(Framework)

1. **What is .NET Framework?**
   The .NET Framework has two main components: the common language runtime and the .NET Framework class library.
   You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that ensure security and robustness.
   The class library, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.
2. **What is CLR?**
   The CLS is simply a specification that defines the rules to support language integration in such a way that programs written in any language, yet can interoperate with one another, taking full advantage of inheritance, polymorphism, exceptions, and other features. These rules and the specification are documented in the ECMA proposed standard document, "Partition I Architecture", available here.
3. **Is .NET a runtime service or a development platform?**
   Ans: It's both and actually a lot more. Microsoft .NET includes a new way of delivering software and services to businesses and consumers. A part of Microsoft.NET is the .NET Frameworks. The .NET frameworks SDK consists of two parts: the .NET common language runtime and the .NET class library. In addition, the SDK also includes command-line compilers for C#, C++, JScript, and VB. You use these compilers to build applications and components. These components require the runtime to execute so this is a development platform.
4. **What are the new features of Framework 1.1 ?**
   1. Native Support for Developing Mobile Web Applications
   2. Enable Execution of Windows Forms Assemblies Originating from the Internet Assemblies originating from the Internet zone—for example, Microsoft Windows® Forms controls embedded in an Internet-based Web page or Windows Forms assemblies hosted on an Internet Web server and loaded either through the Web browser or programmatically using the System.Reflection.Assembly.LoadFrom() method—now receive sufficient permission to execute in a semi-trusted manner. Default security policy has been changed so that assemblies assigned by the common language runtime (CLR) to the Internet zone code group now receive the constrained permissions associated with the Internet permission set. In the .NET Framework 1.0 Service Pack 1 and Service Pack 2, such applications received the permissions associated with the Nothing permission set and could not execute.
   3. Enable Code Access Security for ASP.NET Applications
      Systems administrators can now use code access security to further lock down the permissions granted to ASP.NET Web applications and Web services. Although the operating system account under which an application runs imposes security restrictions on the application, the code access security system of the CLR can enforce additional restrictions on selected application resources based on policies specified by systems administrators. You can use this feature in a shared server environment (such as an Internet service provider (ISP) hosting multiple Web applications on one server) to isolate separate applications from one another, as well as with stand-alone servers where you want applications to run with the minimum necessary privileges.
   4. Native Support for Communicating with ODBC and Oracle Databases
   5. Unified Programming Model for Smart Client Application Development
      The Microsoft .NET Compact Framework brings the CLR, Windows Forms

controls, and other .NET Framework features to small devices. The .NET Compact Framework supports a large subset of the .NET Framework class library optimized for small devices.

6. Support for IPv6
   The .NET Framework 1.1 supports the emerging update to the Internet Protocol, commonly referred to as IP version 6, or simply IPv6. This protocol is designed to significantly increase the address space used to identify communication endpoints in the Internet to accommodate its ongoing growth.

http://msdn.microsoft.com/netframework/technologyinfo/Overview/whatsnew.aspx

5. **What is Code Access Security (CAS)?**
   CAS is the part of the .NET security model that determines whether or not a piece of code is allowed to run, and what resources it can use when it is running. For example, it is CAS that will prevent a .NET web applet from formatting your hard disk.
   How does CAS work?
   The CAS security policy revolves around two key concepts - code groups and permissions. Each .NET assembly is a member of a particular **code group**, and each code group is granted the permissions specified in a **named permission set.**
   For example, using the default security policy, a control downloaded from a web site belongs to the 'Zone - Internet' code group, which adheres to the permissions defined by the 'Internet' named permission set. (Naturally the 'Internet' named permission set represents a very restrictive range of permissions.)
   Who defines the CAS code groups?
   Microsoft defines some default ones, but you can modify these and even create your own. To see the code groups defined on your system, run 'caspol -lg' from the command-line. On my ssystem it looks like this:
6. Level = Machine
7. Code Groups:
8.
9. 1. All code: Nothing
10. 1.1. Zone - MyComputer: FullTrust
11. 1.1.1. Honor SkipVerification requests: SkipVerification
12. 1.2. Zone - Intranet: LocalIntranet
13. 1.3. Zone - Internet: Internet
14. 1.4. Zone - Untrusted: Nothing
15. 1.5. Zone - Trusted: Internet
16. 1.6. StrongName - 002400000480000094000000060200000024000052534131000400000317. 000000CFCB3291AA715FE99D40D49040336F9056D7886FED46775BC7BB5430BA4444FEF8348EBD06
18. F962F39776AE4DC3B7B04A7FE6F49F25F740423EBF2C0B89698D8D08AC48D69CED0FC8F83B465E08
19. 07AC11EC1DCC7D054E807A43336DDE408A5393A48556123272CEEEE72F1660B71927D38561AABF5C
    AC1DF1734633C602F8F2D5: Everything

    Note the hierarchy of code groups - the top of the hierarchy is the most general ('All code'), which is then sub-divided into several groups, each of which in turn can be sub-divided. Also note that (somewhat counter-intuitively) a sub-group can be associated with a more permissive permission set than its parent.
    How do I define my own code group?
    Use caspol. For example, suppose you trust code from www.mydomain.com and you want it have full access to your system, but you want to keep the default restrictions for all other internet sites. To achieve this, you would add a new code group as a sub-group of the 'Zone - Internet' group, like this:
    caspol -ag 1.3 -site www.mydomain.com FullTrust
    Now if you run caspol -lg you will see that the new group has been added as group

1.3.1:
...
1.3. Zone - Internet: Internet
1.3.1. Site - www.mydomain.com: FullTrust
...
Note that the numeric label (1.3.1) is just a caspol invention to make the code groups easy to manipulate from the command-line. The underlying runtime never sees it.

How do I change the permission set for a code group?

Use caspol. If you are the machine administrator, you can operate at the 'machine' level - which means not only that the changes you make become the default for the machine, but also that users cannot change the permissions to be more permissive. If you are a normal (non-admin) user you can still modify the permissions, but only to make them more restrictive. For example, to allow intranet code to do what it likes you might do this:
caspol -cg 1.2 FullTrust
Note that because this is more permissive than the default policy (on a standard system), you should only do this at the machine level - doing it at the user level will have no effect.

Can I create my own permission set?

Yes. Use caspol -ap, specifying an XML file containing the permissions in the permission set. To save you some time, here is a sample file corresponding to the 'Everything' permission set - just edit to suit your needs. When you have edited the sample, add it to the range of available permission sets like this:
caspol -ap samplepermset.xml
Then, to apply the permission set to a code group, do something like this:
caspol -cg 1.3 SamplePermSet (By default, 1.3 is the 'Internet' code group)

I'm having some trouble with CAS. How can I diagnose my problem?

Caspol has a couple of options that might help. First, you can ask caspol to tell you what code group an assembly belongs to, using caspol -rsg. Similarly, you can ask what permissions are being applied to a particular assembly using caspol -rsp.

I can't be bothered with all this CAS stuff. Can I turn it off?

Yes, as long as you are an administrator. Just run:
caspol -s off

20. **What is MSIL, IL?**

When compiling to managed code, the compiler translates your source code into Microsoft intermediate language (MSIL), which is a CPU-independent set of instructions that can be efficiently converted to native code. MSIL includes instructions for loading, storing, initializing, and calling methods on objects, as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling, and other operations. Microsoft intermediate language (MSIL) is a language used as the output of a number of compilers and as the input to a just-in-time (JIT) compiler. The common language runtime includes a JIT compiler for converting MSIL to native code.

21. **Can I write IL programs directly?**

Yes. Peter Drayton posted this simple example to the DOTNET mailing list:

```
.assembly MyAssembly {}
.class MyApp {
  .method static void Main() {
    .entrypoint
    ldstr      "Hello, IL!"
    call       void System.Console::WriteLine(class System.Object)
    ret
  }
}
```

Just put this into a file called hello.il, and then run ilasm hello.il. An exe assembly will be generated.

Can I do things in IL that I can't do in C#?
Yes. A couple of simple examples are that you can throw exceptions that are not derived from System.Exception, and you can have non-zero-based arrays.

22. **What is CTS?**
The common type system defines how types are declared, used, and managed in the runtime, and is also an important part of the runtime's support for cross-language integration.
The common type system supports two general categories of types, each of which is further divided into subcategories:

Value types
Value types directly contain their data, and instances of value types are either allocated on the stack or allocated inline in a structure. Value types can be built-in (implemented by the runtime), user-defined, or enumerations.
Reference types
Reference types store a reference to the value's memory address, and are allocated on the heap. Reference types can be self-describing types, pointer types, or interface types. The type of a reference type can be determined from values of self-describing types. Self-describing types are further split into arrays and class types. The class types are user-defined classes, boxed value types, and delegates.

2. **What is JIT (just in time)? how it works?**
Before Microsoft intermediate language (MSIL) can be executed, it must be converted by a .NET Framework just-in-time (JIT) compiler to native code, which is CPU-specific code that runs on the same computer architecture as the JIT compiler.
Rather than using time and memory to convert all the MSIL in a portable executable (PE) file to native code, it converts the MSIL as it is needed during execution and stores the resulting native code so that it is accessible for subsequent calls.
The runtime supplies another mode of compilation called install-time code generation. The install-time code generation mode converts MSIL to native code just as the regular JIT compiler does, but it converts larger units of code at a time, storing the resulting native code for use when the assembly is subsequently loaded and executed.
As part of compiling MSIL to native code, code must pass a verification process unless an administrator has established a security policy that allows code to bypass verification. Verification examines MSIL and metadata to find out whether the code can be determined to be type safe, which means that it is known to access only the memory locations it is authorized to access.
3. **What is strong name?**
A name that consists of an assembly's identity—its simple text name, version number, and culture information (if provided)—strengthened by a public key and a digital signature generated over the assembly.
4. **What is portable executable (PE)?**
The file format defining the structure that all executable files (EXE) and Dynamic Link Libraries (DLL) must use to allow them to be loaded and executed by Windows. PE is derived from the Microsoft Common Object File Format (COFF). The EXE and DLL files created using the .NET Framework obey the PE/COFF formats and also add additional header and data sections to the files that are only used by the CLR. The specification for the PE/COFF file formats is available at
http://www.microsoft.com/whdc/hwdev/hardware/pecoffdown.mspx
5. **Which namespace is the base class for .net Class library?**
Ans: system.object
6. **What is Event - Delegate? clear syntax for writing a event delegate**
The **event** keyword lets you specify a delegate that will be called upon the

occurrence of some "event" in your code. The delegate can have one or more associated methods that will be called when your code indicates that the event has occurred. An event in one program can be made available to other programs that target the .NET Framework Common Language Runtime.

```
// keyword_delegate.cs
// delegate declaration
delegate void MyDelegate(int i);
class Program
{
  public static void Main()
  {
    TakesADelegate(new MyDelegate(DelegateFunction));
  }
  public static void TakesADelegate(MyDelegate SomeFunction)
  {
    SomeFunction(21);
  }
  public static void DelegateFunction(int i)
  {
    System.Console.WriteLine("Called by delegate with number: {0}.", i);
  }
}
```

21. **What are object pooling and connection pooling and difference? Where do we set the Min and Max Pool size for connection pooling?**
    Object pooling is a COM+ service that enables you to reduce the overhead of creating each object from scratch. When an object is activated, it is pulled from the pool. When the object is deactivated, it is placed back into the pool to await the next request. You can configure object pooling by applying the ObjectPoolingAttribute attribute to a class that derives from the System.EnterpriseServices.ServicedComponent class.
    Object pooling lets you control the number of connections you use, as opposed to connection pooling, where you control the maximum number reached. Following are important differences between object pooling and connection pooling:
    Creation. When using connection pooling, creation is on the same thread, so if there is nothing in the pool, a connection is created on your behalf. With object pooling, the pool might decide to create a new object. However, if you have already reached your maximum, it instead gives you the next available object. This is crucial behavior when it takes a long time to create an object, but you do not use it for very long.
    Enforcement of minimums and maximums. This is not done in connection pooling. The maximum value in object pooling is very important when trying to scale your application. You might need to multiplex thousands of requests to just a few objects. (TPC/C benchmarks rely on this.)

COM+ object pooling is identical to what is used in .NET Framework managed SQL Client connection pooling. For example, creation is on a different thread and minimums and maximums are enforced.

22. **What is Application Domain?**
    The primary purpose of the AppDomain is to isolate an application from other applications. Win32 processes provide isolation by having distinct memory address spaces. This is effective, but it is expensive and doesn't scale well. The

.NET runtime enforces AppDomain isolation by keeping control over the use of memory - all memory in the AppDomain is managed by the .NET runtime, so the runtime can ensure that AppDomains do not access each other's memory. Objects in different application domains communicate either by transporting copies of objects across application domain boundaries, or by using a proxy to exchange messages.
MarshalByRefObject is the base class for objects that communicate across application domain boundaries by exchanging messages using a proxy. Objects that do not inherit from **MarshalByRefObject** are implicitly marshal by value. When a remote application references a marshal by value object, a copy of the object is passed across application domain boundaries.

