

What is ASP.NET Web API?

ASP.NET Web API is a framework that simplifies building HTTP services for broader range of clients (including browsers as well as mobile devices) on top of .NET Framework.

Using ASP.NET Web API, we can create non-SOAP based services like plain XML or JSON strings, etc. with many other advantages including:

WebAPI is a framework which helps you to build/develop HTTP services.

- Create resource-oriented services using the full features of HTTP
- Exposing services to a variety of clients easily like browsers or mobile devices, etc.

What are the Advantages of Using ASP.NET Web API?

Using ASP.NET Web API has a number of advantages, but core of the advantages are:

- It works the HTTP way using standard HTTP verbs like **GET**, **POST**, **PUT**, **DELETE**, etc. for all CRUD operations
- Complete support for routing
- Response generated in JSON or XML format using **MediaTypeFormatter**
- It has the ability to be hosted in IIS as well as self-host outside of IIS
- Supports Model binding and Validation
- Support for OData
- and more....

For implementation on performing all CRUD operations using ASP.NET Web API, [click here](#).

What New Features are Introduced in ASP.NET Web API 2.0?

More new features introduced in ASP.NET Web API framework v2.0 are as follows:

- Attribute Routing
- External Authentication
- CORS (Cross-Origin Resource Sharing)
- OWIN (Open Web Interface for .NET) Self Hosting
- **IHttpActionResult**
- Web API OData

You can follow a good Web API new feature details on [Top 5 New Features in ASP.NET Web API 2](#) here.

WCF Vs ASP.NET Web API?

Actually, **Windows Communication Foundation** is designed to exchange standard SOAP-based messages using variety of transport protocols like HTTP, TCP, NamedPipes or MSMQ, etc.

On the other hand, **ASP.NET API** is a framework for building non-SOAP based services over HTTP only.

For more details, [please follow here](#).

Is it True that ASP.NET Web API has Replaced WCF?

It's a misconception that ASP.NET Web API has replaced WCF. It's another way of building non-SOAP based services, for example, plain XML or JSON string, etc.

Yes, it has some added advantages like utilizing full features of HTTP and reaching more clients such as mobile devices, etc. But WCF is still a good choice for following scenarios:

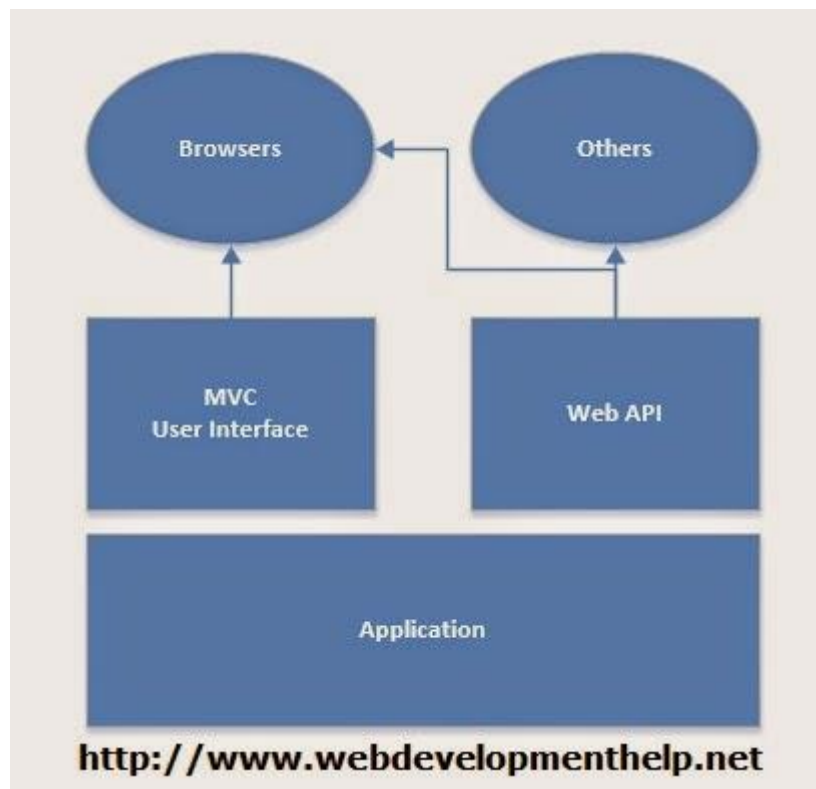
- If we intended to use transport other than HTTP, e.g. TCP, UDP or Named Pipes
- Message Queuing scenario using MSMQ
- One-way communication or Duplex communication

For a good understanding for WCF(Windows Communication Foundation), please follow [WCF Tutorial](#).

MVC Vs ASP.NET Web API?

