**Q #1) What is .Net framework?**

**Ans:** It is a platform for building various applications on windows. It has a list of inbuilt functionalities in the form of class, library, and APIs which are used to build, deploy and run web services and different applications. It supports different languages such as C#, VB .Net, Cobol, Perl, etc.

This framework supports object-oriented programming model.

**Q #2) What are the important components of .Net?**

**Ans:** The components of .Net are Common language run-time, .Net Class library, Application domain, Common Type System, .Net framework, Profiling, etc. However, the two important components are **Class library and Common Language Runtime.**

CLR provides building blocks for a wide variety of applications. The class library consists of a set of classes that are used to access the common functionality. The functionality can be shared among different applications.

**Q #3) What is CTS?**

**Ans:** CTS stands for **Common Type System**. It has a set of rules which state how a data type should be declared, defined and used in the program. It describes the data types that are to be used in the application.

We can design our own classes and values by following the rules that are present in the CTS. The rules are made so that the data type declared using a programming language is callable by an application that is developed using a different language.

**Q #4) What is CLR?**

**Ans:** CLR stands for **Common Language Runtime**. It is one of the most important components of .Net framework. It provides building blocks for many applications.

An application built using C# gets compiled by its own compiler and is converted into an Intermediate language. This is then targeted to CLR. CLR does various operations like memory management, Security checks, assemblies to be loaded and thread management. It provides a secure execution environment for applications.

**Q #5) What is CLS?**

**Ans:** CLS stands for **Common Language Specification**. With the rules mentioned under CLS, the developers are made to use the components that are inter-language compatible. They are reusable across all the .Net Compliant languages.

**Q #6) What is JIT?**

**Ans:** JIT stands for **Just In Time**. JIT is a compiler that converts Intermediate Language to a Native code.

The code is converted into Native language during execution. Native code is nothing but hardware specifications that can be read by the CPU. The native code can be stored so that it is accessible for subsequent calls.

**Q #7) What is MSIL?**

**Ans:** MSIL stands for **Microsoft Intermediate Language**.

MSIL provides instructions for calling methods, initializing and storing values, operations such as memory handling, exception handling and so on. All .Net codes are first compiled to IL.

**Q #8) What is meant by Managed and Unmanaged code?**

**Ans:** The code that is managed by the CLR is called **Managed code**. This code runs inside the CLR. Hence, it is necessary to install the .Net framework in order to execute the managed code. CLR manages the memory through garbage collection and also uses the other features like CAS and CTS for efficient management of the code.

Unmanaged code is any code that does not depend on CLR for execution. It means it is developed by any other language independent of .Net framework. It uses its own runtime environment for compiling and execution.

Though it is not running inside the CLR, the unmanaged code will work properly if all the other parameters are correctly followed.

**Q #9) How is a Managed code executed?**

**Ans:** **Following steps are followed while executing a Managed code:**

* Choosing a language compiler depending on the language in which the code is written.
* Converting the above code into Intermediate Language by its own compiler.
* The IL is then targeted to CLR which converts the code into native code with the help of JIT.
* Execution of Native code.

**Q #10) What is ASP.Net?**

**Ans:** ASP .Net is a part of .Net technology and it comprises of CLR too. It is an open source server-side technology that enables the programmers to build powerful web services, websites and web applications.

ASP stands for **Active Server Pages**.

**Q #11) Explain State management in ASP .Net.**

**Ans:** State Management means maintaining the state of the object. The object here refers to a web page/control.

**There are two types of State management, Client Side, and Server side.**

**Client Side** – Storing the information in the Page or Client’s System. They are reusable, simple objects.

**Server Side** – Storing the information on the Server. It is easier to maintain the information on the Server rather than depending on the client for preserving the state.

**Q #12) What is an Assembly? What are the different types of Assemblies?**

**Ans:** An Assembly is a collection of logical units. Logical units refer to the types and resources which are required to build an application and deploy them using the .Net framework. The CLR uses this information for type implementations. Basically, Assembly is a collection of Exe and Dlls. It is portable and executable.

**There are two types of Assemblies, Private and Shared.**

**Private Assembly**, as the name itself suggests, it is accessible only to the application. It is installed in the installation directory of the Application.

A **Shared assembly** can be shared by multiple applications. It is installed in the GAC.

**Q #13) Explain the different parts of an Assembly.**

**Ans: The different parts of an Assembly are:**

* **Manifest** – It contains the information about the version of an assembly. It is also called as assembly metadata.
* **Type Metadata** – Binary information of the program.
* **MSIL** – Microsoft Intermediate Language code.
* **Resources** – List of related files.

**Q #14) What is an EXE and a DLL?**

**Ans:** Exe and DLLs are Assembly executable modules.

**Exe is an executable file**. This runs the application for which it is designed. An Exe is generated when we build an application. Hence the assemblies are loaded directly when we run an Exe. However, an Exe cannot be shared with the other applications.

**DLL stands for Dynamic Link Library**. It is a library that consists of code which needs be hidden. The code is encapsulated inside this library. An Application can consist of many DLLs. These can be shared with the other applications as well.

Other applications which need to share this DLL need not worry about the code intricacies as long as it is able to call the function on this DLL.

**Q #15) What is Caching?**

**Ans:** Caching means storing data temporarily in the memory so that the application can access the data from the cache instead of looking for its original location. This increases the performance of the application and its speed. System.Runtime.Caching namespace is used for Caching information in .Net.

**Given below are the 3 different types of Caching:**

* Page Caching
* Data Caching
* Fragment Caching

**Q #16) What is MVC?**

**Ans:** MVC stands for Model View Controller. It is an architectural model for building the .Net applications.

**Models** – Model objects store and retrieve data from the database for an application. They are usually the logical parts of an application that is implemented by the application’s data domain.

**View** – These are the components that display the view of the application in the form of UI. The view gets the information from the model objects for their display. They have components like buttons, drop boxes, combo box, etc.

**Controllers** – They handle the User Interactions. They are responsible for responding to the user inputs, work with the model objects, and pick a view to be rendered to the user.

**Q #17) What is the difference between Function and Stored procedure?**

**Ans:**

**Stored Procedure:**

* A Stored procedure is always used to perform a specific task.
* It can return zero, one or more value.
* It can have both Input and Output Parameters.
* Exception handling can be done using a try-catch block.
* A function can be called from a Procedure.

**Functions:**

* Functions must return a single value.
* It can only have the input parameter.
* Exception handling cannot be done using a try-catch block.
* A Stored procedure cannot be called from a function.

**Q #18) Explain CAS (Code Access Security).**

**Ans:** .Net provides a security model that prevents unauthorized access to resources. CAS is a part of that security model. CAS is present in the CLR. It enables the users to set permissions at a granular level for the code.

CLR then executes the code depending on the available permissions. CAS can be applied only to the managed code. Unmanaged code runs without CAS. If CAS is used on assemblies, then the assembly is treated as partially trusted. Such assemblies must undergo checks every time when it tries to access a resource.

**The different components of CAS are Code group, Permissions, and Evidence.**

**Evidence**– To decide what permissions to give, the CAS and CLR depend on the specified evidence by the assembly. The examination of the assembly provides details about the different pieces of evidence. Some common evidence include Zone, URL, Site, Hash Value, Publisher and Application directory.

**Code Group** – Depending on the evidence, codes are put into different groups. Each group has specific conditions attached to it. Any assembly that matches those condition is put into that group.

**Permissions** – Each code group can perform only specific actions. They are called Permissions. When CLR loads an assembly, it matches them to one of the code groups and identifies what actions those assemblies can do. Some of the Permissions include Full Trust, Everything, Nothing, Execution, Skip Verification, and the Internet.

**Q #19) What is GAC?**

**Ans:** GAC stands for**Global Assembly Cache**. Whenever CLR gets installed on the machine, GAC comes as a part of it. GAC specifically stores those assemblies which will be shared by many applications. A Developer tool called Gacutil.exe is used to add any file to GAC.

**Q #20) What is meant by Globalization and Localization?**

**Ans:** Internationalization is the process of designing applications that support multiple languages. This is divided into **Localization and Globalization**.

Globalization is nothing but developing applications to support different languages. Existing applications can also be converted to support multiple cultures.

Whereas Localization means changing the already globalized app to cater to a specific culture or language Microsoft.Extensions.Localization is used for localizing the app content. Some of the other keywords that are used for Localization are IHtmlLocalizer, IStringLocalizer, IViewLocalizer and so on

**Q #21) What is a Garbage Collector?**

**Ans:** Garbage collection is a feature of .Net to free the unused code objects in the memory.

**The memory heap is divided into three generations. Generation 0, Generation 1 and Generation 2.**

**Generation 0** – This is used to store short-lived objects. Garbage Collection happens frequently in this Generation.

**Generation 1** – This is for medium-lived objects. Usually, the objects that get moved from generation 0 are stored in this.

**Generation 2** – This is for long-lived objects.

Collecting a Generation refers to collecting the objects in that generation and all its younger generations. Garbage collection of Generation 2 means full garbage collection, it collects all the objects in Generation 2 as well as Generation 1 and Generation 0.

During the Garbage collection process, as the first phase, list of live objects are identified. In the second phase, references are updated for those objects which will be compacted. And in the last phase, the space occupied by dead objects are reclaimed. The remaining objects are moved to an older segment.

<https://www.softwaretestinghelp.com/dot-net-interview-questions/>

[**Top 40 .NET Interview Questions & Answers**](https://career.guru99.com/net-technology-interview-questions/)

**1) Explain what is .NET Framework?**

The.Net Framework is developed by Microsoft. It provides technologies and tool that is required to build Networked Applications as well as Distributed Web Services and Web Applications.

**2) What does .NET Framework provides?**

.NET Framework renders the necessary compile time and run time foundation to build and run any language that conforms to the Common Language Specification (CLS).