How does an AppDomain get created?
AppDomains are usually created by *hosts*. Examples of hosts are the Windows Shell, ASP.NET and IE. When you run a .NET application from the command-line, the host is the Shell. The Shell creates a new AppDomain for every application.
AppDomains can also be explicitly created by .NET applications. Here is a C# sample which creates an AppDomain, creates an instance of an object inside it, and then executes one of the object's methods. Note that you must name the executable 'appdomaintest.exe' for this code to work as-is.

```
using System;
using System.Runtime.Remoting;

public class CAppDomainInfo : MarshalByRefObject
{
  public string GetAppDomainInfo()
  {
            return "AppDomain = " + AppDomain.CurrentDomain.FriendlyName;
  }
}
public class App
{
  public static int Main()
  {
            AppDomain ad = AppDomain.CreateDomain( "Andy's new domain", null,
null );
            ObjectHandle oh = ad.CreateInstance( "appdomaintest",
"CAppDomainInfo" );
            CAppDomainInfo adInfo = (CAppDomainInfo)(oh.Unwrap());
            string info = adInfo.GetAppDomainInfo();
            Console.WriteLine( "AppDomain info: " + info );
            return 0;
  }
}
```

23. **What is serialization in .NET? What are the ways to control serialization?**
    Serialization is the process of converting an object into a stream of bytes. Deserialization is the opposite process of creating an object from a stream of bytes. Serialization/Deserialization is mostly used to transport objects (e.g. during remoting), or to persist objects (e.g. to a file or database).Serialization can be defined as the process of storing the state of an object to a storage medium. During this process, the public and private fields of the object and the name of the class, including the assembly containing the class, are converted to a stream of bytes, which is then written to a data stream. When the object is subsequently deserialized, an exact clone of the original object is created.

Binary serialization preserves type fidelity, which is useful for preserving the state of an object between different invocations of an application. For example, you can share an object between different applications by serializing it to the clipboard. You can serialize an object to a stream, disk, memory, over the network, and so forth. Remoting uses serialization to pass objects "by value" from one computer or application domain to another.

XML serialization serializes only public properties and fields and does not preserve type fidelity. This is useful when you want to provide or consume data without restricting the application that uses the data. Because XML is an open standard, it is an attractive choice for sharing data across the Web. SOAP is an open standard, which makes it an attractive choice.

There are two separate mechanisms provided by the .NET class library - XmlSerializer and SoapFormatter/BinaryFormatter. Microsoft uses XmlSerializer for Web Services, and uses SoapFormatter/BinaryFormatter for remoting. Both are available for use in your own code.

Why do I get errors when I try to serialize a Hashtable?
XmlSerializer will refuse to serialize instances of any class that implements IDictionary, e.g. Hashtable. SoapFormatter and BinaryFormatter do not have this restriction.

24. **What is the use of trace utility?**
    **
25. **What are server controls?**
    ASP.NET server controls are components that run on the server and encapsulate user-interface and other related functionality. They are used in ASP.NET pages and in ASP.NET code-behind classes.
26. **What is the difference between Web User Control and Web Custom Control?**
    Custom Controls
    Web custom controls are compiled components that run on the server and that encapsulate user-interface and other related functionality into reusable packages. They can include all the design-time features of standard ASP.NET server controls, including full support for Visual Studio design features such as the Properties window, the visual designer, and the Toolbox.
    There are several ways that you can create Web custom controls:

    You can compile a control that combines the functionality of two or more existing controls. For example, if you need a control that encapsulates a button and a text box, you can create it by compiling the existing controls together.

    If an existing server control almost meets your requirements but lacks some required features, you can customize the control by deriving from it and overriding its properties, methods, and events.

    If none of the existing Web server controls (or their combinations) meet your requirements, you can create a custom control by deriving from one of the base control classes. These classes provide all the basic functionality of Web server controls, so you can focus on programming the features you need.

If none of the existing ASP.NET server controls meet the specific requirements of your applications, you can create either a Web user control or a Web custom control that encapsulates the functionality you need. The main difference between the two controls lies in ease of creation vs. ease of use at design time.
Web user controls are easy to make, but they can be less convenient to use in advanced scenarios. You develop Web user controls almost exactly the same way that you develop Web Forms pages. Like Web Forms, user controls can be created in the

visual designer, they can be written with code separated from the HTML, and they can handle execution events. However, because Web user controls are compiled dynamically at run time they cannot be added to the Toolbox, and they are represented by a simple placeholder glyph when added to a page. This makes Web user controls harder to use if you are accustomed to full Visual Studio .NET design-time support, including the Properties window and Design view previews. Also, the only way to share the user control between applications is to put a separate copy in each application, which takes more maintenance if you make changes to the control. Web custom controls are compiled code, which makes them easier to use but more difficult to create; Web custom controls must be authored in code. Once you have created the control, however, you can add it to the Toolbox and display it in a visual designer with full Properties window support and all the other design-time features of ASP.NET server controls. In addition, you can install a single copy of the Web custom control in the global assembly cache and share it between applications, which makes maintenance easier.

Web user controls	Web custom controls
Easier to create	Harder to create
Limited support for consumers who use a visual design tool	Full visual design tool support for consumers
A separate copy of the control is required in each application	Only a single copy of the control is required, in the global assembly cache
Cannot be added to the Toolbox in Visual Studio	Can be added to the Toolbox in Visual Studio
Good for static layout	Good for dynamic layout

27. **What is exception handling?**
    When an exception occurs, the system searches for the nearest catch clause that can handle the exception, as determined by the run-time type of the exception. First, the current method is searched for a lexically enclosing try statement, and the associated catch clauses of the try statement are considered in order. If that fails, the method that called the current method is searched for a lexically enclosing try statement that encloses the point of the call to the current method. This search continues until a catch clause is found that can handle the current exception, by naming an exception class that is of the same class, or a base class, of the run-time type of the exception being thrown. A catch clause that doesn't name an exception class can handle any exception.
    Once a matching catch clause is found, the system prepares to transfer control to the first statement of the catch clause. Before execution of the catch clause begins, the system first executes, in order, any finally clauses that were associated with try statements more nested that than the one that caught the exception.
    Exceptions that occur during destructor execution are worth special mention. If an exception occurs during destructor execution, and that exception is not caught, then the execution of that destructor is terminated and the destructor of the base class (if any) is called. If there is no base class (as in the case of the object type) or if there is no base class destructor, then the exception is discarded.
28. **What is Assembly?**
    Assemblies are the building blocks of .NET Framework applications; they form the fundamental unit of deployment, version control, reuse, activation scoping, and security permissions. An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality. An assembly provides the common language runtime with the information it needs to be aware of type implementations. To the runtime, a type does not exist outside the context of an assembly.

Assemblies are a fundamental part of programming with the .NET Framework. An assembly performs the following functions:

It contains code that the common language runtime executes. Microsoft intermediate language (MSIL) code in a portable executable (PE) file will not be executed if it does not have an associated assembly manifest. Note that each assembly can have only one entry point (that is, **DllMain**, **WinMain**, or **Main**).

It forms a security boundary. An assembly is the unit at which permissions are requested and granted.

It forms a type boundary. Every type's identity includes the name of the assembly in which it resides. A type called MyType loaded in the scope of one assembly is not the same as a type called MyType loaded in the scope of another assembly.

It forms a reference scope boundary. The assembly's manifest contains assembly metadata that is used for resolving types and satisfying resource requests. It specifies the types and resources that are exposed outside the assembly. The manifest also enumerates other assemblies on which it depends.

It forms a version boundary. The assembly is the smallest versionable unit in the common language runtime; all types and resources in the same assembly are versioned as a unit. The assembly's manifest describes the version dependencies you specify for any dependent assemblies.

It forms a deployment unit. When an application starts, only the assemblies that the application initially calls must be present. Other assemblies, such as localization resources or assemblies containing utility classes, can be retrieved on demand. This allows applications to be kept simple and thin when first downloaded.

It is the unit at which side-by-side execution is supported.

Assemblies can be static or dynamic. Static assemblies can include .NET Framework types (interfaces and classes), as well as resources for the assembly (bitmaps, JPEG files, resource files, and so on). Static assemblies are stored on disk in PE files. You can also use the .NET Framework to create dynamic assemblies, which are run directly from memory and are not saved to disk before execution. You can save dynamic assemblies to disk after they have executed.

There are several ways to create assemblies. You can use development tools, such as Visual Studio .NET, that you have used in the past to create .dll or .exe files. You can use tools provided in the .NET Framework SDK to create assemblies with modules created in other development environments. You can also use common language runtime APIs, such as Reflection.Emit, to create dynamic assemblies.

29. **What are the contents of assembly?**

    In general, a static assembly can consist of four elements:

    The assembly manifest, which contains assembly metadata.

    Type metadata.

    Microsoft intermediate language (MSIL) code that implements the types.

    A set of resources.

30. **What are the different types of assemblies?**

    Private, Public/Shared, Satellite

31. **What is the difference between a private assembly and a shared assembly?**

1. **Location and visibility**: A private assembly is normally used by a single application, and is stored in the application's directory, or a sub-directory beneath. A shared assembly is normally stored in the global assembly cache,

which is a repository of assemblies maintained by the .NET runtime. Shared assemblies are usually libraries of code which many applications will find useful, e.g. the .NET framework classes.

2. **Versioning**: The runtime enforces versioning constraints only on shared assemblies, not on private assemblies.

23. **What are Satellite Assemblies? How you will create this? How will you get the different language strings?**
Satellite assemblies are often used to deploy language-specific resources for an application. These language-specific assemblies work in side-by-side execution because the application has a separate product ID for each language and installs satellite assemblies in a language-specific subdirectory for each language. When uninstalling, the application removes only the satellite assemblies associated with a given language and .NET Framework version. No core .NET Framework files are removed unless the last language for that .NET Framework version is being removed.
(For example, English and Japanese editions of the .NET Framework version 1.1 share the same core files. The Japanese .NET Framework version 1.1 adds satellite assemblies with localized resources in a \ja subdirectory. An application that supports the .NET Framework version 1.1, regardless of its language, always uses the same core runtime files.)
http://www.ondotnet.com/lpt/a/2637
**

24. **How will u load dynamic assembly? How will create assemblies at run time?**
**

25. **What is Assembly manifest? what all details the assembly manifest will contain?**
Every assembly, whether static or dynamic, contains a collection of data that describes how the elements in the assembly relate to each other. The assembly manifest contains this assembly metadata. An assembly manifest contains all the metadata needed to specify the assembly's version requirements and security identity, and all metadata needed to define the scope of the assembly and resolve references to resources and classes. The assembly manifest can be stored in either a PE file (an .exe or .dll) with Microsoft intermediate language (MSIL) code or in a standalone PE file that contains only assembly manifest information.
It contains Assembly name, Version number, Culture, Strong name information, List of all files in the assembly, Type reference information, Information on referenced assemblies.

26. **Difference between assembly manifest & metadata?**
assembly manifest - An integral part of every assembly that renders the assembly self-describing. The assembly manifest contains the assembly's metadata. The manifest establishes the assembly identity, specifies the files that make up the assembly implementation, specifies the types and resources that make up the assembly, itemizes the compile-time dependencies on other assemblies, and specifies the set of permissions required for the assembly to run properly. This information is used at run time to resolve references, enforce version binding policy, and validate the integrity of loaded assemblies. The self-describing nature of assemblies also helps makes zero-impact install and XCOPY deployment feasible.
metadata - Information that describes every element managed by the common language runtime: an assembly, loadable file, type, method, and so on. This can include information required for debugging and garbage collection, as well as security attributes, marshaling data, extended class and member definitions, version binding, and other information required by the runtime.

27. **What is Global Assembly Cache (GAC) and what is the purpose of it? (How to make an assembly to public? Steps) How more than one version of an assembly can keep in same place?**
Each computer where the common language runtime is installed has a machine-wide code cache called the global assembly cache. The global assembly cache stores assemblies specifically designated to be shared by several applications on the computer. You should share assemblies by installing them into the global assembly cache only when you need to.
Steps
- Create a strong name using sn.exe tool
eg: sn -k keyPair.snk
- with in AssemblyInfo.cs add the generated file name
eg: [assembly: AssemblyKeyFile("abc.snk")]
- recompile project, then install it to GAC by either
drag & drop it to assembly folder (C:\WINDOWS\assembly OR C:\WINNT\assembly) (shfusion.dll tool)
or
gacutil -i abc.dll

28. **If I have more than one version of one assemblies, then how'll I use old version (how/where to specify version number?)in my application?**
**

29. **How to find methods of a assembly file (not using ILDASM)**
Reflection

30. **What is Garbage Collection in .Net? Garbage collection process?**
The process of transitively tracing through all pointers to actively used objects in order to locate all objects that can be referenced, and then arranging to reuse any heap memory that was not found during this trace. The common language runtime garbage collector also compacts the memory that is in use to reduce the working space needed for the heap.

