# What are similarities between Class and structure?

* Both can have constructors, methods, properties, fields, constants, enumerations, events, and event handlers.
* Structures and classes can implement interface.
* Both of them can have constructors with and without parameter.
* Both can have delegates and events.

# What is the difference between Class and structure’s?

* Structures are value types and classes are reference types. So structures use stack and classes use heap.
* Structures members cannot be declared as protected, but class members can be. You cannot do inheritance in structures.
* Structures do not require constructors while classes require.
* Objects created from classes are terminated using Garbage collector. Structures are not destroyed using GC.

# What is “Sealed” modifier?

* Sealed modifer is used on class to pervent it from inherited from other classes
* Sealed modifer on method prevent the method from override from inherited classes.

# List out the differences between Array and ArrayList in C#?

* Arrays keeps data of same type whereas ArrayList can have data of various data types
* Arrays are of fixed length whereas Arraylist grows as values are added

# Why use “using” keyword?

* “using” keyword is used to invoke namespaces, wherein we can use the classes from various namespaces without using it’s fully qualified name
* “using” is used as code-block on class that implements IDisposable interface
* “using” is used as code-block wherein it calls the “dispose” method internally on objects used in using code block. However, if exception occurs on the using statement then it doesn’t get caught.

Ex: using( ){};

# Explain Namespaces in C#?

* Namespaces are containers to group related classes.

# Difference between Const, readonly and static keywords?

* Const: variable value has to be assigned at declaration. And it’s value cannot be changed later in the code. By default, const is static as it’s value cannot be changed once assigned. Constants can be accessed by using class name directy without creating instances because by default it is static, it cannot be accessed by using the instance of the class but inside the call without class name. Ex: const int fixedpoint = 21;
* Readonly: A ReadOnly variable is similar to constant but we can assign the value either at the time of declaration or in the constructor. values can be changed at runtime via (only constructors). Also, readonly variable can be re-assigned with new value at runtime (via parameterised constructor).
* Static readonly : variable value can be assigned at variable or static constructor. It’s value can be changed only once (via static constructor)
* Static : is a class level variable. Hence, only one copy is shared across all instances.

# Difference between Interface and Class

* Interface contains only member declaration (method,properties,indexes and Events), whereas Abstract class can have abstract and concrete methods.
* All members of Interface are public by default. Access modifiers are not allowed in Interface, whereas Abstract class can have various access modfiers (public,private,protected,protected internal)
* Interface facilitates mutiple inhertences. A class can implement multiple interfaces. Whereas Abstract class do not support multiple inheritence. A class can inherit fron only one base class
* Abstract class cannot be instantiated. Interface can.
* Interface cannot have fields and constructors, whereas Abstract class can have fields and constructors
* Abstract class should be used in situations wherein various related class have common members, the common members can be placed in an abstract class which then can be inherited by the classes. Abstract class also facititate to provide partial implementations for reuseability and productivity. Interfaces should be used in situations wherein various unrelated objects are required to implement certain functionalities. Inferface can also be used in situation when you cannot use class inheritance. For example, structures cannot inherit from classes, but they can implement interfaces.

# What is Enum?

* Enum is a set of named constants. It’s default type is int and starts with “0” for the element constant. Enumerations (enums) make your code much more readable and understandable.

# Difference between ref and out keyword?

* Ref parameter is used when a value type has to be passed as reference type in the callin function.
* Ref parameter value can be changed from within the calling method
* Ref parameter has to be initialized before passing into the calling method.
* Out parameter is used when multiple values has to be returned from the callling method.
* Out parameter has to be initialized/ assigned from within the calling method.

# What is property?

* Property are public data member to access private fields of a class.
* There can be 3 variasions : read/write, read, write.

# Difference between dispose and finalize?

* Finalize method is used to re-claim the memory from unmanaged resources (such as database connection etc). It cannot be called from user code rather it is called by Gargbage Collector. Finalize method cannot be implemented directly it can only be implemented via destructors. After compilation destuctors becomes finalize method.
* Dispose method is also used to re-claim the memory from unmanaged resources, however it can be explicity implement and invoked from user code. Idisposable interface should be implemented to use dispose method in user-defined class.

