.
TempData is used to pass data from the current request to a subsequent request, in other words in the case of redirection.
The life of a TempData is very short and it retains its value for a short period of time.

It requires typecasting for complex data type as I've used in my example:
@foreach (var item in
(List<MVCSample.Models.EmpRegistration>)TempData["EmployeeRegistration"])
You can retain its value using the Keep method for subsequent requests.

36 How to use ViewBag?

ViewBag is dynamic property that takes advantage of new dynamic features in C# 4.0. It's also used to pass data from a controller to a view. In short, The ViewBag property is simply a wrapper around the ViewData that exposes the ViewData dictionary as a dynamic object. Now create an action method "StudentSummary" in the "DisplayDataController" controller that stores a Student class object in ViewBag.

```
public ActionResult StudentSummary()
{
   var student = new Student()
   {
      Name = "Sandeep Singh Shekhawat",
         Age = 24,
         City = "Jaipur"
   };
   ViewBag.Student = student;
   return View();
}
```

Thereafter create a view StudentSummary ("StudentSummary.cshtml") that shows student object data. ViewBag does not require typecasting for complex data type so you can directly access the data from ViewBag.

```
@ {
   ViewBag.Title = "Student Summary";
   var student = ViewBag.Student;
}
< table >
   < tr >
   < th > Name < /th> < th > Age < /th> < th > City < /th> < /tr> < tr >
   < td > @student.Name < /td> < td > @student.Age < /td> < td > @student.City < /td> < /tr>
< /table>
```

Here we used one more thing, "ViewBag.Title", that shows the title of the page.

37 What are the Difference between ViewBag&ViewData?

Difference between ViewBag&ViewData?
ViewData is a dictionary of objects that is derived from ViewDataDictionary class and accessible using strings as keys.
ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0.
ViewData requires typecasting for complex data type and check for null values to avoid error.
ViewBag doesn't require typecasting for complex data type.
Calling of ViewBag is:

```
ViewBag.Name = "Yogesh";
```

Calling of ViewDatais :

ViewData["Name"] = "yogesh";

38 What is Data Annotation Validator Attributes in MVC?

Using the Data Annotation Validator Attributes

DataAnnotation plays a vital role in added validation to properties while designing the model itself. This validation can be added for both the client side and the server side.

You understand that decorating the properties in a model with an Attribute can make that property eligible for Validation.

Some of the DataAnnotation used for validation are given below, Required

Specify a property as required.

1. [Required(ErrorMessage="CustomerName is mandatory")]

RegularExpression

Specifies the regular expression to validate the value of the property.

1. [RegularExpression("[a-z]", ErrorMessage = "Invalid character")]

Range

Specifies the Range of values between which the property values are checked.

1. [Range(1000,10000,ErrorMessage="Range should be between 1k & 10k")]

StringLength

Specifies the Min & Max length for a string property.

1. [StringLength(50, MinimumLength = 5, ErrorMessage = "Minimum char is 5 and maximum char is 10")]

MaxLength

Specifies the Max length for the property value.

1. [MaxLength(10,ErrorMessage="Customer Code is exceeding")]

MinLength

It is used to check for minimum length.

1. [MinLength(5, ErrorMessage = "Customer Code is too small")]

39 How can we done Custom Error Page in MVC?

The HandleErrorAttribute allows you to use a custom page for this error. First you need to update your web.config file to allow your application to handle custom errors.

```
<system.web>
  <customErrors mode="On">
</system.web>
```

Then, your action method needs to be marked with the atttribute.

```
[HandleError]
public class HomeController: Controller
{
  [HandleError]
  publicActionResultThrowException()
  {
    throw new ApplicationException();
  }
}
```

By calling the ThrowException action, this would then redirect the user to the default error page. In our case though, we want to use a custom error page and redirect the user there instead.So, let's create our new custom view page.

40 Server Side Validation in MVC?

The ASP.NET MVC Framework validates any data passed to the controller action that is executing, It populates a ModelState object with any validation failures that it finds and passes that object to the controller. Then the controller actions can query the ModelState to discover whether the request is valid and react accordingly.

I will use two approaches in this article to validate a model data. One is to manually add an error to the ModelState object and another uses the Data Annotation API to validate the model data.

Approach 1 - Manually Add Error to ModelState object

I create a User class under the Models folder. The User class has two properties "Name" and "Email". The "Name" field has required field validations while the "Email" field has Email validation. So let's see the procedure to implement the validation. Create the User Model as in the following,

```
namespace ServerValidation.Models
{
  public class User
  {
    public string Name
   {
      get;
      set;
    }
    public string Email
    {
      get;
      set;
    }
  }
}
```

After that I create a controller action in User Controller (UserController.cs under Controllers folder). That action method has logic for the required validation for Name and Email validation on the Email field. I add an error message on ModelState with a key and that message will be shown on the view whenever the data is not to be validated in the model.

41 What is the use of remote validation in MVC?

Remote validation is the process where we validate specific data posting data to a server without posting the entire form data to the server. Let's see an actual scenario, in one of my projects I had a requirement to validate an email address, whetehr it already exists in the database. Remote validation was useful for that; without posting all the data we can validate only the email address supplied by the user.