31. **Readonly vs. const?**
A **const** field can only be initialized at the declaration of the field. A **readonly** field can be initialized either at the declaration or in a constructor. Therefore, **readonly** fields can have different values depending on the constructor used. Also, while a **const** field is a compile-time constant, the **readonly** field can be used for runtime constants, as in the following example:
public static readonly uint l1 = (uint) DateTime.Now.Ticks;

32. **What is Reflection in .NET? Namespace? How will you load an assembly which is not referenced by current assembly?**
All .NET compilers produce metadata about the types defined in the modules they produce. This metadata is packaged along with the module (modules in turn are packaged together in assemblies), and can be accessed by a mechanism called **reflection**. The System.Reflection namespace contains classes that can be used to interrogate the types for a module/assembly.
Using reflection to access .NET metadata is very similar to using ITypeLib/ITypeInfo to access type library data in COM, and it is used for similar purposes - e.g. determining data type sizes for marshaling data across context/process/machine boundaries.
Reflection can also be used to dynamically invoke methods (see System.Type.InvokeMember), or even create types dynamically at run-time (see System.Reflection.Emit.TypeBuilder).

33. **What is Custom attribute? How to create? If I'm having custom attribute in an assembly, how to say that name in the code?**
A: The primary steps to properly design custom attribute classes are as follows:

a. Applying the AttributeUsageAttribute ([AttributeUsage(AttributeTargets.All, Inherited = false, AllowMultiple = true)])
b. Declaring the attribute. (class public class MyAttribute : System.Attribute { // . . . })
c. Declaring constructors (public MyAttribute(bool myvalue) { this.myvalue = myvalue; })
d. Declaring properties
e. public bool MyProperty
f. {
g. get {return this.myvalue;}
h. set {this.myvalue = value;}
i. }

34. The following example demonstrates the basic way of using reflection to get access to custom attributes.

```
35. class MainClass
36. {
37.     public static void Main()
38.     {
39.         System.Reflection.MemberInfo info = typeof(MyClass);
40.         object[] attributes = info.GetCustomAttributes();
41.         for (int i = 0; i < attributes.Length; i ++)
42.         {
43.             System.Console.WriteLine(attributes[i]);
44.         }
45.     }
}
```

2. **What is the managed and unmanaged code in .net?**
The .NET Framework provides a run-time environment called the Common Language Runtime, which manages the execution of code and provides services that make the development process easier. Compilers and tools expose the runtime's functionality and enable you to write code that benefits from this managed execution environment. Code that you develop with a language compiler that targets the runtime is called *managed code*; it benefits from features such as cross-language integration, cross-language exception handling, enhanced security, versioning and deployment support, a simplified model for component interaction, and debugging and profiling services.
3. **How do you create threading in .NET? What is the namespace for that?**
**
System.Threading.Thread
4. **Serialize and MarshalByRef?**
5. **using** directive vs **using** statement
You create an instance in a **using** statement to ensure that **Dispose** is called on the object when the **using** statement is exited. A **using** statement can be exited either when the end of the **using** statement is reached or if, for example, an exception is thrown and control leaves the statement block before the end of the statement.
The **using** directive has two uses:

Create an alias for a namespace (a **using** alias).

Permit the use of types in a namespace, such that, you do not have to qualify the use of a type in that namespace (a **using** directive).

32. **Describe the Managed Execution Process?**
The managed execution process includes the following steps:

3. Choosing a compiler.
   To obtain the benefits provided by the common language runtime, you must use one or more language compilers that target the runtime.
4. Compiling your code to Microsoft intermediate language (MSIL).
   Compiling translates your source code into MSIL and generates the required metadata.
5. Compiling MSIL to native code.
   At execution time, a just-in-time (JIT) compiler translates the MSIL into native code. During this compilation, code must pass a verification process that examines the MSIL and metadata to find out whether the code can be determined to be type safe.
6. Executing your code.
   The common language runtime provides the infrastructure that enables execution to take place as well as a variety of services that can be used during execution.

46. **What is Active Directory? What is the namespace used to access the Microsoft Active Directories? What are ADSI Directories?**
    Active Directory Service Interfaces (ADSI) is a programmatic interface for Microsoft Windows Active Directory. It enables your applications to interact with diverse directories on a network, using a single interface. Visual Studio .NET and the .NET Framework make it easy to add ADSI functionality with the **DirectoryEntry** and **DirectorySearcher** components.
    Using ADSI, you can create applications that perform common administrative tasks, such as backing up databases, accessing printers, and administering user accounts. ADSI makes it possible for you to:

    Log on once to work with diverse directories. The **DirectoryEntry** component class provides username and password properties that can be entered at runtime and communicated to the Active Directory object you are binding to.
    Use a single application programming interface (API) to perform tasks on multiple directory systems by offering the user a variety of protocols to use. The **DirectoryServices** namespace provides the classes to perform most administrative functions.
    Perform "rich querying" on directory systems. ADSI technology allows for searching for an object by specifying two query dialects: SQL and LDAP.
    Access and use a single, hierarchical structure for administering and maintaining diverse and complicated network configurations by accessing an Active Directory tree.
    Integrate directory information with databases such as SQL Server. The **DirectoryEntry** path may be used as an ADO.NET connection string provided that it is using the LDAP provider.

```
using System.DirectoryServices;
```

33. **How Garbage Collector (GC) Works?**
    The methods in this class influence when an object is garbage collected and when resources allocated by an object are released. Properties in this class provide information about the total amount of memory available in the system and the age category, or generation, of memory allocated to an object. Periodically, the garbage collector performs garbage collection to reclaim memory allocated to objects for which there are no valid references. Garbage collection happens automatically when a request for memory cannot be satisfied using available free memory. Alternatively, an application can force garbage collection using the Collect method.
    Garbage collection consists of the following steps:

7. The garbage collector searches for managed objects that are referenced in managed code.
8. The garbage collector attempts to finalize objects that are not referenced.
9. The garbage collector frees objects that are not referenced and reclaims their memory.

47. **Why do we need to call CG.SupressFinalize?**
    Requests that the system not call the finalizer method for the specified object.
    public static void **SuppressFinalize(**
    object *obj*
    ); The method removes *obj* from the set of objects that require finalization.
    The *obj* parameter is required to be the caller of this method.
    Objects that implement the IDisposable interface can call this method from the IDisposable.Dispose method to prevent the garbage collector from calling Object.Finalize on an object that does not require it.
48. **What is nmake tool?**
    The Nmake tool (Nmake.exe) is a 32-bit tool that you use to build projects based on commands contained in a .mak file.
    usage : **nmake -a all**
49. **What are Namespaces?**
    The **namespace** keyword is used to declare a scope. This namespace scope lets you organize code and gives you a way to create globally-unique types. Even if you do not explicitly declare one, a default namespace is created. This unnamed namespace, sometimes called the global namespace, is present in every file. Any identifier in the global namespace is available for use in a named namespace. Namespaces implicitly have public access and this is not modifiable.
50. **C++ & C# differences**
    **
51. **If you want to write your own dot net language, what steps you will u take care?**
52. **What is custom events? How to create it?**
53. **how dot net compiled code will become platform independent?**
54. **without modifying source code if we compile again, will it be generated MSIL again?**
55. **Describe the difference between inline and code behind - which is best in a loosely coupled solution?**
56. **What is the difference between CONST and READONLY?**
57. **What is the difference between ref & out parameters?**
    What is the difference between arrays and Arraylist?
58. **What are indexers?**
59. **How to create events for a control?**
60. **Explain about SOAP**
61. **Practical Example of Passing an Events to delegates**
62. **How can you read 3rd line from a text file?**
63. **What is Asynchronous call and how it can be implemented using delegates?**
64. **What is the difference between Array and LinkedList?**
65. **What is Jagged Arrays?**
    A jagged array is an array whose elements are arrays. The elements of a jagged array can be of different dimensions and sizes. A jagged array is sometimes called an "array-of-arrays."

(COM)

66. **Interop Services?**
    The common language runtime provides two mechanisms for interoperating with unmanaged code:

    Platform invoke, which enables managed code to call functions exported from an unmanaged library.
    COM interop, which enables managed code to interact with COM objects through interfaces.

Both platform invoke and COM interop use interop marshaling to accurately move method arguments between caller and callee and back, if required.

34. **How does u handle this COM components developed in other programming languages in .NET?**
35. **What is RCW (Runtime Callable Wrappers)?**
    The common language runtime exposes COM objects through a proxy called the runtime callable wrapper (RCW). Although the RCW appears to be an ordinary object to .NET clients, its primary function is to marshal calls between a .NET client and a COM object.
36. **What is CCW (COM Callable Wrapper)**

A proxy object generated by the common language runtime so that existing COM applications can use managed classes, including .NET Framework classes, transparently.

37. **How CCW and RCW is working?**
    **
38. **How will you register com+ services?**
    The .NET Framework SDK provides the .NET Framework Services Installation Tool (Regsvcs.exe - a command-line tool) to manually register an assembly containing serviced components. You can also access these registration features programmatically with the System.EnterpriseServicesRegistrationHelper class by creating an instance of class RegistrationHelper and using the method InstallAssembly
39. **What is use of ContextUtil class?**
    ContextUtil is the preferred class to use for obtaining COM+ context information.
40. **What is the new three features of COM+ services, which are not there in COM (MTS)?**
    **
41. **Is the COM architecture same as .Net architecture? What is the difference between them?**
    **
42. **Can we copy a COM dll to GAC folder?**
    **
43. **What is Pinvoke?**
    Platform invoke is a service that enables managed code to call unmanaged functions implemented in dynamic-link libraries (DLLs), such as those in the Win32 API. It locates and invokes an exported function and marshals its arguments (integers, strings, arrays, structures, and so on) across the interoperation boundary as needed.
44. **Is it true that COM objects no longer need to be registered on the server?**
    Answer: Yes and No. Legacy COM objects still need to be registered on the server before they can be used. COM developed using the new .NET

Framework will not need to be registered. Developers will be able to auto-register these objects just by placing them in the 'bin' folder of the application.

45. **Can .NET Framework components use the features of Component Services?**
    Answer: Yes, you can use the features and functions of Component Services from a .NET Framework component.
    http://msdn.microsoft.com/library/techart/Pahlcompserv.htm

(OOPS)

46. **What are the OOPS concepts?**
    1) Encapsulation: It is the mechanism that binds together code and data in manipulates, and keeps both safe from outside interference and misuse. In short it isolates a particular code and data from all other codes and data. A well-defined interface controls the access to that particular code and data.
    2) Inheritance: It is the process by which one object acquires the properties of another object. This supports the hierarchical classification. Without the use of hierarchies, each object would need to define all its characteristics explicitly. However, by use of inheritance, an object need only define those qualities that make it unique within its class. It can inherit its general attributes from its parent. A new sub-class inherits all of the attributes of all of its ancestors.
    3) Polymorphism: It is a feature that allows one interface to be used for general class of actions. The specific action is determined by the exact nature of the situation. In general polymorphism means "one interface, multiple methods", This means that it is possible to design a generic interface to a group of related activities. This helps reduce complexity by allowing the same interface to be used to specify a general class of action. It is the compiler's job to select the specific action (that is, method) as it applies to each situation.

47. **What is the difference between a Struct and a Class?**
    The struct type is suitable for representing lightweight objects such as Point, Rectangle, and Color. Although it is possible to represent a point as a class, a struct is more efficient in some scenarios. For example, if you declare an array of 1000 Point objects, you will allocate additional memory for referencing each object. In this case, the struct is less expensive.
    When you create a struct object using the new operator, it gets created and the appropriate constructor is called. Unlike classes, structs can be instantiated without using the new operator. If you do not use new, the fields will remain unassigned and the object cannot be used until all of the fields are initialized.
    It is an error to declare a default (parameterless) constructor for a struct. A default constructor is always provided to initialize the struct members to their default values.
    It is an error to initialize an instance field in a struct.
    There is no inheritance for structs as there is for classes. A struct cannot inherit from another struct or class, and it cannot be the base of a class. Structs, however, inherit from the base class Object. A struct can implement interfaces, and it does that exactly as classes do.
    A struct is a value type, while a class is a reference type.

48. **Value type & reference types difference? Example from .NET. Integer & struct are value types or reference types in .NET?**
    Most programming languages provide built-in data types, such as integers and floating-point numbers, that are copied when they are passed as arguments (that is, they are passed by value). In the .NET Framework, these are called value types. The runtime supports two kinds of value types:
    Built-in value types
    The .NET Framework defines built-in value types, such as System.Int32 and System.Boolean, which correspond and are identical to primitive data types used by programming languages.

