2018

Mahindra Finance

PRAMIT JAIN

[TECHNICAL PREPRATION]

[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]

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:

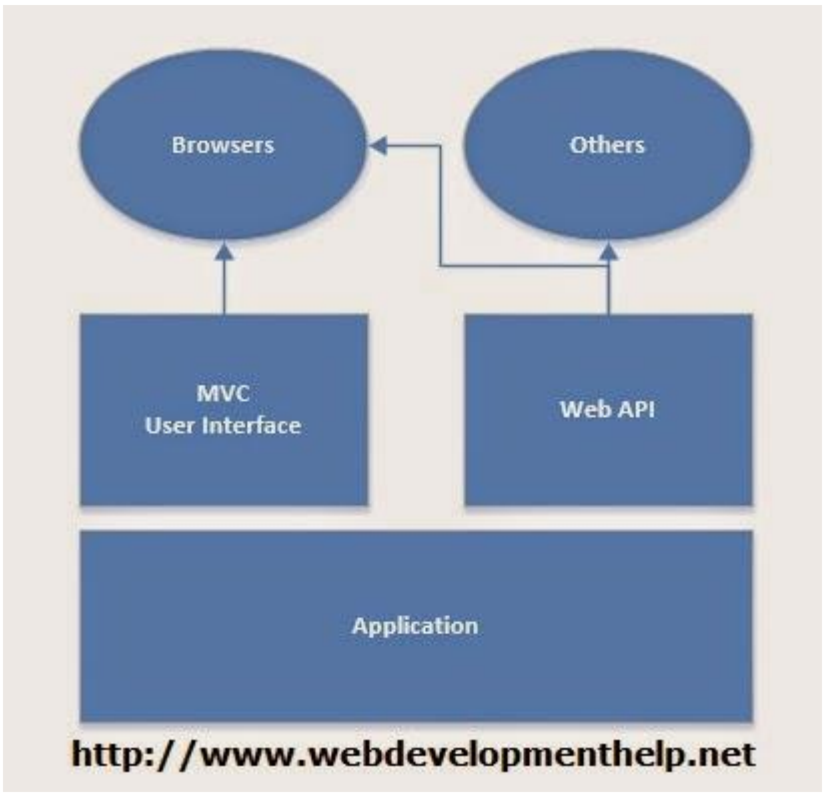
- If we intended to use transport other than HTTP, e.g. TCP, UDP or Named Pipes
- Message Queuing scenario using MSMQ
- One-way communication or Duplex communication

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.

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 a 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#

```
Routes.MapHttpRoute(

Name: "ExampleWebAPIRoute",

routeTemplate: “api/{controller}/{id}

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:

```
[HttpPost]

[ActionName("SaveStudentInfo")]

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

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:

```
[NotImplExceptionFilter]

public TestCustomer GetMyTestCustomer(int custid)

{

```

```
//write the code

}
```

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#

```
Config.Routes.MapHttpRoute(

name: "MyRoute,"//route name

routeTemplate: "api/{controller}/{action}/{id}",//as you can see "API" is at the beginning.

defaults: new { id = RouteParameter.Optional }

);
```

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
- HttpActionResult
- 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

- MVCis used to create a web app, in which we can build web pages.
- For JSONit 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 atleast 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 is 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) { … }
}
```

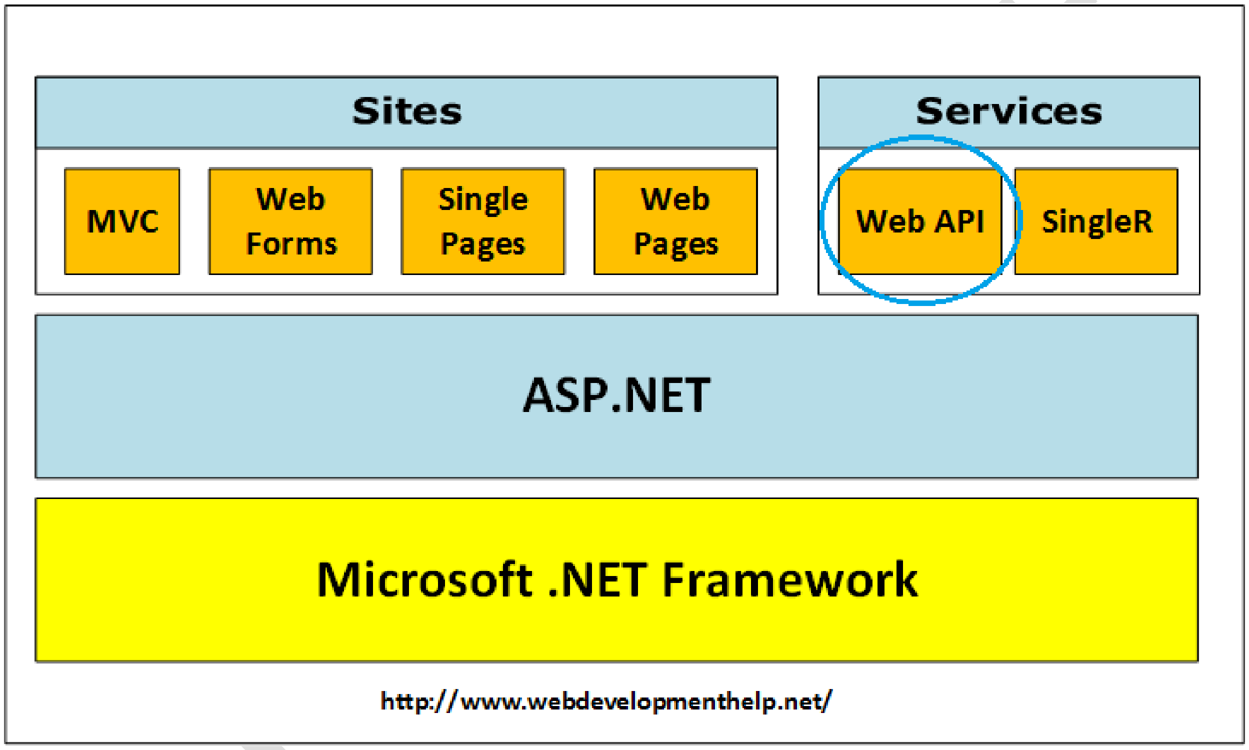
60) Why to use "FromBody" in Web API?
This attribute is used to force Web API to read the simple type from message body. "FromBody" attribute is along with parameter. Eg:

```
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)?

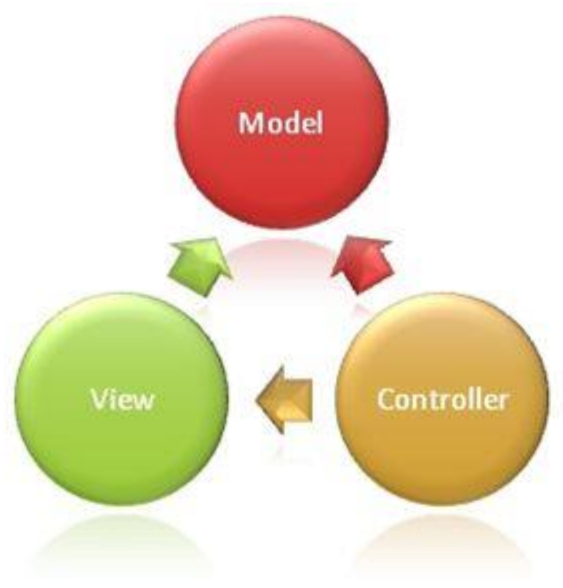
Answer:

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.

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

}
        }
```

- [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 mappped together to a handler,rounting 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,

1. ControllerName
2. ActionMethodName
3. Parammeter

i.e: ControllerName/ActionMethodName/{ParamerName} 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 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.

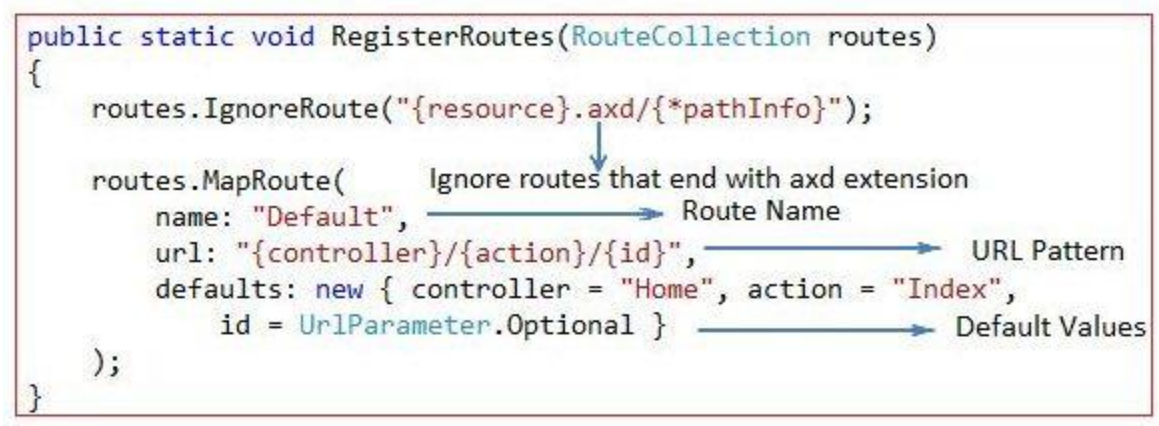
Route definition	Example of matching URL
{controller}/{action}/{id}	/Products/show/beverages
{table}/Details.aspx	/Products/Details.aspx
blog/{action}/{entry}	/blog/show/123
{reporttype}/{year}/{month}/{day}	/sales/2008/1/5
{locale}/{action}	/US/show
{language}-{country}/{action}	/en-US/show

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 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:

```
<%=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:

```
<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

```
public class HomeController: Controller
{
    [Route("Users/about")]
    publicActionResultGotoAbout()
    {
        return View();
    }
}
```

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 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);
    }
```

- 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 theGlobal.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 inApplication_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.

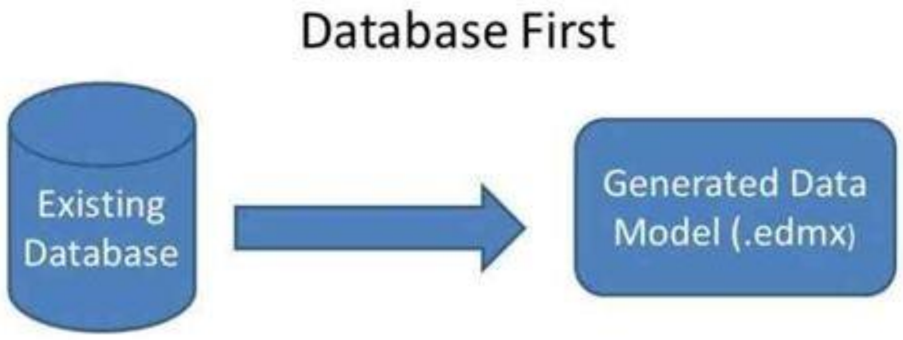
Question 27:What is Database First Approach in MVC using Entity Framework?

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



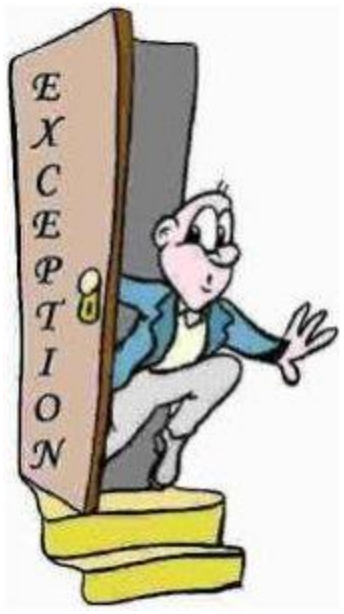
Database first is nothing but only a approach to create web application where database is available first and can interact with the database. In this database, database is created first and after that we manage the code. The Entity Framework is able to generate a business model based on the tables and 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 builtinIExceptionFilter interface. The interface looks as in the following:

```
public interface IExceptionFilter
{
    void OnException(ExceptionContext filterContext)
}
```

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 ExceptionContextpassess 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.