As in previous ASP.NET Web API Interview Questions, we discussed that the purpose of Web API framework is to generate HTTP services that reach more clients by generating data in raw format, for example, plain XML or JSON string. So, ASP.NET Web API creates simple HTTP services that renders raw data.

On the other hand, ASP.NET MVC framework is used to develop web applications that generates Views as well as data. ASP.NET MVC facilitates in rendering HTML easy.



For **ASP.NET MVC Interview Questions**, [follow the link](http://www.webdevelopmenthelp.net).

How to Return View from ASP.NET Web API Method?

(A tricky Interview question) No, we can't return view from ASP.NET Web API method. We discussed in the earlier interview question about the difference between ASP.NET MVC and Web API that ASP.NET Web API creates HTTP services that renders raw data. Although, it's quite possible in ASP.NET MVC application.

How to Restrict Access to Web API Method to Specific HTTP Verb?

Attribute programming plays its role here. We can easily restrict access to an ASP.NET Web API method to be called using a specific HTTP method. For example, we may require in a scenario to restrict access to a Web API method through HTTP **POST** only as follows:

Hide Copy Code

```
[HttpPost]
public void UpdateStudent(Student aStudent)
{
    StudentRepository.AddStudent(aStudent);
}
```

Can we use Web API with ASP.NET Web Form?

Yes, ASP.NET Web API is bundled with ASP.NET MVC framework but still it can be used with ASP.NET Web Form.

It can be done in three simple steps as follows:

1. Create a Web API Controller
2. Add a routing table to **Application_Start** method of *Global.asax*
3. Make a jQuery AJAX Call to Web API method and get data

jQuery call to Web API for all CRUD (Create, Retrieve, Update, Delete) operations can be [found here](#).

How Can We Provide an Alias Name for ASP.NET Web API Action?

We can provide an alias name for ASP.NET Web API action same as in case of ASP.NET MVC by using "**ActionName**" attribute as follows:

[Hide](#) [Copy Code](#)

```
[HttpPost]
[ActionName("SaveStudentInfo")]
public void UpdateStudent(Student aStudent)
{
    StudentRepository.AddStudent(aStudent);
}
```

In this ASP.NET Tutorial, we covered the most important Interview questions on ASP.NET Web API framework. Hopefully, it will be helpful for Web API developer Interview but along with these questions, do the practical implementation as much as you can. In **Practical guide to ASP.NET Web API**, you can find a good step by step approach for understanding and implementing ASP.NET Web API service

2) Why is Web API required? Is it possible to use RESTful services using WCF?

Yes, we can still develop RESTful services with WCF. However, there are two main reasons that prompt users to use Web API instead of RESTful services.

- Web API increases TDD (Test Data Driven) approach in the development of RESTful services.
- If we want to develop RESTful services in WCF, you surely need a lot of config settings, URI templates, contracts & endpoints for developing RESTful services using web API.

3) Why select Web API?

- It is used to create simple, non-SOAP-based HTTP Services
- It is also an easy method for creation with Web API. With WCF REST Services
- It is based on HTTP and easy to define, expose and consume in a REST-ful way.
- It is lightweight architecture and ideal for devices that have limited bandwidth like smartphones.

4) Is it right that ASP.NET Web API has replaced WCF?

It's not at all true that ASP.NET Web API has replaced WCF. In fact, it is another way of building non-SOAP based services, i.e., plain XML or JSON string.

5) What are the advantages of Web API?

Advantages of Web API are:

- OData
- Filters
- Content Negotiation
- Self-Hosting
- Routing
- Model Bindings

6) What are main return types supported in Web API?

A Web API controller action can return following values:

- Void – It will return empty content
- HttpResponseMessage – It will convert the response to an HTTP message.
- IHttpActionResult – internally calls ExecuteAsync to create an HttpResponseMessage
- Other types – You can write the serialized return value into the response body

7) Web API supports which protocol?

Web App supports HTTP protocol.

8) Which .NET framework supports Web API?

NET 4.0 and above version supports web API.

9) Web API uses which of the following open-source library for JSON serialization?

Web API uses Json.NET library for JSON serialization.

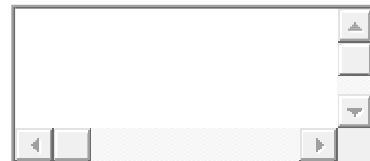
13) What is Web API Routing?

Routing is pattern matching like in MVC.

All routes are registered in Route Tables.

For example:

C#



```
1 Routes.MapHttpRoute(  
2  
3 Name: "ExampleWebAPIRoute",  
4  
5 routeTemplate: "api/{controller}/{id}"  
6  
7 defaults: new { id = RouteParameter.Optional }
```

14) What is SOAP?