**3) Mention what are main components of .Net framework?**

The main components of .Net framework are

* Common Language Runtime (CLR)
* .Net Framework Class Library (FCL)
* Application Domains
* Runtime Host
* Cross-Language Interoperability
* Side-by-Side Execution
* Profiling
* Dynamic Language Runtime (DLR)
* Common Type System
* Metadata and Self-Describing Components
* .Net Framework Security
* Model View Presenter (MVP) Architecture

**4) Mention key characteristics of .NET ?**

* Unlike other programming language, in .NET the program will be compiled into an intermediate language representation known as MSIL (Microsoft Intermediate Language)
* MSIL code does not contain any API calls particular to any platform
* Compiler checks only for syntax and the necessary semantics as such it is
* Libraries used by the program are linked even before generating MSIL. It is linked in an un-compiled form
* Instead of directly calling API of the operating system, the program uses CLR to call API. CLR acts as mediator
* Garbage collection and automatic memory management are done by CLR

**5) Mention what are the languages that .NET supports?**

Languages that .NET supports are,

* NET
* C#
* COBOL
* PERL

**6) Mention how big is the datatype int in .NET?**

Datatype int in .NET is 32 bits.

**7) Mention what is .Net Namespaces?**

Namespaces in .NET is nothing but a way to organize .NET Framework Class Library into a logical grouping according to their usability, functionality as well as category they belong to.

**8) Mention what is MSIL in .NET ?**

* MSIL stands for Microsoft Intermediate Language
* During the compile time, the source code is converted into Microsoft Intermediate Language (MSIL) by compiler
* MSIL is a CPU-independent set of instructions that can be efficiently converted to the native code

**9) Mention what are the functions .NET Assembly performs?**

Assembly is the main unit of deployment in a .NET Framework application executed as .exe or .dll.

An assembly performs following functions

* It consists of an IL code that gets executed by common language runtime
* It forms a security boundary
* By establishing name scope for types at the runtime, it ensures safety
* It carries version information
* It enables side-by-side execution of multiple versions of the same assembly
* Assembly is where permission is requested and granted.

**10) Mention what is .Net Assembly Manifest?**

.Net Assembly Manifest is a file which contains metadata about .NET Assemblies.  It describes how the elements in the assembly relate to each other. In other words, it describes the relationship and dependencies of the components in the Assembly, scope information, versioning information, etc.

**11) Mention what is MSIL in .NET ?**

Microsoft Intermediate Language (MSIL) includes instructions for storing, loading, initializing, and calling methods on objects, as well as instructions for logical and arithmetic operations, direct memory access, control flow, exception handling, and other operations.

**12) Explain what is PE (Portable Executable) File format?**

The Portable Executable (PE) format is a file format for executables, object code, and DLLs, used in 64-bit and 32-bit versions of Windows operating systems.

**13) Mention what is the difference between Assembly and Namespace?**

* Namespace can span multiple assemblies
* Namespace can logically groups class
* Assembly is a physical grouping of logical units

**14) List out the namespace provided by .net for data management?**  
The namespace provided by .net for data management include,

* Data
* Data.SQLClient
* XML

**15) Mention what is GAC in .net ?**

GAC stands for Global Assembly Cache. It is an area of memory reserved to store the assemblies of all .NET applications that are running on a certain machine.

**16) Mention what is STA in .NET?**

STA or single threaded apartment model offers a message-based paradigm for dealing with multiple objects running concurrently. Every thread lives within its own apartment.

**17) Mention what is data access modifier in .NET?**

Data access modifier in .NET provide a class, a function or a variable with accessibility.

**18) Mention what are the types of access modifier in .NET?**

The access modifier in .NET are five types

* Public
* Private
* Protected
* Internal
* Protected Internal

**19) Mention the type of code security available in .NET?**

The type of code security available in .NET are

* Role based security: This authorizes the user.
* Code access security: This protects system resources from unauthorized calls.

**20) Explain how you can implement singleton pattern in .NET?**

To implement singleton pattern in .NET, following steps, has to be implemented.

* Create a class with static members
* Define a private constructor
* To access the singleton object, a static method can be used

**21) Explain how the exception is handled in .NET?**

In .Net, when there is an exception, the .NET framework creates an object of type ‘Exception’ and ‘throws’ it. This Exception object will have all the information about the ‘error’.

If you have enclosed your code within the try-catch block, you will receive the exception object in the ‘catch’ block when the exception occurs.

**22) Explain how can you create and use an array in .NET?**

In .NET, you can create array by following ways,

* Declaring a reference to an array
* Create array of ten Int32elements
* Creating a 2-dimensional array
* Creating a 3-dimensional array

**23) Mention what is user-defined data type?**

A user-defined data type is a named data type created by the user. It can be a structured type which has a sequence of named attributes that each has a type, or It can be a distinct type sharing a common representation with some built-in data type.  Based on this it can be categorized as,

* Distinct type
* Reference type
* Structured type

**24) List out few of the .Net base class library namespace?**

The .Net base class library encapsulates a huge number of common functions and makes them easily accessible to the developer.

Few of the .Net base class library namespace are

* Activities
* Collections
* Configuration
* EnterpriseServices
* Management
* Runtime and so on

**25) Mention what is the difference between structures and classes in .NET?**

|  |  |
| --- | --- |
| **Classes** | **Structures** |
| * Usually, it is used for large volume of data | * It is used for smaller amounts of data |
| * It can be inherited | * It cannot be inherited |
| * It can be NULL | * It cannot be NULL like the class. |
| * For class the keyword used is ‘class’. | * For structure the keyword used is ‘struct’. |
| * By default class member variables are private. | * By default structure, members have public access. |
| * It contains a volatile field. | * It cannot contain the volatile field. |
| * Cannot use size of operator | * Can use size of operator |
| * Fields are automatically initialized | * Fields are not initialized automatically |

**26) Mention the types of multidimensional arrays used in .NET ?**

The types of multidimensional arrays used in .NET are,

* Jagged Arrays: These type of multidimensional arrays have each sub-array as independent arrays of different lengths. For Jagged arrays, you need to use a separate set of square brackets.
* **Rectangular Arrays**: These type of multidimensional arrays have all the sub-arrays with a particular dimension of the same length. For rectangular arrays, you need to use a single set of square brackets.

**27) Explain how to add properties in.NET?**

To add properties in.NET, either you can use property procedures or fields.

**28) Mention what is event bubbling in .NET?**

The event bubbling in .NET is defined as the passing of the control from child to the parent is called as bubbling.  Controls like datalist, datagrid, repeater, etc. can have child controls like listbox, etc.

**29) Mention what are the debugging windows available?**

The windows available while debugging include,

* Breakpoints
* Output
* Immediate

**30) Explain what is Microsoft Silverlight?**

Micro-soft Silverlight is an open-source tool for making and deploying internet applications and media experiences on the web.

Silverlight architecture is made of mainly three components

* **Core presentation framework:**The framework includes the components like data binding, vector graphics, text, animation, images for presenting various features,
* **.NET framework for Silverlight:**It consists of the libraries and components like XLINQ, XML serialization, Syndication, base class libraries, networking and common language runtime,
* **Updater and Installer:**It is a control for installation and provides automatic updates

Important features in Silverlight

* **Tooling:** XAML debugging is included in this version, specially for the binding purpose
* **Media:** You can control volume, pitch, in the soundeffect class
* **Text:** Introduced RichTextBoxOverflow element, it will help in automatically laying out text in situations like multi-column
* **Data binding:** It supports implicit data templates, which means you can specify a Data Template for a specific type in your binding
* **Controls:** “Click Counts” will help basically in doing double-click tracking on elements in your application. Another feature is “Multiple-Window” support, it allows you to create several windows that your application can interact with
* **3D graphics:** 3D graphics API is available with new Silver light with many additional features
* **Trusted Application in Browser:**Without being installed, applications can be used in the browser through this feature
* **General:** Apart from all these features there are enhanced features that are supposed to be seen in Beta version, like vector printing, new DataContextchanged event, COM interop for trusted in-browser applications,

**31) What Silverlight is composed of?**

Silverlight comes with four main components

* Silverlight Plug-in
* Silverlight Host, the Web Page
* Silverlight Application File (.XAP)
* The Interface language, XAML

Silverlight plug in comes with many features like

* Presentation Engine
* XAML Parser
* .NET Parser
* .NET Framework
* Media Features
* Browser Interaction
* Downloader

**32) Explain what is .xap file?**

.xap file is a compressed file for the Silver Application.  The .xap files include AppManifest.xaml, compile output assembly of the Silverlight project (.dll) and any other resources by the Silverlight application.  Visual studio creates two files when creates project; App.xaml and Page.xaml

To run a Silverlight application on a web server, you require to append the extension .XAP with the MIME type application/X-Silverlight in the server supported file types configuration.

The .XAP mime type is: application/x-Silverlight.

**33) List out the tools required to build Silverlight applications?**

Tools used for building Silverlight applications are

* **Microsoft Expression Studio:** This tool is meant for web designers as it is used to create rich visual elements for Silverlight applications with enhanced visual contents and graphics
* **Microsoft Visual Studio:** This tool is meant for a programmer whose application is based on logic or requires programming. It permits programmer to build Silverlight applications in any .NET language ( like C#, NET, etc.)

The .NET application support in Silverlight can be segregated into two parts

* Embedded Common Language Runtime (CLR)
* .NET framework libraries

**34) Mention whether Silverlight allows MPEG4 and H.264 videos or AAC (advanced audio coding) audio or flash video?**

No, Silverlight does not support MPEG4 and H.264 videos. However, contents from many of these formats can be converted into formats that are supported by Silverlight like automated server function, and then inserted into a Silverlight based application.

**35) Explain how you can host a Silverlight Applications?**

Silverlight applications can be hosted on most types of the web server like Apache and IIS (Internet Information Server). To host Silverlight applications from your web server, you need to allow the MIME type as mentioned below

|  |  |
| --- | --- |
| **Extension** | **MIME type** |
| * .XAML * .XAP | * Application/ xaml+xml * Application/ Silverlight-app |

**36) Mention what is the difference between WPF (Windows Presentation Foundation) and Windows Silverlight?**

* In terms of features, silver light is the subset of WPF
* Silverlight competes with Adobes flash and is designed for developing rich browser based internet applications
* While WPF is a Microsoft technology for developing enhanced graphics applications for desktop platform, while Silverlight can be used for building application accessed through internet
* Web browser application can be hosted on web browsers that give rich graphics features for web applications.