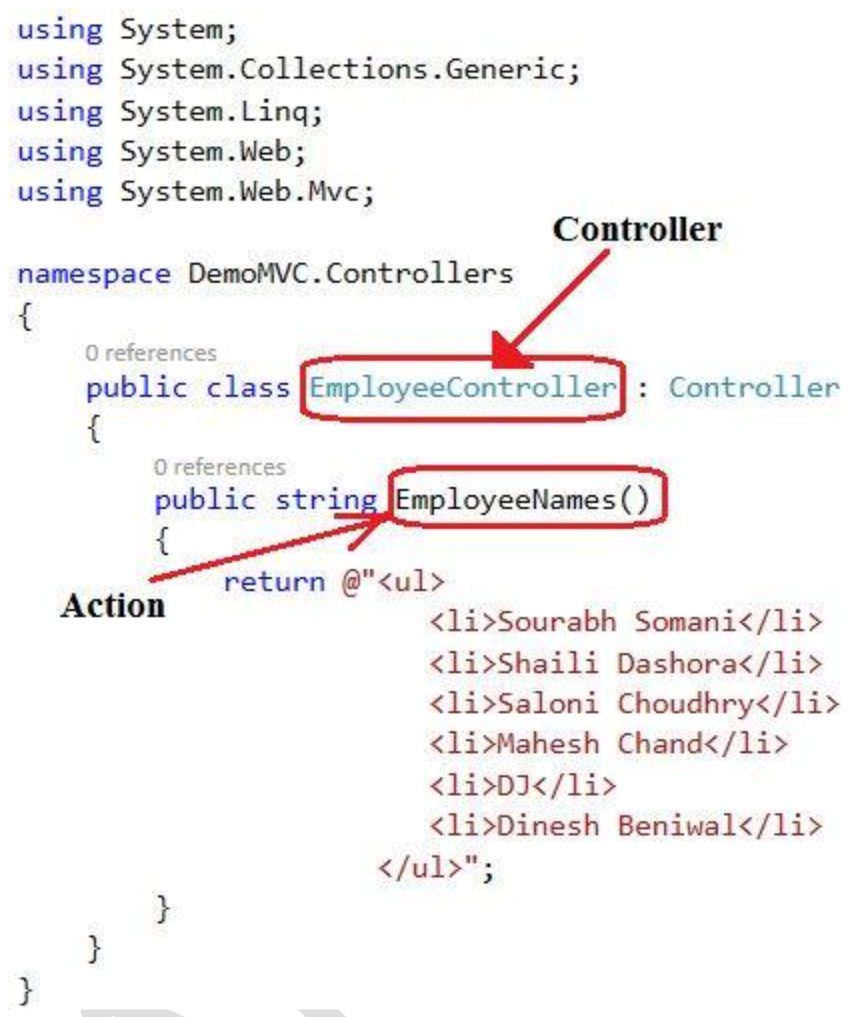
The exception being thrown from the action is detailed by the Exception property and once handled (if), then the property ExceptionHandled can be toggled, so that the other filters would know if the exception has been already handled and cancel the other filter requests to handle. The problem is that 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 CustomExceptionFilter.cs.

```
public class CustomExceptionFilter: FilterAttribute,
    IExceptionFilter
{
        public void OnException(ExceptionContext filterContext)
        {
            if (!filterContext.ExceptionHandled && filterContext.Exception
is NullReferenceException)
            {
                filterContext.Result = new RedirectResult("customErrorPag
e.html");
                filterContext.ExceptionHandled = true;
                }
            }
        }
```

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()
        {
            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 consumtion.

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/{TestCategoryId: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 is MVC?
Section 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

6. What is a primary key?
A primary key is a combination of fields which uniquely specify a row. This is a special kind of unique key, and it has implicit NOT NULL constraint. It means, Primary key values cannot be NULL.

7. What is a unique key?
A Unique key constraint uniquely identified each record in the database. This provides uniqueness for the column or set of columns.
A Primary key constraint has automatic unique constraint defined on it. But not, in the case of Unique Key.
There can be many unique constraint defined per table, but only one Primary key constraint defined per table.

8. What is a foreign key?
A foreign key is one table which can be related to the primary key of another table. Relationship needs to be created between two tables by referencing foreign key with the primary key of another table.
9. What is a join?
This is a keyword used to query data from more tables based on the relationship between the fields of the tables. Keys play a major role when JOINs are used.

10. What are the types of join and explain each?
There are various types of join which can be used to retrieve data and it depends on the relationship between tables.

- **Inner Join.**

Inner join return rows when there is at least one match of rows between the tables.

- **Right Join.**

Right join return rows which are common between the tables and all rows of Right hand side table. Simply, it returns all the rows from the right hand side table even though there are no matches in the left hand side table.

- **Left Join.**

Left join return rows which are common between the tables and all rows of Left hand side table. Simply, it returns all the rows from Left hand side table even though there are no matches in the Right hand side table.

- **Full Join.**

Full join return rows when there are matching rows in any one of the tables. This means, it returns all the rows from the left hand side table and all the rows from the right hand side table.

11. What is normalization?
Normalization is the process of minimizing redundancy and dependency by organizing fields and table of a database. The main aim of Normalization is to add, delete or modify field that can be made in a single table.

12. What is Denormalization.

DeNormalization is a technique used to access the data from higher to lower normal forms of database. It is also process of introducing redundancy into a table by incorporating data from the related tables.

13. What are all the different normalizations?
The normal forms can be divided into 5 forms, and they are explained below -.

- **First Normal Form (1NF):.**

This should remove all the duplicate columns from the table. Creation of tables for the related data and identification of unique columns.

- **Second Normal Form (2NF):.**

Meeting all requirements of the first normal form. Placing the subsets of data in separate tables and Creation of relationships between the tables using primary keys.

- **Third Normal Form (3NF):.**

This should meet all requirements of 2NF. Removing the columns which are not dependent on primary key constraints.

- **Fourth Normal Form (3NF):.**

Meeting all the requirements of third normal form and it should not have multi- valued dependencies.

14. What is a View?
A view is a virtual table which consists of a subset of data contained in a table. Views are not virtually present, and it takes less space to store. View

can have data of one or more tables combined, and it is depending on the relationship.

15. What is an Index?
An index is performance tuning method of allowing faster retrieval of records from the table. An index creates an entry for each value and it will be faster to retrieve data.

16. What are all the different types of indexes?
There are three types of indexes -.

- **Unique Index.**

This indexing does not allow the field to have duplicate values if the column is unique indexed. Unique index can be applied automatically when primary key is defined.

- **Clustered Index.**

This type of index reorders the physical order of the table and search based on the key values. Each table can have only one clustered index.

- **NonClustered Index.**

NonClustered Index does not alter the physical order of the table and maintains logical order of data. Each table can have 999 nonclustered indexes.

17. What is a Cursor?
A database Cursor is a control which enables traversal over the rows or records in the table. This can be viewed as a pointer to one row in a set of rows. Cursor is very much useful for traversing such as retrieval, addition and removal of database records.

18. What is a relationship and what are they?
Database Relationship is defined as the connection between the tables in a database. There are various data basing relationships, and they are as follows:.

- One to One Relationship.
- One to Many Relationship.
- Many to One Relationship.
- Self-Referencing Relationship.

22. What is a stored procedure?

Stored Procedure is a function consists of many SQL statement to access the database system. Several SQL statements are consolidated into a stored procedure and execute them whenever and wherever required.

23. What is a trigger?

A DB trigger is a code or programs that automatically execute with response to some event on a table or view in a database. Mainly, trigger helps to maintain the integrity of the database.

Example: When a new student is added to the student database, new records should be created in the related tables like Exam, Score and Attendance tables.

24. What is the difference between DELETE and TRUNCATE commands?

DELETE command is used to remove rows from the table, and WHERE clause can be used for conditional set of parameters. Commit and Rollback can be performed after delete statement.

TRUNCATE removes all rows from the table. Truncate operation cannot be rolled back.

25. What are local and global variables and their differences?

Local variables are the variables which can be used or exist inside the function. They are not known to the other functions and those variables cannot be referred or used. Variables can be created whenever that function is called.

Global variables are the variables which can be used or exist throughout the program. Same variable declared in global cannot be used in functions. Global variables cannot be created whenever that function is called.

26. What is a constraint?

Constraint can be used to specify the limit on the data type of table. Constraint can be specified while creating or altering the table statement. Sample of constraint are.

- NOT NULL.
- CHECK.
- DEFAULT.
- UNIQUE.
- PRIMARY KEY.

- FOREIGN KEY.
-

37. Advantages and Disadvantages of Stored Procedure?

Stored procedure can be used as a modular programming – means create once, store and call for several times whenever required. This supports faster execution instead of executing multiple queries. This reduces network traffic and provides better security to the data.

- Disadvantage is that it can be executed only in the Database and utilizes more memory in the database server.
- **Question #17) What is SQL Injection?**
- SQL Injection is a type of database attack technique where malicious SQL statements are inserted into an entry field of database such that once it is executed the database is opened for an attacker. This technique is usually used for attacking Data-Driven Applications to have an access to sensitive data and perform administrative tasks on databases.

What is the difference between view and tables in sql
- Tables are the actual database entities that hold your rows.

- Views are "imaginary" tables that are constructed based on the actual tables

Tables as you know store the actual data. Views will require and use actual columns from the actual tables.

Q55. What are Views used for?

A view refers to a logical snapshot based on a table or another view. It is used for the following reasons:

- Restricting access to data.
- Making complex queries simple.
- Ensuring data independence.
- Providing different views of same data.

```
CREATE view [dbo].[vw_getStudent]
as
        select top 100 * from t_BVR_Visit_Hdr a inner join t_BVR_Visit_Dtl b  on
a.f_Details_HdrSequence = b.f_Details_HdrSequence order by 1 desc
GO
```

Index

```
CREATE NONCLUSTERED INDEX [IDX_NC_T_BVR_VISIT_HDR_F_ACTIVE] ON
[dbo].[t_BVR_Visit_Hdr]
(
        [f_Active] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB
= OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO
```

Transaction

```
 BEGIN TRY
  BEGIN TRANSACTION;

COMMIT TRANSACTION;
    END TRY

BEGIN CATCH
   IF @@TRANCOUNT > 0
    ROLLBACK TRANSACTION;

 DECLARE @ErrorNumber INT = ERROR_NUMBER();
    DECLARE @ErrorLine INT = ERROR_LINE();
    DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
    DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
    DECLARE @ErrorState INT = ERROR_STATE();

    RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);
  END CATCH
```

Foreign Key Reference

```
Alter table table_name add foreign key(column1) references table_Name(column2)
```

DELETE	TRUNCATE

Delete command is used to delete a row in a table.	Truncate is used to delete all the rows from a table.
You can rollback data after using delete statement.	You cannot rollback data.
It is a DML command.	It is a DDL command.
It is slower than truncate statement.	It is faster.

Q21. What is the difference between DROP and TRUNCATE commands?

DROP command removes a table and it cannot be rolled back from the database whereas TRUNCATE command removes all the rows from the table.

Q42. List the ways in which Dynamic SQL can be executed?

Following are the ways in which dynamic SQL can be executed:

- Write a query with parameters.
- Using EXEC.
- Using sp_executesql.

23. What is the difference between Rename and Alias?
'Rename' is a permanent name given to a table or column
'Alias' is a temporary name given to a table or column.

36. What is a *NULL* value?
A field with a *NULL* value is a field with no value. A *NULL*value is different from a zero value or a field that contains spaces. A field with a *NULL* value is one that has been left blank during record creation. Assume, there is a field in a table is optional and it is possible to insert a record without adding a value to the optional field then the field will be saved with a *NULL* value.