Practical Explanation

Let's create a MVC project and name it accordingly, for me its “TestingRemoteValidation”. Once the project is created let's create a model named UserModel that will look like:

```
public class UserModel
{
   [Required]
   public string UserName
   {
      get;
      set;
   }
   [Remote("CheckExistingEmail", "Home", ErrorMessage = "Email already exists!")]
   public string UserEmailAddress
   {
      get;
      set;
   }
}
```

Let's get some understanding of the remote attribute used, so the very first parameter “CheckExistingEmail” is the the name of the action. The second parameter “Home” is referred to as controller so to validate the input for the UserEmailAddress the “CheckExistingEmail” action of the “Home” controller is called and the third parameter is the error message. Let's implement the “CheckExistingEmail” action result in our home controller.

```
public ActionResult CheckExistingEmail(string UserEmailAddress)
{
   bool ifEmailExist = false;
   try
   {
      ifEmailExist = UserEmailAddress.Equals("mukeshknayak@gmail.com") ? true : false;
      return Json(!ifEmailExist, JsonRequestBehavior.AllowGet);
   } catch (Exception ex)
   {
      return Json(false, JsonRequestBehavior.AllowGet);
   }
}
```

42 What are the Exception filters in MVC?

Exception are part and parcel of an application. They are a boon and a ban for an application too. Isn't it? This would be controversial, for developers it helps them track minor and major defects in an application and sometimes they are frustrating when it lets users land on the Yellow screen of death each time. This would make the users mundane to the application. Thus to avoid this, developers handle the exceptions. But still sometimes there are a few unhandled exceptions.

Now what is to be done for them? MVC provides us with built-in "Exception Filters" about which we will explain here.

43 What is MVC HTML- Helpers and its Methods?

Helper methods are used to render HTML in the view. Helper methods generates HTML output that is part of the view. They provide an advantage over using the HTML elements since they can be reused across the views and also requires less coding. There are several builtin helper methods that are used to generate the HTML for some commonly used HTML elements, like form, checkbox, dropdownlist etc. Also we can create our own helper methods to generate custom HTML. First we will see how to use the builtin helper methods and then we will see how to create custom helper methods.

Standard HtmlHelper methods

Some of the standard helper methods are,
ActionLink: Renders an anchor.
BeginForm: Renders HTML form tag
CheckBox: Renders check box.
DropDownList: Renders drop-down list.
Hidden: Renders hidden field
ListBox: Renders list box.
Password: Renders TextBox for password input
RadioButton: Renders radio button.
TextArea: Renders text area.
TextBox: Renders text box.

44 What is Attribute Routing in MVC?

A route attribute is defined on top of an action method. The following is the example of a Route Attribute in which routing is defined where the action method is defined.

In the following example, I am defining the route attribute on top of the action method

```
public class HomeController: Controller
{
   //URL: /Mvctest
   [Route(“Mvctest”)]
   public ActionResult Index()
   ViewBag.Message = "Welcome to ASP.NET MVC!";
```

```
   return View();
}
}
```

Attribute Routing with Optional Parameter

We can also define an optional parameter in the URL pattern by defining a mark (“?") to the route parameter. We can also define the default value by using parameter=value.

```
public class HomeController: Controller
{
      // Optional URI Parameter
      // URL: /Mvctest/
      // URL: /Mvctest/0023654
      [Route(“Mvctest /
      {
         customerName ?
      }”)]
      public ActionResult OtherTest(string customerName)
      ViewBag.Message = "Welcome to ASP.NET MVC!";
      return View();
   }
   // Optional URI Parameter with default value
   // URL: /Mvctest/
   // URL: /Mvctest/0023654
   [Route(“Mvctest /
   {
      customerName = 0036952
   }”)]
public ActionResult OtherTest(string customerName)
{
   ViewBag.Message = "Welcome to ASP.NET MVC!";
   return View();
}
}
```

45 Explain RenderSection in MVC?

RenderSection() is a method of the WebPageBase class. Scott wrote at one point, The first parameter to the "RenderSection()" helper method specifies the name of the section we want to render at that location in the layout template. The second parameter is optional, and allows us to define whether the section we are rendering is required or not. If a section is "required", then Razor will throw an error at runtime if that section is not implemented within a view template that is based on the layout file (that can make it easier to track down content errors). It returns the HTML content to render.

```
<div id="body">
   @RenderSection("featured", required: false)
   <section class="content-wrapper main-content clear-fix">
      @RenderBody()
   </section>
</div>
```

46 What is GET and POST Actions Types?

GET

GET is used to request data from a specified resource. With all the GET request we pass the URL which is compulsory, however it can take the following overloads.

.get(url [, data] [, success(data, textStatus, jqXHR)] [, dataType]).done/.fail

POST

POST is used to submit data to be processed to a specified resource. With all the POST requests we pass the URL which is compulsory and the data, however it can take the following overloads.