**37) Explain how you can pass parameters to silver light controls from ASP.NET pages?**

You can pass parameters from you aspx pages and HTML pages to the Silverlight controls by setting InitParameters. The Xaml page user control has a property known as InitParameters. From your ASPX pages, you can set a value in the form of key value pairs. Since this property accepts key-value pairs, you can pass any set of string values.

**38) Explain how Silverlight and ASP.NET AJAX can be used by consumers?**

Silverlight synchronizes with existing web applications, including ASP.NET AJAX applications.  Apparently, ASP.NET AJAX and Silverlight are built as complementary technologies.  In short, Silverlight and ASP.NET AJAX can communicate with any AJAX application. Also, ASP.NET AJAX can additionally be used to control Silverlight-based visualization of data or delivery of rich experiences.

**39) Explain how you can check the internet connection status in Silverlight?**

By using the following code, you can check the internet connection



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | If (NetworkInterface.GetIsNetworkAvailable())    {  Messagebox.Show(“ Network available”);  }  else  {  Message.box.Show(“ Network not available”);  } |

**40) Explain what is RIA?**

RIA stands for rich internet applications, and they are web applications with rich features. Rich features include built in AJAX support, layouts, animations, audio, and video components. Silverlight is an example of RIA.

**41) Mention what are the different layout controls available in Silverlight?**

There are three controls available like

* **StackPanel:** It will position the child elements either in a vertical or horizontal manner
* **Grid:** It will place the child elements in either rows or columns
* **Canvas:** It will place the child elements according to X, Y space

**42) What is the syntax for Net?**

The syntax for ASP.Net usually consists of HTML file.  However, an ASP file can contain server script with delimiters.  An example of “Hello World !”

C#



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | < ! DOCTYPE html>    < html >    < body >    < %    Response.write (“Hello World!”)    %>      </body>    </html> |

<https://career.guru99.com/net-technology-interview-questions/>

## **Beginner .NET interview Questions**

**Q-1: What is the .NET framework?**

The .NET framework supports an object-oriented approach that is used for building applications on windows. It supports various languages like C#, VB, Cobol, Perl, .NET, etc. It has a wide variety of tools and functionalities like class, library and APIs that are used to build, deploy and run web services and different applications.

****

**Q-2: What are the different components of .NET?**

Following are the components of .NET

* Common Language run-time
* Application Domain
* Common Type System
* .NET Class Library
* .NET Framework
* Profiling

**Q-3: What do you know about CTS?**

CTS stands for Common Type System. It follows certain rules according to which a data type should be declared and used in the program code. CTS also describes the data types that are going to be used in the application. We can even make our own classes and functions following the rules in the CTS, it helps in calling the data type declared in one program language by other programming languages.

**Q-4: What is CLR?**

CLR stands for common language run-time, it is an important component of the .NET framework. We can use CLR as a building block of various applications and provides a secure execution environment for applications.

Whenever an application written in C# is compiled, the code is converted into an intermediate language. After this, the code is targeted to CLR which then performs several operations like memory management, security checks, loading assemblies, and thread management.

**Q-5: Explain CLS.**

Common language specification helps the developers to use the components that are inter-language compatible with certain rules that come with CLS. It then helps in reusing the code in other .NET compatible languages.

**Q-6: What do you know about JIT?**

JIT is a compiler which stands for Just In Time. It is used to convert the intermediate code into the native language. During the execution, the intermediate code is converted into the native language.

**Q-7: Why do we use Response.Output.Write()?**

Response.Output.Write() is used to get the formatted output.

**Q-8: What is the difference between Response.Redirect and Server.Transfer?**

Response.Redirect basically redirects the user’s browser to another page or site. The history of the user’s browser is updated to reflect the new address as well. It also performs a trip back to the client where the client’s browser is redirected to the new page.

Whereas, Server.Transfer transfers from one page to the other without making any round-trip back to the client’s browser. The history does not get updated in the case of Server.Transfer.

**Q-9: What is the difference between managed and unmanaged code?**

|  |  |
| --- | --- |
| **Managed code** | **Unmanaged code** |
| Managed code is managed by **CLR** | Any code that is not managed by **CLR** |
| .NET framework is necessary to execute managed code | Independent of .NET framework |
| **CLR**manages memory management through garbage collection | Own runtime environment for compilation and execution |

**Q-10: Explain the difference between a class and an object?**

[[](https://www.edureka.co/microsoft-dotnet-framework-self-paced)](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \t "_blank)

### [Microsoft .NET Framework Certification Training](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \t "_blank)

* *[Course Duration](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \t "_blank)*
* *[Real-life Case Studies](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \t "_blank)*
* *[Assignments](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \t "_blank)

|  |  |
| --- | --- |
| **Class** | **Object** |
| Class is the definition of an object | An object is an instance of a class. |
| It is a template of the object | A class does not become an object unless instantiated |
| It describes all the methods, properties, etc | An object is used to access all those properties from the class. |

**Q-11: What do you know about boxing and unboxing?**

|  |  |
| --- | --- |
| **Boxing** | **Unboxing** |
| Implicit | Explicit |
| Converting a value type to the type object | Extracting the value type from the object |
| eg : obj myObject = i; | eg : i = (int)myObject; |

**Q-12: Differentiate between constants and read-only variables.**

|  |  |
| --- | --- |
| **Constants** | **Read-only Variables** |
| Evaluated at compile time | Evaluated at run-time |
| Support only value type variables | They can hold the reference type variables |
| They are used when the value is not changing at compile time | Used when the actual value is unknown before the run-time |
| Cannot be initialized at the time of declaration or in a constructor | Can be initialized at the time of declaration or in a constructor |

**Q-13: What is BCL?**

* BCL is a base class library of classes, interfaces and value types
* It is the foundation of .NET framework applications, components, and controls
* Encapsulates a huge number of common functions and make them easily available for the developers
* It provides functionality like threading, input/output, security, diagnostics, resources, globalization, etc.
* Also serves the purpose of interaction between user and runtime
* It also provides namespaces that are used very frequently. for eg: system, system.Activities, etc.

**Q-14: What are the different versions of the .NET framework?**

|  |  |  |
| --- | --- | --- |
| **Version** | **.NET Framework** | **Visual Studio** |
| C# 1.0 | .NET Framework 1.0/1.1 | Visual Studio .NET 2002 |
| C# 2.0 | .NET Framework 2.0 | Visual Studio 2005 |
| C# 3.0 | .NET Framework 3.0/3.5 | Visual Studio 2008 |
| C# 4.0 | .NET Framework 4.0 | Visual Studio 2010 |
| C# 5.0 | .NET Framework 4.5 | Visual Studio 2012/2013 |
| C# 6.0 | .NET Framework 4.6 | Visual Studio 2013/2015 |
| C# 7.0 | .NET CORE | Visual Studio 2017 |

**Q-15: What is the difference between namespace and assembly?**

An assembly is a physical grouping of logical units whereas namespace groups classes. Also, a namespace can span multiple assemblies as well.

**Q-16: What is LINQ?**

It is an acronym for Language integrated query which was introduced with visual studio 2008. LINQ is a set of features that extend query capabilities to the .NET framework language syntax that allows data manipulation irrespective of the data source. LINQ bridges the gap between the world of objects and the world of data.

**Q-17: What is MSIL?**

MSIL is the Microsoft Intermediate Language, it provides instructions for calling methods, storing and initializing values, memory handling, exception handling and so on. All the .NET codes are first compiled to Intermediate Language.

**Q-18: From which base class all web Forms are inherited?**

All web forms are inherited from **page class**.

## **Intermediate .NET interview questions**

**Q-1: Explain the different parts of the assembly.**

Following are the different parts of assembly:

* Manifest: It has the information about the version of the assembly.
* Type Metadata: Contains the binary information of the program
* MSIL: Microsoft Intermediate Language Code
* Resources: List of related files

**Q-2: How do you prevent a class from being inherited?**

In C#, we can use the sealed keyword to prevent a class from being inherited.

**Q-3: What are the different types of constructors in c#?**

Following are the types of constructors in C#:

* Default Constructor
* Parameterized constructor
* Copy Constructor
* Static Constructor
* Private Constructor

**Q-4: What are the different types of assemblies?**

There are two types of assemblies:

* Private Assembly: It is accessible only to the application, it is installed in the installation directory of the application.
* Shared Assembly: It can be shared by multiple applications, it is installed in the GAC.

**Q-5: What are MDI and SDI?**

* MDI( Multiple Document Interface): An MDI lets you open multiple windows, it will have one parent window and as many child windows. The components are shared from the parent window like menubar, toolbar, etc.
* SDI( Single Document Interface): It opens each document in a separate window. Each window has its own components like menubar, toolbar, etc. Therefore it is not constrained to the parent window.

**Q-6: What is the difference between custom and user control?**

|  |  |
| --- | --- |
| **Custom Control** | **User Control** |
| Derives from control | Derives from UserControl |
| Dynamic Layout | Static Layout |
| Defines a single control | Defines a set of con |
| It has full toolbox support | Cannot be added to the toolbox |
| Loosely coupled control | Tightly coupled control |

**Q-7: What is a garbage collector?**

Garbage collector feature in .NET frees the unused code objects in the memory. The memory head is divided into 3 generations:

* Generation 0: It stores short-lived objects.
* Generation 1: This is for medium-lived objects.
* Generation 2: It stores long-lived objects.

Collection of garbage refers to the collection of objects stored in the generations.

**Q-8: What is caching?**

Caching simply means storing the data temporarily in the memory so that the data can be accessed from the memory instead of searching for it in the original location. It increases the efficiency of the application and also increases its speed.

Following are the types of caching:

* Page caching
* Data caching
* Fragment caching

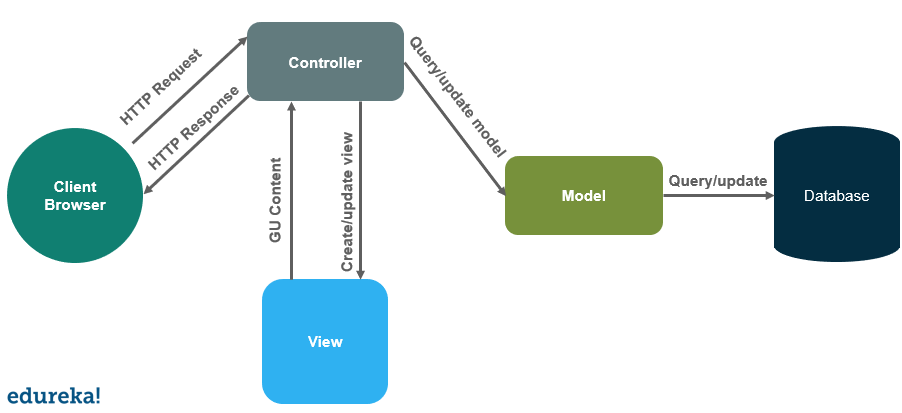
**Q-9: Explain MVC.**

MVC stands for model view controller which is an architecture to build .NET applications.