SOAP is an XML message format used in web service interactions. It allows to send messages over HTTP or JMS, but other transport protocols can be used. It is also an XML-based messaging protocol for exchanging information among computers.

15) What is the benefit of using REST in Web API?

REST is used to make fewer data transfers between client and server which make it an ideal for using it in mobile apps. Web API also supports HTTP protocol. Therefore, it reintroduces the traditional way of the HTTP verbs for communication.

16) How can we use Web API with ASP.NET Web Form?

Web API can be used with ASP.NET Web Form

It can be performed in three simple steps:

1. Create a Web API Controller,
2. Add a routing table to Application_Start method of Global.sax
3. Then you need to make a jQuery AJAX Call to Web API method and get data.

17) How to you can limit Access to Web API to Specific HTTP Verb?

Attribute programming plays a important role. It is easy to restrict access to an ASP.NET Web API method to be called using a particular HTTP method.

18) Can you use Web API with ASP.NET Web Form?

Yes, It is possible to use Web API with ASP.Net web form. As it is bundled with ASP.NET MVC framework. However, it can be used with ASP.NET Web Form.

19) How Can assign alias name for ASP.NET Web API Action?

We can give alias name for Web API action same as in case of ASP.NET MVC by using "ActionName" attribute as follows:

```
1 [HttpPost]
2
3 [ActionName("SaveStudentInfo")]
4
5 public void UpdateStudent(Student aStudent)
6 {
7     StudentRepository.AddStudent(aStudent);
8 }
```

21) Explain exception filters?

It will be executed when exceptions are unhandled and thrown from a controller method. The reason for the exception can be anything. Exception filters will implement "IExceptionFilter" interface.

22) How can we register exception filter from the action?

We can register exception filter from action using following code:

```
1 [NotImplExceptionFilter]
2
3 public TestCustomer GetMyTestCustomer(int custid)
4
5 {
6 }
```

```
7 //write the code
8
9 }
```

23) How you can return View from ASP.NET Web API method?

No, we can't return a view from ASP.NET Web API Method. Web API creates HTTP services that render raw data. However, it's also possible in ASP.NET MVC application.

24) How to register exception filter globally?

It is possible to register exception filter globally using following code-

```
GlobalConfiguration.Configuration.Filters.Add(new
MyTestCustomerStore.NotImplExceptionFilterAttribute());
```

25) Explain what is REST and RESTFUL?

REST represents REpresentational State Transfer; it is entirely a new aspect of writing a web app.

RESTFUL: It is term written by applying REST architectural concepts is called RESTful services. It focuses on system resources and how the state of the resource should be transported over HTTP protocol.

26) Give me one example of Web API Routing?

C#

```
1 Config.Routes.MapHttpRoute(
2
3 name: "MyRoute", //route name
4
5 routeTemplate: "api/{controller}/{action}/{id}", //as you can see "API" is
6 at the beginning.
7
8 defaults: new { id = RouteParameter.Optional }
9 );
```

27) How can you handle errors in Web API?

Several classes are available in Web API to handle errors. They are HttpError, Exception Filters, HttpResponseException, and Registering Exception Filters.

28) What New Features comes with ASP.NET Web API 2.0?

The latest features of ASP.NET Web API framework v2.0 are as follows:

- Attribute Routing
- Cross-Origin Resource Sharing
- External Authentication
- Open Web Interface NET
- IHttpActionResult
- Web API OData

32) Name the tools or API for developing or testing web api?

Testing tools for web services for REST APIs include:

1. Jersey API
2. CFX
3. Axis
4. Restlet

33) What is REST?

REST is architectural style. It has defined guidelines for creating services which are scalable. REST used with HTTP protocol using its verbs GET, PUT, POST and DELETE.

34) How to unit test Web API?

We can perform a Unit test using Web API tools like Fiddler.

Here, are some setting to be done if you are using

Fiddler –Compose Tab -> Enter Request Headers -> Enter the Request Body and execute

40) Web API supports which protocol?

Web App support HTTP protocol

41) Which of the following .NET framework supports Web API?

Web API is supported by NET 4.0 version

42) Web API uses which library for JSON serialization?

Web API uses Json.NET library for JSON serialization.

43) By default, Web API sends HTTP response with which of the following status code for all uncaught exception?

500 – Internal Server Error

3) Explain Web API Routing?

Routing is the mechanism of pattern matching as we have in MVC. These routes will get registered in Route Tables. Below is the sample route in Web API –

```
Routes.MapHttpRoute(  
    Name: "MyFirstWebAPIRoute",  
    routeTemplate: "api/{controller}/{id}"  
    defaults: new { id = RouteParameter.Optional }  
);
```

7) List out differences between MVC and Web API?