.post(url [, data] [, success(data, textStatus, jqXHR)] [, dataType])

47 Mention what "beforFilter()","beforeRender" and "afterFilter" functions do in Controller?

beforeFilter(): This function is run before every action in the controller. It's the right place to check for an active session or inspect user permissions.
beforeRender(): This function is called after controller action logic, but before the view is rendered. This function is not often used, but may be required If you are calling render() manually before the end of a given action
afterFilter(): This function is called after every controller action, and after rendering is done. It is the last controller method to run

48 Explain the role of components Presentation, Abstraction and Control in MVC?

Presentation: It is the visual representation of a specific abstraction within the application
Abstraction: It is the business domain functionality within the application
Control: It is a component that keeps consistency between the abstraction within the system and their presentation to the user in addition to communicating with other controls within the system

49 Mention the advantages and disadvantages of MVC model?

Advantages

Disadvantages

It represents clear separation between business logic and presentation logic
Each MVC object has different responsibilities
The development progresses in parallel
Easy to manage and maintain
All classes and object are independent of each other
The model pattern is little complex
Inefficiency of data access in view
With modern user interface, it is difficult to use MVC

You need multiple programmers for parallel development
Multiple technologies knowledge is required

50 Explain what are the steps for the execution of an MVC project?

The steps for the execution of an MVC project includes

Receive first request for the application
Performs routing
Creates MVC request handler
Create Controller
Execute Controller
Invoke action
Execute Result

51 Explain what is routing? What are the three segments for routing is important?

Routing helps you to decide a URL structure and map the URL with the Controller.

The three segments that are important for routing is

ControllerName
ActionMethodName
Parameter

52 Explain how routing is done in MVC pattern?

There is a group of routes called the RouteCollection, which consists of registered routes in the application. The RegisterRoutes method records the routes in this collection. A route defines a URL pattern and a handler to use if the request matches the pattern. The first parameter to the MapRoute method is the name of the route. The second parameter will be the pattern to which the URL matches. The third parameter might be the default values for the placeholders if they are not determined.

53 Explain using hyperlink how you can navigate from one view to other view?

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

Collapse / Copy Code

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

54 Mention how can maintain session in MVC?

Session can be maintained in MVC by three ways tempdata, viewdata, and viewbag.

55 Explain how you can implement Ajax in MVC?

In MVC, Ajax can be implemented in two ways

Ajax libraries
Jquery

56 Mention what is the difference between "ActionResult" and "ViewResult" ?

"ActionResult" is an abstract class while "ViewResult" is derived from "AbstractResult" class. "ActionResult" has a number of derived classes like "JsonResult", "FileStreamResult" and "ViewResult" .

"ActionResult" is best if you are deriving different types of view dynamically.

57 Explain how you can send the result back in JSON format in MVC?

In order to send the result back in JSON format in MVC, you can use "JSONRESULT" class.

58 Mention what is the importance of NonActionAttribute?

All public methods of a controller class are treated as the action method if you want to prevent this default method then you have to assign the public method with NonActionAttribute.

59 Mention what is the use of the default route {resource}.axd/{*pathinfo} ?

This default route prevents request for a web resource file such as Webresource.axd or ScriptResource.axd from being passed to the controller.

60 Mention the order of the filters that get executed, if the multiple filters are implemented?

The filter order would be like

Authorization filters
Action filters
Response filters
Exception filters

61 Mention what filters are executed in the end?

In the end "Exception Filters" are executed.

62 Mention what are the file extensions for razor views?

For razor views the file extensions are

.cshtml: If C# is the programming language
.vbhtml: If VB is the programming language

63 Mention what are the two ways for adding constraints to a route?

Two methods for adding constraints to route is

Using regular expressions
Using an object that implements IRouteConstraint interface

64 Mention two instances where routing is not implemented or required?

Two instance where routing is not required are

When a physical file is found that matches the URL pattern
When routing is disabled for a URL pattern

65 Mention what are main benefits of using MVC?

There are two key benefits of using MVC

As the code is moved behind a separate class file, you can use the code to a great extent
As behind code is simply moved to.NET class, it is possible to automate UI testing. This gives an opportunity to automate manual testing and write unit tests.

66 What is the use of ViewModel in MVC?

ViewModel is a plain class with properties, which is used to bind it to a strongly-typed view. ViewModel can have the validation rules defined for its properties using data annotation.

67 What is Database first approach in MVC using Entity Framework?

Database First Approach is an alternative or substitutes to the Code First and Model First approaches to the Entity Data Model. The Entity Data Model creates model codes (classes, properties, DbContext, etc.) from the database in the project and that class behaves as the link between database and controller.

There are the following approaches, which are used to connect the database with the application.

Database First

Model First

Code First

68 What is the use of Keep and Peek in “TempData”?

Once “TempData” is read in the current request, it’s not available in the subsequent request. If we want “TempData” to be read and also available in the subsequent request then after reading we need to call “Keep” method as shown in the code below.

```
1
2
@TempData["MyData"];
TempData.Keep("MyData");
```

The more shortcut way of achieving the same is by using “Peek”. This function helps to read as well advices MVC to maintain “TempData” for the subsequent request.

```
1
string str = TempData.Peek("Td").ToString();
```