**Model:**They are the logical part of any application that handles the object storage and retrieval from the databases for an application.

**View:**View handles the UI part of an application. They get the information from the models for their display.

**Controller:**They handle the user interactions, figure out the responses for the user input and also render the view that is required for the user interaction.

****

**Q-10: What is CAS?**

### Programming & Frameworks Training

### [JAVA, J2EE & SOA CERTIFICATION TRAINING](https://www.edureka.co/java-j2ee-training-course" \o "Java, J2EE & SOA Certification Training" \t "_blank)

### [Java, J2EE & SOA Certification Training](https://www.edureka.co/java-j2ee-training-course" \o "Java, J2EE & SOA Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/java-j2ee-training-course" \o "Java, J2EE & SOA Certification Training" \t "_blank)*

**[4](https://www.edureka.co/java-j2ee-training-course" \o "Java, J2EE & SOA Certification Training" \t "_blank)**[(35536)](https://www.edureka.co/java-j2ee-training-course" \o "Java, J2EE & SOA Certification Training" \t "_blank)

### [PYTHON SCRIPTING CERTIFICATION TRAINING](https://www.edureka.co/python-scripting" \o "Python Scripting Certification Training" \t "_blank)

### [Python Scripting Certification Training](https://www.edureka.co/python-scripting" \o "Python Scripting Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/python-scripting" \o "Python Scripting Certification Training" \t "_blank)*

**[5](https://www.edureka.co/python-scripting" \o "Python Scripting Certification Training" \t "_blank)**[(7601)](https://www.edureka.co/python-scripting" \o "Python Scripting Certification Training" \t "_blank)

### [PYTHON DJANGO TRAINING AND CERTIFICATION](https://www.edureka.co/python-django" \o "Python Django Training and Certification" \t "_blank)

### [Python Django Training and Certification](https://www.edureka.co/python-django" \o "Python Django Training and Certification" \t "_blank)

*[Reviews](https://www.edureka.co/python-django" \o "Python Django Training and Certification" \t "_blank)*

**[5](https://www.edureka.co/python-django" \o "Python Django Training and Certification" \t "_blank)**[(3512)](https://www.edureka.co/python-django" \o "Python Django Training and Certification" \t "_blank)

### [COMPREHENSIVE JAVA COURSE CERTIFICATION TRAINING](https://www.edureka.co/java-course-self-paced" \o "Comprehensive Java Course Certification Training" \t "_blank)

### [Comprehensive Java Course Certification Training](https://www.edureka.co/java-course-self-paced" \o "Comprehensive Java Course Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/java-course-self-paced" \o "Comprehensive Java Course Certification Training" \t "_blank)*

**[5](https://www.edureka.co/java-course-self-paced" \o "Comprehensive Java Course Certification Training" \t "_blank)**[(15399)](https://www.edureka.co/java-course-self-paced" \o "Comprehensive Java Course Certification Training" \t "_blank)

### [ADVANCED JAVA CERTIFICATION TRAINING](https://www.edureka.co/advanced-java-sp" \o "Advanced Java Certification Training" \t "_blank)

### [Advanced Java Certification Training](https://www.edureka.co/advanced-java-sp" \o "Advanced Java Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/advanced-java-sp" \o "Advanced Java Certification Training" \t "_blank)*

**[5](https://www.edureka.co/advanced-java-sp" \o "Advanced Java Certification Training" \t "_blank)**[(2928)](https://www.edureka.co/advanced-java-sp" \o "Advanced Java Certification Training" \t "_blank)

### [PHP & MYSQL WITH MVC FRAMEWORKS CERTIFICATION TRAINING](https://www.edureka.co/php-mysql-self-paced" \o "PHP & MySQL with MVC Frameworks Certification Training" \t "_blank)

### [PHP & MySQL with MVC Frameworks Certification Training](https://www.edureka.co/php-mysql-self-paced" \o "PHP & MySQL with MVC Frameworks Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/php-mysql-self-paced" \o "PHP & MySQL with MVC Frameworks Certification Training" \t "_blank)*

**[5](https://www.edureka.co/php-mysql-self-paced" \o "PHP & MySQL with MVC Frameworks Certification Training" \t "_blank)**[(2552)](https://www.edureka.co/php-mysql-self-paced" \o "PHP & MySQL with MVC Frameworks Certification Training" \t "_blank)

### [SPRING FRAMEWORK CERTIFICATION TRAINING](https://www.edureka.co/spring-certification-course" \o "Spring Framework Certification Training " \t "_blank)

### [Spring Framework Certification Training](https://www.edureka.co/spring-certification-course" \o "Spring Framework Certification Training " \t "_blank)

*[Reviews](https://www.edureka.co/spring-certification-course" \o "Spring Framework Certification Training " \t "_blank)*

**[5](https://www.edureka.co/spring-certification-course" \o "Spring Framework Certification Training " \t "_blank)**[(7429)](https://www.edureka.co/spring-certification-course" \o "Spring Framework Certification Training " \t "_blank)

### [MICROSOFT .NET FRAMEWORK CERTIFICATION TRAINING](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \o "Microsoft .NET Framework Certification Training" \t "_blank)

### [Microsoft .NET Framework Certification Training](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \o "Microsoft .NET Framework Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \o "Microsoft .NET Framework Certification Training" \t "_blank)*

**[5](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \o "Microsoft .NET Framework Certification Training" \t "_blank)**[(2666)](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \o "Microsoft .NET Framework Certification Training" \t "_blank)

### [NODE.JS CERTIFICATION TRAINING](https://www.edureka.co/nodejs-certification-training" \o "Node.js Certification Training" \t "_blank)

### [Node.js Certification Training](https://www.edureka.co/nodejs-certification-training" \o "Node.js Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/nodejs-certification-training" \o "Node.js Certification Training" \t "_blank)*

**[5](https://www.edureka.co/nodejs-certification-training" \o "Node.js Certification Training" \t "_blank)**[(5485)](https://www.edureka.co/nodejs-certification-training" \o "Node.js Certification Training" \t "_blank)

Next

CAS stands for code access security, CAS is a part of a security model that prevents unauthorized access to the resources. It also enables the users to set permissions for the code. CLR then executes the code depending upon the permissions.

CAS can only be used for managed code. If an assembly uses CAS it is treated as partially trusted. Although it goes through checks each time an assembly tries to access the resources.

**Q-11: Explain localization and globalization.**

|  |  |
| --- | --- |
| **Localization** | **Globalization** |
| It means changing the already globalized application to cater to a specific language or culture. | Globalization is the process of developing applications to support multiple languages. |
| Microsoft.Extensions.Localization is used to localize the application content. | Existing applications can also be converted to support multiple languages. |

**Q-12: What is the application domain?**

ASP.NET introduces a concept of application domain or AppDomain which is like a lightweight process that acts like both container and boundary. The .NET run-time uses the AppDomain as a container for data and code. The CLR allows multiple .NET applications to run in a single AppDomain.

**Q-13: What is delegate in .NET?**

A delegate in .NET is similar to a function pointer in other programming languages like C or C++. A delegate allows the user to encapsulate the reference of a method in a delegate object. A delegate object can then be passed in a program, which will call the referenced method. We can even use a delegate method to create a custom event in a class.

**Q-14: Difference between interface and abstract class in .NET?**

|  |  |
| --- | --- |
| **Interface** | **Abstract Class** |
| An interface merely declares a contract or behavior that implementing classes should have. | An abstract class provides a partial implementation for a functionality that must be implemented by the inheriting entities. |
| An interface may declare only properties, methods and events with no access modifier. | An abstract class declares fields too. |

Neither interface nor an abstract class can be instantiated.

**Q-15: What is the difference between a stack and a heap?**

|  |  |
| --- | --- |
| **Stack** | **Heap** |
| Stored value type | Stored reference type |
| A stack is responsible for keeping track of each executing thread and its location. | The heap is responsible for keeping track of the more precise objects or data. |

**Q-16: What are the different validators in ASP.NET?**

* **Client-side validation –**When the validation takes place on the client-side browser, it is called client-side validation. Usually, JavaScript is used for client-side validation.
* **Server-side validation –**When the validation takes place on the server then it is called server-side validation. Server-side validation is considered as a secure form of validation because even if the user bypasses the client-side validation we can still catch it in server-side validation.

## **Advanced .NET Interview Questions**

**Q-1:  What are EXE and DLL?**

EXE and DLL are assembly executable modules.

**EXE:**It is an executable file that runs the application for which it is designed. When we build an application, an exe file is generated. Therefore the assemblies are loaded directly when we run an exe. But an exe file cannot be shared with other applications.

**DLL:** It stands for dynamic link library that consists of code that needs to be hidden. The code is encapsulated in this library, an application can have many DLLs and can also be shared with other applications.

