**Basic** **Questions:**

**Please briefly explain the usage of Global.asax?**

Global.asax is basically ASP.NET Application file. It’s a place to write code for Application-level events such as Application start, Application end, Session start and end, Application error etc. raised by ASP.NET or by HTTP Modules.

There is a good list of events that are fired but following are few of the important events in Global.asax:

* Application\_Init occurs in case of application initialization for the very first time.
* Application\_Start fires on application start.
* Session\_Start fires when a new user session starts
* Application\_Error occurs in case of an unhandled exception generated from application.
* Session\_End fires when user session ends.
* Application\_End fires when application ends or time out.

**What are the types of Authentication in ASP.NET?**

There are three types of authentication available in ASP.NET:

* *Windows Authentication:* This authentication method uses built-in windows security features to authenticate user.
* *Forms Authentication:* authenticate against a customized list of users or users in a database.
* *Passport Authentication:* validates against Microsoft Passport service which is basically a centralized authentication service.

**What is the difference between Label Control and Literal Control?**

A Label control in ASP.NET renders text inside <span> tags while a Literal Control renders just the text without any tags.  
With Label controls we can easily apply styles using it’s CssClass property, however, if we don’t want to apply style/formatting, it’s better to go for a Literal control.

**Hyperlink Vs LinkButton in ASP.NET?**

A Hyperlink just redirects to a given URL identified by “NavigateURL” property. However a LinkButton which actually displays a Hyperlink style button causes a postback to the same page but it doesn’t redirect to a given URL.

Hopefully, this pool of ASP.NET Interview Questions and Answers along with previous list of Top 10 will be helpful for ASP.NET Developers.

**HTML Server Controls Vs Web Server Controls, Please define?**

**HTML Server Controls** are server-side mapped form of HTML elements. In order to make HTML elements programmable on server-side, **ASP.NET** framework added runat=”server” attribute, so it’s accessible in ASP.NET code-behind. A typical HTML Server Control is as follows:

*<input type=”text” id=”txtFirstName” runat=”server” />*

On the other hand, **Web Server Controls** are more feature-rich as compared to HTML server controls and truly designed to provide Win Apps development experience for ASP.NET Web developers. Also, it provides comparatively high level of abstraction. Apart from mapping to existing HTML elements, web server controls provide more rich and complex functionality like Calendar, Grid, Repeater, menu, tree view etc.

**So, which one you prefer to use while developing an ASP.NET Web Application?**

Although both of these types of controls render HTML elements but Web Server controls being rich in functionality can render additional HTML tags that collectively fulfill control’s functionality. On the other hand, HTML Server Controls are comparatively lightweight only producing the corresponding HTML element.So, preference depends totally on your circumstances. If you are migrating from a classic ASP application to ASP.NET, then only choice is using HTML server controls but if you are going to develop a new application and you wanted to provide rich functionality with ease (i.e. calendar, grid, tree view etc), web server controls are the only choice because we don’t have it in HTML server controls.

**What is the role of ValidationSummary Control?**

Sometimes, we have a requirement that all validation messages need to display at one location (may be top or bottom of a web form). In such scenario, ValidationSummary control displays all validation messages at one place.  
DisplayMode property of this control can be used to display in different formats as follows:

* BulletList
* List
* SingleParagraph

**Globalization Vs Localization**

There are situations when we need to build an application that work for multiple cultures e.g *en-US*, *ar-SA* etc, this process of designing and building applications that work for more than one cultures is called globalization. However, customizing an application for a specific culture is localization. Both globalization and localization normally go together.

**How to access information about a user’s locale in ASP.NET?**

User’s locale information can be accessed through ***System.Web.UI.Page.Culture*** property.

**Difference between Culture and UICulture properties in ASP.NET?**

***CultureInfo*** class plays an important role for localizing our application pages. Culture is specific in localizing non-visual parts of the page like DateTime, Currency, number formatting etc. while on the other hand, ***UICulture*** is specific in localizing visual part of a webpage like Language being used to display the contents of the web page.

**Difference between Local and Global resources?**

Resources for a localized ASP.NET application can be stored locally as well as globally. Local resources are specific to web page and stored in a folder ***App\_LocalResources***. It can be accessible to that specific page only. Global resources are accessed by almost all application pages and stored in ***App\_GlobalResources*** folder.

**What is a Neutral Culture? and how its different from a Specific Culture?**

As we have seen earlier that a culture has two things i.e. Language and Country/Region. For example, en-Us represents English – United States; en-GB represents English – Great Britain; and ar-SA represents Arabic – Saudi Arabia. So, we can easily understand that first page is language while second is country/region.

A neutral culture is one that is associated only with part-1 i.e. language and not with part-2 i.e. country/region. For example, ar is the neutral name for Arabic culture while ar-SA is specific to Saudi Arabian Arabic culture.