37. What is the difference between NULL value, Zero, and Blank space?
As I mentioned earlier, Null value is field with no value which is different from zero value and blank space.
Null value is a field with no value.

Zero is a number
Blank space is the value we provide. The ASCII value of space is CHAR(32).

38. How to Test for *NULL* Values?
A field with a *NULL* value is a field with no value. *NULL*value cannot be compared with other NULL values. Hence, It is not possible to test for *NULL* values with comparison operators, such as =, <, or <>. For this, we have to use the *IS NULL* and *IS NOT NULL* operators.

```
SELECT column_names FROM table_name WHERE column_name IS
NULL;
```

```
SELECT column_names FROM table_name WHERE column_name IS
NOT NULL;
```

39. What is SQL *NOT NULL* constraint?
NOT NULL constraint is used to ensure that the value in the filed cannot be a NULL

47. What is the largest value that can be stored in a *BYTE* data field?
The largest number that can be represented in a single byte is 11111111 or 255. The number of possible values is 256 (i.e. 255 (the largest possible value) plus 1 (zero), or 2^8**).**

Which TCP/IP port does SQL Server run?
By default it is 1433

Improve stored procedure performance in SQL Server

This blog covers simple yet useful tips and optimization to improve stored procedure performance.

1. Use SET NOCOUNT ON

SQL Server returns informational messages when running select or DML operations. In case a procedure has many such statements a cursor or a while loop SQL Server will display lot of such messages increasing network traffic. These messages can be suppressed with SET NOCOUNT ON and can increase performance by decreasing network traffic.

2. Use fully qualified procedure name

A fully qualified object name is database.schema.objectname. When stored procedure is called as schemaname.procedurename, SQL Server can swiftly find the compiled plan instead of looking for procedure in other schemas when schemaname is not specified. This may not be a great boost to the performance but should be followed as best practice. All objects inside procedure should also be referred as schemaname.objectname.

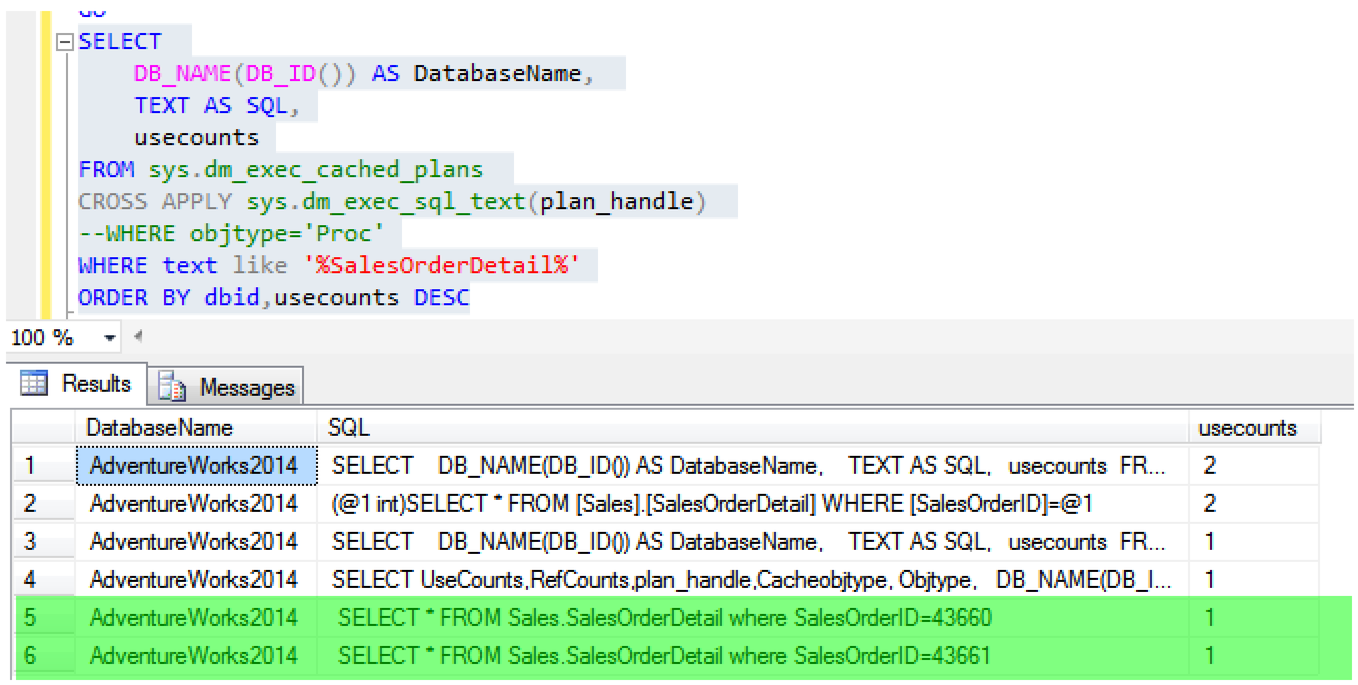
3. sp_executesql instead of Execute for dynamic queries

The sp_executesql allows for cache plan reuse and protects from SQL Injection. Let's see an example of the plan reuse.

```
DBCC FREEPROCCACHE
GO
Declare
            @dynamic_sql varchar(max),
            @salesorderid int
SET @salesorderid=43660

SET @dynamic_sql=' SELECT * FROM Sales.SalesOrderDetail where SalesOrderID='
            + CAST(@salesorderid AS VARCHAR(100))
EXECUTE(@dynamic_sql)
```

The above query executes a dynamic query using EXECUTE command for two values of salesorderid 43660 and 43661. Let's analyze the cached plans.



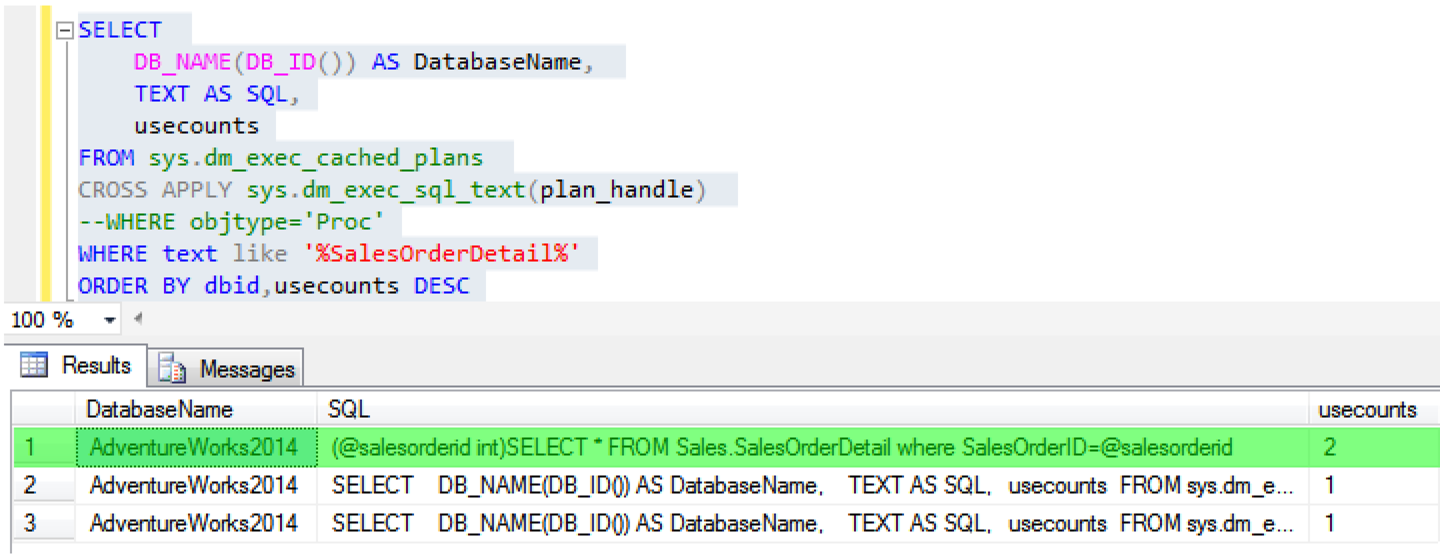
```
SELECT
    DB_NAME(DB_ID()) AS DatabaseName,
    TEXT AS SQL,
    usecounts
FROM sys.dm_exec_cached_plans
CROSS APPLY sys.dm_exec_sql_text(plan_handle)
--WHERE objtype='Proc'
WHERE text like '%SalesOrderDetail%'
ORDER BY dbid,usecounts DESC
```

	DatabaseName	SQL	usecounts
1	AdventureWorks2014	SELECT DB_NAME(DB_ID()) AS DatabaseName, TEXT AS SQL, usecounts FR...	2
2	AdventureWorks2014	(@1 int)SELECT * FROM [Sales].[SalesOrderDetail] WHERE [SalesOrderID]=@1	2
3	AdventureWorks2014	SELECT DB_NAME(DB_ID()) AS DatabaseName, TEXT AS SQL, usecounts FR...	1
4	AdventureWorks2014	SELECT UseCounts,RefCounts,plan_handle,Cacheobjtype, Objtype, DB_NAME(DB_I...	1
5	AdventureWorks2014	SELECT * FROM Sales.SalesOrderDetail where SalesOrderID=43660	1
6	AdventureWorks2014	SELECT * FROM Sales.SalesOrderDetail where SalesOrderID=43661	1

As shown in above snapshot, there are two separate plans for the two salesorderids. Let's now execute the same query with sp_execute SQL and analyze the cached plans.

```
DECLARE @dynamic_sql NVARCHAR(100)
SET @dynamic_sql = N'SELECT * FROM Sales.SalesOrderDetail where SalesOrderID=@salesorderid'
EXECUTE sp_executesql @dynamic_sql, N'@salesorderid int', @salesorderid = 43661
```

The above query uses sp_executesql to execute the dynamic query for 2 different values of salesorderid. Let's analyze the cached plans.



```
SELECT
    DB_NAME(DB_ID()) AS DatabaseName,
    TEXT AS SQL,
    usecounts
FROM sys.dm_exec_cached_plans
CROSS APPLY sys.dm_exec_sql_text(plan_handle)
--WHERE objtype='Proc'
WHERE text like '%SalesOrderDetail%'
ORDER BY dbid,usecounts DESC
```

100 %

Results | Messages

	DatabaseName	SQL	usecounts
1	AdventureWorks2014	(@salesorderid int)SELECT * FROM Sales.SalesOrderDetail where SalesOrderID=@salesorderid	2
2	AdventureWorks2014	SELECT DB_NAME(DB_ID()) AS DatabaseName, TEXT AS SQL, usecounts FROM sys.dm_e...	1
3	AdventureWorks2014	SELECT DB_NAME(DB_ID()) AS DatabaseName, TEXT AS SQL, usecounts FROM sys.dm_e...	1

As shown in above snapshot, only one plan is cached and is used for different values of salesorderid.

4. Using IF EXISTS AND SELECT

IF EXISTS is used to check existence of a record, object etc.. And is a handy statement to improve performance of queries where in one only wants to check existence of a record in a table instead of using that record/row in the query. When doing so use IF EXISTS(SELECT 1 from mytable) instead of IF EXISTS(Select * from mytable) as only thing we are interested in is to check the presence of record/s. So, if the query return 1 then record is present else it's not. It's needless to return all column values.

5. Avoid naming user stored procedure as sp_procedurename.

If a stored procedure begins with sp_ then SQL Server first searches it in master database and then in the current user database. This might cause slight performance issues and moreover it may result in wrong results if a stored procedure with same name exists in master database.

6. Use set based queries wherever possible.

T-SQL is a set based language and thus loops don't work well in here. Cursors and while loop are to be used only when a set based query is either expensive or can't be formulated.