**Q-2: What is the difference between function and stored procedure?**

|  |  |
| --- | --- |
| **Function** | **Stored Procedure** |
| Must return a single value | Always used to perform a specific task |
| It can only have the input parameter | It can have both input and output parameters |
| Exception handling is not possible using a try-catch block | Exception handling can be done using a try-catch block |
| A stored procedure cannot be called from a function | A function can be called from a procedure |

**Q-3: List the events in the page life cycle.**

Following are the events in the page life cycle:

* Page\_PreInit
* Page\_Init
* Page\_InitComplete
* Page\_PreLoad
* Page\_Load
* Page\_LoadComplete
* Page\_PreRender
* Render

**Q-4: What is the code to send an email from an ASP.NET application?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | mail message = new mail();  message.From = "abc@gmail.com";  message.To = "xyz@gmail.com";  message.Subject = "Test";  message.Body = "hello";    SmtpMail.SmtpServer = "localhost";  SmtpMail.Send(message); |

**Q-5: What are the event handlers that we have for the Global.asax file?**

* **Application Events:**
  + Application\_Start, Application\_End, Application\_AcquireRequestState, Application\_AuthenticateRequest, Application\_AuthorizeRequest, Application\_BeginRequest, Application\_Disposed, Application\_EndRequest, Application\_Error, Application\_PostRequestHandlerExecute, Application\_PreRequestHandlerExecute, Application\_PreSendRequestContent, Application\_PreSendRequestHeaders, Application\_ReleaseRequestState, Application\_ResolveRequestCache, Application\_UpdateRequestCache
* **Session Events:**
  + Session\_Start, Session\_End

**Q-6: Explain role-based security.**

Role-based security is used to implement security measures based on the role assigned to the users in the organization. Then we can authorize users based on their roles in the organization. For example, windows have role-based access like user, administrators, and guests.

**Q-7: What is cross-page posting?**

Whenever we click on a submit button on a page, the data is stored on the same page. But if the data is stored on a different page, it is known as a cross-page posting.

Cross-page posting can be achieved by POSTBACKURL property which causes the postback.

FindControl method can be used to get the values that are posted on this page to which the page has been posted.

**Q-8: How can we apply themes to an ASP.NET application?**

We can use the web.config file to specify the themes

|  |  |
| --- | --- |
| 1  2  3  4  5 | <cofiguration>  <system.web>  <pages theme="windows"/>  </system.web>  </configuration> |

**Q-9: Explain passport authentication.**

During the passport authentication, it first checks the passport authentication cookie, if the cookie is not available the application redirects to the passport sign on page. Passport service then authenticates the details of the user on the sign on page and if they are valid, stores them on the client machine and then redirects the user to the requested page.

**Q-10: What are ASP.NET security controls?**

* <asp: Login>: Provides a login capability that enables the users to enter their credentials.
* <asp: LoginName>: Allows you to display the name of the logged-in user.
* <asp: LoginStatus>: Displays if the user is authenticated or not.
* <asp: LoginView>: provides various login views depending on the template that has been selected.
* <asp: PasswordRecovery>: Emails the users the lost passwords.

**Q-11: List all the templates of the Repeater control.**

* ItemTemplate
* AlternatingItemTemplate
* SeparatorTemplate
* HeaderTemplate
* FooterTemplate

**Q-12: What is the appSettings section in the web.config file?**

If we want to set the user-defined values for the whole applications, we can use the appSettings block in the web.config file. For example the code below uses the ConnectionString throughout the project for the database connection:

|  |  |
| --- | --- |
| 1  2  3  4 | <em><configuration>  <appsettings>  <add key= "ConnectionString" value="server=local; pwd=password; database=default" />  </appSettings></em> |

**Q-13: What is MIME?**

[[](https://www.edureka.co/microsoft-dotnet-framework-self-paced)](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \t "_blank)

### [Microsoft .NET Framework Certification Training](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \t "_blank)

[Weekday / Weekend BatchesSee Batch Details](https://www.edureka.co/microsoft-dotnet-framework-self-paced" \t "_blank)

MIME stands for multipurpose internet mail extensions, it is the extension of the e-mail protocol which lets users use the protocol to exchange files over the internet.

Servers insert the MIME header at the beginning of the web transmission. Then the clients use this header to select an appropriate ‘player’ for the type of data that the header indicates. Some of these players are built into the web browser.

**Q-14: What is HTTP Handler?**

Every request into an ASP.NET application is handled by a specialized component called HTTP handler. It is the most important component for handling ASP.NET application requests.

It uses different handlers to serve different files. The handler for web page creates the page and control objects, runs your code and then renders the final HTML.

Following are the default HTTP handlers for ASP.NET:

* Page Handler(.aspx): Handles web pages
* User Control Handler(.ascx): It handles web user control pages
* Web Service Handler(.asmx): Handles web service pages
* Trace Handler(trace.axd): It handles trace functionality

**Q-15: What are the different types of cookies in ASP.NET?**

* **Session Cookie:**It resides on the client machine for a single session until the user logs out.
* **Persistent Cookie:**Resides on the user machine for a period specified for its expiry. It may be an hour, a month or never.

**Q-16: What is the difference between ExecuteScalar and ExecuteNonQuery?**

|  |  |
| --- | --- |
| **ExecuteScalar** | **ExecuteNonQuery** |
| Returns the output value | Does not return any value |
| Used for fetching a single value | Used to execute insert and update statements |
| Does not return the number of affected rows | Returns the number of affected rows. |

This brings us to the end of this tutorial. Hope all the questions shared in the article are clear to you.

If you found this article on “.NET Interview Questions” relevant, check out the Edureka*[.NET Certification Training](https://www.edureka.co/microsoft-dotnet-framework-self-paced?qId=9547cca2cd1f4fee76012ed9f2afe6de&index_name=prod_courses&objId=81&objPos=1" \t "_blank)*, a trusted online learning company with a network of more than 250,000 satisfied learners spread across the globe. We are here to help you with every step on your journey and come up with a curriculum that is designed for students and professionals who want to be a .NET developer.

If you come across any questions, feel free to ask all your questions in the comments section of “.NET Interview Questions” and our team will be glad to answer.

<https://www.edureka.co/blog/interview-questions/dot-net-interview-questions/>

**Why .NET Framework?**

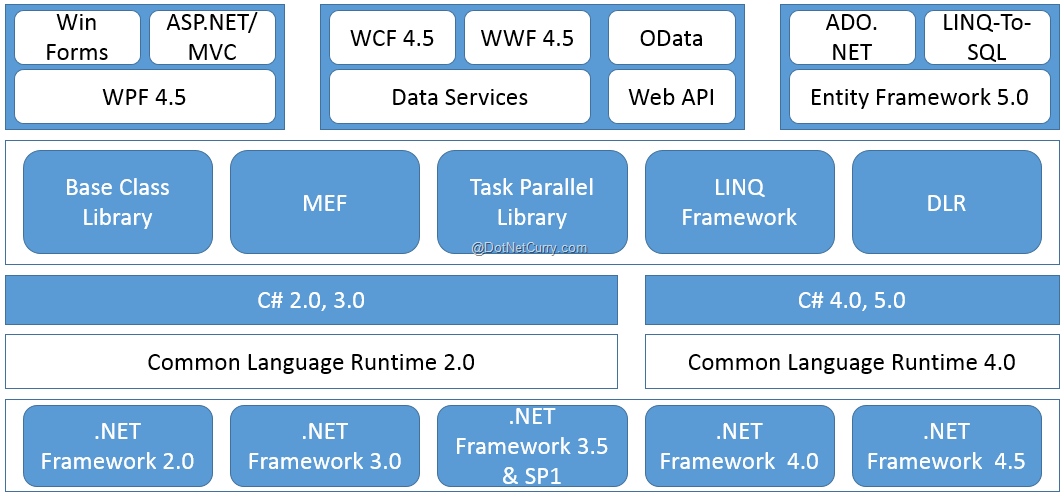
.NET Framework is the most powerful development platform for building a variety of solutions on Windows. For example, using the .NET Framework, you can create applications for Windows Desktop, Web applications, Windows Phone applications, Windows Store applications, Windows Server, Windows Azure (cloud) as well as Bot frameworks and cognitive services.

Before exploring the .NET Framework, we first have to understand the issues/pain areas which developers have faced in other technologies -

* Programming with Win32 API - Earlier we were using Win32 API and C language. This paradigm does not offer Object Oriented Features as C is a structured language. There is no better way of memory management either as it is done manually. Using Pointers is a big pain area for developers.
* Programming with VC++ - Many of us use C++ and MFS as a programming language which offers the features of Object Oriented Features. However C++ is built on top of C language and hence the development can still cumbersome for many novice programmers who are dealing with memory management and pointers.
* Programming with Visual Basic 6.0 - Way back in 1998, VB 6.0 as a development language had made development efforts easier. For example designing complex UI, Data Centric applications and much more. But VB 6.0 is not an object oriented language and is outdated now. When designing Multi-threaded application, we still have to go back to Win32 APIs.
* Programming with COM - Component Object Model is a specification which is used for developing reusable components which can be called across the languages like VC++, Delphi. But with COM we also get "DLL Hell". DLL Hell is the issue of maintaining multiple versions of a component for multiple applications.

Under .NET Framework, many of these problems have been addressed and resolved.

Microsoft .NET Framework provides a huge no. of benefits compared with the legacy languages -

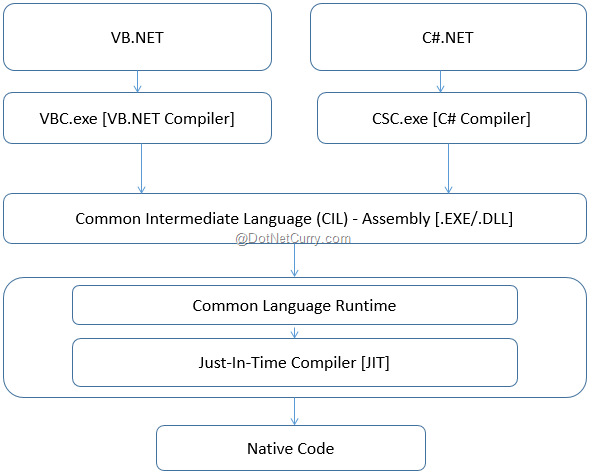
* No more COM Specifications in .NET. So, we automatically get away from DLL Hell.
* Microsoft .NET supports language integration.
* Microsoft .NET Base Class Library offers a wrapper over many raw API calls which can be used in various .NET Languages.
* You can call COM components in .NET and .NET Components in COM using interoperability.
* A Common Runtime Engine shared by all the .NET Languages.   
  A single framework to develop Windows/Web applications.
* We have no. of languages which make developers to adapt the .NET Framework for application development.

**What is BCL?**

The Base Class Library is a Common Language Infrastructure. BCL encapsulates a large number of common functionalities which are available to all the .NET Languages. BCL makes the developers life much simpler while implementing various  functionalities like I/O operations, Data access operations, graphical user interfaces and interfaces to various hardware devices by encapsulating them into various namespaces and classes. It also encapsulates the services which are required by the latest real world applications. .NET Framework applications, components and the controls are built on BCL.