# Difference between String and StringBuilder?

* String: is immutable object means it takes new memory whenever a new value is assigned/append. It belongs to System namespace.
* StringBuilder: is mutable objects. It belongs to System.Text

# Partial Classes

* A class defination can split in multiple files with the same namespace. Partial keyword should be used to split class defination, and it is complied as single class.

# Difference between IEnumerable and IQueryable ?

* IEnumerable exists in System.Collection Namespace. VS . Iqueryable exists in System.Linq namespace
* IEnumerable suits to query in-memory data (such as collections). Vs IQueryable is used to query out-of-memory data (like database).
* IEnumerable do not supports Lazy Loading. VS. IQueryable supports Lazy support. Hence suitable to Paging like senario.
* Extention methods supports by IEnumerable takes functional objects. VS. Extension methods supports by IQueryable takes expression objects means expression tree.

# Difference between IEnumerable and IList ?

1. IEnumerable exists in System.Collections Namespace.
2. IEnumerable can move forward only over a collection, it can’t move backward and between the items.
3. IEnumerable is best to query data from in-memory collections like List, Array etc.
4. IEnumerable doesn't support add or remove items from the list.
5. Using IEnumerable we can find out the no of elements in the collection after iterating the collection.
6. IEnumerable supports deferred execution.
7. IEnumerable supports further filtering.
8. IList exists in System.Collections Namespace.
9. IList is used to access an element in a specific position/index in a list.
10. Like IEnumerable, IList is also best to query data from in-memory collections like List, Array etc.
11. IList is useful when you want to Add or remove items from the list.
12. IList can find out the no of elements in the collection without iterating the collection.
13. IList supports deferred execution.
14. IList doesn't support further filtering.

# What is connection pooling and how do you make your application use it?

Opening database connection is a time consuming operation.  
Connection pooling increases the performance of the applications by reusing the active database connections instead of create new connection for every request.  
Connection pooling Behaviour is controlled by the connection string parameters.  
Follwing the the 4 parameters that control most of the connection pooling behaviour.  
1. Connect Timeout  
2. Max Pool Size  
3. Min Pool Size  
4. Pooling

# What are different methods of session maintenance in**ASP.NET**?

In-process storage.  
Session State Service. (ASP.NET State Service)  
Microsoft SQL Server.

**In-Process Storage**The default location for session state storage is in the ASP.NET process itself

**Session State Service**  
As an alternative to using in-process storage for session state, ASP.NET provides the ASP.NET State Service. The State Service gives you an out-of-process alternative for storing session state that is not tied quite so closely to ASP. Net’s own process.

Advantage:   
Session state can be shared in web garden (multiple processes on same machince.) and web farm (multiple servers running the application).

Application crash do not impact the session data.

Disadvantage:

It’s slow to access session data from different state process (session state service)

**Microsoft SQL Server**The final choice for storing state information is to save it in a Microsoft SQL Server database. To use SQL Server for storing session state, you need to perform several setup steps

# What are HTTP handlers in ASP.NET?

An ASP.NET HTTP handler is the process (frequently referred to as the "endpoint") that runs in response to a request made to an ASP.NET Web application. The most common handler is an ASP.NET page handler that processes .aspx files. When users request an .aspx file, the request is processed by the page through the page handler. You can create your own HTTP handlers that render custom output to the browser.

An HTTP module is an assembly that is called on every request that is made to your application. HTTP modules are called as part of the ASP.NET request pipeline and have access to life-cycle events throughout the request. HTTP modules let you examine incoming and outgoing requests and take action based on the request.