7. Keep transaction short and crisp

The longer the transaction the longer the locks will be held based on isolation level. This may result in deadlocks and blocking. Open a new query window and execute the below query

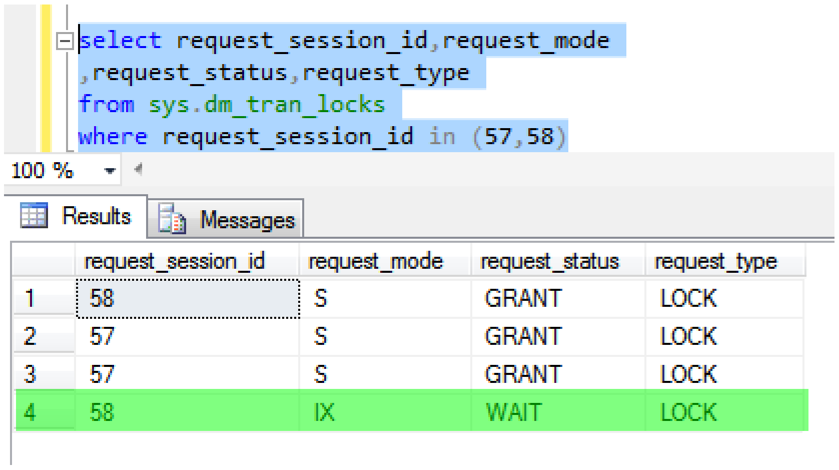
```
use AdventureWorks2014
GO
BEGIN TRANSACTION
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

SELECT * FROM Sales.SalesOrderDetail
```

Note the session id for the query. Open a new query window and execute the below query. Note down the session id of the query.

```
begin tran
Update Sales.SalesOrderDetail
SET OrderQty=50 WHERE SalesOrderDetailID=1
```

The above update query will wait on the select query on shared lock. Let's analyze the locks for these two sessions.



```
select request_session_id,request_mode
,request_status,request_type
from sys.dm_tran_locks
where request_session_id in (57,58)
```

	request_session_id	request_mode	request_status	request_type
1	58	S	GRANT	LOCK
2	57	S	GRANT	LOCK
3	57	S	GRANT	LOCK
4	58	IX	WAIT	LOCK

Do follow these tips and let me know how does it increases procedure performance. Will come back with some more tips and best practices.

Using Index in temp table and Table Variables also

```
DECLARE @PAN_FD_Cluster TABLE(PAN_NO
NVARCHAR(20),QUICK_BUY_Cluster_Cnt BIGINT,Customer_Portal_Cluster_Cnt
BIGINT,Offline_Portal_Cluster_Cnt BIGINT,UNIQUE NONCLUSTERED (PAN_NO))
```

What SCOPE_IDENTITY is

SCOPE_IDENTITY is:

1. SCOPE_IDENTITY returns the last IDENTITY value inserted into an IDENTITY column in the same scope.
2. SCOPE_IDENTITY returns the last identity value generated for any table in the current session and the current scope.

3. A scope is a module; a Stored Procedure, trigger, function, or batch.
4. Thus, two statements are in the same scope if they are in the same Stored Procedure, function, or batch.
5. The SCOPE_IDENTITY() function will return the NULL value if the function is invoked before any insert statements into an identity column occur in the scope.

What IDENT_CURRENT is

IDENT_CURRENT is:

1. IDENT_CURRENT returns the last identity value generated for a specific table in any session and any scope.
2. IDENT_CURRENT is not limited by scope and session; it is limited to a specified table.

What @@IDENTITY is

@@IDENTITY is:

1. @@IDENTITY returns the last identity value generated for any table in the current session, across all scopes.
2. After an INSERT, SELECT INTO, or bulk copy statement completes, @@IDENTITY contains the last identity value generated by the statement.
3. If the statement did not affect any tables with identity columns, @@IDENTITY returns NULL.
4. If multiple rows are inserted, generating multiple identity values, @@IDENTITY returns the last identity value generated.
5. The @@IDENTITY value does not revert to a previous setting if the INSERT or SELECT INTO statement or bulk copy fails, or if the transaction is rolled back.

Differences

The differences between them are:

1. SCOPE_IDENTITY, IDENT_CURRENT, and @@IDENTITY are similar functions in that they return values inserted into IDENTITY columns.
2. SCOPE_IDENTITY and @@IDENTITY will return the last identity values generated in any table in the current session. However, SCOPE_IDENTITY returns values inserted only within the current scope; @@IDENTITY is not limited to a specific scope. A scope is a module; a Stored Procedure, trigger, function, or batch.

Example

1. Create two tables as below:

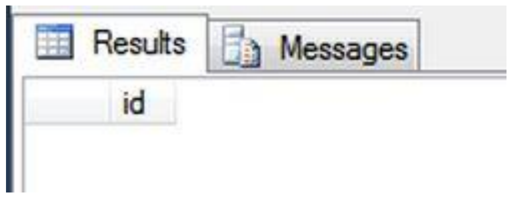
```
CREATE TABLE Table1(id int IDENTITY)
CREATE TABLE Table2(id int IDENTITY(100,1))
```

2. Create a trigger on table1 as below:

```
CREATE TRIGGER TG_Table1 ON Table1 FOR INSERT
AS
BEGIN
	INSERT table2 DEFAULT VALUES
END
```

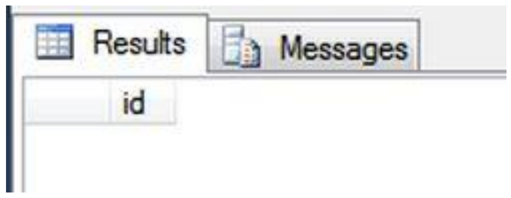
3. Do a select statement of both the tables:

```
SELECT * FROM Table1
```



```
id
```

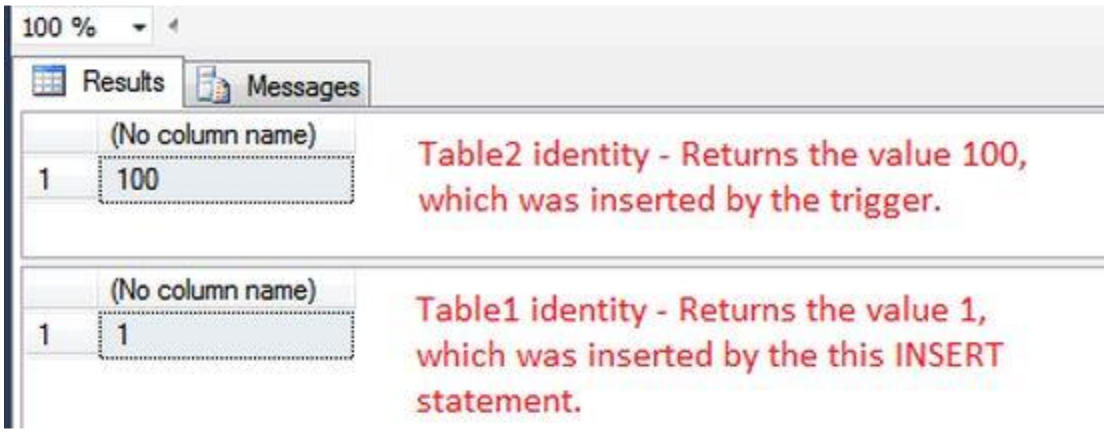
```
SELECT * FROM Table2
```



```
id
```

4. Run the following SQL statements and observe the output

```
INSERT Table1 DEFAULT VALUES
SELECT @@IDENTITY      -- It will consider identity value changed by trigger as
trigger is another scope.
SELECT SCOPE_IDENTITY() -- It will NOT consider identity value changed by trigger
as trigger is another scope.
```



```
100 %
Results | Messages
   (No column name)
1  100
```

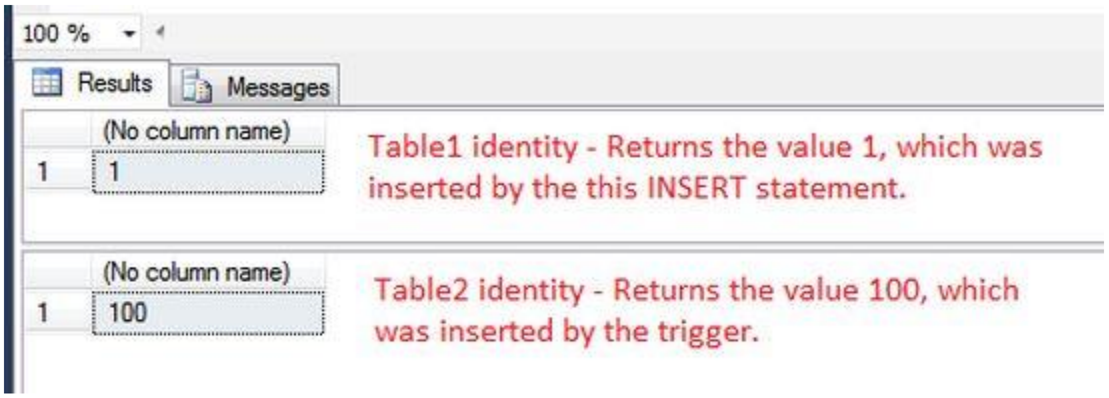
Table2 identity - Returns the value 100, which was inserted by the trigger.

```
   (No column name)
1  1
```

Table1 identity - Returns the value 1, which was inserted by the this INSERT statement.

5. Run the following SQL statements for ident_current:

```
SELECT IDENT_CURRENT('Table1')
SELECT IDENT_CURRENT('Table2')
```



```
100 %
Results | Messages
   (No column name)
1  1
```

Table1 identity - Returns the value 1, which was inserted by the this INSERT statement.

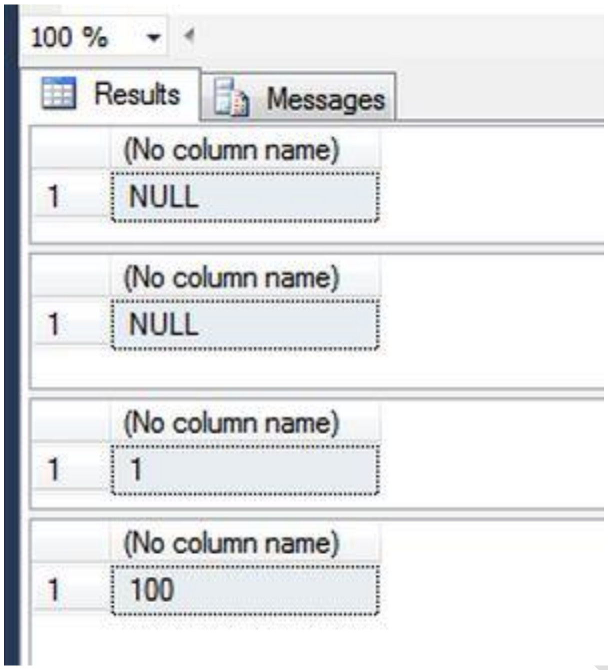
```
   (No column name)
1  100
```

Table2 identity - Returns the value 100, which was inserted by the trigger.

6. Run the following SQL statements in a different query window, in other words a different session:

```
SELECT @@IDENTITY
SELECT SCOPE_IDENTITY()

SELECT IDENT_CURRENT('Table1')
SELECT IDENT_CURRENT('Table2')
```



```
(No column name)
1  NULL

(No column name)
1  NULL

(No column name)
1  1

(No column name)
1  100
```

```
ALTER procedure [dbo].[usp_FD_QB_CSE_GetCustomerInfo_V3](@PanNo
nvarchar(100)='', @AppNo nvarchar(100),@TemplateType int OUTPUT        )
as
begin

 SET NOCOUNT ON;

select * from t_FD_Qb_Cust_Investor_Info info

FOR XML RAW ('EmpData'), ROOT
 select @TemplateType = 1;
 SET NOCOUNT OFF;

 End


select * from t_BVR_Students where 1=1  -- Will return all the records from the
table because conditions  satisfied

select * from t_BVR_Students where 1=NULL -- Will return NO records from the
table as conditions  doesn't satisfied
```

ASP.NET

Question 1: What is ASP.NET?