There are a number of namespaces and types available under various class libraries in .NET framework which can be found here <https://msdn.microsoft.com/en-us/library/gg145045(v=VS.110).aspx>

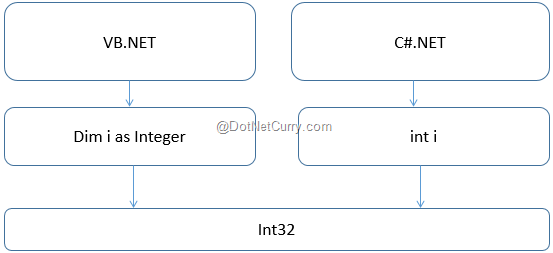
**Explain CLR, CTS, CLS under .NET Framework?**

Common Language Runtime is one of the main building blocks of Microsoft .NET Framework which is responsible for performing various operations under .NET Framework.  
 

When you design the application using Microsoft .NET languages like C#.NET or VB.NET, the language has its own complier which compiles the code into common format that is CIL [Common Intermediate Language]. Then CIL gets targeted to CLR which in turn performs many operations. It converts CIL into Native code with the help of JIT.

**CLR** also performs various other operations like Memory Management, Application Execution, Thread Management, Security checks, load the required assemblies and their types. The code which is managed by CLR is also known as Managed Code. All the managed languages are handled by a single runtime that is CLR.

**Common Type System (CTS)** defines how the types are defined and used in the Common Language Runtime. CTS provides cross-language integration, type safety, and high-performance code execution. The Common Type System for language integration works as follows -

  
   
When you define an integer variable in VB.NET and declare integer in C#.NET, both the languages share the same type which is Int32, available under .NET Framework.

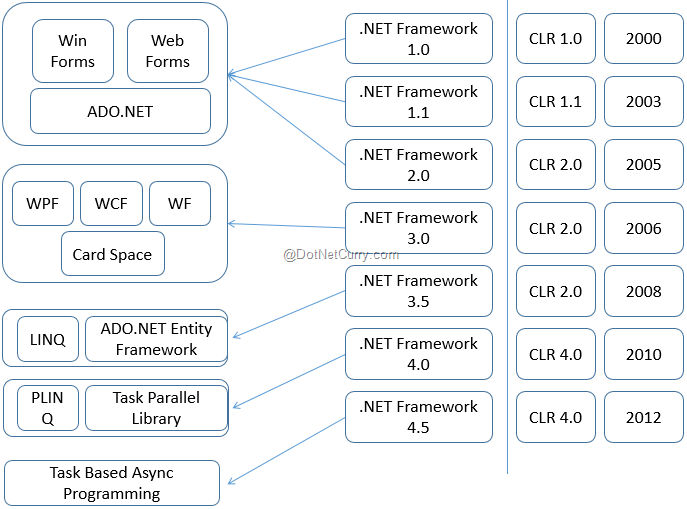
CTS defines rules which must be followed by languages in order to make objects written in one language callable in other languages.

Microsoft .NET Framework provides various primitive data types which can be used while developing applications using various languages.

**Common Language Specification (CLS)** is a set of rules. CLS rules also define a subset of CTS. By defining components using CLS features only, developers are guaranteed to make use of these components in all the other languages which are CLS compliant languages. Most of the types which are defined under .NET framework are CLS complaint.

**What are the different versions of .NET Framework?**

Following are the different versions of the Microsoft .NET Framework -



There are two more versions of .NET Framework released after v4.5. .NET Framework 4.6 was released in July 2015 and came with support for a new just-in-time compiler (JIT) for 64-bit systems called RyuJIT. .NET Framework 4.6.1 was announced in November 2015.

**What is an application server?**

As defined in Wikipedia, an application server is a [software engine](http://en.wikipedia.org/wiki/Software_engine) that delivers [applications](http://en.wikipedia.org/wiki/Computer_application) to [client computers](http://en.wikipedia.org/wiki/Client_computer) or devices. The application server runs your server code. Some well known application servers are IIS (Microsoft), WebLogic Server (BEA), JBoss (Red Hat), WebSphere (IBM).

**Compare C# and VB.NET**

A detailed comparison can be found over [here](http://msdn2.microsoft.com/en-us/library/czz35az4(vs.80).aspx).

**What is a base class and derived class?**

A class is a template for creating an object. The class from which other classes derive fundamental functionality is called a base class. For e.g. If Class Y derives from Class X, then Class X is a base class.

The class which derives functionality from a base class is called a derived class. If Class Y derives from Class X, then Class Y is a derived class.

**What is an extender class?**

An extender class allows you to extend the functionality of an existing control. It is used in Windows forms applications to add properties to controls.

A demonstration of extender classes can be found over [here](http://www.code-magazine.com/Article.aspx?quickid=0301041).

**What is inheritance?**

Inheritance represents the relationship between two classes where one type derives functionality from a second type and then extends it by adding new methods, properties, events, fields and constants.

C# support two types of inheritance:

· Implementation inheritance

· Interface inheritance

**What is implementation and interface inheritance?**

When a class (type) is derived from another class(type) such that it inherits all the members of the base type it is Implementation Inheritance.

When a type (class or a struct) inherits only the signatures of the functions from another type it is Interface Inheritance.

In general Classes can be derived from another class, hence support Implementation inheritance. At the same time Classes can also be derived from one or more interfaces. Hence they support Interface inheritance.

Source: [Exforsys](http://www.exforsys.com/tutorials/csharp/inheritance-in-csharp.html" \t "_blank).

**What is inheritance hierarchy?**

The class which derives functionality from a base class is called a derived class. A derived class can also act as a base class for another class. Thus it is possible to create a tree-like structure that illustrates the relationship between all related classes. This structure is known as the inheritance hierarchy.

**How do you prevent a class from being inherited?**

In VB.NET you use the **NotInheritable**modifier to prevent programmers from using the class as a base class. In C#, use the **sealed**keyword.

**When should you use inheritance?**

Read [this.](http://msdn2.microsoft.com/en-us/library/27db6csx(VS.80).aspx#Mtps_DropDownFilterText)

**Explain Different Types of Constructors in C#?**

There are four different types of constructors you can write in a class -

1. Default Constructor

2. Parameterized Constructor

3. Copy Constructor

4. Static Constructor

Read more about it at <https://www.dotnetcurry.com/csharp/1193/csharp-constructor-types-interview-question>

**Define Overriding?**

Overriding is a concept where a method in a derived class uses the same name, return type, and arguments as a method in its base class. In other words, if the derived class contains its own implementation of the method rather than using the method in the base class, the process is called overriding.

**Can you use multiple inheritance in .NET?**

.NET supports only single inheritance. However the purpose is accomplished using multiple interfaces.

**Why don’t we have multiple inheritance in .NET?**

There are several reasons for this. In simple words, the efforts are more, benefits are less. Different languages have different implementation requirements of multiple inheritance. So in order to implement multiple inheritance, we need to study the implementation aspects of all the languages that are CLR compliant and then implement a common methodology of implementing it. This is too much of efforts. Moreover multiple interface inheritance very much covers the benefits that multiple inheritance has.

**What is an Interface?**

An interface is a standard or contract that contains only the signatures of methods or events. The implementation is done in the class that inherits from this interface. Interfaces are primarily used to set a common standard or contract.

**When should you use abstract class vs interface or What is the difference between an abstract class and interface?**

I would suggest you to read [this](http://msdn2.microsoft.com/en-us/library/scsyfw1d(VS.71).aspx). There is a good comparison given over [here](http://en.csharp-online.net/Should_I_use_an_abstract_class_or_an_interface%3F).

**What are events and delegates?**

An event is a message sent by a control to notify the occurrence of an action. However it is not known which object receives the event. For this reason, .NET provides a special type called Delegate which acts as an intermediary between the sender object and receiver object.

**What is business logic?**

It is the functionality which handles the exchange of information between database and a user interface.

**What is a component?**

Component is a group of logically related classes and methods. A component is a class that implements the IComponent interface or uses a class that implements IComponent interface.

**What is a control?**

A control is a component that provides user-interface (UI) capabilities.

**What are the differences between a control and a component?**

The differences can be studied over[here.](http://msdn2.microsoft.com/en-us/library/0b1dk63b(vs.80).aspx)

**What are design patterns?**

Design patterns are common solutions to common design problems.

**What is a connection pool?**

A connection pool is a ‘collection of connections’ which are shared between the clients requesting one. Once the connection is closed, it returns back to the pool. This allows the connections to be reused.

**What is a flat file?**

A flat file is the name given to text, which can be read or written only sequentially.

**What are functional and non-functional requirements?**

Functional requirements defines the behavior of a system whereas non-functional requirements specify how the system should behave; in other words they specify the quality requirements and judge the behavior of a system.

E.g.

Functional - Display a chart which shows the maximum number of products sold in a region.

Non-functional – The data presented in the chart must be updated every 5 minutes.

**What is an Assembly? Explain different types of Assemblies?**

An assembly is a final deployable unit which can versioned and secured. The assembly can also be termed as a reusable collection of types and resources which work together as a logical unit of functionalities in .NET. .NET assemblies can be designed as executable (.EXE) or reusable component (.DLL). An assembly contains one or more managed types which can be made accessible to the unit or outside the unit.

Assembly gets divided into four different parts.  
•    Manifest.  
•    Type Metadata.  
•    Intermediate Language.  
•    Resources.

Manifest - contains information about the assembly like Version of an assembly, the public key in case the assembly is shared assembly and the culture information. It also contains the security demands to verify this assembly. CLR uses the manifest to load the assembly.

Type Metadata - gives the complete information about the types which are available in the assembly like Class, Structure, Interface, Enum, and the methods, their parameters. The compilers automatically generate this metadata. You can make use of this Type Metadata to dynamically load the types using .NET Reflection.

Intermediate Language - It a code generated by the language specific compiler. It is also known as Machine independent code which can be compiled on one machine and can be deployed on another. CLR targets this code to JIT to convert it into processor depend on code which can be further executed.

Resources - Assembly can also contain the embedded resources like icons, images, string tables media clips.

You can see this information using a .NET framework utility called ILDASM.EXE

Microsoft .NET support different types of assemblies.