You can define HTTP handlers in the **<httpHandlers>** element of a configuration file. The **<add>** element tag is used to add new handlers and the **<remove>**element tag is used to remove existing handlers. To create an HTTP handler, you need to define a class that implements the **IHttpHandler** interface. The interfaces require that you implement the [**IsReusable**](https://msdn.microsoft.com/en-us/library/system.web.ihttphandler.isreusable.aspx)property and the [**ProcessRequest**](https://msdn.microsoft.com/en-us/library/system.web.ihttphandler.processrequest.aspx) method. The [**IsReusable**](https://msdn.microsoft.com/en-us/library/system.web.ihttphandler.isreusable.aspx) property specifies whether the [**IHttpHandlerFactory**](https://msdn.microsoft.com/en-us/library/system.web.ihttphandlerfactory.aspx)object (the object that actually calls the appropriate handler) can put the handler in a pool and reuse it to increase performance. If the handler cannot be pooled, the factory must create a new instance of the handler every time that the handler is needed.

The [ProcessRequest](https://msdn.microsoft.com/en-us/library/system.web.ihttphandler.processrequest.aspx) method is responsible for processing individual HTTP requests. In this method, you write the code that produces the output for the handler.

# What is the lifespan for items stored in ViewState?

Viewstate keeps page state data in an hidden field in Browser. It expires with the page expiration or postback event on the page.

# How information about the user's locale can be accessed?

System.Web.UI.Page.Culture

# Parent class of web server controls?

System.Web.UI.Control

# How do you sign out from forms authentication?

The FormsAuthentication.Signout() method is used to sign out from the forms authentication.

# Why do you use theApp\_Codefolder in ASP.NET?

In a **Web site project**, you can store source code in the App\_Code folder, and it will be automatically compiled at run time. The resulting assembly is accessible to any other code in the Web application. The App\_Code folder therefore works much like the Bin folder, except that you can store source code in it instead of compiled code. The App\_Code folder and its special status in an ASP.NET Web application makes it possible to create custom classes and other source-code-only files and use them in your Web application without having to compile them independently.

# What is ASP.NET Skin? Difference between default skin and Named skin?

ASP.Net skins can only be used to apply the styles to  the ASP.Net controls. **.**skin file created in App\_themes folder contains style defination for controls.

1. A control Id cannot be assigned to ASP.Net controls inside the SkinFile.skin.
2. SkinId must be assigned to the ASP.Net controls inside the SkinFile.skin.
3. The SkinId should be uniquely defined because duplicate SkinId's per control type are not allowed in the same theme.
4. Only one default control skin per control type is allowed in the same theme.

The default skin is applied to all the Web server controls in a Web form, which are of similar type, and it does not provide a Skin ID attribute. The named skin provides a Skin ID attribute and users have to set the Skin ID property to apply it.

Named skin:: -  
<asp:Label runat="server" ForeColor="#585880" Font-Size="0.9em" Font-Names="Verdana" SkinID="LabelHeader" />

<asp:Label runat="server" ForeColor="#585980" Font-Size="0.8em" Font-Names="Arial" SkinID="LabelFooter" />

Apply skin:: -   
<asp:Label id="Header" runat="server" SkinID="LabelHeader" />  
<asp:Label id="Header" runat="server" SkinID="LabelFooter" />

# What are the validation controls in ASP.NET 2.0?

* RequiredFieldValidator.
* CompareValidator.
* RangeValidator.
* RegularExpressionValidator.
* CustomValidator.
* ValidationSummary.

# How can you display all validation messages in one control?

The ValidationSummary control displays all validation messages in one control.

# Which two new properties are added in ASP.NET 4.0 Page class?

The two new properties added in the Page class are MetaKeyword and MetaDescription.

# What is tracing? Where is it used?

ASP.NET tracing enables you to view diagnostic information about a request for an ASP.NET page. ASP.NET tracing enables you to follow a page's execution path, display diagnostic information at run time, and debug your application. ASP.NET tracing can be integrated with system-level tracing to provide multiple levels of tracing output in distributed and multi-tier applications.

Tracing appends diagnostic information and custom tracing messages to the output of the page and sends this information to the requesting browser. Optionally, you can view this information from a separate trace viewer (Trace.axd) that displays trace information for every page in an ASP.NET Web application. Tracing information can help you investigate errors or unwanted results while ASP.NET processes a page request.

You can configure individual pages to display trace information. Alternatively, you can configure the application's Web.config file so that all pages display trace information unless the page explicitly disables tracing. Setting application-level tracing is useful because you do not have to change individual pages to enable and disable it.