Answer: ASP.NET was developed in direct response to the problems that developers had with classic ASP. Since ASP is in such wide use, however, Microsoft ensured that ASP scripts execute without modification on a machine with the .NET Framework (the ASP engine, ASP.DLL, is not modified when installing the .NET Framework). Thus, IIS can house both ASP and ASP.NET scripts on the same machine.

Advantages of ASP.NET

1. **Separation of Code from HTML:**

   To make a clean sweep, with ASP.NET you have the ability to completely separate layout and business logic. This makes it much easier for teams of programmers and designers to collaborate efficiently.

2. **Support for compiled languages:**

   Developer can use VB.NET and access features such as strong typing and object-oriented programming. Using compiled languages also means that ASP.NET pages do not suffer the performance penalties associated with interpreted code. ASP.NET pages are precompiled to byte-code and Just In Time (JIT) compiled when first requested. Subsequent requests are directed to the fully compiled code, which is cached until the source changes.

3. **Use services provided by the .NET Framework:**

   The .NET Framework provides class libraries that can be used by your application. Some of the key classes help you with input/output, access to operating system services, data access, or even debugging. We will go into more detail on some of them in this module.

4. **Graphical Development Environment:**

   Visual Studio .NET provides a very rich development environment for web developers. You can drag and drop controls and set properties the way you do in Visual Basic 6. And you have full IntelliSense support, not only for your code, but also for HTML and XML.

5. **State management:**

   To refer to the problems mentioned before, ASP.NET provides solutions for session and application state management. State information can, for example, be kept in memory or stored in a database. It can be shared across web farms, and state information can be recovered, even if the server fails or the connection breaks down.

6. **Update files while the server is running:**

   Components of your application can be updated while the server is online and clients are connected. The framework will use the new files as soon as they are copied to the application. Removed or old files that are still in use are kept in memory until the clients have finished.

7. **XML-Based Configuration Files:**

   Configuration settings in ASP.NET are stored in XML files that you can easily read and edit. You can also easily copy these to another server, along with the other files that comprise your application.

ASP.NET Overview

Here are some points that give the quick overview of ASP.NET.

- ASP.NET provides services to allow the creation, deployment, and execution of Web Applications and Web Services.

- Like ASP, ASP.NET is a server-side technology.

- Web Applications are built using Web Forms. ASP.NET comes with built-in Web Forms controls, which are responsible for generating the user interface. They mirror typical HTML widgets like text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls.

- Web Forms are designed to make building web-based applications as easy as building Visual Basic applications.

For further information click on the link:

Question 11: What are the Advantages of ASP.NET?

Answer: ASP.NET provides services to allow the creation, deployment, and execution of Web Applications and Web Services like ASP, ASP.NET is a server-side technology. Web Applications are built using Web Forms. ASP.NET comes with built-in Web Form controls, which are responsible for generating the user interface. They mirror typical HTML widgets such as text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls.

Advantages of ASP.NET:

- Separation of Code from HTML
- Support for compiled languages
- Use services provided by the .NET Framework
- Graphical Development Environment
- Update files while the server is running

- XML-Based Configuration Files

Question 2: What are the different validators in ASP.NET?

Answer: ASP.NET validation controls define an important role in validating the user input data. Whenever the user gives the input, it must always be validated before sending it across to various layers of an application. If we get the user input with validation, then chances are that we are sending the wrong data. So, validation is a good idea to do whenever we are taking input from the user.

There are the following two types of validation in ASP.NET:

- Client-Side Validation
- Server-Side Validation

Client-Side Validation:

When validation is done on the client browser, then it is known as Client-Side Validation. We use JavaScript to do the Client-Side Validation.

Server-Side Validation:

When validation occurs on the server, then it is known as Server-Side Validation. Server-Side Validation is a secure form of validation. The main advantage of Server-Side Validation is if the user somehow bypasses the Client-Side Validation, we can still catch the problem on server-side.

The following are the Validation Controls in ASP.NET:

- RequiredFieldValidator Control
- CompareValidator Control
- RangeValidator Control
- RegularExpressionValidator Control
- CustomFieldValidator Control
- ValidationSummary

For further information click on the link:

- Validation Controls in ASP.Net

Question 3: What is View State?

Answer: View State is the method to preserve the Value of the Page and Controls between round trips. It is a Page-Level State Management technique. View State is turned on by default and normally serializes the data in every control on the page regardless of whether it is actually used during a post-back.

A web application is stateless. That means that a new instance of a page is created every time when we make a request to the server to get the page and after the round trip our page has been lost immediately

Features of View State

These are the main features of view state:

1. Retains the value of the Control after post-back without using a session.
2. Stores the value of Pages and Control Properties defined in the page.
3. Creates a custom View State Provider that lets you store View State Information in a SQL Server Database or in another data store.

Advantages of View State

1. Easy to Implement.
2. No server resources are required: The View State is contained in a structure within the page load.
3. Enhanced security features: It can be encoded and compressed or Unicode implementation.

For further information click on the link:

- What is View State and How it Works in ASP.NET

Question 4: What are the different Session state management options available in ASP.NET?

Answer: State Management in ASP.NET

- A new instance of the Web page class is created each time the page is posted to the server.

- In traditional Web programming, all information that is associated with the page, along with the controls on the page, would be lost with each roundtrip.

- The Microsoft ASP.NET framework includes several options to help you preserve data on both a per-page basis and an application-wide basis. These options can be broadly divided into the following two categories:

    - Client-Side State Management Options
    - Server-Side State Management Options

Client-Side State Management

- Client-based options involve storing information either in the page or on the client computer.

- Some client-based state management options are:

  - Hidden fields
  - View state
  - Cookies
  - Query strings

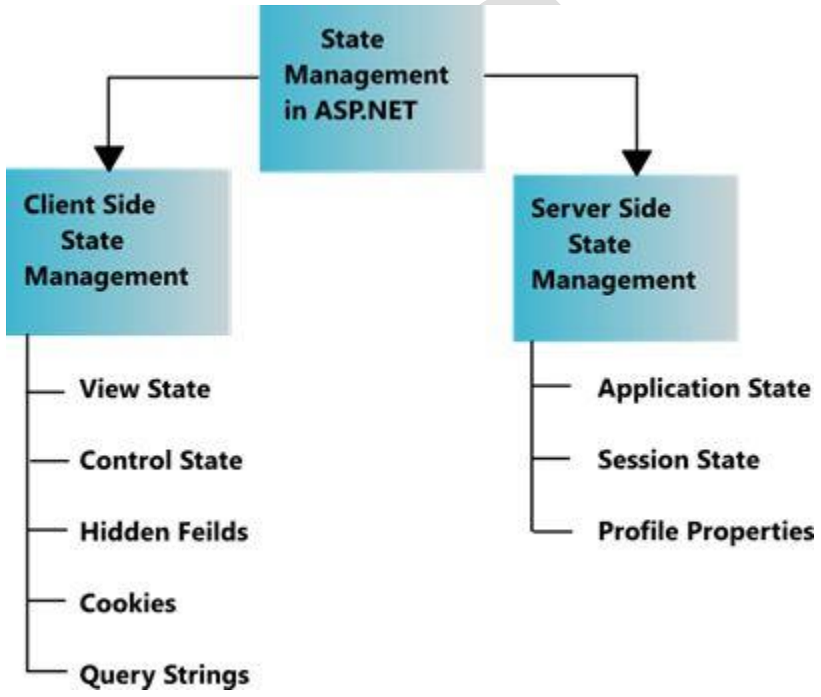
Server-Side State Management

- There are situations where you need to store the state information on the server side.

- Server-side state management enables you to manage application-related and session-related information on the server.

- ASP.NET provides the following options to manage state at the server side:

  - Application state
  - Session state



For further information click on the link:

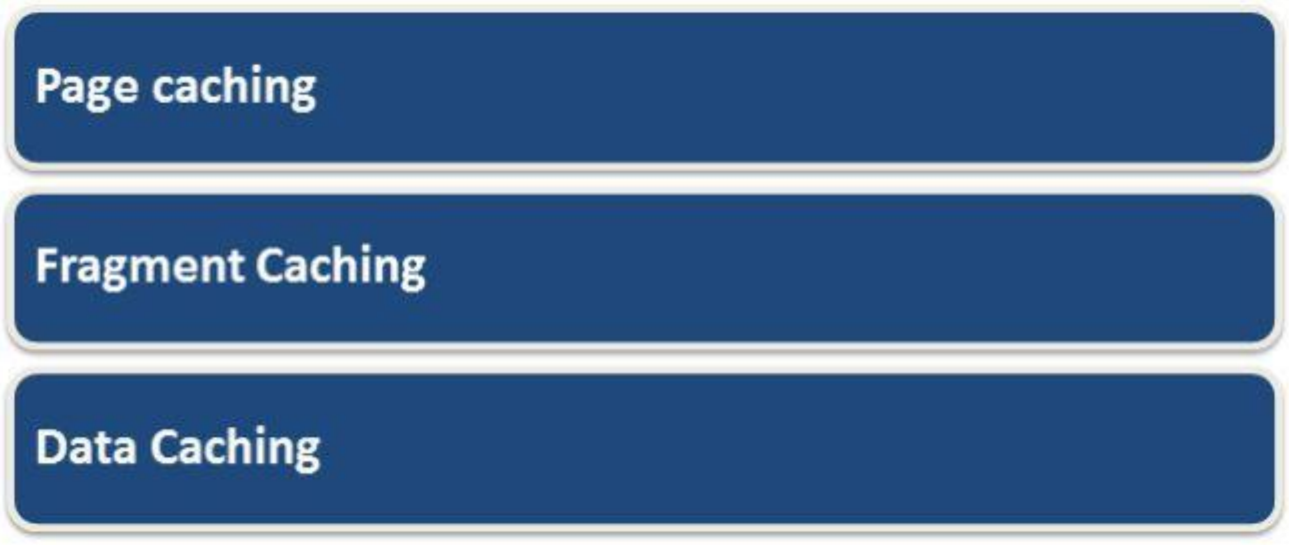
- State Management in ASP.Net

Question 5: What is caching in ASP.NET?

Answer: Caching is one of the most interesting concept and operation in ASP.NET. If you can handle it, you can run any web application by applying the caching concept depending on the requirements.

Caching is for providing solutions or the results to the users depending on their request, admin needs to recreate the pages often depending on user requests…STOP!!! "A cache simply stores the output generated by a page in the memory and this saved output (cache) will serve us (users) in the future.".

Types



For further information click on the link:

Question 8: What are Cookies in ASP.NET?

Answer: Cookies are a State Management Technique that can store the values of control after a post-back. Cookies can store user-specific Information on the client's machine like when the user last visited your site. Cookies are also known by many names, such as HTTP Cookies, Browser Cookies, Web Cookies, Session Cookies and so on. Basically cookies are a small text file sent by the web server and saved by the Web Browser on the client's machine.

List of properties containing the HttpCookies Class:

1. **Domain:** Using these properties we can set the domain of the cookie.
2. **Expires:** This property sets the Expiration time of the cookies.
3. **HasKeys:** If the cookies have a subkey then it returns True.
4. **Name:** Contains the name of the Key.
5. **Path:** Contains the Virtual Path to be submitted with the Cookies.

6. **Secured:** If the cookies are to be passed in a secure connection then it only returns True.
7. **Value:** Contains the value of the cookies.

Limitation of the Cookies

1. The size of cookies is limited to 4096 bytes.
2. A total of 20 cookies can be used in a single website.

For further info click on the link:

- Introduction To Cookies in ASP.Net

Question 9: What is Ajax in ASP.NET?

Answer. Ajax stands for Asynchronous JavaScript and XML; in other words Ajax is the combination of various technologies such as a JavaScript, CSS, XHTML, DOM, etc.

AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the entire page.

We can also define Ajax is a combination of client side technologies that provides asynchronous communication between the user interface and the web server so that partial page rendering occurs instead of complete page post back.