•    Private Assembly - Private Assemblies are the assemblies which are only known to the application to which it has been referenced. By default, the assemblies are always private when you create and reference them. The private assembly gets stored in the application folder which is using the assembly.

Private assemblies are identified with the help of name of an assembly and version of an assembly. But the versions does not really come into the picture as the referenced assembly is in the same folder of your application.

•    Shared/Global/Public Assembly - Global/Public Assemblies are the assemblies which are shared across multiple applications. These assemblies are installed into Global Assembly Cache which acts as a shared area for all the assemblies.

Global Assembly is identified with the four-part assembly name - Name of an Assembly, Version of an assembly, and Culture of an Assembly and Public Key Token of an assembly. Global Assembly Cache can contain different versions of an assembly.

You can install a public/global assembly into Global Assembly Cache [GAC] using GACUTIL.EXE tool.

•    Satellite Assembly - Satellite Assemblies are used to build multi-lingual assemblies for applications. Satellite assemblies contain information about the cultures. Satellite assemblies are used to display the data in multiple languages according to Country/Region.

**What is the global assembly cache (GAC)?**

GAC is a machine-wide cache of assemblies that allows .NET applications to share libraries. GAC solves some of the problems associated with dll’s (DLL Hell).

**What is a stack? What is a heap? Give the differences between the two?**

Stack is a place in the memory where value types are stored. Heap is a place in the memory where the reference types are stored.

Check [this](http://www.c-sharpcorner.com/UploadFile/rmcochran/csharp_memory01122006130034PM/csharp_memory.aspx?ArticleID=9adb0e3c-b3f6-40b5-98b5-413b6d348b91) link for the differences.

**What is instrumentation?**

It is the ability to monitor an application so that information about the application’s progress, performance and status can be captured and reported.

**What is code review?**

The process of  examining the source code generally through a peer, to verify it against best practices.

**What is logging?**

Logging is the process of persisting information about the status of an application.

**What are mock-ups?**

Mock-ups are a set of designs in the form of screens, diagrams, snapshots etc., that helps verify the design and acquire feedback about the application’s requirements and use cases, at an early stage of the design process.

**What is a Form?**

A form is a representation of any window displayed in your application. Form can be used to create standard, borderless, floating, modal windows.

**What is a multiple-document interface(MDI)?**

A user interface container that enables a user to work with more than one document at a time. E.g. Microsoft Excel.

**What is a single-document interface (SDI) ?**

A user interface that is created to manage graphical user interfaces and controls into single windows. E.g. Microsoft Word

**What is BLOB ?**

A BLOB (binary large object) is a large item such as an image or an exe  represented in binary form.

**What is ClickOnce?**

ClickOnce is a new deployment technology that allows you to create and publish self-updating applications that can be installed and run with minimal user interaction.

**What is object role modeling (ORM) ?**

It is a logical model for designing and querying database models. There are various ORM tools in the market like CaseTalk, Microsoft Visio for Enterprise Architects, Infagon etc.

**What is a private assembly?**

A private assembly is local to the installation directory of an application and is used only by that application.

**What is a shared assembly?**

A shared assembly is kept in the global assembly cache (GAC) and can be used by one or more applications on a machine.

**What is the difference between user and custom controls?**

User controls are easier to create whereas custom controls require extra effort.

User controls are used when the layout is static whereas custom controls are used in dynamic layouts.

A user control cannot be added to the toolbox whereas a custom control can be.

A separate copy of a user control is required in every application that uses it whereas since custom controls are stored in the GAC, only a single copy can be used by all applications.

**Where do custom controls reside?**

In the global assembly cache (GAC).

**What is a third-party control ?**

A third-party control is one that is not created by the owners of a project. They are usually used to save time and resources and reuse the functionality developed by others (third-party).

**What is a binary formatter?**

Binary formatter is used to serialize and deserialize an object in binary format.

**What is Boxing/Unboxing?**

Boxing is used to convert value types to object.

E.g. int x = 1;

object obj = x ;

Unboxing is used to convert the object back to the value type.

E.g. int y = (int)obj;

Boxing/unboxing is quiet an expensive operation.

**What is a COM Callable Wrapper (CCW)?**

CCW is a wrapper created by the common language runtime(CLR) that enables COM components to access .NET objects.

**What is a Runtime Callable Wrapper (RCW)?**

RCW is a wrapper created by the common language runtime(CLR) to enable .NET components to call COM components.

**What is a digital signature?**

A digital signature is an electronic signature used to verify/gurantee the identity of the individual who is sending the message.

**What is garbage collection and explain its different generations?**

Garbage collector is a part of Common Language Runtime, which does automatic memory management for your application. When you create an object in your application, CLR allocates the memory for that object on Managed Heap.

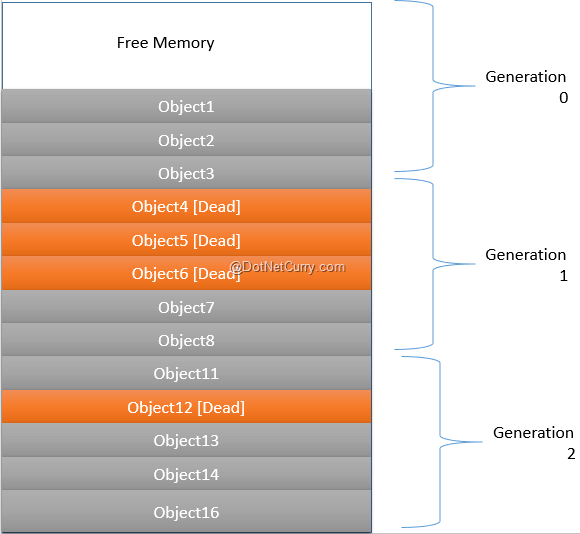
Garbage collector gives number of benefits like -

Automatic Memory Management - You can build your application without thinking about how to free the memory as Garbage Collector gets called automatically by CLR.

Garbage Collector does proficient memory management for your objects.

Garbage Collector does automatic reclaim of the memory for those objects which are not in use and which are marked for deletion.

Garbage collector allocates the memory for objects in such a way that one object will not be able to use other object data accidently.

  
   
The Managed Heap is a memory area where a series of managed objects are stored and managed. As shown in the above diagram, the managed heap gets divided into three different sections which are known as Generations.

**Generation 0** - This is the first generation under managed heap which is used to store and manage short-lived objects. All small size objects usually get allocated on Generation 0. When garbage collector reclaims the memory, it always reclaims the memory from generation 0 by default. The objects which are survived in Generation 0 will be pushed towards the generation 1.

**Generation 1** - This generation again contain the short-lived objects and the objects which are survived from Generation 0. The objects which are survived in Generation 1 will be pushed towards the generation 2.

**Generation 2** - This generation contains the long lived objects which are survived from multiple generations and are used till the process is running.

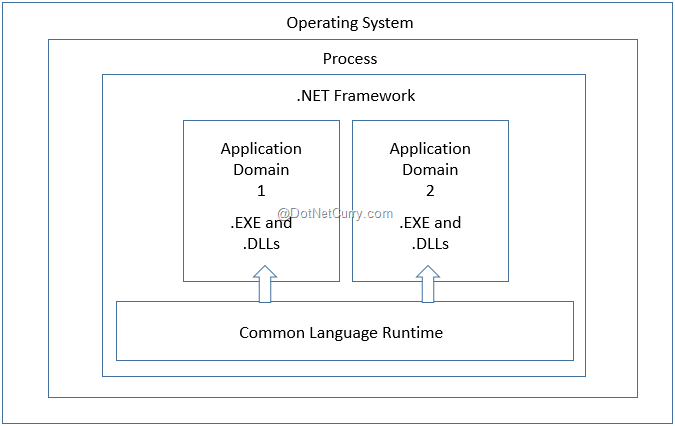
Garbage Collector first looks for the free memory in generation 0 which can be used to allocate the memory for the new object. The objects are always created in contagious memory. If the garbage collector finds sufficient memory for the new object, it does not search for the dead object and hence memory allocation process is always faster. But in case, sufficient memory is not available; then garbage collector reclaims the memory of the objects which are dead or not getting used for a long time.

When Garbage collector gets executed, it first of all, finds all the live objects. After this garbage collector updates the addresses of objects which will be compacted. Then it reclaims the memory of all the dead objects. Once the memory is reclaimed, it moves the lived objects to the next Generation. For example, the objects which live through in Generation 0 will be moved towards Generation 1. The objects which survived in generation 1 will be moved towards Generation 2. The objects which survived in Generation 2 will stay in Generation 2 only.

**What is Application Domain and how does it work?**

Windows Operating Systems load a set of resources like .EXE, DLLs and allocate the memory for those resources in an area called as Process. Windows OS creates a separate and isolated area for each running application. Making separate isolation area for each application, makes the process more secure and stable. In case, one process fails, it does not affect the other process.

.NET applications, however, are not hosted like traditional applications by Windows Operating System. Under .NET, .EXEs are hosted under a process by logical partitioning which is known as "Application Domain". Now you can host multiple application domains under one single process.

  
   
Application Domains consume less memory and power for processing the applications compared to the traditional processes for each application. In case one application domain fails, the other application domain will work as-is without any effects. You can also stop one application domain without affecting another application domain or without stopping an entire process.

One Application Domain cannot share/access the data from other Application Domain which is running within the same Domain or other Domain.

**What is globalization?**

Globalization is the process of customizing applications that support multiple cultures and regions.

**What is localization?**

Localization is the process of customizing applications that support a given culture and regions.

**What is MIME?**

The definition of MIME or Multipurpose Internet Mail Extensions as stated in MSDN is “MIME is a standard that can be used to include content of various types in a single message. MIME extends the Simple Mail Transfer Protocol (SMTP) format of mail messages to include multiple content, both textual and non-textual. Parts of the message may be images, audio, or text in different character sets. The MIME standard derives from RFCs such as 2821 and 2822”. Quoted from [here.](http://support.microsoft.com/kb/836555)

I hope you liked these questions and I thank you for viewing them. I thank [Pravin Dabade](https://www.dotnetcurry.com/author/pravinkumar-dabade) for contributing some of the .NET Interview question and answers.