Below are some of the differences between MVC and Web API

MVC

- MVC is used to create a web app, in which we can build web pages.
- For JSON it will return JsonResult from action method.
- All requests are mapped to the respective action methods.

Web API

- This is used to create a service using HTTP verbs.
- This returns XML or JSON to client.
- All requests are mapped to actions using HTTP verbs.
 - **11) Can we return view from Web API?**
 - No. We cannot return view from Web API.
 - **16) What is the difference between MVC Routing and Web API Routing?**
 - There should be at least one route defined for MVC and Web API to run MVC and Web API application respectively. In Web API pattern we can find "api/" at the beginning which makes it distinct from MVC routing. In Web API routing "action" parameter is not mandatory but it can be a part of routing.

41) What are media types?

It is also called MIME, which is used to identify the data. In HTML, media types are used to describe message format in the body.

42) List out few media types of HTTP?

Below are the list of media types –

- Image/Png
- Text/HTML

- Application/Json

43) Explain Media Formatters in Web API?

Media Formatters in Web API can be used to read the CLR object from our HTTP body and Media formatters are also used for writing CLR objects of message body of HTTP.

54) How to apply custom action filter in WebAPI.config?

Add a new action filter in "Register" method as shown –

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        config.Filters.Add(new MyCustomModelAttribute());
        // ...
    }
}
```

58) How parameter binding works in Web API?

Below are the rules followed by WebAPI before binding parameters –

- If it is simple parameters like – bool,int, double etc. then value will be obtained from the URL.
- Value read from message body in case of complex types.

59) Why to use "FromUri" in Web API?

In Web API to read complex types from URL we will use "FromUri" attribute to the parameter in action method. Eg:

```
public MyValuesController : ApiController
{
    public HttpResponseMessage Get([FromUri] MyCustomer c) { ... }
}
```

60) Why to use "FromBody" in Web API?

This attribute is used to force Web API to read the simple type from message body. "FromBody" attribute is along with parameter. Eg:

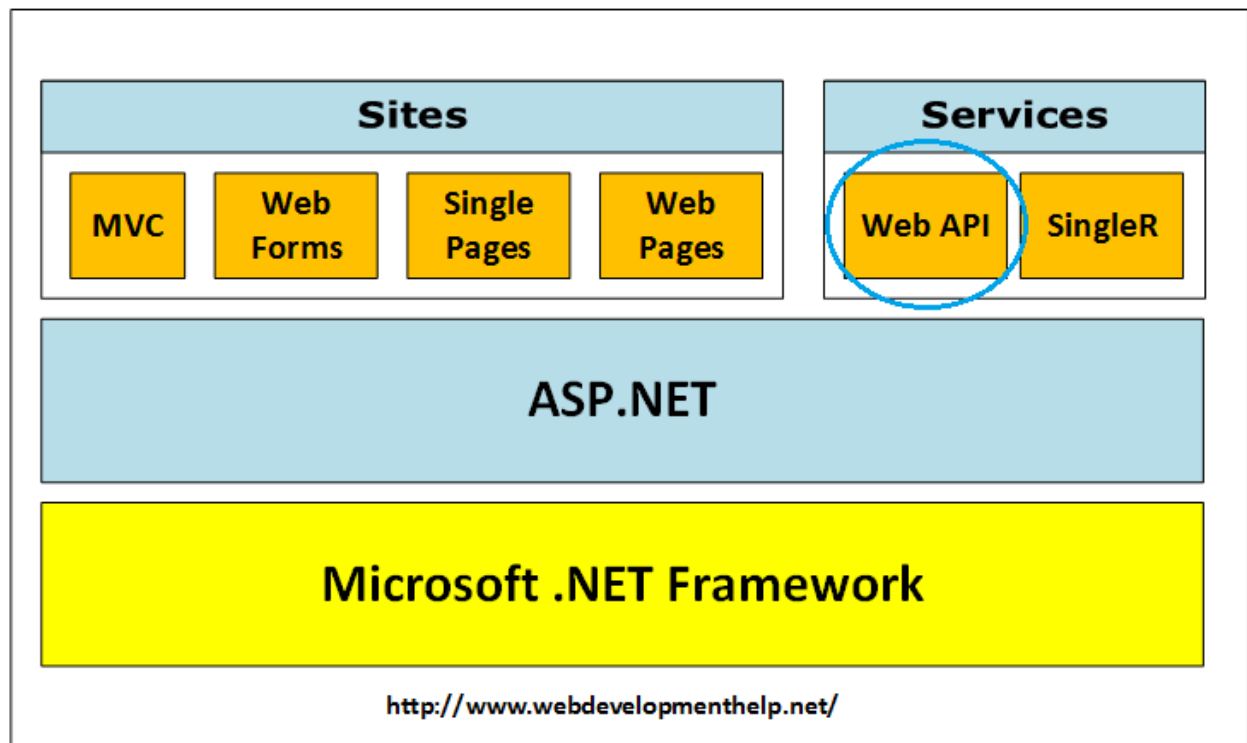
```
public HttpResponseMessage Post([FromBody] int customerid, [FromBody]
string customername) { ... }
```

What is ASP.NET Web API?

ASP.NET Web API is a framework that simplifies building HTTP services for broader range of clients (including browsers as well as mobile devices) on

top of .NET Framework. Using ASP.NET Web API we can create non-SOAP based services like plain XML or JSON strings etc. with many other advantages including:

- Create resource-oriented services using the full features of HTTP.
- Exposing services to a variety of clients easily like browsers or mobile devices etc.



MVC

Question 1: What is MVC (Model view controller)?

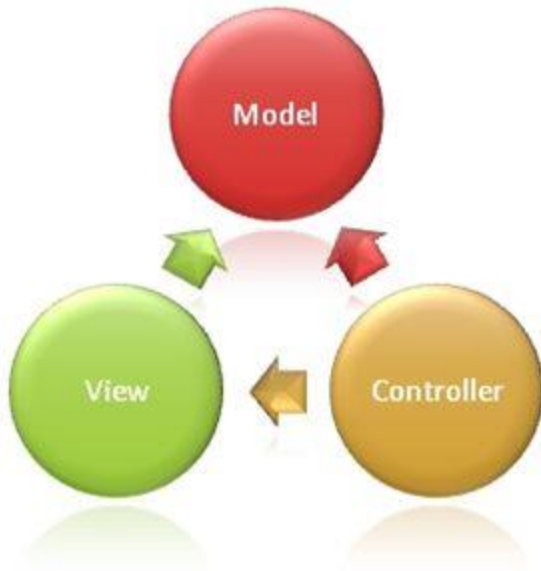
Answer:

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 a 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.

Question 2: What are the advantages of MVC?

Answer: Benefits 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 the 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 ASP.NET framework and hence most of the features of the ASP.NET like membership providers, roles, etc can still be used.

Question 3: Explain MVC application life cycle?

Answer: Any web application has two main execution steps, first understanding the request and depending on the type of the request sending out 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 the 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 route table with routes collection. This filling of route table happens in 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 Response object

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

4)Can you explain the page life cycle of MVC?

Below are the processed followed in the sequence -

→App initialization

- Routing
- Instantiate and execute controller
- Locate and invoke controller action
- Instantiate and render view.

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

Answer: There are total nine return types we can use to return results from controller to view.

The base type of all these result types is ActionResult.

1. **ViewResult (View):** This return type is used to return a webpage from an action method.
2. **PartialviewResult (Partialview):** This return type is used to send a part of a view which will be rendered in another view.
3. **RedirectResult (Redirect):** This return type is used to redirect to any other controller and action method depending on the URL.
4. **RedirectToRouteResult (RedirectToAction, RedirectToRoute):** This return type is used when we want to redirect to any other action method.
5. **ContentResult (Content):** This return type is used to return HTTP content type like text/plain as the result of the action.
6. **jsonResult (json):** This return type is used when we want to return a JSON message.
7. **javascriptResult (javascript):** This return type is used to return JavaScript code that will run in browser.
8. **FileResult (File):** This return type is used to send binary output in response.
9. **EmptyResult:** This return type is used to return nothing (void) in the result.

- [Various Return Types From MVC Controller](#)

Question 5: What are Filters in MVC?

Answer: 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 feature to add pre and post action behaviors on controller's action methods.

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 most commonly used filters to perform additional data processing, or manipulating the return values or cancelling the execution of action or modifying the view structure at run time.

- [Understanding Filters in MVC](#)

Question 6: What are Action Filters in MVC?

Answer: 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

E.g.: Specifies the return value to be cached for 10 seconds.