Ajax is platform-independent; in other words AJAX is a cross-platform technology that can be used on any Operating System since it is based on XML & JavaScript. It also supports open source implementation of other technology. It partially renders the page to the server instead of complete page post back. We use AJAX for developing faster, better and more interactive web applications. AJAX uses a HTTP request between web server & browser.

- With AJAX, when a user clicks a button, you can use JavaScript and DHTML to immediately update the UI, and spawn an asynchronous request to the server to fetch results.

- When the response is generated, you can then use JavaScript and CSS to update your UI accordingly without refreshing the entire page. While this is happening, the form on the users screen doesn't flash, blink, disappear, or stall.

- The power of AJAX lies in its ability to communicate with the server asynchronously, using a XMLHttpRequest object without requiring a browser refresh.

- Ajax essentially puts JavaScript technology and the XMLHttpRequest object between your Web form and the server.

For further info click on the link:

- Introduction to Ajax and Ajax Control Toolkit

Question 10: What are Web Services in ASP.NET?

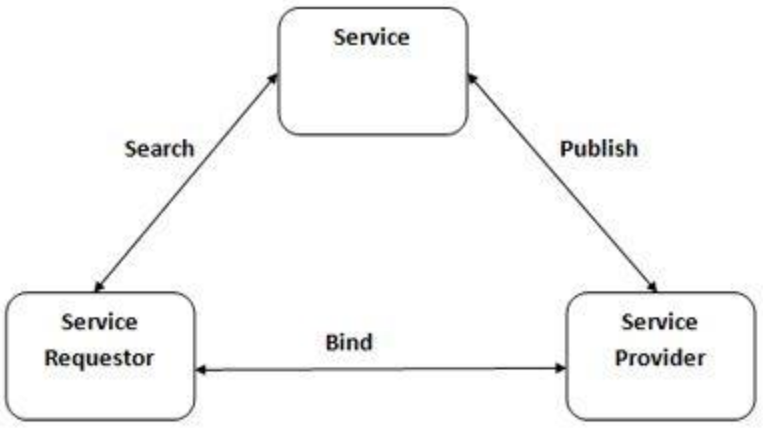
Answer: A Web Service is a software program that uses XML to exchange information with other software via common internet protocols. In a simple sense, Web Services are a way for interacting with objects over the Internet.

A web service is:

- Language Independent.
- Protocol Independent.
- Platform Independent.
- It assumes a stateless service architecture.
- Scalable (e.g. multiplying two numbers together to an entire customer-relationship management system).
- Programmable (encapsulates a task).
- Based on XML (open, text-based standard).
- Self-describing (metadata for access and use).
- Discoverable (search and locate in registries)- ability of applications and developers to search for and locate desired Web services through registries. This is based on UDDI.

Key Web Service Technologies:

- **XML-** Describes only data. So, any application that understands XML-regardless of the application's programming language or platform-has the ability to format XML in a variety of ways (well-formed or valid).
- **SOAP-** Provides a communication mechanism between services and applications.
- **WSDL-** Offers a uniform method of describing web services to other programs.
- **UDDI-** Enables the creation of searchable Web services registries.



Question 13: What is the Web.config file in ASP?

Answer: Configuration file is used to manage various settings that define a website. The settings are stored in XML files that are separate from your application code. In this way you can configure settings independently from your code. Generally a website contains a single Web.config file stored inside the application root directory. However there can be many configuration files that manage settings at various levels within an application.

Usage of configuration file

ASP.NET Configuration system is used to describe the properties and behaviors of various aspects of ASP.NET applications. Configuration files help you to manage the settings related to your website. Each file is an XML file (with the extension .config) that contains a set of configuration elements. Configuration information is stored in XML-based text files.

Benefits of XML-based Configuration files:

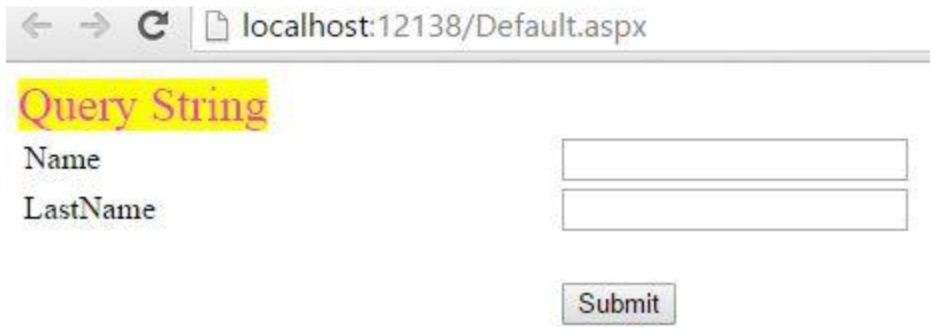
- ASP.NET Configuration system is extensible and application specific information can be stored and retrieved easily. It is human readable.
- You need not restart the web server when the settings are changed in configuration file. ASP.NET automatically detects the changes and applies them to the running ASP.NET application.
- You can use any standard text editor or XML parser to create and edit ASP.NET configuration files.

Question 15: What is Query String in ASP?

Answer: A QueryString is a collection of characters input to a computer or web browser. A Query String is helpful when we want to transfer a value from one page to another. When we need to pass content between the HTML pages or aspx Web Forms in the context of ASP.NET, a Query String is Easy to use and the Query String follows a separating character, usually a Question Mark (?). It is basically used for identifying data appearing after this separating symbol. A Query String Collection is used to retrieve the variable values in the HTTP query string. If we want to transfer a large amount of data then we can't use the Request.QueryString. Query Strings are also generated by form submission or can be used by a user typing a query into the address bar of the browsers.

Syntax of Query String

Request.QueryString(variable)[(index).count]



Advantages:

- Simple to Implement
- Easy to get information from Query string.
- Used to send or read cross domain (from different domain).

Disadvantages:

- Human Readable
- Client browser limit on URL length
- Cross paging functionality makes it redundant
- Easily modified by end user

Question 18: What are the data controls available in ASP.NET?

Answer: The Controls having DataSource Property are called Data Controls in ASP.NET. ASP.NET allows powerful feature of data binding, you can bind any server control to simple properties, collections, expressions and/or methods. When you use data binding, you have more flexibility when you use data from a database or other means.

Data Bind controls are container controls.

Controls -> Child Control

Data Binding is binding controls to data from databases. With data binding we can bind a control to a particular column in a table from the database or we can bind the whole table to the data grid.

Data binding provides simple, convenient, and powerful way to create a read/write link between the controls on a form and the data in their application.

Data binding allows you to take the results of properties, collection, method calls, and database queries and integrate them with your ASP.NET code. You can combine data binding with Web control rendering to relieve much of the programming burden surrounding Web control creation. You can also use data binding with ADO.NET and Web controls to populate control contents from SQL select statements or stored procedures.

Data binding uses a special syntax:

<%# %>

The *<%#,* which instructs ASP.NET to evaluate the expression. The difference between a data binding tags and a regular code insertion tags <% and %> becomes apparent when the expression is evaluated. Expressions within the data binding tags are evaluated only when the DataBind method in the Page objects or Web control is called.

Data Bind Control can display data in connected and disconnected model.

Following are data bind controls in ASP.NET:

- Repeater Control
- DataGrid Control
- DataList Control
- GridView Control
- DetailsView
- FormView
- DropDownList
- ListBox
- RadioButtonList
- CheckBoxList
- BulletList etc.

Question 19: What are the major events in global.aspx?

Answer: The Global.asax file, which is derived from the HttpApplication class, maintains a pool of HttpApplication objects, and assigns them to applications as needed. The Global.asax file contains the following events:

- *Application_Init*

- *Application_Disposed*
- *Application_Error*
- *Application_Start*
- *Application_End*
- *Application_BeginReques*

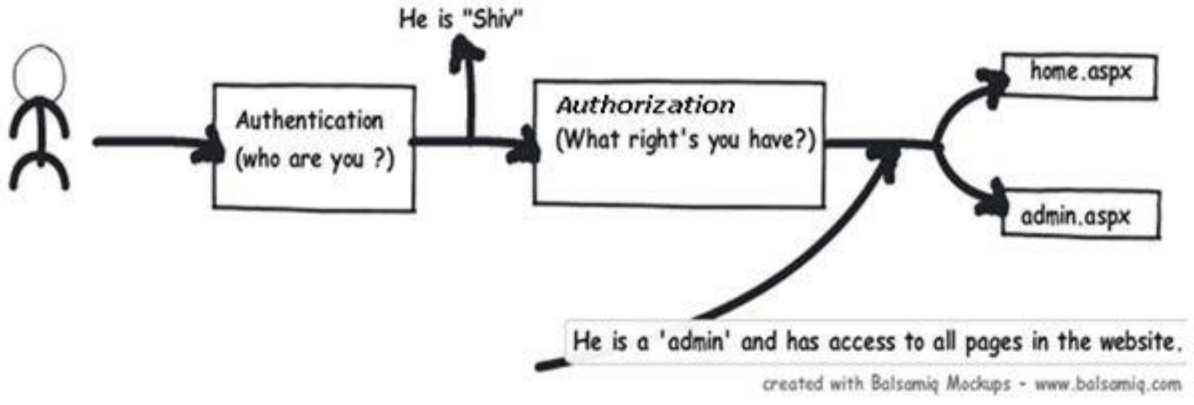
Question 21: What is the authentication and authorization in ASP.NET?

Answer

- **Authentication:** Prove genuineness
- **Authorization:** process of granting approval or permission on resources.

In ASP.NET authentication means to identify the user or in other words its nothing but to validate that he exists in your database and he is the proper user.

Authorization means does he have access to a particular resource on the IIS website. A resource can be an ASP.NET web page, media files (MP4, GIF, JPEG etc), compressed file (ZIP, RAR) etc.


He is "Shiv"
Authentication (who are you ?)
Authorization (What right's you have?)
home.aspx
admin.aspx
He is a 'admin' and has access to all pages in the website.
created with Balsamiq Mockups - www.balsamiq.com

Types of authentication and authorization in ASP.NET

There are three ways of doing authentication and authorization in ASP.NET:

- **Windows authentication:** In this methodology ASP.NET web pages will use local windows users and groups to authenticate and authorize resources.

- **Forms Authentication:** This is a cookie based authentication where username and password are stored on client machines as cookie files or they are sent through URL for every request. Form-based authentication presents the user with an HTML-based Web page that prompts the user for credentials.

- **Passport authentication:** Passport authentication is based on the passport website provided by the Microsoft .So when user logins with credentials it will be reached to the passport website (i.e. hotmail,devhood,windows live etc) where authentication will happen. If Authentication is successful it will return a token to your website.

- **Anonymous access:** If you do not want any kind of authentication then you will go for Anonymous access.

In 'web.config' file set the authentication mode to 'Windows' as shown in the below code snippets.

```
<authentication mode="Windows"/>
```

We also need to ensure that all users are denied except authorized users. The below code snippet inside the authorization tag that all users are denied. '?' indicates any unknown user.

```
<authorization>
  <deny users="?"/>
</authorization>
```

Question 27: What is the ASP.NET page life Cycle?

Answer: When a page is requested by the user from the browser, the request goes through a series of steps and many things happen in the background to produce the output or send the response back to the client. The periods between the request and response of a page is called the "Page Life Cycle".

- **Request:** Start of the life cycle (sent by the user).
- **Response:** End of the life cycle (sent by the server).

There are four stages that occur during the Page Life Cycle before the HTML Response is returned to the client. Later in this article we"ll study all these stages and their sub events.

1. Initialization
2. Loading
3. Rendering
4. Unloading

Initialization	During this stage the IsPostback property is set. The page determines whether the request is a Postback (old request) or if this is the first time the page is being processed (new request). Controls on the page are available and each control's UniqueID property is set. Now if the current request is a postback then the data has not been loaded and the value of the controls have not yet been restored

	from the view state.
Loading	At this stage if the request is a Postback then it loads the data from the view state.
Rendering	Before rendering, the View State is saved for the page and its controls. During this phase, the page calls the render method for each control, providing a text writer that writes its output to the OutputStream of the page's Response property.
Unloading	Unload is called after the page has been fully rendered, sent to the client and is ready to be discarded. At this point also the page properties such as Response and Request are unloaded.