User-defined value types
Your language will provide ways to define your own value types, which derive from System.ValueType. If you want to define a type representing a value that is small, such as a complex number (using two floating-point numbers), you might choose to define it as a value type because you can pass the value type efficiently by value. If the type you are defining would be more efficiently passed by reference, you should define it as a class instead.

Variables of reference types, referred to as objects, store references to the actual data. This following are the reference types:

class
interface
delegate

This following are the built-in reference types:

object
string

49. **What is Inheritance, Multiple Inheritance, Shared and Repeatable Inheritance?**
    **
50. **What is Method overloading?**
    Method overloading occurs when a class contains two methods with the same name, but different signatures.
51. **What is Method Overriding? How to override a function in C#?**
    Use the override modifier to modify a method, a property, an indexer, or an event. An override method provides a new implementation of a member inherited from a base class. The method overridden by an override declaration is known as the overridden base method. The overridden base method must have the same signature as the override method.
    You cannot override a non-virtual or static method. The overridden base method must be virtual, abstract, or override.
52. **Can we call a base class method without creating instance?**
    Its possible If its a static method.
    Its possible by inheriting from that class also.
    Its possible from derived classes using base keyword.
53. **You have one base class virtual function how will call that function from derived class?**
    Ans:
54. class a
55. {
56. public virtual int m()
57. {
58. return 1;
59. }
60. }
61. class b:a
62. {
63. public int j()
64. {
65. return m();
66. }

}

67. **In which cases you use override and new base?**
    Use the new modifier to explicitly hide a member inherited from a base class. To hide an inherited member, declare it in the derived class using the same name, and modify it with the new modifier.

C# Language features

68. **What are Sealed Classes in C#?**
    The sealed modifier is used to prevent derivation from a class. A compile-time error occurs if a sealed class is specified as the base class of another class. (A sealed class cannot also be an abstract class)
69. **What is Polymorphism? How does VB.NET/C# achieve polymorphism?**
    **

```
class Token
        {
            public string Display()
            {
                //Implementation goes here
                return "base";
            }
        }
        class IdentifierToken:Token
        {
            public new string Display() //What is the use of new keyword
            {
                //Implementation goes here
                return "derive";
            }
        }
        static void Method(Token t)
        {
            Console.Write(t.Display());
        }
        public static void Main()
        {
            IdentifierToken Variable=new IdentifierToken();
            Method(Variable); //Which Class Method is called here
            Console.ReadLine();
        }
        For the above code What is the "new" keyword and Which Class Method is
          called here
```

A: it will call base class Display method

```
class Token
        {
            public virtual string Display()
            {
                //Implementation goes here
                return "base";
            }
        }
```

```
class IdentifierToken:Token
{
    public override string Display() //What is the use of new keyword
    {
        //Implementation goes here
        return "derive";
    }
}
static void Method(Token t)
{
    Console.Write(t.Display());
}
public static void Main()
{
    IdentifierToken Variable=new IdentifierToken();
    Method(Variable); //Which Class Method is called here
    Console.ReadLine();
}
```

A: Derive

In which Scenario you will go for Interface or Abstract Class?

Interfaces, like classes, define a set of properties, methods, and events. But unlike classes, interfaces do not provide implementation. They are implemented by classes, and defined as separate entities from classes. Even though class inheritance allows your classes to inherit implementation from a base class, it also forces you to make most of your design decisions when the class is first published.

Abstract classes are useful when creating components because they allow you specify an invariant level of functionality in some methods, but leave the implementation of other methods until a specific implementation of that class is needed. They also version well, because if additional functionality is needed in derived classes, it can be added to the base class without breaking code.

Interfaces vs. Abstract Classes

Feature	Interface	Abstract class
Multiple inheritance	A class may implement several interfaces.	A class may extend only one abstract class.
Default implementation	An interface cannot provide any code at all, much less default code.	An abstract class can provide complete code, default code, and/or just stubs that have to be overridden.
Constants	Static final constants only, can use them without qualification in classes that implement the interface. On the other paw, these unqualified names pollute the namespace. You can use them and it is not obvious where they are coming from since the qualification is optional.	Both instance and static constants are possible. Both static and instance intialiser code are also possible to compute the constants.
Third party	An interface	A third party class must be

convenience	implementation may be added to any existing third party class.	rewritten to extend only from the abstract class.
is-a vs -able or can-do	Interfaces are often used to describe the peripheral abilities of a class, not its central identity, e.g. an Automobile class might implement the Recyclable interface, which could apply to many otherwise totally unrelated objects.	An abstract class defines the core identity of its descendants. If you defined a Dog abstract class then Damamation descendants are Dogs, they are not merely dogable. Implemented interfaces enumerate the general things a class can do, not the things a class is.
Plug-in	You can write a new replacement module for an interface that contains not one stick of code in common with the existing implementations. When you implement the interface, you start from scratch without any default implementation. You have to obtain your tools from other classes; nothing comes with the interface other than a few constants. This gives you freedom to implement a radically different internal design.	You must use the abstract class as-is for the code base, with all its attendant baggage, good or bad. The abstract class author has imposed structure on you. Depending on the cleverness of the author of the abstract class, this may be good or bad. Another issue that's important is what I call "heterogeneous vs. homogeneous." If implementors/subclasses are homogeneous, tend towards an abstract base class. If they are heterogeneous, use an interface. (Now all I have to do is come up with a good definition of hetero/homogeneous in this context.) If the various objects are all of-a-kind, and share a common state and behavior, then tend towards a common base class. If all they share is a set of method signatures, then tend towards an interface.
Homogeneity	If all the various implementations share is the method signatures, then an interface works best.	If the various implementations are all of a kind and share a common status and behavior, usually an abstract class works best.
Maintenance	If your client code talks only in terms of an interface, you can easily change the concrete implementation behind it, using a factory method.	Just like an interface, if your client code talks only in terms of an abstract class, you can easily change the concrete implementation behind it, using a factory method.
Speed	Slow, requires extra indirection to find the corresponding method in the actual class. Modern JVMs are discovering ways to reduce this speed penalty.	Fast
Terseness	The constant declarations	You can put shared code into an

	in an interface are all presumed public static final, so you may leave that part out. You can't call any methods to compute the initial values of your constants. You need not declare individual methods of an interface abstract. They are all presumed so.	abstract class, where you cannot into an interface. If interfaces want to share code, you will have to write other bubblegum to arrange that. You may use methods to compute the initial values of your constants and variables, both instance and static. You must declare all the individual methods of an abstract class abstract.
Adding functionality	If you add a new method to an interface, you must track down all implementations of that interface in the universe and provide them with a concrete implementation of that method.	If you add a new method to an abstract class, you have the option of providing a default implementation of it. Then all existing code will continue to work without change.

126. see the code

```
127. interface ICommon
128.         {
129.             int getCommon();
130.         }
131.         interface ICommonImplements1:ICommon
132.         {
133.         }
134.         interface ICommonImplements2:ICommon
135.         {
136.         }
137.         public class a:ICommonImplements1,ICommonImplements2
138.         {
    }
```

How to implement getCommon method in class a? Are you seeing any problem in the implementation?
Ans:

```
public class a:ICommonImplements1,ICommonImplements2
        {
           public int getCommon()
           {
                    return 1;
           }
        }
```

```
139. interface IWeather
140.         {
141.             void display();
142.         }
143.         public class A:IWeather
144.         {
145.             public void display()
146.             {
```

```
                    MessageBox.Show("A");
                }
            }
            public class B:A
            {
            }
            public class C:B,IWeather
            {
                public void display()
                {
                    MessageBox.Show("C");
                }
            }
```

When I instantiate C.display(), will it work?

```
interface IPrint
            {
                string Display();
            }
            interface IWrite
            {
                string Display();
            }
            class PrintDoc:IPrint,IWrite
            {
                //Here is implementation
            }
```

how to implement the Display in the class printDoc (How to resolve the naming Conflict) A: no naming conflicts

```
class PrintDoc:IPrint,IWrite
        {
           public string Display()
           {
                    return "s";
           }
        }
interface IList
            {
              int Count { get; set; }
            }
            interface ICounter
            {
              void Count(int i);
            }
            interface IListCounter: IList, ICounter {}
            class C
            {
              void Test(IListCounter x)
              {
```

```
            x.Count(1);              // Error
            x.Count = 1;             // Error
            ((IList)x).Count = 1;    // Ok, invokes IList.Count.set
            ((ICounter)x).Count(1);  // Ok, invokes ICounter.Count
        }
    }
```

192. **Write one code example for compile time binding and one for run time binding? What is early/late binding?**
An object is *early bound* when it is assigned to a variable declared to be of a specific object type. Early bound objects allow the compiler to allocate memory and perform other optimizations before an application executes.
' Create a variable to hold a new object.
Dim FS As FileStream
' Assign a new object to the variable.
FS = New FileStream("C:\tmp.txt", FileMode.Open)
By contrast, an object is *late bound* when it is assigned to a variable declared to be of type **Object**. Objects of this type can hold references to any object, but lack many of the advantages of early-bound objects.
Dim xlApp As Object
xlApp = CreateObject("Excel.Application")

193. **Can you explain what inheritance is and an example of when you might use it?**

194. **How can you write a class to restrict that only one object of this class can be created (Singleton class)?**

(Access specifiers)

195. **What are the access-specifiers available in c#?**
Private, Protected, Public, Internal, Protected Internal.

196. **Explain about Protected and protected internal, "internal" access-specifier?**
protected - Access is limited to the containing class or types derived from the containing class.
internal - Access is limited to the current assembly.
protected internal - Access is limited to the current assembly or types derived from the containing class.

(Constructor / Destructor)

197. **Difference between type constructor and instance constructor? What is static constructor, when it will be fired? And what is its use?**
(Class constructor method is also known as type constructor or type initializer) Instance constructor is executed when a new instance of type is created and the class constructor is executed after the type is loaded and before any one of the type members is accessed. (It will get executed only 1st time, when we call any static methods/fields in the same class.) Class constructors are used for static field initialization. Only one class constructor per type is permitted, and it cannot use the vararg (variable argument) calling convention.
A static constructor is used to initialize a class. It is called automatically to initialize the class before the first instance is created or any static members are referenced.

198. **What is Private Constructor? and it's use? Can you create instance of a class which has Private Constructor?**
A: When a class declares only private instance constructors, it is not possible for classes outside the program to derive from the class or to directly create instances of it. (Except Nested classes)
Make a constructor private if:

- You want it to be available only to the class itself. For example, you might have a special constructor used only in the implementation of your class' Clone method.
- You do not want instances of your component to be created. For example, you may have a class containing nothing but Shared utility functions, and no instance data. Creating instances of the class would waste memory.

199. **I have 3 overloaded constructors in my class. In order to avoid making instance of the class do I need to make all constructors to private?**
(yes)

200. **Overloaded constructor will call default constructor internally?**
(no)

201. **What are virtual destructors?**

202. **Destructor and finalize**
Generally in C++ the destructor is called when objects gets destroyed. And one can explicitly call the destructors in C++. And also the objects are destroyed in reverse order that they are created in. So in C++ you have control over the destructors.
In C# you can never call them, the reason is one cannot destroy an object. So who has the control over the destructor (in C#)? it's the .Net frameworks Garbage Collector (GC). GC destroys the objects only when necessary. Some situations of necessity are memory is exhausted or user explicitly calls System.GC.Collect() method.
Points to remember:
1. Destructors are invoked automatically, and cannot be invoked explicitly.
2. Destructors cannot be overloaded. Thus, a class can have, at most, one destructor.
3. Destructors are not inherited. Thus, a class has no destructors other than the one, which may be declared in it.
4. Destructors cannot be used with structs. They are only used with classes.
5. An instance becomes eligible for destruction when it is no longer possible for any code to use the instance.
6. Execution of the destructor for the instance may occur at any time after the instance becomes eligible for destruction.
7. When an instance is destructed, the destructors in its inheritance chain are called, in order, from most derived to least derived.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconfinalizemethodscdestructors.asp

203. **What is the difference between Finalize and Dispose (Garbage collection)**
Class instances often encapsulate control over resources that are not managed by the runtime, such as window handles (HWND), database connections, and so on. Therefore, you should provide both an explicit and an implicit way to free those resources. Provide implicit control by implementing the protected Finalize Method on an object (destructor syntax in C# and the Managed Extensions for C++). The garbage collector calls this method at some point after there are no longer any valid references to the object.
In some cases, you might want to provide programmers using an object with the ability to explicitly release these external resources before the garbage collector frees the object. If an external resource is scarce or expensive, better performance can be achieved if the programmer explicitly releases resources when they are no longer being used. To provide explicit control, implement the Dispose method provided by the IDisposable Interface. The consumer of the object should call this method when it is done using the object. **Dispose** can be called even if other references to the object are alive.
Note that even when you provide explicit control by way of **Dispose**, you should provide implicit cleanup using the **Finalize** method. **Finalize** provides a backup to prevent resources from permanently leaking if the programmer fails to call **Dispose**.