**Difference between Response.Write() and Response.Output.Write()?**

Difference between Response.Write() and Response.Output.Write() is that the later provide formatting capability through String.Format-Style which the former doesn’t have.

In case of Response.Write(), HttpResponse calls directly the following method:

*public void Write(object obj)*

*{*

*this.\_writer.Write(obj);*

*}*

Above method internally calls it TextWriter’s write() method.

However, in case of Response.Output.Writer(), HttpResponse actually get reference to TextWriter through

Reponse.Output and then after getting control TextWtiter, call other overloaded method that helps to format the string.

*Response.Output.Write(“This is an {0} test at {1:d}”, “amazing”, DateTime.Now);*

**What is Caching and what are the benefits of using it?**

Performance has always been a concern for web based applications. So, Caching is a mechanism that improve performance for an application by storing data in memory for fast access. When the application will access data from Cache (i.e. in-memory) instead of fetching it from original data store (may be a database), it will definitely improve performance.But Caching benefits are not limited to performance only, it also improve application Scalability as well as Availability.

* Load on server is reduced when data is fetched from Cache instead of original source, thus improving scalability of an application.
* Caching normally keep serving application data even if the original source is temporarily down, thus improving availability of an application.

**Authentication Vs Authorization?**

Authentication and Authorization are two key security related concepts that are independent but normally go together.

Authentication is a process that verifies the identity of a user. On ther hand, Authorization is the process of assigning rights/privileges to already authenticated user.

For example, when a user tries to login a web application. First of all, user identity is verified that either he/she is valid registered user of application. If his/her identity validated successfully then appropriate privileges are assigned accordingly. Different users may have different privileges on same application, for example, user1 can only view/read some records while user2 may have privileges for all CRUD (Create, Read, Update, Delete) operations on same data.

**What are the available Authentication modes in ASP.NET?**

We have already explained this question in previous [ASP.NET Interview Questions and Answers](http://www.webdevelopmenthelp.net/2013/12/top-10-asp-net-interview-questions-and-answers.html) post, don’t skip this important questions details.

**What is the difference between Windows Authentication and Forms Authentication in ASP.NET?**

**Windows Authentication** is a way to authenticate a user against Windows accounts. Windows authentication mode is suitable for corporate users in windows environment.

In case of **Forms Authentication**, a separate list of users is maintained for authentication. For example, we can maintain the list in database and authenticate user against it.

We can set authentication mode in web.config as follows:

*<authentication mode=”Forms”>*

**What is Protected Configuration in ASP.NET?**

While developing an ASP.NET application, we normally store a number of important sensitive information in our config files like encryption keys, connection strings etc. Application vulnerability increases if this sensitive information is stored as plane text. So ***Protected Configuration*** is an ASP.NET feature that enables to encrypt such sensitive information in configuration files.

**What is Passport Authentication?**

As we have discussed [previously](http://www.webdevelopmenthelp.net/2013/12/top-10-asp-net-interview-questions-and-answers.html) that there are three types of authentications in ASP.NET i.e.

* Windows Authentication
* Forms Authentication
* Passport Authentication

Windows and Forms Authentications are already explained.  
Passport Authentication actually validates against a centralized authentication service i.e. Microsoft Passport Service. We don’t need to implement our own custom authentication mechanism if implementing .NET Passport Single Sign-In (SSI) service.

**Can you briefly explain how Passport Authentication works?**

As discussed above that Passport Authentication is a central service. It just authenticate (validate the credentials), no authorization (grant or deny access to a site). So, implementing application will check for the Passport Authentication Cookie. In case of unavailability of Passport Cookie, user is redirected to passport Sign-In page. User provides the credentials on Sign-In page, if validated,  Authentication Cookie is stored on client machine and redirected to the requested page.

**What are the advantages of using Passport Authentication?**

Advantages of Passport Authentication are:

* We don’t need to care of authentication mechanism our self, Passport SSI does this for us.
* Single login credentials can be used to access multiple sites. User don’t need to remember separate credentials for individual site.

**What is Role-based Security?**

We have discussed about authentication in above questions but another different but related concept is Authorization. Authorization is a process of granting privileges or permissions on resources to an authenticated user. So,  
“*Role Based Security is a technique we use to implement authorization on the basis of user’s roles within an   organization. It’s more granular approach to grant or revoke permissions on resources through user’s roles.*“

An example of granting or revoking permissions in configuration file using windows built-in groups as follows:

*<authorization >*  
*<allow roles=”MyDomain1Administrators” / >   < !– Allow Admin of this domain — >*  
*<deny users=”\*”  / >                                          < !– Deny anyone else. — >*  
*</authorization >*

**What are the different Security Controls in ASP.NET?**

ASP.NET provides several security controls which are actually Web Server controls. You can find those in your Visual Studio Toolbox.  
 **Login Control:**  
In almost every application we need to take user credentials on a typical login page. Login control provides the same standard functionality and reduces the effort for building it from scratch.