Question 28: What is the ASP.NET page life cycle events?

Answer: We have many events in ASP.NET page life cycle let’s see some most important events:

- **Page request**

  When ASP.NET gets a page request, it decides whether to parse and compile the page or there would be a cached version of the page; accordingly the response is sent,

- **Starting of page life cycle**

  At this stage, the Request and Response objects are set. If the request is an old request or post back, the IsPostBack property of the page is set to true. The UICulture property of the page is also set.

- **Page initialization**

  At this stage, the controls on the page are assigned unique ID by setting the UniqueID property and themes are applied. For a new request postback data is loaded and the control properties are restored to the view-state values.

- **Page load**

  At this stage, control properties are set using the view state and control state values.

- **Validation**

  Validate method of the validation control is called and if it runs successfully, the IsValid property of the page is set to true.

- **Postback event handling**

If the request is a postback (old request), the related event handler is called.

- **Page rendering**

  At this stage, view state for the page and all controls are saved. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Page's Response property.

- **Unload**

  The rendered page is sent to the client and page properties, such as Response and Request are unloaded and all cleanup done.

ASP.NET Page Life Cycle Events

Following are the page life cycle events:

- **PreInit**

  PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls and gets and sets profile property values. This event can be handled by overloading the OnPreInit method or creating a Page_PreInit handler.

- **Init**

  Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page_Init handler.

- **InitComplete**

  InitComplete event allows tracking of view state. All the controls turn on view-state tracking.

- **LoadViewState**

  LoadViewState event allows loading view state information into the controls.

- **LoadPostData**

  During this phase, the contents of all the input fields defined with the <form> tag are processed.

- **PreLoad**

  PreLoad occurs before the post back data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page_PreLoad handler.

- **Load**

  The Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page_Load handler.

- **LoadComplete**

  The loading process is completed, control event handlers are run and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page_LoadComplete handler.

- **PreRender**

  The PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.

- **PreRenderComplete**

  as the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.

- **SaveStateComplete**

  State of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page_Render handler.

- **UnLoad**

  The UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed. This event can be handled by modifying the OnUnLoad method or creating a Page_UnLoad handler.

Question 32: What is the Difference between session and caching?

Answer: The first main difference between session and caching is: a session is per-user based but caching is not per-user based, So what does that mean? Session data is stored at the user level but caching data is stored at the application level and shared by all the users. It means that it is simply session data that will be different for the various users for all the various users, session memory will be allocated differently on the server but for the caching only one memory will be allocated on the server and if one user modifies the data of the cache for all, the user data will be modified.

For further info click on the link:

- Difference Between Session and Caching

Question 33: What is the difference between HttpContext.Current.Items and HttpContext.Current.Session in ASP.NET?

Answer: Session state is one of the popular state management techniques in ASP.NET environment. We developer people play with session storage every now and then. It’s pretty simple to manage session if you understand the basic concept. Here is the syntax to do that.

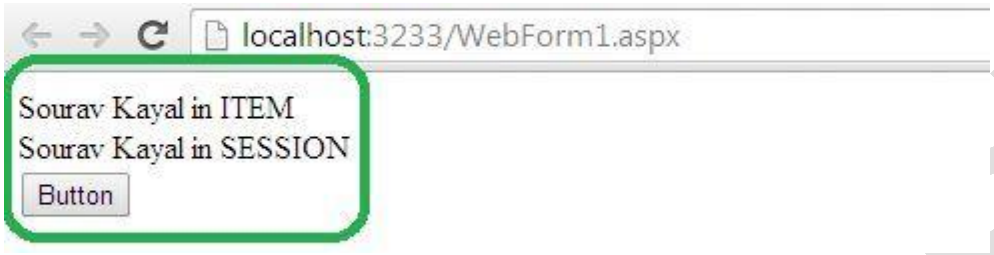
```
Session[“KEY”] =”Value”;
```

Or

```
Session[index] = ”Value”;
Let’ s a have an example: using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebApp
{
    public partial class WebForm1: System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if(!IsPostBack)
            {
                HttpContext.Current.Items["Value"] = "Sourav Kayal in ITEM";
                HttpContext.Current.Session["Value"] = "Sourav Kayal in SESSION";
                Response.Write((string)(HttpContext.Current.Items["Value"]) + "<br>");
                Response.Write((string)(HttpContext.Current.Session["Value"]));

            }
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
```

```
        Response.Write((string)(HttpContext.Current.Items["Value"]) + "<b
r>");
        Response.Write((string)(HttpContext.Current.Session["Value"]));
    }
  }
}
```



localhost:3233/WebForm1.aspx

Sourav Kayal in ITEM
Sourav Kayal in SESSION

For further info click on the link:

- Difference Between HttpContext.Current.Items and HttpContext.Current.Session in ASP.Net

Question 34: What is the difference between Server.Transfer and Response.redirect?

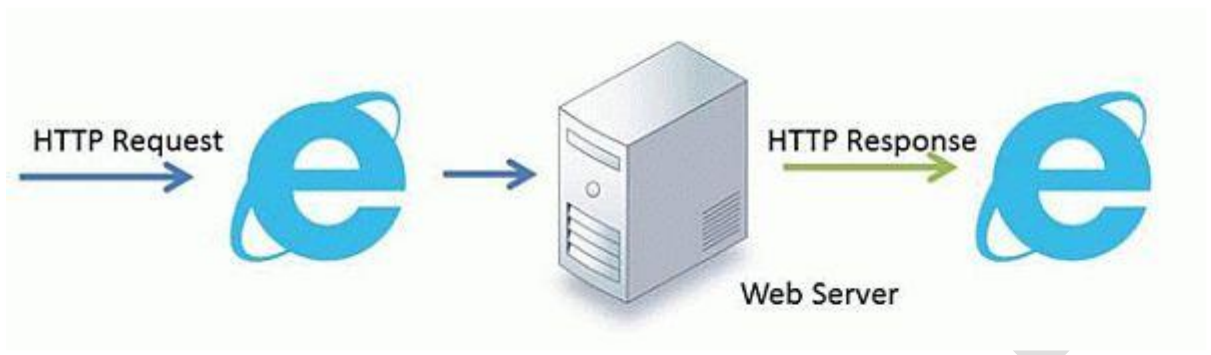
Answer: Both Response.Redirect and Server.Transfer methods are used to transfer a user from one web page to another web page. Both methods are used for the same purpose but still there are some differences as follows.

The Response.Redirect method redirects a request to a new URL and specifies the new URL while the Server.Transfer method for the current request, terminates execution of the current page and starts execution of a new page using the specified URL path of the page.

Both Response.Redirect and Server.Transfer has same syntax like:

```
Response.Redirect("UserDetail.aspx");
Server.Transfer("UserDetail.aspx");
```

Before touching on more points I want to explain some HTTP status codes, these are important for the understanding of the basic differences between these two. The HTTP status codes are the codes that the Web server uses to communicate with the Web browser or user agent.



For further info click on the link:

Question 36: What is HTTP Handler?

Answer: Every request into an ASP.NET application is handled by a specialized component known as an HTTP handler. The HTTP handler is the most important ingredient while handling ASP.NET requests.

Examples: ASP.NET uses different HTTP handlers to serve different file types. For example, the handler for web Page creates the page and control objects, runs your code, and renders the final HTML.

ASP.NET default handlers:

1. Page Handler (.aspx) - Handles Web pages.
2. User Control Handler (.ascx) - Handles Web user control pages.
3. Web Service Handler (.asmx) - Handles Web service pages.
4. Trace Handler (trace.axd) - Handles trace functionality.

Why we need to create our own HTTP Handler: Sometime we need to avoid ASP.NET full page processing model, which saves lot of overheads, as ASP.NET web form model has to go through many steps such as creating web page objects, persisting view state etc. What we are interested into is to develop some low level interface that provides access to objects like Request and Response but doesn't use the full control based web form model discussed above.

Examples:

1. Dynamic image creator - Use the System.Drawing classes to draw and size your own images.
2. RSS - Create a handler that responds with RSS-formatted XML. This would allow you to add RSS feed capabilities to your sites.
3. Render a custom image,
4. Perform an ad hoc database query,
5. Return some binary data.

All HTTP handlers are defined in the <httpHandlers> section of a configuration file which is nested in the <system.web> element.

```
<httpHandlers>
    <add verb="*" path="trace.axd" validate="true" type="System.Web.Handlers.TraceHandler" />
    <add verb="*" path="*.config" validate="true" type="System.Web.HttpForbiddenHandler" />
    <add verb="*" path="*.cs" validate="true" type="System.Web.HttpForbiddenHandler" />
    <add verb="*" path="*.aspx" validate="true" type="System.Web.UI.PageHandlerFactory" /> </httpHandlers>
```

For further info click on the link:

- Create your first HTTP Handler in ASP.NET 3.5

Question 37: What are Differences between ASP.NET HttpHandler and HttpModule?

Answer: The user requests for a resource on web server. The web server examines the file name extension of the requested file, and determines which ISAPI extension should handle the request. Then the request is passed to the appropriate ISAPI extension. For example when an .aspx page is requested it is passed to ASP.NET page handler. Then Application domain is created and after that different ASP.NET objects like Httpcontext, HttpRequest, HttpResponse are created. Then instance of HttpApplication is created and also instance of any configured modules. One can register different events of HttpApplication class like BeginRequest, AuthenticateRequest, AuthorizeRequest, ProcessRequest etc.

HTTP Handler

HTTP Handler is the process which runs in response to a HTTP request. So whenever user requests a file it is processed by the handler based on the extension. So, custom http handlers are created when you need to special handling based on the file name extension. Let's consider an example to create RSS for a site. So, create a handler that generates RSS-formatted XML. Now bind the .rss extension to the custom handler.

HTTP Modules

HTTP Modules are plugged into the life cycle of a request. So when a request is processed it is passed through all the modules in the pipeline of the request. So generally http modules are used for:

- **Security:** For authenticating a request before the request is handled.
- **Statistics and Logging:** Since modules are called for every request they can be used for gathering statistics and for logging information.

- **Custom header:** Since response can be modified, one can add custom header information to the response.

Question 42: What is the PostBack property in ASP.NET?

Answer: If we create a web Page, which consists of one or more Web Controls that are configured to use AutoPostBack (Every Web controls will have their own AutoPostBack property), the ASP.NET adds a special JavaScipt function to the rendered HTML Page. This function is named _doPostBack() . When Called, it triggers a PostBack, sending data back to the web Server.

ASP.NET also adds two additional hidden input fields that are used to pass information back to the server. This information consists of ID of the Control that raised the event and any additional information if needed. These fields will empty initially as shown below,

```
<input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />
```

The following actions will be taken place when a user changes a control that has the AutoPostBack property set to true:

1. On the client side, the JavaScript _doPostBack function is invoked, and the page is resubmitted to the server.
2. ASP.NET re-creates the Page object using the .aspx file.
3. ASP.NET retrieves state information from the hidden view state field and updates the controls accordingly.
4. The Page.Load event is fired.
5. The appropriate change event is fired for the control. (If more than one control has been changed, the order of change events is undetermined.)
6. The Page.PreRender event fires, and the page is rendered (transformed from a set of objects to an HTML page).
7. Finally, the Page.Unload event is fired.
8. The new page is sent to the client.

Cookies is a small piece of information stored on the client machine. This file is located on client machines "C:\Document and Settings\Currently_Login user\Cookie" path. Its is used to store user preference information like Username, Password,City and PhoneNo etc on client machines. We need to import namespace called Systen.Web.HttpCookie before we use cookie.

Type of Cookies?

1. Persist Cookie - A cookie has not have expired time Which is called as Persist Cookie
2. Non-Persist Cookie - A cookie has expired time Which is called as Non-Persist Cookie

How to create a cookie?