204. **What is close method? How its different from Finalize & Dispose?**
     **
205. **What is boxing & unboxing?**
206. **What is check/uncheck?**
207. **What is the use of base keyword? Tell me a practical example for base keyword's usage?**
208. What are the different .net tools which u used in projects?
209. try
     {
     ...
     }
     catch
     {
     ...//exception occurred here. What'll happen?
     }
     finally
     {
     ..
     }
     Ans : It will throw exception.
210. What will do to avoid prior case?
     Ans:
211. try
212. {
213. try
214. {
215. ....
216. }
217. catch
218. {
219. ....
220. //exception occurred here.
221. }
222. finally
223. {
224. ....
225. }
226. }
227. catch
228. {
229. ....
230. }
231. finally
232. {
233. ....
}
234. try
235. {
236. ....
237. }
238. catch
239. {

240. ...
241. }
242. finally
243. {
244. ...
245. }
246. Will it go to finally block if there is no exception happened?
Ans: Yes. The **finally** block is useful for cleaning up any resources allocated in the try block. Control is always passed to the finally block regardless of how the try block exits.
247. **Is goto statement supported in C#? How about Java?**
Gotos are supported in C#to the fullest. In Java goto is a reserved keyword that provides absolutely no functionality.
248. **What's different about switch statements in C#?**
No fall-throughs allowed. Unlike the C++ **switch** statement, C# does not support an explicit fall through from one case label to another. If you want, you can use **goto** a switch-case, or **goto default**.

```
case 1:
cost += 25;
break;
case 2:
cost += 25;
goto case 1;
```

(ADO.NET)
249. **Advantage of ADO.Net?**
ADO.NET Does Not Depend On Continuously Live Connections
Database Interactions Are Performed Using Data Commands
Data Can Be Cached in Datasets
Datasets Are Independent of Data Sources
Data Is Persisted as XML
Schemas Define Data Structures
250. **How would u connect to database using .NET?**

```
SqlConnection nwindConn = new SqlConnection("Data Source=localhost;
Integrated Security=SSPI;" +
                                    "Initial Catalog=northwind");
nwindConn.Open();
```

251. **What are relation objects in dataset and how & where to use them?**
In a **DataSet** that contains multiple **DataTable** objects, you can use **DataRelation** objects to relate one table to another, to navigate through the tables, and to return child or parent rows from a related table. Adding a **DataRelation** to a **DataSet** adds, by default, a **UniqueConstraint** to the parent table and a **ForeignKeyConstraint** to the child table.
The following code example creates a **DataRelation** using two **DataTable** objects in a **DataSet**. Each **DataTable** contains a column named **CustID**, which serves as a link between the two **DataTable** objects. The example adds a single **DataRelation** to the **Relations** collection of the **DataSet**. The first argument in the example specifies the name of the **DataRelation** being created. The second argument sets the parent **DataColumn** and the third argument sets the child **DataColumn**.

```
custDS.Relations.Add("CustOrders",
custDS.Tables["Customers"].Columns["CustID"],
custDS.Tables["Orders"].Columns["CustID"]);
```

OR

```
private void CreateRelation()
{
// Get the DataColumn objects from two DataTable objects in a DataSet.
DataColumn parentCol;
DataColumn childCol;
// Code to get the DataSet not shown here.
parentCol = DataSet1.Tables["Customers"].Columns["CustID"];
childCol = DataSet1.Tables["Orders"].Columns["CustID"];
// Create DataRelation.
DataRelation relCustOrder;
relCustOrder = new DataRelation("CustomersOrders", parentCol, childCol);
// Add the relation to the DataSet.
DataSet1.Relations.Add(relCustOrder);
}
```

252. **Difference between OLEDB Provider and SqlClient ?**

Ans: SQLClient .NET classes are highly optimized for the .net / sqlserver combination and achieve optimal results. The SqlClient data provider is fast. It's faster than the Oracle provider, and faster than accessing database via the OleDb layer. It's faster because it accesses the native library (which automatically gives you better performance), and it was written with lots of help from the SQL Server team.

253. **What are the different namespaces used in the project to connect the database? What data providers available in .net to connect to database?**

System.Data.OleDb – classes that make up the .NET Framework Data Provider for OLE DB-compatible data sources. These classes allow you to connect to an OLE DB data source, execute commands against the source, and read the results.

System.Data.SqlClient – classes that make up the .NET Framework Data Provider for SQL Server, which allows you to connect to SQL Server 7.0, execute commands, and read results. The **System.Data.SqlClient** namespace is similar to the **System.Data.OleDb** namespace, but is optimized for access to SQL Server 7.0 and later.

System.Data.Odbc - classes that make up the .NET Framework Data Provider for ODBC. These classes allow you to access ODBC data source in the managed space.

System.Data.OracleClient - classes that make up the .NET Framework Data Provider for Oracle. These classes allow you to access an Oracle data source in the managed space.

254. **Difference between DataReader and DataAdapter / DataSet and DataAdapter?**

You can use the ADO.NET DataReader to retrieve a read-only, forward-only stream of data from a database. Using the DataReader can increase application performance and reduce system overhead because only one row at a time is ever in memory.

After creating an instance of the **Command** object, you create a **DataReader** by calling **Command.ExecuteReader** to retrieve rows from a data source, as shown in the following example.

```
SqlDataReader myReader = myCommand.ExecuteReader();
```

You use the **Read** method of the **DataReader** object to obtain a row from the results of the query.

```
while (myReader.Read())
  Console.WriteLine("\t{0}\t{1}", myReader.GetInt32(0),
myReader.GetString(1));
myReader.Close();
```

The DataSet is a memory-resident representation of data that provides a consistent relational programming model regardless of the data source. It can be used with multiple and differing data sources, used with XML data, or used

to manage data local to the application. The **DataSet** represents a complete set of data including related tables, constraints, and relationships among the tables. The methods and objects in a **DataSet** are consistent with those in the relational database model. The **DataSet** can also persist and reload its contents as XML and its schema as XML Schema definition language (XSD) schema.
The DataAdapter serves as a bridge between a DataSet and a data source for retrieving and saving data. The DataAdapter provides this bridge by mapping Fill, which changes the data in the DataSet to match the data in the data source, and Update, which changes the data in the data source to match the data in the DataSet. If you are connecting to a Microsoft SQL Server database, you can increase overall performance by using the SqlDataAdapter along with its associated SqlCommand and SqlConnection. For other OLE DB-supported databases, use the DataAdapter with its associated OleDbCommand and OleDbConnection objects.

255. **Which method do you invoke on the DataAdapter control to load your generated dataset with data?**
Fill()

256. **Explain different methods and Properties of DataReader which you have used in your project?**
Read
GetString
GetInt32
while (myReader.Read())
Console.WriteLine("\t{0}\t{1}", myReader.GetInt32(0),
myReader.GetString(1));
myReader.Close();

257. **What happens when we issue Dataset.ReadXml command?**
Reads XML schema and data into the DataSet.

258. **In how many ways we can retrieve table records count? How to find the count of records in a dataset?**
foreach(DataTable thisTable in myDataSet.Tables){
// For each row, print the values of each column.
foreach(DataRow myRow in thisTable.Rows){

259. **How to check if a datareader is closed or opened?**
IsClosed()

260. **What happens when u try to update data in a dataset in .NET while the record is already deleted in SQL SERVER as backend?**
OR What is concurrency? How will you avoid concurrency when dealing with dataset? (One user deleted one row after that another user through his dataset was trying to update same row. What will happen? How will you avoid the problem?)
**

261. **How do you merge 2 datasets into the third dataset in a simple manner? OR If you are executing these statements in commandObject. "Select * from Table1;Select * from Table2" how you will deal result set?**
**

262. **How do you sort a dataset?**
**

263. **If a dataset contains 100 rows, how to fetch rows between 5 and 15 only?**
**

264. **Differences between dataset.clone and dataset.copy?**
Clone - Copies the structure of the DataSet, including all DataTable schemas, relations, and constraints. Does not copy any data.
Copy - Copies both the structure and data for this DataSet.

265. **What is the use of parameter object?**
**

266. **How to generate XML from a dataset and vice versa?**
**

267. **What is method to get XML and schema from Dataset?**
ans: getXML () and get Schema ()

268. **How do u implement locking concept for dataset?**
**

(ASP.NET)

269. **Asp.net and asp – differences?**

Code Render Block	Code Declaration Block
	Compiled
Request/Response	Event Driven
	Object Oriented - Constructors/Destructors, Inheritance, overloading..
	Exception Handling - Try, Catch, Finally
	Down-level Support
	Cultures
	User Controls
	In-built client side validation
Session - weren't transferable across servers	It can span across servers, It can survive server crashes, can work with browsers that don't support cookies
built on top of the window & IIS, it was always a separate entity & its functionality was limited.	its an integral part of OS under the .net framework. It shares many of the same objects that traditional applications would use, and all .net objects are available for asp.net's consumption.
	Garbage Collection
	Declare variable with datatype
	In built graphics support
	Cultures

270. **How ASP and ASP.NET page works? Explain about asp.net page life cycle?**
**

271. **Order of events in an asp.net page? Control Execution Lifecycle?**

Phase	What a control needs to do	Method or event to override
Initialize	Initialize settings needed during the lifetime of the incoming Web request.	**Init** event (**OnInit** method)
Load view state	At the end of this phase, the **ViewState** property of a control is automatically populated as described in Maintaining State in a Control. A control can override the	**LoadViewState** method

	default implementation of the **LoadViewState** method to customize state restoration.	
Process postback data	Process incoming form data and update properties accordingly.	**LoadPostData** method (if **IPostBackDataHandler** is implemented)
Load	Perform actions common to all requests, such as setting up a database query. At this point, server controls in the tree are created and initialized, the state is restored, and form controls reflect client-side data.	**Load** event (**OnLoad** method)
Send postback change notifications	Raise change events in response to state changes between the current and previous postbacks.	**RaisePostDataChangedEvent** method (if **IPostBackDataHandler** is implemented)
Handle postback events	Handle the client-side event that caused the postback and raise appropriate events on the server.	**RaisePostBackEvent** method(if **IPostBackEventHandler** is implemented)
Prerender	Perform any updates before the output is rendered. Any changes made to the state of the control in the prerender phase can be saved, while changes made in the rendering phase are lost.	**PreRender** event (**OnPreRender** method)
Save state	The **ViewState** property of a control is automatically persisted to a string object after this stage. This string object is sent to the client and back as a hidden variable. For improving efficiency, a control can override the **SaveViewState** method to modify the **ViewState** property.	**SaveViewState** method
Render	Generate output to be rendered to the client.	**Render** method
Dispose	Perform any final cleanup before the control is torn down. References to expensive resources such as database connections must be released in this phase.	**Dispose** method
Unload	Perform any final cleanup before the control is torn down. Control authors generally perform cleanup in **Dispose** and do not handle this event.	**UnLoad** event (**On UnLoad** method)

272. **Note** To override an *EventName* event, override the On*EventName* method (and call base. On*EventName*).

(Session/State)

273. **Application and Session Events**

The ASP.NET page framework provides ways for you to work with events that can be raised when your application starts or stops or when an individual user's session starts or stops:

Application events are raised for all requests to an application. For example, **Application_BeginRequest** is raised when any Web Forms page or XML Web service in your application is requested. This event allows you to initialize resources that will be used for each request to the application. A corresponding event, **Application_EndRequest**, provides you with an opportunity to close or otherwise dispose of resources used for the request.

Session events are similar to application events (there is a **Session_OnStart** and a **Session_OnEnd** event), but are raised with each unique session within the application. A session begins when a user requests a page for the first time from your application and ends either when your application explicitly closes the session or when the session times out.

You can create handlers for these types of events in the Global.asax file.

274. **Difference between ASP Session and ASP.NET Session?**

asp.net session supports cookie less session & it can span across multiple servers.

275. **What is cookie less session? How it works?**

By default, ASP.NET will store the session state in the same process that processes the request, just as ASP does. If cookies are not available, a session can be tracked by adding a session identifier to the URL. This can be enabled by setting the following:
<sessionState cookieless="true" />
http://samples.gotdotnet.com/quickstart/aspplus/doc/stateoverview.aspx

276. **How you will handle session when deploying application in more than a server? Describe session handling in a webfarm, how does it work and what are the limits?**

By default, ASP.NET will store the session state in the same process that processes the request, just as ASP does. Additionally, ASP.NET can store session data in an external process, which can even reside on another machine. To enable this feature:
Start the ASP.NET state service, either using the Services snap-in or by executing "net start aspnet_state" on the command line. The state service will by default listen on port 42424. To change the port, modify the registry key for the service:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\aspnet_state\Parameters\Port

Set the **mode** attribute of the **<sessionState>** section to "StateServer".