```

1. public class ActionFilterDemoController: Controller
2. {
3.     [HttpGet]
4.     OutputCache(Duration = 10)]
5.     public string Index()
6.     {
7.         return DateTime.Now.ToString("T");
8.     }
9. }
10. }
```

- [ASP.NET MVC with Action Filters](#)

Question 7: Explain what is routing in MVC? What are the three segments for routing important?

Answer: Routing is a mechanism to process the incoming url that is more descriptive and give 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).

1. **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.
2. **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 words 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,

1. ControllerName
2. ActionMethodName
3. Parameter

i.e: ControllerName/ActionMethodName/{ParameterName} and also route map coding written in a Global.asax file.

- [Routing in MVC](#)

Question 8: What is Route in MVC? What is Default Route in MVC?

Answer: 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 Routes property 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 MvcApplication 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

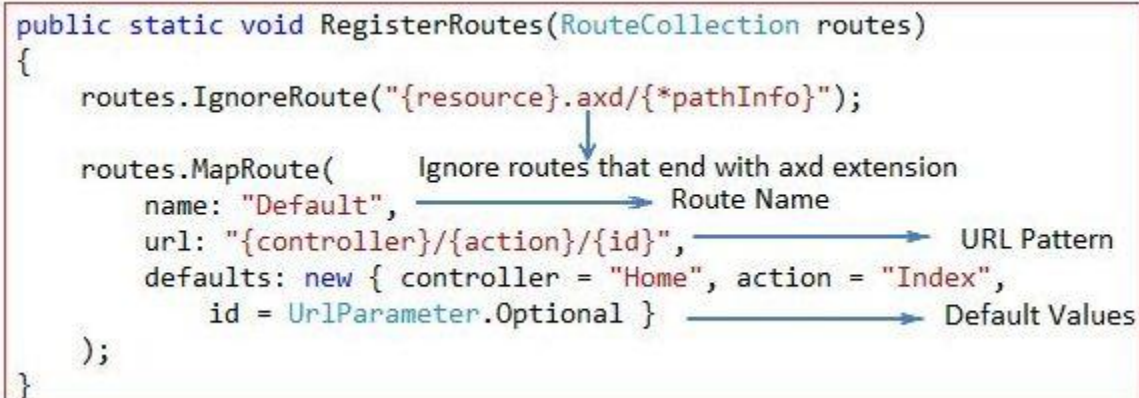
Default Route

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 call to the **MapRoute()** extension method of **RouteCollection**

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index",
                       id = UrlParameter.Optional }
    );
}
```



Question 9: Mention what is the difference between Temp data, View, and View Bag?

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

ViewData

1. ViewData is used to pass data from controller to view.
2. It is derived from ViewDataDictionary class.
3. It is available for the current request only.
4. Requires typecasting for complex data type and checks for null values to avoid error.
5. If redirection occurs, then its value becomes null.

ViewBag

1. ViewBag is also used to pass data from the controller to the respective view.
2. ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0

3. It is also available for the current request only.
4. If redirection occurs, then its value becomes null.
5. Doesn't require typecasting for complex data type.

TempData

1. TempData is derived from TempDataDictionary class
2. TempData is used to pass data from the current request to the next request
3. 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
4. It requires typecasting for complex data type and checks for null values to avoid error. Generally, it is used to store only one time messages like the error messages and validation messages

Question 12: What are HTML helpers in MVC?

Answer:

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:

1. `<%=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:

```
1. <a href="/Home/About">About this Website</a>
```

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()`

Question 13: Explain attribute based routing in MVC?

Answer:

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

```
1. public class HomeController: Controller
2. {
3.     [Route("Users/about")]
4.     public ActionResult GotoAbout()
5.     {
6.         return View();
7.     }
8. }
```

Question 14: What is TempData in MVC?

Answer: 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 HTTP 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

```
1. public ActionResult FirstRequest()
2. {
3.     List < string > TempDataTest = new List < string > ();
4.     TempDataTest.Add("Tejas");
5.     TempDataTest.Add("Jignesh");
6.     TempDataTest.Add("Rakesh");
7.     TempData["EmpName"] = TempDataTest;
8.     return View();
9. }
10. public ActionResult ConsecutiveRequest()
11. {
```



```
12.         List < string > modelData = TempData["EmpName"] as List  
13.         < string > ;  
14.         return View(modelData);  
15.     }
```

- [All About the TempData in MVC](#)

Question 15: What is Razor in MVC?

Answer:

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?

1. Compact & Expressive.
2. 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.
3. 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.
4. 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).
5. Has great Intellisense:
6. 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).

Question 19: Explain Areas in MVC?

Answer: 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 the Global.asax file that can automatically find the area routes in the AreaRegistration file.

Benefits of Area in MVC

1. Allows us to organize models, views and controllers into separate functional sections of the application, such as administration, billing, customer support and much more.
2. Easy to integrate with other Areas created by another.
3. Easy for unit testing.

Question 20: Explain the need of display mode in MVC?

Answer:

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:

1. We should register Display Mode with a suffix for particular browser using “DefaultDisplayMode”e class in Application_Start() method in the Global.asax file.
2. View name for particular browser should be appended with suffix mentioned in first step.

Question 24: What is Output Caching in MVC?

Answer:

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.

1. Any
2. Client
3. Downstream
4. Server
5. None
6. 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".

- [Output Caching in MVC](#)

Question 25: 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.