Its really easy to create a cookie in the Asp.Net with help of Response object or HttpCookie

Example 1:

```
HttpCookie userInfo = new HttpCookie("userInfo");
userInfo["UserName"] = "Annathurai";
userInfo["UserColor"] = "Black";
userInfo.Expires.Add(new TimeSpan(0, 1, 0));
Response.Cookies.Add(userInfo);
```

Example 2:

```
Response.Cookies["userName"].Value = "Annathurai";
Response.Cookies["userColor"].Value = "Black";
```

How to retrieve from cookie?

Its easy way to retrieve cookie value form cookes by help of Request object.

Example 1:

```
string User_Name = string.Empty;
string User_Color = string.Empty;
User_Name = Request.Cookies["userName"].Value;
User_Color = Request.Cookies["userColor"].Value;
```

Example 2:

```
string User_name = string.Empty;
string User_color = string.Empty;
HttpCookie reqCookies = Request.Cookies["userInfo"];
if (reqCookies != null)
{
   User_name = reqCookies["UserName"].ToString();
   User_color = reqCookies["UserColor"].ToString();
}
```

When we make request from client to web server , the web server process the request and give the lot of information with big pockets which will have Header information, Metadata, cookies etc., Then repose object can do all the things with browser.

Cookie's common property:

1. Domain => Which is used to associate cookies to domain.
2. Secure => We can enable secure cookie to set true(HTTPs).
3. Value => We can manipulate individual cookie.
4. Values => We can manipulate cookies with key/value pair.
5. Expires => Which is used to set expire date for the cookies.

Advantages of Cookie:

1. Its clear text so user can able to read it.
2. We can store user preference information on the client machine.
3. Its easy way to maintain.
4. Fast accessing.

Disadvantages of Cookie

1. If user clear cookie information we can't get it back.
2. No security.
3. Each request will have cookie information with page.

How to clear the cookie information?

1. we can clear cookie information from client machine on cookie folder
2. To set expires to cookie object
   userInfo.Expires = DateTime.Now.AddHours(1);
   It will clear the cookie with one hour duration.

3. In which event of page cycle is the ViewState available?

After the Init() and before the Page_Load().

5. From which base class all Web Forms are inherited?

Page class.

8. What is ViewState?

ViewState is used to retain the state of server-side objects between page post backs.

9. Where the viewstate is stored after the page postback?

ViewState is stored in a hidden field on the page at client side. ViewState is transported to the client and back to the server, and is not stored on the server or any other external source.

10. How long the items in ViewState exists?

They exist for the life of the current page.

OOPS

Four Pillars of Object Oriented Programming (OOP)

There are four Pillars of Object Oriented Programming:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Variables declared with **var** are implicitly but statically typed. Variables declared with**dynamic** are **dynamically** typed. ... When using the '**var**' **keyword**, the type is decided by the compiler at compile time, whereas when using the '**dynamic**'**keyword**, the type is decided by the runtime.

Nvarchar stores UNICODE data. If you have requirements to store UNICODE or multilingual data, **nvarchar is** the choice. **Varchar**stores ASCII data and should be your data type of choice for normal use. Regarding memory usage, **nvarchar** uses 2 bytes per character, whereas **varchar** uses 1

An **Array list** is not a strongly-typed collection. It can store the values of different data types or same datatype. The size of an **array list** increases or decreases dynamically so it can take any size of values from any data type. **ArrayList** is one **of the**most flexible data structures from **C#** Collections.

Difference between a **Value Type** and a **Reference Type**. ... A **Value Type** holds the data within its own memory allocation and a **Reference Type** contains a pointer to another memory location that holds the real data. **Reference Type** variables are stored **in the**heap while **Value Type** variables are stored **in the** stack

Stack is used for static memory allocation and **Heap**for dynamic memory allocation, both stored **in the**computer's RAM . ... Variables allocated on the **heap**have their memory allocated at run time and accessing this memory is a bit slower, but the **heap**size is only limited by the size of virtual memory .

The **stack** is **faster** because the access pattern makes it trivial to allocate and deallocate memory from it (a pointer/integer is simply incremented or decremented), while the **heap** has much more complex bookkeeping involved in an allocation or free.

Difference between Classes and Structures. ... **Class** can create a subclass that will inherit parent's properties and methods, whereas **Structure** does not support the inheritance.
A **class** has all members private by default. A**struct** is a **class** where members are public by default.

dispose is an object method invoked to execute code required for memory cleanup and release and reset unmanaged resources, such as file handles and database connections. ...
The **Dispose** method, provided by the IDisposable interface, implements **Dispose** calls.

BASIS FOR COMPARISON	**DISPOSE( )**	**FINALIZE( )**
Defined	The method dispose( ) is defined in the interface IDisposable interface.	The method finalize( ) id defined in java.lang.object class.

Syntax	public void Dispose( ){
	// Dispose code here
	}
		protected void finalize( ){
		// finalization code here
		}

Invoked	The method dispose( ) is invoked by the user.	The method finalize( ) is invoked by the garbage

BASIS FOR COMPARISON	DISPOSE( )	FINALIZE( )
		collector.
Purpose	Method dispose( ) is used to free unmanaged resources whenever it is invoked.	Method finalize( ) is used to free unmanaged resources before the object is destroyed.
Implementation	The method dispose( ) is to be implemented whenever there is a close( ) method.	The method finalize( ) is to be implemented for unmanaged resources.
Access specifier	The method dispose( ) is declared as public.	The method finalize( ) is declared as private.
Action	The method dispose( ) is faster and instantly disposes an object.	The method finalize is slower as compared to dispose
Performance	The method disposes( ) performs the instantaneous	The method finalize( ) being slower affects the

BASIS FOR COMPARISON	DISPOSE( )	FINALIZE( )
	action hence, does not effect the performance of websites.	performance of the websites.

Garbage Collector to call Manually

```
static void Main(string[] args)
        {
            //Fibooniee();
            //Reverestring();
            //ReverseInteger();
            //GetData();


            Program p = new ConsoleApplication1.Program();

            p.SetValues("PRamit");
            p.SetValues("PRamit", "abbd");

            GC.Collect();
            GC.WaitForPendingFinalizers();

        }
```

There might be times in your application when you want to force the .NET Garbage Collector (GC) to spin through all unused objects and de-allocate them. The method for accomplishing this task is the **GC.Collect** method. When you call **GC.Collect**, the GC will run each object's finalizer on a separate thread. Therefore, another method to keep in mind is **GC.WaitForPendingFinalizers**. This synchronous method that will not return until the **GC.Collect** has finished its work.

what is garbage collection in c#

When you create any object in C#, CLR (*common language runtime*) allocates memory for the object from heap. This process is repeated for each newly created object, but there is a limitation to everything, Memory is not un-limited and we need to clean some used space in order to make room for new objects, Here, the concept of *garbage collection* is introduced, *Garbage collector* manages allocation and reclaiming of memory. GC (Garbage collector) makes a trip to the heap and collects all objects that are no longer used by the application and then makes them free from memory.

How GC Works?

GC works on *managed heap*, which is nothing but a block of memory to store objects, when garbage collection process is put in motion, it checks for dead objects and the objects which are no longer used, then it compacts the space of live object and tries to free more memory.

Basically, heap is managed by different '*Generations*', it stores and handles long-lived and short-lived objects, see the below generations of Heap:

- **0 Generation (Zero)**: This generation holds short-lived objects, e.g., Temporary objects. GC initiates garbage collection process frequently in this generation.
- **1 Generation (One)**: This generation is the buffer between short-lived and long-lived objects.
- **2 Generation (Two)**: This generation holds long-lived objects like a static and global variable, that needs to be persisted for a certain amount of time. Objects which are not collected in generation Zero, are then moved to generation 1, such objects are known as *survivors*, similarly objects which are not collected in generation One, are then moved to generation 2 and from there onwards objects remain in the same generation.

What is constructor?

A special method of the class that will be automatically invoked when an instance of the class is created is called a constructor.

Some of the key points regarding the Constructor are:

- A class can have any number of constructors.
- A constructor doesn't have any return type, not even void.
- A static constructor can not be a parametrized constructor.
- Within a class you can create only one static constructor.

Constructors can be divided into 5 types:

1. Default Constructor
2. Parametrized Constructor
3. Copy Constructor
4. Static Constructor
5. Private Constructor

36. What is an Object Pool in .Net?

Answer:

Object Pooling is something that tries to keep a pool of objects in memory to be re-used later and hence it will reduce the load of object creation to a great extent. This article will try to explain this in detail. The example is for an Employee object, but you can make it general by using Object base class.

What does it mean?

Object Pool is nothing but a container of objects that are ready for use. Whenever there is a request for a new object, the pool manager will take the request and it will be served by allocating an object from the pool.

How it works?

We are going to use Factory pattern for this purpose. We will have a factory method, which will take care about the creation of objects. Whenever there is a request for a new object, the factory method will look into the object pool (we use Queue object). If there is any object available within the allowed limit, it will return the object (value object), otherwise a new object will be created and give you back.

CLASS

Internal is the **default** if no **access modifier** is specified

```
@media screen and (min-width: 480px) {
    body {
        background-color: lightgreen;
    }
}

@media screen and (max-width: 992px) {
  body {
    background-color: blue;
  }
}

/* On screens that are 600px or less, set the background color to olive */
@media screen and (max-width: 600px) {
  body {
    background-color: olive;
  }
}.

Bootstrap 4.1.2
Jquery 3.3.1
```

n previous ASP.NET MVC Tutorial, we discussed about different available options for passing data from Controller to View in ASP.NET MVC. We implemented passing data using `ViewBag` and `ViewData` in ASP.NET MVC application. Now, in this part of the tutorial, we are going to discuss about the third available option i.e. `TempData`.

`TempData` in ASP.NET MVC is basically a dictionary object derived from `TempDataDictionary`. `TempData` stays for a subsequent HTTP Request as opposed to other options (`ViewBag` and `ViewData`) those stay only for current request. So, `TempdData` can be used to maintain data between controller actions as well as redirects.

Note: Just like `ViewData`, typecasting and null checks required for `TempData` also in order to avoid errors.

Let's see how we can use `TempData` in a practical scenario to pass data from one controller action to another.

Hide   Copy Code

```
 //Controller Action 1 (TemporaryEmployee)
 public ActionResult TemporaryEmployee()
{
```

```
                Employee employee = new Employee
                {
                        EmpID = "121",
                        EmpFirstName = "Imran",
                        EmpLastName = "Ghani"
                };
                TempData["Employee"] = employee;
                return RedirectToAction("PermanentEmployee");
}

 //Controller Action 2(PermanentEmployee)
 public ActionResult PermanentEmployee()
{
               Employee employee = TempData["Employee"] as Employee;
               return View(employee);
 }
```

As in above example, we store an employee object in `TempData` in Controller Action 1 (i.e. `TemporaryEmployee`) and retrieve it in another Controller Action 2 (i.e. PermanentEmployee). But If we try to do the same using `ViewBag` or `ViewData`, we will get null in Controller Action 2 because only `TempData` object maintains data between controller actions.

An important thing about `TempData` is that it stores contents in Session object. Then one may raise a question that "*What's the difference between TempData in ASP.NET MVC and Session?*" or "*Why we have TempData dictionary object?, Why can't we use the Session object directly?*"

ASP.NET MVC TempData Vs Sessions

Although ASP.NET MVC TempData stores it's content in Session state but it gets destroyed earlier than a session object. TempData gets destroyed immediately after it's used in subsequent HTTP request, so no explicit action required. If we use a Session object for this purpose, we would have to destroy the object explicitly.

It's best fit for scenarios when:

- we need to preserve data from current to subsequent request.
- passing error message to an error page.

With the above discussion on using TempData in ASP.NET MVC, we have successfully covered different options for passing data from Controller to View in ASP.NET MVC. Hopefully reader will have a better understanding of using ViewBag, ViewData and TempData in ASP.NET MVC.

Previous: ViewBag and ViewData in ASP.NET MVC