Configure the **stateConnectionString** attribute with the values of the machine on which you started aspnet_state.

The following sample assumes that the state service is running on the same machine as the Web server ("localhost") and uses the default port (42424):

```
<sessionState
            mode="StateServer"
            stateConnectionString="tcpip=localhost:42424"
          />
```

Note that if you try the sample above with this setting, you can reset the Web server (enter iisreset on the command line) and the session state value will persist. **

277. **What method do you use to explicitly kill a users session?**
Abandon()
278. **What are the different ways you would consider sending data across pages in ASP (i.e between 1.asp to 2.asp)?**
Session
public properties
279. **What is State Management in .Net and how many ways are there to maintain a state in .Net? What is view state?**
Web pages are recreated each time the page is posted to the server. In traditional Web programming, this would ordinarily mean that all information associated with the page and the controls on the page would be lost with each round trip.
To overcome this inherent limitation of traditional Web programming, the ASP.NET page framework includes various options to help you preserve changes — that is, for managing state. The page framework includes a facility called view state that automatically preserves property values of the page and all the controls on it between round trips.
However, you will probably also have application-specific values that you want to preserve. To do so, you can use one of the state management options.
Client-Based State Management Options:
View State
Hidden Form Fields
Cookies
Query Strings
Server-Based State Management Options
Application State
Session State
Database Support
280. **What are the disadvantages of view state / what are the benefits?**
Automatic view-state management is a feature of server controls that enables them to repopulate their property values on a round trip (without you having to write any code). This feature does impact performance, however, since a server control's view state is passed to and from the server in a hidden form field. You should be aware of when view state helps you and when it hinders your page's performance.
281. **When maintaining session through Sql server, what is the impact of Read and Write operation on Session objects? will performance degrade?**
Maintaining state using database technology is a common practice when storing user-specific information where the information store is large. Database storage is particularly useful for maintaining long-term state or state that must be preserved even if the server must be restarted.
**
282. **What are the contents of cookie?**
**
283. **How do you create a permanent cookie?**
**
284. **What is ViewState? What does the "EnableViewState" property do? Why would I want it on or off?**
**
285. **Explain the differences between Server-side and Client-side code?**
Server side code will process at server side & it will send the result to client. Client side code (javascript) will execute only at client side.
286. **Can you give an example of what might be best suited to place in the Application_Start and Session_Start subroutines?**
**

287. **Which ASP.NET configuration options are supported in the ASP.NET implementation on the shared web hosting platform?**
A: Many of the ASP.NET configuration options are not configurable at the site, application or subdirectory level on the shared hosting platform. Certain options can affect the security, performance and stability of the server and, therefore cannot be changed. The following settings are the only ones that can be changed in your site's web.config file (s):
browserCaps
clientTarget
pages
customErrors
globalization
authorization
authentication
webControls
webServices
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconaspnetconfiguration.asp
288. **Briefly describe the role of global.asax?**
289. **How can u debug your .net application?**
290. **How do u deploy your asp.net application?**
291. **Where do we store our connection string in asp.net application?**
292. **Various steps taken to optimize a web based application (caching, stored procedure etc.)**
293. **How does ASP.NET framework maps client side events to Server side events.**

(Security)
294. **Security types in ASP/ASP.NET? Different Authentication modes?**
295. **How .Net has implemented security for web applications?**
296. **How to do Forms authentication in asp.net?**
297. **Explain authentication levels in .net ?**
298. **Explain autherization levels in .net ?**
299. **What is Role-Based security?**
A role is a named set of principals that have the same privileges with respect to security (such as a teller or a manager). A principal can be a member of one or more roles. Therefore, applications can use role membership to determine whether a principal is authorized to perform a requested action.
**
300. **How will you do windows authentication and what is the namespace? If a user is logged under integrated windows authentication mode, but he is still not able to logon, what might be the possible cause for this? In ASP.Net application how do you find the name of the logged in person under windows authentication?**
301. **What are the different authentication modes in the .NET environment?**
302. <authentication mode="*Windows*|*Forms*|*Passport*|*None*">
303. <forms name="*name*"
304. loginUrl="*url*"
305. protection="*All*|*None*|*Encryption*|*Validation*"
306. timeout="*30*" *path*="*/*" >
307. requireSSL="*true*|*false*"
308. slidingExpiration="*true*|*false*">
309. <credentials passwordFormat="*Clear*|*SHA1*|*MD5*">
310. <user name="*username*" *password*="*password*"/>

311. </credentials>
312. </forms>
313. <passport redirectUrl="*internal*"/>
</authentication>

Attribute	Option	Description
mode		Controls the default authentication mode for an application.
	Windows	Specifies Windows authentication as the default authentication mode. Use this mode when using any form of Microsoft Internet Information Services (IIS) authentication: Basic, Digest, Integrated Windows authentication (NTLM/Kerberos), or certificates.
	Forms	Specifies ASP.NET forms-based authentication as the default authentication mode.
	Passport	Specifies Microsoft Passport authentication as the default authentication mode.
	None	Specifies no authentication. Only anonymous users are expected or applications can handle events to provide their own authentication.

314. **How do you specify whether your data should be passed as Query string and Forms (Mainly about POST and GET)**
Through attribute tag of form tag.
315. **What is the other method, other than GET and POST, in ASP.NET?**
316. **What are validator? Name the Validation controls in asp.net? How do u disable them? Will the asp.net validators run in server side or client side? How do you do Client-side validation in .Net? How to disable validator control by client side JavaScript?**
A set of server controls included with ASP.NET that test user input in HTML and Web server controls for programmer-defined requirements. Validation controls perform input checking in server code. If the user is working with a browser that supports DHTML, the validation controls can also perform validation ("EnableClientScript" property set to true/false) using client script.
The following validation controls are available in asp.net:
RequiredFieldValidator Control, CompareValidator Control, RangeValidator Control, RegularExpressionValidator Control, CustomValidator Control, ValidationSummary Control.
317. **Which two properties are there on every validation control?**
ControlToValidate, ErrorMessage
318. **How do you use css in asp.net?**
Within the <HEAD> section of an HTML document that will use these styles, add a link to this external CSS style sheet that
follows this form:
<LINK REL="STYLESHEET" TYPE="text/css" HREF="MyStyles.css">
MyStyles.css is the name of your external CSS style sheet.
319. **How do you implement postback with a text box? What is postback and usestate?**
Make AutoPostBack property to true
320. **How can you debug an ASP page, without touching the code?**
321. **What is SQL injection?**
An SQL injection attack "injects" or manipulates SQL code by adding unexpected SQL to a query.
Many web pages take parameters from web user, and make SQL query to the database. Take for instance when a user login, web page that user name and

password and make SQL query to the database to check if a user has valid name and password.
Username: ' or 1=1 ---
Password: [Empty]
This would execute the following query against the users table:
select count(*) from users where userName='' or 1=1 --' and userPass=''

322. **How can u handle Exceptions in Asp.Net?**
323. **How can u handle Un Managed Code Exceptions in ASP.Net?**
324. **Asp.net - How to find last error which occurred?**
A: **Server.GetLastError();**
[C#]
Exception LastError;
String ErrMessage;
LastError = **Server.GetLastError();**
if (LastError != null)
ErrMessage = LastError.Message;
else
ErrMessage = "No Errors";
Response.Write("Last Error = " + ErrMessage);
325. **How to do Caching in ASP?**
A: <%@ OutputCache Duration="60" VaryByParam="None" %>

VaryByParam value	Description
none	One version of page cached (only raw GET)
*****	*n* versions of page cached based on query string and/or POST body
v1	*n* versions of page cached based on value of *v1* variable in query string or POST body
v1;v2	*n* versions of page cached based on value of *v1* and *v2* variables in query string or POST body

326. <%@ OutputCache Duration="60" VaryByParam="none" %>
<%@ OutputCache Duration="60" VaryByParam="*" %>
<%@ OutputCache Duration="60" VaryByParam="name;age" %>
The **OutputCache** directive supports several other cache varying options
VaryByHeader - maintain separate cache entry for header string changes (**UserAgent**, **UserLanguage**, etc.)
VaryByControl - for user controls, maintain separate cache entry for properties of a user control
VaryByCustom - can specify separate cache entries for browser types and version *or* provide a custom **GetVaryByCustomString** method in **HttpApplication**derived class
327. **What is the Global ASA(X) File?**
328. **Any alternative to avoid name collisions other then Namespaces.**
A scenario that two namespaces named N1 and N2 are there both having the same class say A. now in another class i ve written
using N1;using N2;
and i am instantiating class A in this class. Then how will u avoid name collisions?
Ans: using alias
Eg: using MyAlias = MyCompany.Proj.Nested;
329. **Which is the namespace used to write error message in event Log File?**
330. **What are the page level transaction and class level transaction?**
331. **What are different transaction options?**

332. **What is the namespace for encryption?**
333. **What is the difference between application and cache variables?**
334. **What is the difference between control and component?**
335. **You ve defined one page_load event in aspx page and same page_load event in code behind how will prog run?**
336. **Where would you use an IHttpModule, and what are the limitations of any approach you might take in implementing one?**
337. Can you edit data in the Repeater control? Which template must you provide, in order to display data in a Repeater control? How can you provide an alternating color scheme in a Repeater control? What property must you set, and what method must you call in your code, in order to bind the data from some data source to the Repeater control?
338. **What is the use of web.config? Difference between machine.config and Web.config?**
    ASP.NET configuration files are XML-based text files--each named web.config--that can appear in any directory on an ASP.NET
    Web application server. Each web.config file applies configuration settings to the directory it is located in and to all
    virtual child directories beneath it. Settings in child directories can optionally override or modify settings specified in
    parent directories. The root configuration file--WinNT\Microsoft.NET\Framework\<version>\config\machine.config--provides default configuration settings for the entire machine. ASP.NET configures IIS to prevent direct browser access to web.config
    files to ensure that their values cannot become public (attempts to access them will cause ASP.NET to return 403: Access
    Forbidden).
    At run time ASP.NET uses these web.config configuration files to hierarchically compute a unique collection of settings for
    each incoming URL target request (these settings are calculated only once and then cached across subsequent requests; ASP.NET
    automatically watches for file changes and will invalidate the cache if any of the configuration files change).
    http://samples.gotdotnet.com/quickstart/aspplus/doc/configformat.aspx
339. **What is the use of sessionstate tag in the web.config file?**
    Configuring session state: Session state features can be configured via the **<sessionState>** section in a web.config file. To double the default timeout of 20 minutes, you can add the following to the web.config file of an application:
    <sessionState
    timeout="40"
    />
340. **What are the different modes for the sessionstates in the web.config file?**

Off	Indicates that session state is not enabled.
Inproc	Indicates that session state is stored locally.
StateServer	Indicates that session state is stored on a remote server.
SQLServer	Indicates that session state is stored on the SQL Server.

341. **What is smart navigation?**
    When a page is requested by an Internet Explorer 5 browser, or later, smart navigation enhances the user's experience of the page by performing the following:
    eliminating the flash caused by navigation.
    persisting the scroll position when moving from page to page.
    persisting element focus between navigations.
    retaining only the last page state in the browser's history.

Smart navigation is best used with ASP.NET pages that require frequent postbacks but with visual content that does not change dramatically on return. Consider this carefully when deciding whether to set this property to **true**.
Set the **SmartNavigation** attribute to **true** in the @ Page directive in the .aspx file. When the page is requested, the dynamically generated class sets this property.