The System.Diagnostics namespace contains the predefined interfaces, classes, and structures that are used for tracing. It supplies two classes, Trace and Debug, which allow you to write errors and logs related to the application execution

# How can you register a custom server control to a Web page?

Using @Register directive

# Which method is used to force all the validation controls to run?

Page.Validate()

# Which method has been introduced in ASP.NET 4.0 to redirect a page permanently?

RedirectPermanent() method added in ASP.NET 4.0 to redirect a page permanently. This method performs same redirect as Response.Redirect() but it uses response code 301. Response.Redirect() returns 302 to browser meaning that asked resource is temporarily moved to other location.  
Use RedirectPermanent if the resource has been moved permanently and will no longer be accessible in its previous location. Most browsers will cache this response and perform the redirect automatically without requesting the original resource again.

# How can you send an email message from an ASP.NET Web page?

You can use the System.Net.Mail.MailMessage and the System.Net.Mail.SmtpMail classes to send an email in your Web pages. In order to send an email through your mail server, you need to create an object of the SmtpClient class and set the server name, port, and credentials

# What is the difference between theResponse.Write()andResponse.Output.Write()methods?

Response.Output.Write() gives you String.Format-style formatted output and the Response.Write() doesn't.   
FOR Example  
Response.Write("Current Date Time is "+DateTime.Now.ToString());   
Response.Output.Write("{0} is {1:d}", "Current Date Time is: ",DateTime.Now);

# What is the default timeout for a Cookie?

The default time duration for a Cookie is 30 minutes

# What is Caching?

Caching is a technique where we can store frequently used data, and web pages are stored temporarily on the local hard disk for later retrieval. This technique improves the access time when multiple users access a web site simultaneously, or a single user accesses a web site multiple times. Caching for web applications can occur on the client (browser caching), on a server between the client and the web server, (proxy caching / reverse proxy caching), and on the web server itself (page caching or data caching).

# Different Caching Locations

1. Client Caching
2. Proxy Caching
3. Reverse Proxy Caching
4. Web Server Caching

Client Caching: In Client Caching, the client browser performs caching by storing cached data on the local disk as a temporary file or in the browser internal memory

Proxy Caching: is a decidated server between client browser and webserver.  The proxy server (e.g., Microsoft Proxy Server) fulfills all the requests for the web page without sending out the request to the actual web server over the internet, resulting in faster access

Reverse Proxy Server: Some proxy cache servers can be placed in front of the web server to reduce the number of requests that they receive. This allows the proxy server to respond to frequently received requests and only pass other requests to the web server. This is called a reverse proxy

Web Server Caching: In web server caching, cached data is stored inside the web server. ASP.NET Data caching and page caching uses the web sever caching mechanism.

# Different Types of Caching?

1. Page Output Caching : directive can be added to any ASP.NET page, specifying the duration (in seconds) that the page should be cached.

<%@ OutputCache Duration='300' VaryByParam='none' %>

We can also set the caching property from the code-behind:

void Page\_Load(Object sender, EventArgs e) {

Response.Cache.SetExpires(DateTime.Now.AddSeconds(360));

Response.Cache.SetCacheability(

HttpCacheability.Public);

Response.Cache.SetSlidingExpiration(true);

\_msg.Text = DateTime.Now.ToString();

}

We have to mention the Duration and VaryByParam attribute.   
Duration defines how long the cache will persist.  
VaryByParam defines if there the cache varies with parameter value

The following table shows you the most commonly used and most important attributes of output cache:

| Attribute | Values | Description |
| --- | --- | --- |
| Duration | Number | Defines how long the page will be cached (in seconds). |
| Location | 'Any'  'Client'  'Downstream'  'Server'  'None' | It defines the page cache location. I have discussed it later in detail. |
| VaryByCustom | 'Browser' | Vary the output cache either by browser name and version or by a custom string. |
| VaryByParam | 'none' '\*' | This is a required attribute, which is required for a parameter for the page. I have already discussed this. |