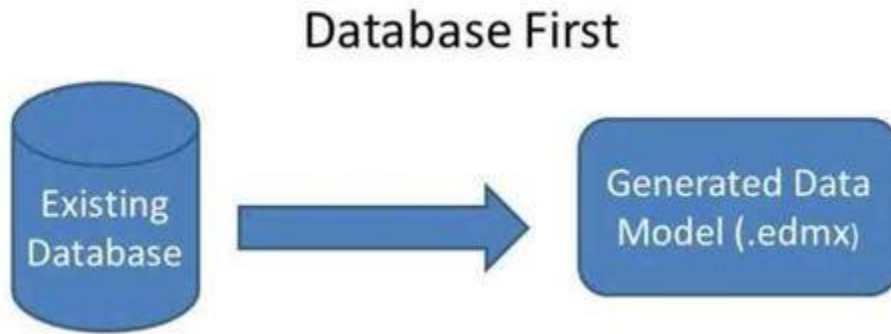
Question 27:What is Database First Approach in MVC using Entity Framework?

Answer:

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.

Question 42: What are the Exception filters in MVC?

Answer:

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.



cc: Google

Let's start!

A Yellow screen of Death can be said is as a wardrobe malfunction of our application.

Get Started

Exception filters run when some of the exceptions are unhandled and thrown from an invoked action. The reason for the exception can be anything and so is the source of the exception.

Creating an Exception Filter

Custom Exception Filters must implement the builtin `IExceptionHandler` interface. The interface looks as in the following:

```
1. public interface IExceptionHandler
2. {
3.     void OnException(ExceptionContext filterContext)
4. }
```

Whenever an unhandled exception is encountered, the `OnException` method gets invoked. The parameter as we can see, `ExceptionContext` is derived from the `ControllerContext` and has a number of built-in properties that can be used to get the information about the request causing the exception.

Their property's `ExceptionContext` passess are shown in the following table:

Name	Type	Detail
Result	ActionResult	The result returned by the action being invoked.
Exception	Exception	The unhandled exceptions caused from the actions in the applications.
ExceptionHandled	BOOL	This is a very handy property that returns a bool value (true/false) based on if the exception is handled by any of the filters in the applicaiton or not.

The exception being thrown from the action is detailed by the `Exception` property and once handled (if), then the property `ExceptionHandled` can be toggled, so that the other filters would know if the exception has been already handled and cancel the other filter requests to handle. The problem is that if the exceptions are not handled, then the default MVC behavior shows the dreaded yellow screen of death. To the users, that makes a very impression on the users and more importantly, it exposes the application's handy and secure information to the outside world that may have hackers and then the application gets into the road to hell. Thus, the exceptions need to be dealt with very carefully. Let's show one small custom exception filter. This filter can be stored inside the `Filters` folder in the web project of the solution. Let's add a file/class called `CustomExceptionHandler.cs`.

```
1. public class CustomExceptionHandler: FilterAttribute,
2.     IExceptionHandler
3. {
4.     public void OnException(ExceptionContext filterContext)
5.     {
6.         if (!filterContext.ExceptionHandled && filterContext.Exception
7.             is NullReferenceException)
8.         {
9.             filterContext.Result = new RedirectResult("customErrorPage.html");
10.            filterContext.ExceptionHandled = true;
11.        }
12.    }
```

Question 44: Define Controller in MVC?

Answer:

The controller provides model data to the view, and interprets user actions such as button clicks. The controller depends on the view and the model. In some cases, the controller and the view are the same object.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace DemoMVC.Controllers
{
    0 references
    public class EmployeeController : Controller
    {
        0 references
        public string EmployeeNames()
        {
            Action return @"<ul>
                <li>Sourabh Somani</li>
                <li>Shaili Dashora</li>
                <li>Saloni Choudhry</li>
                <li>Mahesh Chand</li>
                <li>DJ</li>
                <li>Dinesh Beniwal</li>
            </ul>";
        }
    }
}
```

The Controllers Folder:

The Controllers Folder contains the controller classes responsible for handling user input and responses. MVC requires the name of all controllers to end with "Controller".

In our example, Visual Web Developer has created the following files:
HomeController.cs (for the Home and About pages) and AccountController.cs
(For the Log On pages):

Question 45: Explain Model in MVC?

Answer:

The model represents the data, and does nothing else. The model does NOT depend on the controller or the view. The MVC Model contains all application logic (business logic, validation logic, and data access logic), except pure view and controller logic. With MVC, models both hold and manipulate application data.

The Models Folder:

The Models Folder contains the classes that represent the application model.

Visual Web Developer automatically creates an AccountModels.cs file that contains the models for application security.

- [Model in ASP.Net MVC : Part 1](#)

Question 46: Explain View in MVC?

Answer:

A view is responsible for displaying all of, or a portion of, data for users. In simple terms, whatever we see on the output screen is a view.

The Views Folder:

The Views folder stores the files (HTML files) related to the display of the application (the user interfaces). These files may have the extensions html, asp, aspx, cshtml, and vbhtml, depending on the language content.

The Views folder contains one folder for each controller. Visual Web Developer has created an Account folder, a Home folder, and a Shared folder (inside the Views folder). The Account folder contains pages for registering and logging in to user accounts. The Home folder is used for storing application pages like the home page and the about page. The Shared folder is used to store views shared between controllers (master pages and layout pages).

Question 49: What is GET and POST Actions Types?

Answer:

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 ] )
```