342. **In what order do the events of an ASPX page execute. As a developer is it important to undertsand these events?**
343. **How would you get ASP.NET running in Apache web servers - why would you even do this?**
344. **What tags do you need to add within the asp:datagrid tags to bind columns manually**
345. **What base class do all Web Forms inherit from?**
     System.Web.UI.Page
346. **How can we create pie chart in asp.net?**
347. **Is it possible for me to change my aspx file extension to some other name?**
     Yes.
     Open IIS->Default Website -> Properties
     Select HomeDirectory tab
     Click on configuration button
     Click on add. Enter aspnet_isapi details
     (C:\WINDOWS\Microsoft.NET\Framework\v1.0.3705\aspnet_isapi.dll | GET,HEAD,POST,DEBUG)

     Open machine.config(C:\WINDOWS\Microsoft.NET\Framework\v1.0.3705\CONFIG) & add new extension under <httpHandlers> tag
     <add verb="*" path="*.santhosh" type="System.Web.UI.PageHandlerFactory"/>
348. **What is AutoEventWireup attribute for ?**

     (WEBSERVICE & REMOTING)
349. **What is a WebService and what is the underlying protocol used in it? Namespace?**
     Web Services are applications delivered as a service on the Web. Web services allow for programmatic access of business logic over the Web. Web services typically rely on XML-based protocols, messages, and interface descriptions for communication and access. Web services are designed to be used by other programs or applications rather than directly by end user. Programs invoking a Web service are called clients. SOAP over HTTP is the most commonly used protocol for invoking Web services.
350. **Why Web Services?**
     By exposing data and functionality using standard protocols, Web services make it easy to build sophisticated applications that integrate many features and content. There are three main uses of Web services. Application integration Web services within an intranet are commonly used to integrate business applications running on disparate platforms. For example, a .NET client running on Windows 2000 can easily invoke a Java Web service running on a mainframe or Unix machine to retrieve data from a legacy application. Business integration Web services allow trading partners to engage in e-business leveraging the existing Internet infrastructure. Organizations can send electronic purchase orders to suppliers and receive electronic invoices. Doing e-business with Web services means a low barrier to entry because Web services can be added to existing applications running on any platform without changing legacy code. Commercial Web services focus on selling content and business services to clients over the Internet similar to familiar Web pages.

Unlike Web pages, commercial Web services target applications not humans as their direct users. Continental Airlines exposes flight schedules and status Web services for travel Web sites and agencies to use in their applications. Like Web pages, commercial Web services are valuable only if they expose a valuable service or content. It would be very difficult to get customers to pay you for using a Web service that creates business charts with the customers? data. Customers would rather buy a charting component (e.g. COM or .NET component) and install it on the same machine as their application. On the other hand, it makes sense to sell real-time weather information or stock quotes as a Web service. Technology can help you add value to your services and explore new markets, but ultimately customers pay for contents and/or business services, not for technology

351. **In a Webservice, need to display 10 rows from a table. So DataReader or DataSet is best choice?**
A: WebService will support only DataSet.

352. **Are Web Services a replacement for other distributed computing platforms?**
No. Web Services is just a new way of looking at existing implementation platforms.

353. **What is SOAP, WSDL, UDDI and the concept behind Web Services? What are various components of WSDL? What is the use of WSDL.exe utility?**
SOAP is an XML-based messaging framework specifically designed for exchanging formatted data across the Internet, for example using request and reply messages or sending entire documents. SOAP is simple, easy to use, and completely neutral with respect to operating system, programming language, or distributed computing platform.
After SOAP became available as a mechanism for exchanging XML messages among enterprises (or among disparate applications within the same enterprise), a better way was needed to describe the messages and how they are exchanged. The Web Services Description Language (WSDL) is a particular form of an XML Schema, developed by Microsoft and IBM for the purpose of defining the XML message, operation, and protocol mapping of a web service accessed using SOAP or other XML protocol. WSDL defines web services in terms of "endpoints" that operate on XML messages. The WSDL syntax allows both the messages and the operations on the messages to be defined abstractly, so they can be mapped to multiple physical implementations. The current WSDL spec describes how to map messages and operations to SOAP 1.1, HTTP GET/POST, and MIME. WSDL creates web service definitions by mapping a group of endpoints into a logical sequence of operations on XML messages. The same XML message can be mapped to multiple operations (or services) and bound to one or more communications protocols (using "ports"). The Universal Description, Discovery, and Integration (UDDI) framework defines a data model (in XML) and SOAP APIs for registration and searches on business information, including the web services a business exposes to the Internet. UDDI is an independent consortium of vendors, founded by Microsoft, IBM, and Ariba, for the purpose of developing an Internet standard for web service description registration and discovery. Microsoft, IBM, and Ariba also are hosting the initial deployment of a UDDI service, which is conceptually patterned after DNS (the Internet service that translates URLs into TCP addresses). UDDI uses a private agreement profile of SOAP (i.e. UDDI doesn't use the SOAP serialization format because it's not well suited to passing complete XML documents (it's aimed at RPC style interactions). The main idea is that businesses use the SOAP APIs to register themselves with UDDI, and other businesses search UDDI when they want to discover a trading partner, for example someone from whom they wish to procure sheet metal, bolts, or transistors. The information in UDDI is categorized according to industry type and geographical location, allowing UDDI consumers to search through lists of

potentially matching businesses to find the specific one they want to contact. Once a specific business is chosen, another call to UDDI is made to obtain the specific contact information for that business. The contact information includes a pointer to the target business's WSDL or other XML schema file describing the web service that the target business publishes.

354. **How to generate proxy class other than .net app and wsdl tool?**
To access an XML Web service from a client application, you first add a Web reference, which is a reference to an XML Web service. When you create a Web reference, Visual Studio creates an XML Web service proxy class automatically and adds it to your project. This proxy class exposes the methods of the XML Web service and handles the marshalling of appropriate arguments back and forth between the XML Web service and your application. Visual Studio uses the Web Services Description Language (WSDL) to create the proxy.
To generate an XML Web service proxy class:
From a command prompt, use Wsdl.exe to create a proxy class, specifying (at a minimum) the URL to an XML Web service or a service description, or the path to a saved service description.
Wsdl /language:*language* /protocol:*protocol* /namespace:*myNameSpace* /out:*filename*
/username:*username* /password*:password* /domain*:domain <url or path>*
355. **asynchronous web service means?**
356. **What are the events fired when web service called?**
357. **How does SOAP transport happen and what is the role of HTTP in it? How you can access a webservice using soap?**
358. **How will do transaction in Web Services?**
359. **What are the different formatters can be used in both? Why?.. binary/soap**
360. **What is a proxy in web service? How do I use a proxy server when invoking a Web service?**
If you are using the SOAP Toolkit, you need to set some connector properties to use a proxy server: Dim soap As SoapClient Set soap=New SoapClient soap.ConnectorProperty("ProxyServer") = ?proxyservername? soap.ConnectorProperty("ProxyPort") = ?8080? soap.ConnectorProperty("UseProxy") = True While with .NET , you just need to create a System.Net.WebProxy object and use it to set the Proxy property Dim webs As localhost.MyService() webs.Proxy=New System.Net.WebProxy(?http://proxyserver:8080?)
361. **How you will protect / secure a web service?**
For the most part, things that you do to secure a Web site can be used to secure a Web Service. If you need to encrypt the data exchange, you use Secure Sockets Layer (SSL) or a Virtual Private Network to keep the bits secure. For authentication, use HTTP Basic or Digest authentication with Microsoft® Windows® integration to figure out who the caller is.
these items cannot:
Parse a SOAP request for valid values
Authenticate access at the Web Method level (they can authenticate at the Web Service level)
Stop reading a request as soon as it is recognized as invalid

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpcontransactionsupportinaspnetwebservices.asp

362. **How will you expose/publish a webservice?**
363. **What's the attribute for webservice method? What is the namespace for creating webservice?**
364. **What is disco file?**

365. **What is Remoting?**

The process of communication between different operating system processes, regardless of whether they are on the same computer. The .NET remoting system is an architecture designed to simplify communication between objects living in different application domains, whether on the same computer or not, and between different contexts, whether in the same application domain or not.

366. **Difference between web services & remoting?**

	ASP.NET Web Services	.NET Remoting
Protocol	Can be accessed only over HTTP	Can be accessed over any protocol (including TCP, HTTP, SMTP and so on)
State Management	Web services work in a stateless environment	Provide support for both stateful and stateless environments through Singleton and SingleCall objects
Type System	Web services support only the datatypes defined in the XSD type system, limiting the number of objects that can be serialized.	Using binary communication, .NET Remoting can provide support for rich type system
Interoperability	Web services support interoperability across platforms, and are ideal for heterogeneous environments.	.NET remoting requires the client be built using .NET, enforcing homogenous environment.
Reliability	Highly reliable due to the fact that Web services are always hosted in IIS	Can also take advantage of IIS for fault isolation. If IIS is not used, application needs to provide plumbing for ensuring the reliability of the application.
Extensibility	Provides extensibility by allowing us to intercept the SOAP messages during the serialization and deserialization stages.	Very extensible by allowing us to customize the different components of the .NET remoting framework.
Ease-of-Programming	Easy-to-create and deploy.	Complex to program.

367. Though both the .NET Remoting infrastructure and ASP.NET Web services can enable cross-process communication, each is designed to benefit a different target audience. ASP.NET Web services provide a simple programming model and a wide reach. .NET Remoting provides a more complex programming model and has a much narrower reach.
As explained before, the clear performance advantage provided by TCPChannel-remoting should make you think about using this channel whenever you can afford to do so. If you can create direct TCP connections from your clients to your server and if you need to support only the .NET platform, you should go for this channel. If you are going to go cross-platform or you have the requirement of supporting SOAP via HTTP, you should definitely go for ASP.NET Web services.
Both the .NET remoting and ASP.NET Web services are powerful technologies that provide a suitable framework for developing distributed applications. It is important to understand how both technologies work and then choose the one that is right for your application. For applications that require interoperability and must function over public networks, Web services are probably the best bet. For those that require communications with other .NET components and

where performance is a key priority, .NET Remoting is the best choice. In short, use Web services when you need to send and receive data from different computing platforms, use .NET Remoting when sending and receiving data between .NET applications. In some architectural scenarios, you might also be able to use.NET Remoting in conjunction with ASP.NET Web services and take advantage of the best of both worlds.
The Key difference between ASP.NET webservices and .NET Remoting is how they serialize data into messages and the format they choose for metadata. ASP.NET uses XML serializer for serializing or Marshalling. And XSD is used for Metadata. .NET Remoting relies on System.Runtime.Serialization.Formatter.Binary and System.Runtime.Serialization.SOAPFormatter and relies on .NET CLR Runtime assemblies for metadata.

368. **Can you pass SOAP messages through remoting?**
369. **CAO and SAO.**

Client Activated objects are those remote objects whose Lifetime is directly Controlled by the client. This is in direct contrast to SAO. Where the server, not the client has complete control over the lifetime of the objects.
Client activated objects are instantiated on the server as soon as the client request the object to be created. Unlike as SAO a CAO doesn't delay the object creation until the first method is called on the object. (In SAO the object is instantiated when the client calls the method on the object)

370. **singleton and singlecall.**

Singleton types never have more than one instance at any one time. If an instance exists, all client requests are serviced by that instance.
Single Call types always have one instance per client request. The next method invocation will be serviced by a different server instance, even if the previous instance has not yet been recycled by the system.

371. **What is Asynchronous Web Services?**
372. **How to generate WebService proxy?**
373. **Web Client class and its methods?**
374. **Flow of remoting?**

(XML)

375. **Explain the concept of data island?**
376. **How to use XML DOM model on client side using JavaScript.**
377. **What are the ways to create a tree view control using XML, XSL & JavaScript?**
378. **Questions on XPathNavigator, and the other classes in System.XML Namespace?**
379. **What is Use of Template in XSL?**
380. **What is "Well Formed XML" and "Valid XML"**
381. **How you will do SubString in XSL**
382. **Can we do sorting in XSL ? how do you deal sorting columns dynamically in XML.**
383. **What is "Async" property of XML Means ?**
384. **What is XPath Query ?**
385. **Difference Between Element and Node.**
386. **What is CDATA Section.**
387. **DOM & SAX parsers explanation and difference**
388. **What is GetElementbyname method will do?**
389. **What is selectnode method will give?**
390. **What is valid xml document? What a well formed xml document?**

391. **What is the Difference between XmlDocument and XmlDataDocument?**
392. **Explain what a DiffGram is, and a good use for one?**

A DiffGram is an XML format that is used to identify current and original versions of data elements. When sending and retrieving a **DataSet** from an XML Web service, the DiffGram format is implicitly used.

The **DataSet** uses the DiffGram format to load and persist its contents, and to serialize its contents for transport across a network connection. When a **DataSet** is written as a DiffGram, it populates the DiffGram with all the necessary information to accurately recreate the contents, though not the schema, of the **DataSet**, including column values from both the **Original** and **Current** row versions, row error information, and row order.

DiffGram Format

The DiffGram format is divided into three sections: the current data, the original (or "before") data, and an errors section, as shown in the following example.

```
393. <?xml version="1.0"?>
394.                     <diffgr:diffgram
395.                         xmlns:msdata="urn:schemas-microsoft-com:xml-
     msdata"
396.                         xmlns:diffgr="urn:schemas-microsoft-com:xml-
     diffgram-v1"
397.                         xmlns:xsd="http://www.w3.org/2001/XMLSchema">
398.
399.                       <DataInstance>
400.                       </DataInstance>
401.
402.                      <diffgr:before>
403.                      </diffgr:before>
404.
405.                      <diffgr:errors>
406.                      </diffgr:errors>
                 </diffgr:diffgram>
```

The DiffGram format consists of the following blocks of data:

<DataInstance>

The name of this element, ***DataInstance***, is used for explanation purposes in this documentation. A ***DataInstance*** element represents a **DataSet** or a row of a **DataTable**. Instead of *DataInstance*, the element would contain the name of the **DataSet** or **DataTable**. This block of the DiffGram format contains the current data, whether it has been modified or not. An element, or row, that has been modified is identified with the **diffgr:hasChanges** annotation.