**LoginName:**  
After a user successfully logged in to an application, we normally display his/her username to top right or some other place on the page. Now, this functionality is provided by LoginName control.

**LoginView Control:**  
LoginView control displays different view for different users. Using AnonymousTemplate and LoggedInTemplate, different information can be presented to different users.

**LoginStatus Control:**  
LoginStatus control implies whether a user is authenticated or not. For an unathenticated user, it displays a link to login page. On the other hand, for authenticated user, a logout link is displayed.

**LoginRecovery Control:**  
Password recovery is another important functionality simplified through PasswordRecovery control. It sends an email with login credentials to registered user email.

**What is Code-Access Security (CAS)?**

In one of above ASP.NET security related interview questions, we discussed about Role Based Security that restrict access to resources on the basis of user’s role. CAS (**Code Access Security**) is entirely a different concept. It’s .NET CLR’s security system that restrict the code to perform an unwanted task by applying security policies. Using CAS (Code Access Security), we can restrict *what our code can do?* and also *what code can call our code?*

**What are the key functions of Code Access Security?**

* Defines permissions and permission sets that represent the right to access various system resources.
* Enables code to demand that its callers have specific permissions.
* Enables code to demand that its callers possess a digital signature, thus allowing only callers from a particular organization or site to call the protected code.
* Enforces restrictions on code at run time by comparing the granted permissions of every caller on the call stack to the permissions that callers must have.

Code Access Security Tool (Caspol.exe) can be used to turn Code Access Security ON or OFF as follows:

* caspol -security on
* caspol -security off

We can also list all code groups using following command.

* caspol -listgroups

**What is Impersonation in ASP.NET?**

Impersonation is an act of a user to pretend itself to be another user. By default, ASP.NET executes application code using the same user account as that of ASP.NET process i.e. Network Service. But with impersonation enabled, it executes code with the windows identity of the user making the request.

For example, if a user ‘user1′ logged in and IIS is setup to run as Network Service. If ‘user1′ call a piece of code on another computer (may be a web service call), the other computer will see the IIS user instead of ‘user1′. But we can enable impersonation to allow ‘user1′ to access the web service using its windows identity instead of Network Service.

**How to configure Impersonation in ASP.NET?**

By default, impersonation is disabled in ASP.NET. Impersonation can be Enabled/Disabled as follows:

*</configuration>*  
*<system.web>*  
*<identity impersonate=”true”/> <! — To disable set impersonate=”false” –>*  
*</system.web>*  
*</configuration>*

Impersonate a specific user account as:

*<identity impersonate=”true” userName=”user” password=”pwd” />*

**Define AJAX?**

AJAX stands for “Asynchronous JavaScript and XML”. It’s basically a technique for creating Rich Internet Applications (RIA) that are faster as well as more interactive, using a combination of commonly used techniques as HTML/XHTML, CSS, Document Object Model (DOM), JavaScript, XML/XSLT and XMLHttpRequest object.

**XMLHttpRequest** object is the key basis of AJAX and makes it possible to communicate with the web server asynchronously and exchange data.

**Please elaborate XMLHttpRequest Object further?**

XMLHttpRequest is the core object in AJAX technology regardless of any implementation. XMLHttpRequest object is used to exchange data with a server seamlessly. Basically JavaScript uses this Object to exchange XML as well as text data between client and server. An AJAX implementation uses this object and communicate with server but it doesn’t require the complete page to be refreshed.

**How to send a request to server using XMLHttpRequest Object?**

We can send a request to server using HTTP GET and POST methods as follows:

***//Simple GET Request***

*var xmlHttp = new XMLHttpRequest();*

*xmlHttp.open(“GET”, “TestFile.txt”, true);*

*xmlHttp.send();*

***//Simple POST Request***

*var xmlHttp = new XMLHttpRequest();*

*xmlHttp.open(“POST”, “TestFile.txt”, true);*

*xmlHttp.send();*

**What is ASP.NET AJAX?**

Microsoft provided an implementation of AJAX functionality known as **ASP.NET AJAX**.  
As we discussed in above interview question that AJAX is a combination of various techniques, so Microsoft simplified the usage of these techniques with its own implementation. ASP.NET AJAX is a set of extensions to ASP.NET and comes with reusable AJAX controls. Using ASP.NET AJAX, we can develop applications that can update partial page instead of a complete page refresh.

**Difference between Synchronous and Asynchronous Postback?**

In Synchronous postback, complete web page is sent to server and in return rendering the output (i.e. complete page), whereas in case of Asynchronous postback, partial page goes to the server and renders only partial (required) part of the page.