1. Page Fragement Caching : ASP.NET provides a mechanism for caching portions of pages, called page fragment caching. To cache a portion of a page, you must first encapsulate the portion of the page you want to cache into a user control. In the user control source file, add an OutputCache directive specifying the Duration and VaryByParam attributes. When that user control is loaded into a page at runtime, it is cached, and all subsequent pages that reference that same user control will retrieve it from the cache.
2. Data Caching : Caching data can dramatically improve the performance of an application by reducing database contention and round-trips. Simply, data caching stores the required data in cache so that the web server will not send requests to the DB server every time for each and every request, which increases web site performance. The data cache is a full-featured cache engine that enables you to store and retrieve data between multiple HTTP requests and multiple sessions within the same application.

There are three different ways to add data or objects into cache. But based on the situation, we have to access it differently. These methods are Cache[], Cache.add(), cache.insert(). The following table will show you the clear difference of the there methods.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Stores data in cache | Supports dependency | Supports expiration | Support priority settings | Returns object |
| cache[] | Yes | No | No | No | No |
| cache.insert() | Yes | Yes | Yes | Yes | No |
| cache.add() | Yes | Yes | Yes | Yes | Yes |

cache[] is a property that is very simple to use but cache.insert() and cache.add() give us more control on the cached data.

Now we should look into the details of the Cache.Insert() and Cache.Add() methods. Cache.Insert()has four overloads whereas Cache.Add() has no overloaded methods. The following table shows the most commonly used properties for those methods.

| Property | Type | Description |
| --- | --- | --- |
| Key | String | A unique key used to identify this entry in the cache. |
| Dependency | CacheDependency | A dependency this cache entry has—either on a file, a directory, or another cache entry—that, when changed, should cause this entry to be flushed. |
| Expires | DateTime | A fixed date and time after which this cache entry should be flushed. |
| Sliding Expiration | TimeSpan | The time between when the object was last accessed and when the object should be flushed from the cache. |
| Priority | CacheItemPriority | How important this item is to keep in the cache compared with other cache entries (used when deciding how to remove cache objects during scavenging). |
| OnRemoveCallback | CacheItem RemovedCallback | A delegate that can be registered with a cache entry for invocation upon removal. |

#### **Cache Dependency**

Using cache dependency, we can set the dependency of the cache with some data or entity that might change. So we can set the dependency of cache by which we can update/remove cache. There are three types of dependencies supported in ASP.NET:

* File based dependency
* Key based dependency
* Time based dependency

**File Based Dependency**: File-based dependency invalidates a particular cache item when a file(s) on the disk changes.

Using cache dependency, we can force ASP.NET to expire cached data items from the cache when the dependency file changes. We can set the dependency to multiple files also. On such cases, the dependency should be built from an array of files or directories.

**Use**: File based dependency is very useful when you need to update data that is displayed to the user based on some changes on a file. For example, a news site always shows data from a file, and if some breaking news comes, they just update the file and the cache should expire, and during the expiry time, we can reload the cache with updated data using OnRemoveCallBack.

**Key Based Dependency**: Key-based dependency invalidates a particular cache item when another cache item changes.

**Use**: This is useful when we have multiple interrelated objects in the cache and if one of the objects changes, we need to updated or expire all of them.

**Time Based Dependency**: Time-based dependency causes an item to expire at a defined time. The Cache.Insert() method of the Cache class is used to create a time-based dependency. Two types of time based dependency are available.

* Absolute
* Sliding

# What are the event handlers that can be included in the Global.asax file?

The Global.asax file contains some of the following important event handlers:

* Application\_Error
* Application\_Start
* Application\_End
* Session\_Start
* Session\_End

# Make a list of all templates of the Repeater control.

The Repeater control contains the following templates:

* ItemTemplate
* AlternatingltemTemplate
* SeparatorTemplate
* HeaderTemplate
* FooterTemplate

# In which database is the information, such as membership, role management, profile, and Web parts personalization, stored?

The aspnetdb database stores all information.