<diffgr:before>

This block of the DiffGram format contains the original version of a row. Elements in this block are matched to elements in the ***DataInstance*** block using the **diffgr:id** annotation.

<diffgr:errors>

This block of the DiffGram format contains error information for a particular row in the ***DataInstance*** block. Elements in this block are matched to elements in the ***DataInstance*** block using the **diffgr:id** annotation.

407. **If I replace my Sqlserver with XML files and how about handling the same?**
408. **Write syntax to serialize class using XML Serializer?**

(IIS)

409. **In which process does IIS runs (was asking about the EXE file)**
inetinfo.exe is the Microsoft IIS server running, handling ASP.NET requests among other things. When an ASP.NET request is received (usually a file with .aspx extension), the ISAPI filter aspnet_isapi.dll takes care of it by passing the request to the actual worker process aspnet_wp.exe.

410. **Where are the IIS log files stored?**
C:\WINDOWS\system32\Logfiles\W3SVC1
OR
c:\winnt\system32\LogFiles\W3SVC1

411. **What are the different IIS authentication modes in IIS 5.0 and Explain? Difference between basic and digest authentication modes?**
IIS provides a variety of authentication schemes:
Anonymous (enabled by default)
Basic
Digest
Integrated Windows authentication (enabled by default)
Client Certificate Mapping

Anonymous

Anonymous authentication gives users access to the public areas of your Web site without prompting them for a user name or password. Although listed as an authentication scheme, it is not technically performing any client authentication because the client is not required to supply any credentials. Instead, IIS provides stored credentials to Windows using a special user account, IUSR_*machinename*. By default, IIS controls the password for this account. Whether or not IIS controls the password affects the permissions the anonymous user has. When IIS controls the password, a sub authentication DLL (iissuba.dll) authenticates the user using a network logon. The function of this DLL is to validate the password supplied by IIS and to inform Windows that the password is valid, thereby authenticating the client. However, it does not actually provide a password to Windows. When IIS does not control the password, IIS calls the LogonUser() API in Windows and provides the account name, password and domain name to log on the user using a local logon. After the logon, IIS caches the security token and impersonates the account. A local logon makes it possible for the anonymous user to access network resources, whereas a network logon does not.

Basic Authentication

IIS Basic authentication as an implementation of the basic authentication scheme found in section 11 of the HTTP 1.0 specification.
As the specification makes clear, this method is, in and of itself, non-secure. The reason is that Basic authentication assumes a trusted connection between client and server. Thus, the username and password are transmitted in clear text. More specifically, they are transmitted using Base64 encoding, which is trivially easy to decode. This makes Basic authentication the wrong choice to use over a public network on its own.
Basic Authentication is a long-standing standard supported by nearly all browsers. It also imposes no special requirements on the server side -- users can authenticate against any NT domain, or even against accounts on the local machine. With SSL to shelter the security credentials while they are in transmission, you have an authentication solution that is both highly secure and quite flexible.

Digest Authentication

The Digest authentication option was added in Windows 2000 and IIS 5.0. Like Basic authentication, this is an implementation of a technique suggested by Web standards, namely RFC 2069 (superceded by RFC 2617).
Digest authentication also uses a challenge/response model, but it is much more secure than Basic authentication (when used without SSL). It achieves this greater security not by encrypting the secret (the password) before sending it, but rather by following a different design pattern -- one that does not require the client to transmit the password over the wire at all.

Instead of sending the password itself, the client transmits a one-way message digest (a checksum) of the user's password, using (by default) the MD5 algorithm. The server then fetches the password for that user from a Windows 2000 Domain Controller, reruns the checksum algorithm on it, and compares the two digests. If they match, the server knows that the client knows the correct password, even though the password itself was never sent. (If you have ever wondered what the default ISAPI filter "md5filt" that is installed with IIS 5.0 is used for, now you know.

Integrated Windows Authentication

Integrated Windows authentication (formerly known as NTLM authentication and Windows NT Challenge/Response authentication) can use either NTLM or Kerberos V5 authentication and only works with Internet Explorer 2.0 and later.
When Internet Explorer attempts to access a protected resource, IIS sends two WWW-Authenticate headers, Negotiate and NTLM.

> If Internet Explorer recognizes the Negotiate header, it will choose it because it is listed first. When using Negotiate, the browser will return information for both NTLM and Kerberos. At the server, IIS will use Kerberos if both the client (Internet Explorer 5.0 and later) and server (IIS 5.0 and later) are running Windows 2000 and later, and both are members of the same domain or trusted domains. Otherwise, the server will default to using NTLM.
> If Internet Explorer does not understand Negotiate, it will use NTLM.

So, which mechanism is used depends upon a negotiation between Internet Explorer and IIS.
When used in conjunction with Kerberos v5 authentication, IIS can delegate security credentials among computers running Windows 2000 and later that are trusted and configured for delegation. Delegation enables remote access of resources on behalf of the delegated user.
Integrated Windows authentication is the best authentication scheme in an intranet environment where users have Windows domain accounts, especially when using Kerberos. Integrated Windows authentication, like digest authentication, does not pass the user's password across the network. Instead, a hashed value is exchanged.

Client Certificate Mapping

A certificate is a digitally signed statement that contains information about an entity and the entity's public key, thus binding these two pieces of information together. A trusted organization (or entity) called a Certification Authority (CA) issues a certificate after the CA verifies that the entity is who it says it is. Certificates can contain different types of data. For example, an X.509 certificate includes the format of the certificate, the serial number of the certificate, the algorithm used to sign the certificate, the name of the CA that issued the certificate, the name and public key of the entity requesting the certificate, and the CA's signature. X.509 client certificates simplify authentication for larger user bases because they do not rely on a centralized account database. You can verify a certificate simply by examining the certificate.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsent7/html/vxconIISAuthentication.asp

412. **How to configure the sites in Web server (IIS)?**
413. **Advantages in IIS 6.0?**
    http://www.microsoft.com/windowsserver2003/iis/evaluation/features/default.mspx
    http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windowsserver2003/proddocs/datacenter/gs_whatschanged.asp
414. **IIS Isolation Levels?**
    Internet Information Server introduced the notion "Isolation Level", which is also present in IIS4 under a different name. IIS5 supports three isolation levels, that you can set from the Home Directory tab of the site's Properties dialog:

Low (IIS Process): ASP pages run in INetInfo.Exe, the main IIS process, therefore they are executed in-process. This is the fastest setting, and is the default under IIS4. The problem is that if ASP crashes, IIS crashes as well and must be restarted (IIS5 has a reliable restart feature that automatically restarts a server when a fatal error occurs).

Medium (Pooled): In this case ASP runs in a different process, which makes this setting more reliable: if ASP crashes IIS won't. All the ASP applications at the Medium isolation level share the same process, so you can have a web site running with just two processes (IIS and ASP process). IIS5 is the first Internet Information Server version that supports this setting, which is also the default setting when you create an IIS5 application. Note that an ASP application that runs at this level is run under COM+, so it's hosted in DLLHOST.EXE (and you can see this executable in the Task Manager).

High (Isolated): Each ASP application runs out-process in its own process space, therefore if an ASP application crashes, neither IIS nor any other ASP application will be affected. The downside is that you consume more memory and resources if the server hosts many ASP applications. Both IIS4 and IIS5 supports this setting: under IIS4 this process runs inside MTS.EXE, while under IIS5 it runs inside DLLHOST.EXE.

When selecting an isolation level for your ASP application, keep in mind that out-process settings - that is, Medium and High - are less efficient than in-process (Low). However, out-process communication has been vastly improved under IIS5, and in fact IIS5's Medium isolation level often deliver better results than IIS4's Low isolation. In practice, you shouldn't set the Low isolation level for an IIS5 application unless you really need to serve hundreds pages per second.

Controls

415. **How will you do Redo and Undo in a TextControl?**
416. **How to implement DataGrid in .NET? How would u make a combo-box appear in one column of a DataGrid? What are the ways to show data grid inside a data grid for a master details type of tables? If we write any code for DataGrid methods, what is the access specifier used for that methods in the code behind file and why?**
417. **How can we create Tree control in asp.net?**

Programming

418. **Write a program in C# for checking a given number is PRIME or not.**
419. **Write a program to find the angle between the hours and minutes in a clock**
420. **Write a C# program to find the Factorial of n**
421. **How do I upload a file from my ASP.NET page?**

A: In order to perform file upload in your ASP.NET page, you will need to use two classes: the System.Web.UI.HtmlControls.HtmlInputFile class and the System.Web.HttpPostedFile class. The HtmlInputFile class represents and HTML input control that the user will use on the client side to select a file to upload. The HttpPostedFile class represents the uploaded file and is obtained from the PostedFile property of the HtmlInputFile class. In order to use the HtmlInputFile control, you need to add the enctype attribute to your form tag as follows:

<form id="upload" method="post" runat="server" enctype="multipart/form-data">

Also, remember that the /data directory is the only directory with Write permissions enabled for the anonymous user. Therefore, you will need to make sure that the your code uploads the file to the /data directory or one of its

subdirectories.
Below is a simple example of how to upload a file via an ASP.NET page in C# and VB.NET.

C#

```
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Web" %>
<%@ Import Namespace="System.Web.UI.HtmlControls" %>
<%@ Import Namespace="System.IO" %>
<%@ Import Namespace="System.Drawing" %>
<html>
<head>
<title>upload_cs</title>
</head>
<script language="C#" runat="server">
public void UploadFile(object sender, EventArgs e)
{
if (loFile.PostedFile != null)
{
try
{
string strFileName, strFileNamePath, strFileFolder;
strFileFolder = Context.Server.MapPath(@"data\");
strFileName = loFile.PostedFile.FileName;
strFileName = Path.GetFileName(strFileName);
strFileNamePath = strFileFolder + strFileName;
loFile.PostedFile.SaveAs(strFileNamePath);
lblFileName.Text = strFileName;
lblFileLength.Text = loFile.PostedFile.ContentLength.ToString();
lblFileType.Text = loFile.PostedFile.ContentType;
pnStatus.Visible = true;
}
catch (Exception x)
{
Label lblError = new Label();
lblError.ForeColor = Color.Red;
lblError.Text = "Exception occurred: " + x.Message;
lblError.Visible = true;
this.Controls.Add(lblError);
}
}
}
</script>
<body>
<form id="upload_cs" method="post" runat="server"
enctype="multipart/form-data">
<P>
<INPUT type="file" id="loFile" runat="server">
</P>
<P>
<asp:Button id="btnUpload" runat="server" Text=" Upload "
OnClick="UploadFile"></asp:Button></P>
<P>
<asp:Panel id="pnStatus" runat="server" Visible="False">
<asp:Label id="lblFileName" Font-Bold="True" Runat="server"></asp:Label>
uploaded<BR>
<asp:Label id="lblFileLength" Runat="server"></asp:Label> bytes<BR>
<asp:Label id="lblFileType" Runat="server"></asp:Label>
</asp:Panel></P>
</form>
```

```
</body>
</html>
```

422. **How do I send an email message from my ASP.NET page?**

A: You can use the System.Web.Mail.MailMessage and the System.Web.Mail.SmtpMail class to send email in your ASPX pages. Below is a simple example of using this class to send mail in C# and VB.NET. In order to send mail through our mail server, you would want to make sure to set the static SmtpServer property of the SmtpMail class to mail-fwd.

C#

```
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Web" %>
<%@ Import Namespace="System.Web.Mail" %>
<HTML>
<HEAD>
<title>Mail Test</title>
</HEAD>
<script language="C#" runat="server">
private void Page_Load(Object sender, EventArgs e)
{
try
{
MailMessage mailObj = new MailMessage();
mailObj.From = "sales@joeswidgets.com";
mailObj.To = "ringleader@forexample-domain.com";
mailObj.Subject = "Your Widget Order";
mailObj.Body = "Your order was processed.";
mailObj.BodyFormat = MailFormat.Text;
SmtpMail.SmtpServer = "mail-fwd";
SmtpMail.Send(mailObj);
Response.Write("Mail sent successfully");
}
catch (Exception x)
{
Response.Write("Your message was not sent: " + x.Message);
}
}
</script>
<body>
<form id="mail_test" method="post" runat="server">
</form>
</body>
</HTML>
```

423. **Write a program to create a user control with name and surname as data members and login as method and also the code to call it. (Hint use event delegates)**