- [GET and POST Calls to Controller's Method in MVC](#)

Question 50: What's new in MVC 6?

Answer:

In MVC 6 Microsoft removed the dependency of System.Web.Dll from MVC6 because it's so expensive that typically it consume 30k of memory per request and response, whereas now MVC 6 only requires 2k of memory per request and the response is a very small memory consumption.

The advantage of using the cloud-optimized framework is that we can include a copy of the mono CLR with your website. For the sake of one website we do not need to upgrade the .NET version on the entire machine. A different version of the CLR for a different website running side by side.

MVC 6 is a part of ASP.NET 5 that has been designed for cloud-optimized applications. The runtime automatically picks the correct version of the library when our MVC application is deployed to the cloud

6) What is Separation of Concerns in ASP.NET MVC?

It's is the process of breaking the program into various distinct features which overlaps in functionality as little as possible. MVC pattern concerns on separating the content from presentation and data-processing from content.

. 8) What is the meaning of Unobtrusive JavaScript?

This is a general term that conveys a general philosophy, similar to the term REST (Representational State Transfer). Unobtrusive JavaScript doesn't intermix JavaScript code in your page markup.

Eg : Instead of using events like onclick and onsubmit, the unobtrusive JavaScript attaches to elements by their ID or class based on the HTML5 data- attributes.

9) What is the use of ViewModel in MVC?

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

10) What you mean by Routing in MVC?

Routing is a pattern matching mechanism of incoming requests to the URL patterns which are registered in route table. Class—"UrlRoutingModule" is used for the same process.

11) What are Actions in MVC?

Actions are the methods in Controller class which is responsible for returning the view or json data. Action will mainly have return type—"ActionResult" and it will be invoked from method—"InvokeAction()" called by controller.

12) What is Attribute Routing in MVC?

ASP.NET Web API supports this type routing. This is introduced in MVC5. In this type of routing, attributes are being used to define the routes. This type of routing gives more control over classic URI Routing. Attribute Routing can be defined at controller level or at Action level like –

[Route("{action = TestCategoryList}")]—Controller Level

[Route("customers/{TestId:int:min(10)}")]—Action Level

16) Explain Bundle.Config in MVC4?

"BundleConfig.cs" in MVC4 is used to register the bundles by the bundling and minification system. Many bundles are added by default including jQuery libraries like—jquery.validate, Modernizr, and default CSS references

17) How route table has been created in ASP.NET MVC?

Method—"RegisterRoutes()" is used for registering the routes which will be added in "Application_Start()" method of global.asax file, which is fired when the application is loaded or started.

18) Which are the important namespaces used in MVC?

Below are the important namespaces used in MVC -

System.Web.Mvc
System.Web.Mvc.Ajax
System.Web.Mvc.Html
System.Web.Mvc.Async

26) Explain Sections in MVC?

Sections are the part of HTML which is to be rendered in layout page. In Layout page we will use the below syntax for rendering the HTML –

```
@RenderSection("TestSection")
```

And in child pages we are defining these sections as shown below –

```
@section TestSection{  
<h1>Test Content</h1>  
}
```

If any child page does not have this section defined then error will be thrown so to avoid that we can render the HTML like this –

```
@RenderSection("TestSection", required: false)
```

39) What is RouteConfig.cs in MVC 4?

"RouteConfig.cs" holds the routing configuration for MVC. RouteConfig will be initialized on Application_Start event registered in Global.asax.

44) Why to use "{resource}.axd/{*pathInfo}" in routing in MVC?

Using this default route—{resource}.axd/{*pathInfo}, we can prevent the requests for the web resources files like—WebResource.axd or ScriptResource.axd from passing to a controller.

45) Can we add constraints to the route? If yes, explain how we can do it?

Yes we can add constraints to route in following ways –

Using Regular Expressions

Using object which implements interface—IRouteConstraint.

If we have multiple filters, what's the sequence for execution?

1. Authorization filters
2. Action filters
3. Response filters
4. Exception filters