# What is State Management? How many ways are there to maintain a state in .NET?

State management means to preserve state of a control, web page, object/data, and user in the application explicitly because all ASP.NET web applications are stateless.  
  
There are two ways to maintain a state in .NET, Client-Based state management and Server-Based state management.  
  
The following techniques can be used to implement the Client-Based state management:

* View State
* Hidden Fields
* Cookies
* Query Strings
* Control State

The following techniques can be used to implement Server-Based state management:

* Application State
* Session State
* Profile Properties

# What is a Cookie? Where is it used in ASP.NET?

Cookie is a lightweight executable program, which the server posts to client machines. Cookies store the identity of a user at the first visit of the Web site and validate them later on the next visits for their authenticity. The values of a cookie can be transferred between the user's request and the server's response.

# How can you ensure that no one has tampered with ViewState in a Web page?

To ensure that no one has tampered with ViewState in a Web page, set the EnableViewStateMac property to True.

# Explain the cookie less session and its working.

If ASP.NET manages the session state in the same process that processes the request and does not create a cookie. It is known as a cookie less session. If cookies are not available, a session is tracked by adding a session identifier to the URL. The cookie less session is enabled using the following code snippet: <sessionState cookieless="true" />

# What are the major built-in objects in ASP.NET?

The major built-in objects in ASP.NET are as follows:

* Application
* Request
* Response
* Server
* Session
* Context
* Trace

# To which class a Web form belongs to in the .NET Framework class hierarchy?

Web form belongs to the System.Web.UI.Page class.

# Which data types does the RangeValidator control support?

Data types supported by the RangeValidator control are Integer, Double, String, Currency, and Date.

# Which namespaces are necessary to create a localized application?

System.Globalization and System.Resources namespaces are essential to develop a localized application.

# Explain the validation controls. How many validation controls in ASP.NET 4.0?

ASP.NET 4.0 contains the following six types of validation controls:

* CompareValidator - Performs a comparison between the values contained in two controls.
* CustomValidator - Writes your own method to perform extra validation.
* RangeValidator- Checks value according to the range of value.
* RegularExpressionValidator - Ensures that input is according to the specified pattern or not.
* RequiredFieldValidator - Checks either a control is empty or not.
* ValidationSummary - Displays a summary of all validation error in a central location.

# What is difference between a Label control and a Literal control?

The Label control's final html code has an HTML tag; whereas, the Literal control's final html code contains only text, which is not surrounded by any HTML tag.

# What are the Culture and UICulture values?

The Culture value determines the functions, such as Date and Currency, which are used to format data and numbers in a Web page. The UICulture value determines the resources, such as strings or images, which are loaded for a Web page in a Web application.

# Which method do you use to kill explicitly a users session?

The Session.Abandon() method kills the user session explicitly.

# Which class is inherited when an ASP.NET server control is added to a Web form?

The System.Web.UI.WebControls class is inherited when an ASP.NET server control is added to a Web form.

# What events are fired when a page loads?

The following events fire when a page loads:

* Init() - Fires when the page is initializing.
* LoadViewState() - Fires when the view state is loading.
* LoadPostData() - Fires when the postback data is processing.
* Load() - Fires when the page is loading.
* PreRender() - Fires at the brief moment before the page is displayed to the user as HTML.
* Unload() - Fires when the page is destroying the instances of server controls.

# Common properties of all validation controls.

Three common properties of validation controls are as follows:

* ControlToValidate - Provides a control to validate
* ErrorMessage - Displays an error message
* IsValid - Specifies if the control's validation has succeeded or not
* Text - Displays a text for validation control before validation

# How do you prevent a validation control from validating data at the client end?

You can prohibit a validation control to validate data at the client side by setting the EnableClientScript property to False.

# What is cross-page posting in ASP.NET?

The Server.Transfer() method is used to post data from one page to another. In this case, the URL remains the same. However, in cross page posting, data is collected from different Web pages and is displayed on a single page. To do so, you need to set the PostBackUrl property of the control, which specifies the target page. In the target page, you can access the PreviousPage property. For this, you need to use the @PreviousPageType directive. You can access the controls of previous page by using the FindControl() method.