**What are the basic controls in ASP.NET AJAX?**

Following controls can be considered as core AJAX controls in ASP.NET.

* ScriptManager
* ScriptManagerProxy
* UpdatePanel
* UpdateProgress
* Timer

Later more controls are added to ASP.NET AJAX library e.g. Script Loader, Client Data Context, Client Data Access, jQuery Integration etc.

**What is a ScriptManager in ASP.NET AJAX?**

In order to use AJAX functionality on a web page, we add a ScriptManager control to the page in most of the scenarios, because ScriptManager control register AJAX library scripts to that particular web page. We can have only one ScriptManager per page.

*<asp:ScriptManager ID=”ScriptManager1″ runat=”server”></asp:ScriptManager>*

ScriptManager basically manages all ASP.NET AJAX resources of a web page, creates proxies for asynchronous web service call and also manages partial page updates… etc.

**ScriptManager Vs ScriptManagerProxy?**

As we understand that we can have only one **ScriptManager** control on a page but we can have multiple **ScriptManagerProxy** controls.  
Consider a scenario that we have ScriptManager in our MasterPage that is available for all content pages. Now, we wanted to register a web service in a particular page. So, we will not add another ScriptManager to that page instead we will add ScriptManagerProxy to it in order to avoid error.

**What is the role of UpdatePanel in ASP.NET AJAX?**

UpdatePanel is the control that facilitate the partial page rendering functionality in an ASP.NET application. As discussed earlier that using ASP.NET AJAX, we can communicate with a web server asynchronously and update a part of a page without a complete page postback. In order to apply partial page update/rendering, we can add one or more UpdatePanel controls to our ASP.NET Page as follows:

*<asp:ScriptManager ID=”ScriptManager1″ runat=”server”></asp:ScriptManager>*

*<asp:UpdatePanel ID=”UpdatePanel1″ runat=”server”>*

*<ContentTemplate>*

*<asp:Label ID=”lblPanel” runat=”server” Text=”Update Panel Added.”></asp:Label><br />*

*<asp:Button ID=”btnTestButton”*  
*runat=”server”*  
*OnClick=”btnTestButton\_Click”*  
*Text=”Test Button” />*

*</ContentTemplate>*

*</asp:UpdatePanel>*

**What are the limitations of AJAX?**

* AJAX on an application will not work if JavaScript is disabled.
* In some scenarios, it exposes vulnerability.
* It will always be difficult to bookmark application state.
* Application behavior may be slow in some scenarios, because of different loading time of controls on a single page.

**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.

**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….

**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

**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.

**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.

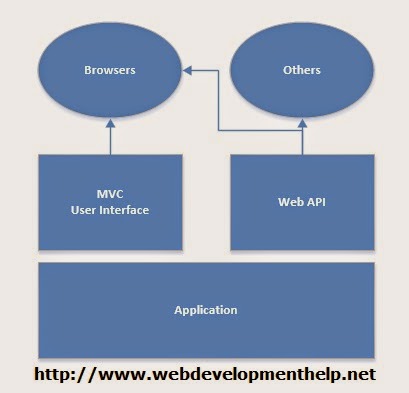
But WCF is still a good choice for following scenarios:

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

A good understanding for WCF(Windows Communication Foundation), please follow [WCF Tutorial](http://www.topwcftutorials.net/).

**MVC Vs ASP.NET Web API?**

As in previous ASP.NET Web API Interview Questions, we discussed that purpose of Web API framework is to generate HTTP services that reaches 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.

[](http://www.webdevelopmenthelp.net/wp-content/uploads/2014/05/MVCVsWebAPI.jpg)

**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. As we discussed in earlier interview question about 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 it’s 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 required in a scenario to restrict access to a Web API method through HTTP POST only as follows:

**[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](http://www.webdevelopmenthelp.net/2014/01/performing-crud-operations-using-asp-net-web-api-part-2.html) for all CRUD (Create, Retrieve, Update, Delete) operations can be [found here](http://www.webdevelopmenthelp.net/2014/01/performing-crud-operations-using-asp-net-web-api-part-2.html).

**How we can 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:

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

**8. What are manipulators?**  
Manipulators are the functions which can be used in conjunction with the insertion (<<) and extraction (>>) operators on an object. Examples are endl and setw.  
  
  
**11. What is Inline function?**  
Inline function is a technique used by the compilers and instructs to insert complete body of the function wherever that function is used in the program source code.  
  
**12. What is avirtual function?**  
Virtual function is a member function ofclass and its functionality can be overridden in its derived class. This function can be implemented by using a keyword called virtual, and it can be given during function declaration.  
  
Virtual function can be achieved in C++, and it can be achieved in C Languageby using function pointers or pointers to function.  
  
**13. What isfriend function?**  
Friend function is a friend of a class that is allowed to access to Public, private or protected data in that same class. If the function is defined outside the class cannot access such information.  
  
Friend can be declared anywhere in the class declaration, and it cannot be affected by access control keywords like private, public or protected.  
  
**18. What is the use of finalize method?**  
Finalize method helps to perform cleanup operations on the resources which are not currently used. Finalize method is protected , and it is accessible only through this class or by a derived class.  
  
**19. What are different types of arguments?**  
A parameter is a variable used during the declaration of the function or subroutine and arguments are passed to the function , and it should match with the parameter defined. There are two types of Arguments.   
Call by Value – Value passed will get modified only inside the function , and it returns the same value whatever it is passed it into the function.   
Call by Reference – Value passed will get modified in both inside and outside the functions and it returns the same or different value.   
  
**20. What is super keyword?**  
Super keyword is used to invoke overridden method which overrides one of its superclass methods. This keyword allows to access overridden methods and also to access hidden members of the superclass.  
  
It also forwards a call from a constructor to a constructor in the superclass.  
  
  
**24. What are tokens?**  
Token is recognized by a compiler and it cannot be broken down into component elements. Keywords, identifiers, constants, string literals and operators are examples of tokens.  
  
Even punctuation characters are also considered as tokens – Brackets, Commas, Braces and Parentheses.  
  
**27. What is an abstraction?**  
Abstraction is a good feature of OOPS , and it shows only the necessary details to the client of an object. Means, it shows only necessary details for an object, not the inner details of an object. Example – When you want to switch On television, it not necessary to show all the functions of TV. Whatever is required to switch on TV will be showed by using abstract class.  
  
**32. What are the various types of constructors?**  
There are three various types of constructors , and they are as follows:.  
  
- Default Constructor – With no parameters.  
- Parametric Constructor – With Parameters. Create a new instance of a class and also passing arguments simultaneously.  
- Copy Constructor – Which creates a new object as a copy of an existing object.  
  
**33. What is early and late binding?**  
Early binding refers to assignment of values to variables during design time whereas late binding refers to assignment of values to variables during run time.  
  
**34. What is ‘this’ pointer?**  
THIS pointer refers to the current object of a class. THIS keyword is used as a pointer which differentiates between the current object with the global object. Basically, it refers to the current object.  
  
  
**37. What is pure virtual function?**  
A pure virtual function is a function which can be overridden in the derived classbut cannot be defined. A virtual function can be declared as Pure by using the operator =0.  
  
Example -.  
Virtual void function1() // Virtual, Not pure  
Virtual void function2() = 0 //Pure virtual  
  
**38. What are all the operators that cannot be overloaded?**  
Following are the operators that cannot be overloaded -.   
Scope Resolution (:: )   
Member Selection (.)   
Member selection through a pointer to function (.\*)   
  
**41. What is a copy constructor?**  
This is a special constructor for creating a new object as a copy of an existing object. There will be always only on copy constructor that can be either defined by the user or the system.  
  
**45. What is static and dynamic binding?**  
Binding is nothing but the association of a name with the class. Static binding is a binding in which name can be associated with the class during compilation time , and it is also called as early Binding.  
  
Dynamic binding is a binding in which name can be associated with the class during execution time , and it is also called as Late Binding.

**47. Which keyword can be used for overloading?**  
Operator keyword is used for overloading.

### C# Interview Questions and **3. Can multiple catch blocks be executed?** -->  No, Multiple catch blocks can’t be executed. Once the proper catch code executed, the control is transferred to the finally block and then the code that follows the finally block gets executed. **4. What is the difference between public, static and void?** -->  All these are access modifiers in C#. Public declared variables or methods are accessible anywhere in the application. Static declared variables or methods are globally accessible without creating an instance of the class. The compiler stores the address of the method as the entry point and uses this information to begin execution before any objects are created. And Void is a type modifier that states that the method or variable does not return any value. **5. What is an object?** -->  An object is an instance of a class through which we access the methods of that class. “New” keyword is used to create an object. A class that creates an object in memory will contain the information about the methods, variables and behavior of that class.

**6. Define Constructors?**   
-->  A constructor is a member function in a class that has the same name as its class. The constructor is automatically invoked whenever an object class is created. It constructs the values of data members while initializing the class.  
  
**7. What is Jagged Arrays?**  
-->  The array which has elements of type array is called jagged array. The elements can be of different dimensions and sizes. We can also call jagged array as Array of arrays.  
  
**8. What is the difference between ref & out parameters?**  
-->  An argument passed as ref must be initialized before passing to the method whereas out parameter needs not to be initialized before passing to a method.  
  
**9. What is the use of using statement in C#?**   
-->  The using block is used to obtain a resource and use it and then automatically dispose of when the execution of block completed.  
  
**10. What is serialization?**   
-->  When we want to transport an object through network then we have to convert the object into a stream of bytes. The process of converting an object into a stream of bytes is called Serialization. For an object to be serializable, it should inherit ISerialize Interface.  
De-serialization is the reverse process of creating an object from a stream of bytes.

**11. Can “this” be used within a static method?**   
-->  We can’t use ‘This’ in a static method because we can only use static variables/methods in a static method.  
  
**12. What is difference between constants and read-only?**   
-->  Constant variables are declared and initialized at compile time. The value can’t be changed after wards. Read-only variables will be initialized only from the Static constructor of the class. Read only is used only when we want to assign the value at run time.  
  
**13. What is an interface class?**   
-->  Interface is an abstract class which has only public abstract methods and the methods only have the declaration and not the definition. These abstract methods must be implemented in the inherited classes.  
  
**14. What are value types and reference types?**   
-->  Value types are stored in the Stack whereas reference types stored on heap.  
  
Value types:  
[csharp] int, enum , byte, decimal, double, float, long[/csharp]  
  
Reference Types:  
[csharp] string , class, interface, object.[/csharp]  
  
**15. What are Custom Control and User Control?**   
-->  Custom Controls are controls generated as compiled code (Dlls), those are easier to use and can be added to toolbox. Developers can drag and drop controls to their web forms. Attributes can be set at design time. We can easily add custom controls to Multiple Applications (If Shared Dlls), If they are private then we can copy to dll to bin directory of web application and then add reference and can use them.  
  
User Controls are very much similar to ASP include files, and are easy to create. User controls can’t be placed in the toolbox and dragged – dropped from it. They have their design and code behind. The file extension for user controls is ascx.  
  
**20. Describe the accessibility modifier “protected internal”.**  
-->  Protected Internal variables/methods are accessible within the same assembly and also from the classes that are derived from this parent class.  
  
**21. What are the differences between System.String and System.Text.StringBuilder classes?**  
-->  System.String is immutable. When we modify the value of a string variable then a new memory is allocated to the new value and the previous memory allocation released. System.StringBuilder was designed to have concept of a mutable string where a variety of operations can be performed without allocation separate memory location for the modified string.  
  
**22. What’s the difference between the System.Array.CopyTo() and System.Array.Clone() ?**  
-->  Using Clone() method, we creates a new array object containing all the elements in the original array and using CopyTo() method, all the elements of existing array copies into another existing array. Both the methods perform a shallow copy.  
  
  
**26. What is the difference between Finalize() and Dispose() methods?**  
-->  Dispose() is called when we want for an object to release any unmanaged resources with them. On the other hand Finalize() is used for the same purpose but it doesn’t assure the garbage collection of an object.  
  
**27. What are circular references?**  
-->  Circular reference is situation in which two or more resources are interdependent on each other causes the lock condition and make the resources unusable.  
  
**28. What are generics in C#.NET?**  
-->  Generics are used to make reusable code classes to decrease the code redundancy, increase type safety and performance. Using generics, we can create collection classes. To create generic collection, System.Collections.Generic namespace should be used instead of classes such as ArrayList in the System.Collections namespace. Generics promotes the usage of parameterized types.  
  
**29. What is an object pool in .NET?**  
-->  An object pool is a container having objects ready to be used. It tracks the object that is currently in use, total number of objects in the pool. This reduces the overhead of creating and re-creating objects.  
  
**30. List down the commonly used types of exceptions in .Net?**  
-->  ArgumentException, ArgumentNullException , ArgumentOutOfRangeException, ArithmeticException, DivideByZeroException ,OverflowException , IndexOutOfRangeException ,InvalidCastException ,InvalidOperationException , IOEndOfStreamException , NullReferenceException , OutOfMemoryException , StackOverflowException etc.

**31. What are Custom Exceptions?**  
-->  Sometimes there are some errors that need to be handeled as per user requirements. Custom exceptions are used for them and are used defined exceptions.  
  
**32. What are delegates?**  
-->  Delegates are same are function pointers in C++ but the only difference is that they are type safe unlike function pointers. Delegates are required because they can be used to write much more generic type safe functions.  
  
**38. How can we set class to be inherited, but prevent the method from being over-ridden?**  
-->  Declare the class as public and make the method sealed to prevent it from being overridden.  
  
**39. What happens if the inherited interfaces have conflicting method names?**  
-->  Implement is up to you as the method is inside your own class. There might be problem when the methods from different interfaces expect different data, but as far as compiler cares you’re okay.

**41. How to use nullable types in .Net?**  
-->  Value types can take either their normal values or a null value. Such types are called nullable types.  
  
[csharp]Int? someID = null;  
If(someID.HasVAlue)  
{  
}  
[/csharp]  
  
**42. How we can create an array with non-default values?**  
-->  We can create an array with non-default values using Enumerable.Repeat.  
  
**43. What is difference between is and as operators in c#?**  
-->  “is” operator is used to check the compatibility of an object with a given type and it returns the result as Boolean.  
“as” operator is used for casting of object to a type or a class.  
  
**44. What’s a multicast delegate?**  
-->  A delegate having multiple handlers assigned to it is called multicast delegate. Each handler is assigned to a method.  
  
**45. What are indexers in C# .NET?**  
-->  Indexers are known as smart arrays in C#. It allows the instances of a class to be indexed in the same way as array.  
  
Eg:  
[csharp]public int this[int index] // Indexer declaration[/csharp]

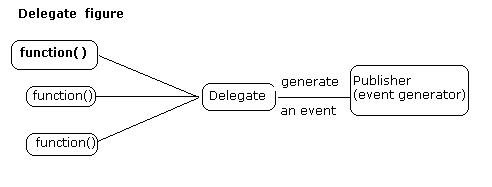
**46. What is difference between the “throw” and “throw ex” in .NET?**  
-->  “Throw” statement preserves original error stack whereas “throw ex” have the stack trace from their throw point. It is always advised to use “throw” because it provides more accurate error information.  
  
**47. What are C# attributes and its significance?**  
-->  C# provides developers a way to define declarative tags on certain entities eg. Class, method etc. are called attributes. The attribute’s information can be retrieved at runtime using Reflection.  
  
**48. How to implement singleton design pattern in C#?**  
-->  In singleton pattern, a class can only have one instance and provides access point to it globally.  
Public sealed class Singleton  
{  
Private static readonly Singleton \_instance = new Singleton();  
}  
  
**49. What is the difference between directcast and ctype?**  
-->  DirectCast is used to convert the type of an object that requires the run-time type to be the same as the specified type in DirectCast.  
  
Ctype is used for conversion where the conversion is defined between the expression and the type.  
  
**50. Is C# code is managed or unmanaged code?**  
-->  C# is managed code because Common language runtime can compile C# code to Intermediate language.

**19. Define enumeration?**

* Enumeration is defined as a value type that consists of a set of named values. These values are constants and are called enumerators. An enumeration type is declared using the enum keyword. Each enumerator in an enumeration is associated with an underlying type that is set, by default, on the enumerator. The following is an example that creates an enumeration to store different varieties of fruits:  
    
  enum Fruits {Mango, Apple, orange, Guava};   
    
  In the preceding example, an enumeration Fruits is created, where number 0 is associated with Mango, number 1 with Apple, number 2 with Orange, and number 3 with Guava. You can access the enumerators of an enumeration by these values.

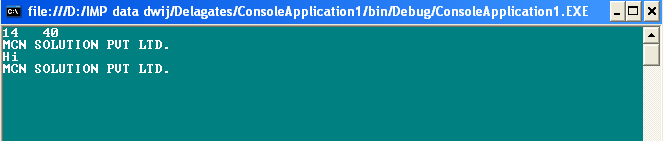
**Basic delegate in C#**

A delegate is similar to a class that is used for storing the reference to a method and invoking that method at runtime, as required. A delegate can hold the reference of only those methods whose signatures are same as that of the delegate. Some of the examples of delegates are type-safe functions, pointers, or callbacks.

An interesting and useful property of a delegate is that it does not know or care about the class of the object that it references. Any object will do; all that matters is that the method's argument types and return type match the delegate's. This makes delegates perfectly suited for "anonymous" invocation.A **delegate** is a form of type-safe function used by the .NET Framework.  Delegates specify a method to call and optionally an object to call the method on.  
  
or  
  
Shifting of work from one's control to another.  
delegate is keyword to make delegates in our program.  
Delegate is also work as class which has a base class Delegate (abstract class).  
Delegates are used to pass methods as arguments to other methods.  
In event handling its too difficult to have a person for every event differently:  
  
  
**Delegates have the following properties**

1. Delegates are similar to C++ function pointers, but are type safe.
2. Delegates allow methods to be passed as parameters.
3. Delegates can be used to define callback methods.
4. Delegates can be chained together; for example, multiple methods can be called on a single event.

**Syntax**  
public delegate <return type> DelegateName();  
  
**Program**using System;  
namespace Deligate  
{  
    public delegate int MyfirstDeligate(int i, int j);  
    public delegate void MySecondDeligate();  
    class Program  
    {  
        public int Add(int i, int j)  
        {  
            return i + j;  
        }  
        public int Sub(int i, int j)  
        {  
            return i - j;  
        }  
        public void Show()  
        {  
            Console.WriteLine("MCN SOLUTION PVT LTD.");  
        }  
        public void Display()  
        {  
            Console.WriteLine("Hi");  
        }  
        static void Main(string[] args)  
        {  
            Program p = new Program();  
            MyfirstDeligate m = new MyfirstDeligate(p.Add);  
            int result = m(9, 5);  
            m = new MyfirstDeligate(p.Sub);  
            int result1 = m(50, 10);  
            Console.WriteLine(result + "   " + result1);  
            MySecondDeligate m2 = new MySecondDeligate(p.Show);  
            MySecondDeligate m3 = new MySecondDeligate(p.Display);  
            m2 = m2 + m3;  
            m3 = m2 - m3;  
            m2();  
            m3();  
            Console.Read();  
       }  
    }  
}

**Output   
  
 **

**4) Define Static Members in C# ?**

If an attributes value had to be same across all the instances of the same class , static keyword is used. For example if the Minimum salary should be set for all employees in the employee class, use the following code

private static double MinSalary = 30000;

To access a private or public attributes or methods in a class, at first an object of the class should be created . Then using that instance of class , attributes or methods can be accessed. To access a static variable, we don't want to create an instance of the class having the static variable. We can directly refer that static variable as shown below.

double var = Employee.MinSalary

**8) Define Constructor Overloading in C# .net ?**

In Constructor overloading, n number of constructors can be created for same class. But the signatures of each constructor should vary. For example

public class Employee {

public Employee() { }

public Employee(String Name) { }

}

**16) How has exception handling changed in .NET Framework 4.0?**

In .NET 4.0, a new namespace, System.Runtime.ExceptionServices, has been introduced which contains the following classes for handling exceptions in a better and advanced manner:

* HandleProcessCorruptedStateExceptionsAttribute Class - Enables managed code to handle the corrupted state exceptions that occur in an operating system. These exceptions cannot be caught by specifying the try...catch block. To handle such exceptions, you can apply this attribute to the method that is assigned to handle these exceptions.
* FirstChanceExceptionEventArgs Class - Generates an event whenever a managed exception first occurs in your code, before the common language runtime begins searching for event handlers.

**29. Can you use the 'throws' clause to raise an exception?**

* No, the throws clause cannot be used to raise an exception. The throw statement signals the occurrence of an exception during the execution of a program. When the program encounters a throw statement, the method terminates and returns the error to the calling method.

**34. What is the function of the Try-Catch-Finally block?**

* The try block encloses those statements that can cause exception and the catch block handles the exception, if it occurs. Catch block contains the statements that have to be executed, when an exception occurs. The finally block always executes, irrespective of the fact whether or not an exception has occurred. The finally block is generally used to perform the cleanup process. If any exception occurs in the try block, the program control directly transfers to its corresponding catch block and later to the finally block. If no exception occurs inside the try block, then the program control transfers directly to the finally block.

**39. You have defined a destructor in a class that you have developed by using the C# programming language, but the destructor never executed. Why did the destructor not execute?**

* The runtime environment automatically invokes the destructor of a class to release the resources that are occupied by variables and methods of an object. However, in C#, programmers cannot control the timing for invoking destructors, as Garbage Collector is only responsible for releasing the resources used by an object. Garbage Collector automatically gets information about unreferenced objects from .NET's runtime environment and then invokes the Finalize() method.  
    
  Although, it is not preferable to force Garbage Collector to perform garbage collection and retrieve all inaccessible memory, programmers can use the Collect() method of the Garbage Collector class to forcefully execute Garbage Collector.

**40.What is a hashtable?**

* Hashtable is a data structure that implements the IDictionary interface. It is used to store multiple items and each of these items is associated with a unique string key. Each item can be accessed using the key associated with it. In short, hashtable is an object holding the key-value pairs.

**48. What is a static constructor?**

Static constructors are introduced with C# to initialize the static data of a class. CLR calls the static constructor before the first instance is created.  
  
The static constructor has the following features:

* No access specifier is required to define it.
* You cannot pass parameters in static constructor.
* A class can have only one static constructor.
* It can access only static members of the class.
* It is invoked only once, when the program execution begins.

**50. Differentiate between an abstract class and an interface.**

**Abstract Class**:

1. A class can extend only one abstract class
2. The members of abstract class can be private as well as protected.
3. Abstract classes should have subclasses
4. Any class can extend an abstract class.
5. Methods in abstract class can be abstract as well as concrete.
6. There can be a constructor for abstract class.
7. The class extending the abstract class may or may not implement any of its method.
8. An abstract class can implement methods.

**Interface**

1. A class can implement several interfaces
2. An interface can only have public members.
3. Interfaces must have implementations by classes
4. Only an interface can extend another interface.
5. All methods in an interface should be abstract
6. Interface does not have constructor.
7. All methods of interface need to be implemented by a class implementing that interface.
8. Interfaces cannot contain body of any of its method.