*This article has been editorially reviewed by [Suprotim Agarwal.](https://www.dotnetcurry.com/author/suprotim-agarwal)*

## ASP.NET Interview Questions and Answers – Experienced Developers

**What is XHTML? Are ASP.NET Pages compliant with XHTML?**

In simple words, XHTML is a stricter and cleaner version of HTML. XHTML stands for EXtensible Hypertext Markup Language and is a W3C Recommendation.

Yes, ASP.NET 2.0 Pages are XHTML compliant. However the freedom has been given to the user to include the appropriate document type declaration.

*More info can be found at*[*http://msdn2.microsoft.com/en-us/library/exc57y7e.aspx*](http://msdn2.microsoft.com/en-us/library/exc57y7e.aspx)

**Can I deploy the application without deploying the source code on the server?**

Yes. You can obfuscate your code by using a new precompilation process called ‘precompilation for deployment’. You can use the aspnet\_compiler.exe to precompile a site. This process builds each page in your web application into a single application DLL and some placeholder files. These files can then be deployed to the server.

You can also accomplish the same task using Visual Studio 2005 by using the Build->Publish menu.

**Does ViewState affect performance? What is the ideal size of a ViewState? How can you compress a viewstate?**

Viewstate stores the state of controls in HTML hidden fields. At times, this information can grow in size. This does affect the overall responsiveness of the page, thereby affecting performance. The ideal size of a viewstate should be not more than 25-30% of the page size.

[Viewstate can be compressed](https://www.dotnetcurry.com/ShowArticle.aspx?ID=67) to almost 50% of its size. .NET also provides the **GZipStream** or**DeflateStream** to compress viewstate. Another option is explained by Scott Hanselmann over [here](http://www.hanselman.com/blog/ZippingCompressingViewStateInASPNET.aspx).

**How can you detect if a viewstate has been tampered?**

By setting the EnableViewStateMac to true in the @Page directive. This attribute checks the encoded and encrypted viewstate for tampering.

You can also view ViewState information as described in this article [How to view information in ViewState using ASP.NET 2.0 and 3.5](https://www.dotnetcurry.com/ShowArticle.aspx?ID=112)

**Can I use different programming languages in the same application?**

Yes. Each page can be written with a different programming language in the same application. You can create a few pages in C# and a few in VB.NET.

**Can the App\_Code folder contain source code files in different programming languages?**

No. All source code files kept in the root App\_Code folder must be in the same programming language.

Update: However, you can create two subfolders inside the App\_Code and then add both C# and VB.NET in the respective subfolders.  You also have to add configuration settings in the web.config for this to work.

**How do you secure your connection string information?**

By using the Protected Configuration feature.

**How do you secure your configuration files to be accessed remotely by unauthorized users?**

ASP.NET configures IIS to deny access to any user that requests access to the Machine.config or Web.config files.

**How can I configure ASP.NET applications that are running on a remote machine?**

You can use the Web Site Administration Tool to configure remote websites.

**How many web.config files can I have in an application?**

You can keep multiple web.config files in an application. You can place a Web.config file inside a folder or wherever you need (apart from some exceptions) to override the configuration settings that are inherited from a configuration file located at a higher level in the hierarchy.

**I have created a configuration setting in my web.config and have kept it at the root level. How do I prevent it from being overridden by another web.config that appears lower in the hierarchy?**

By setting the element's Override attribute to false.

**What is the difference between Response.Write and Response.Output.Write?**

As quoted by Scott Hanselman, the short answer is that the latter gives you String.Format-style output and the former doesn't.

To get a detailed explanation, follow this [link](http://www.hanselman.com/blog/ASPNETResponseWriteAndResponseOutputWriteKnowTheDifference.aspx)

**What is Cross Page Posting? How is it done?**

By default, ASP.NET submits a form to the same page. In cross-page posting, the form is submitted to a different page. This is done by setting the ‘PostBackUrl’ property of the button(that causes postback) to the desired page. In the code-behind of the page to which the form has been posted, use the ‘FindControl’method of the ‘PreviousPage’ property to reference the data of the control in the first page.

**Can you change a Master Page dynamically at runtime? How?**

Yes. To change a master page, set the MasterPageFile property to point to the .master page during the PreInit page event.

**How do you apply Themes to an entire application?**

By specifying the theme in the web.config file.

|  |
| --- |
| <configuration>    <system.web>    <pages theme=”BlueMoon” />    </system.web>    </configuration> |

**How do you exclude an ASP.NET page from using Themes?**

To remove themes from your page, use the EnableTheming attribute of the Page directive.

**Your client complains that he has a large form that collects user input. He wants to break the form into sections, keeping the information in the forms related. Which control will you use?**

The ASP.NET Wizard Control.

*To learn more about this control, visit this*[*link*](http://weblogs.asp.net/scottgu/archive/2006/02/21/438732.aspx)*.*

**Do webservices support data reader?**

No. However it does support a dataset.

**What is use of the AutoEventWireup attribute in the Page directive ?**

The AutoEventWireUp is a boolean attribute that allows automatic wireup of page events when this attribute is set to true on the page. It is set to True by default for a C# web form whereas it is set as False for VB.NET forms. Pages developed with Visual Studio .NET have this attribute set to false, and page events are individually tied to handlers.

**What happens when you change the web.config file at run time?**

ASP.NET invalidates the existing cache and assembles a new cache. Then ASP.NET automatically restarts the application to apply the changes.

**Can you programmatically access IIS configuration settings?**

Yes. You can use ADSI, WMI, or COM interfaces to configure IIS programmatically.

**How does Application Pools work in IIS 6.0?**

As explained under the [IIS documentation](http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/cde42787-982a-478c-b319-e703b270d8ee.mspx?mfr=true), when you run IIS 6.0 in worker process isolation mode, you can separate different Web applications and Web sites into groups known as *application pools*. An *application pool* is a group of one or more URLs that are served by a worker process or set of worker processes. Any Web directory or virtual directory can be assigned to an application pool.

Every application within an application pool shares the same worker process. Because each worker process operates as a separate instance of the worker process executable, W3wp.exe, the worker process that services one application pool is separated from the worker process that services another. Each separate worker process provides a process boundary so that when an application is assigned to one application pool, problems in other application pools do not affect the application. This ensures that if a worker process fails, it does not affect the applications running in other application pools.

Use multiple application pools when you want to help ensure that applications and Web sites are confidential and secure. For example, an enterprise organization might place its human resources Web site and its finance Web site on the same server, but in different application pools. Likewise, an ISP that hosts Web sites and applications for competing companies might run each company’s Web services on the same server, but in different application pools. Using different application pools to isolate applications helps prevent one customer from accessing, changing, or using confidential information from another customers site.

In HTTP.sys, an application pool is represented by a *request queue*, from which the user-mode worker processes that service an application pool collect the requests. Each pool can manage requests for one or more unique Web applications, which you assign to the application pool based on their URLs. Application pools, then, are essentially worker process configurations that service groups of namespaces.

Multiple application pools can operate at the same time. An application, as defined by its URL, can only be served by one application pool at any time. While one application pool is servicing a request, you cannot route the request to another application pool. However, you can assign applications to another application pool while the server is running.

**Note:** This ASP.NET Questions and Answers for Experienced Developers have been taken from forums, my colleagues and my own experience of conducting interviews. I have tried to mention the contributor wherever possible. If you would like to contribute, kindly use the [Contact form](https://www.dotnetcurry.com/Contact.aspx). If you think that a credit for a contribution is missing somewhere, kindly use the same contact form and I will do the needful.

This article has been editorially reviewed by [Suprotim Agarwal.](https://www.dotnetcurry.com/author/suprotim-agarwal)

## ASP.NET Interview Q&A for Freshers

**What is ASP.NET?**

Microsoft ASP.NET is an open source server side technology that enables programmers to build dynamic Web sites, web applications, and Web services. It is a part of the .NET based environment and is built on the Common Language Runtime (CLR) . So programmers can write ASP.NET code using any .NET compatible language.

[ASP.NET Core](https://www.dotnetcurry.com/aspnet/1239/aspnet-5-introduction-conceptual-overview) is the next successor to ASP.NET. It was first called as ASP.NET 5, but to avoid confusing developers that it is an update to the existing ASP.NET framework, Microsoft renamed it to ASP.NET Core. It has been re-implemented as a modular framework. ASP.NET Core uses the .NET Compiler Platform ("Roslyn") which is open source and cross platform. You can read about ASP.NET Core over here <https://www.dotnetcurry.com/aspnet/1239/aspnet-5-introduction-conceptual-overview>

**What are the differences between ASP.NET 1.1 and ASP.NET 2.0?**

A comparison chart containing the differences between ASP.NET 1.1 and ASP.NET 2.0 can be found over [here](http://www.codeproject.com/aspnet/ComparisonASP1xASP20.asp).

**Which is the latest version of ASP.NET? What were the previous versions released?**

The latest version of ASP.NET is 2.0. There have been 3 versions of ASP.NET released as of date. They are as follows :

v1.0 – Released on January 16, 2002.

v1.1 – Released on April 24, 2003.

v2.0 – Released on November 7, 2005.

v3.0 – Released on November 1, 2006

v3.5 – Released on November 19, 2007

v3.5 SP1 - Released on August 11, 2008

v4.0 – Released on April 12, 2010

v4.5 – Released on August 15, 2012

v4.5.1 – Released on October 17, 2013

v4.5.2 -  Released on May 5, 2014

v4.6 - Released on July 20, 2015

ASP.NET Core (Currently in RC1) – Released on November 18, 2015

**Explain the Event Life cycle of ASP.NET 2.0?**

The events occur in the following sequence. Its best to turn on tracing(<% @Page Trace=”true”%>) and track the flow of events :

**PreInit**– This event represents the entry point of the page life cycle. If you need to change the Master page or theme programmatically, then this would be the event to do so. Dynamic controls are created in this event.

**Init** – Each control in the control collection is initialized.

**Init Complete**\* - Page is initialized and the process is completed.

**PreLoad**\* - This event is called before the loading of the page is completed.

**Load** – This event is raised for the Page and then all child controls. The controls properties and view state can be accessed at this stage. This event indicates that the controls have been fully loaded.

**LoadComplete**\* - This event signals indicates that the page has been loaded in the memory. It also marks the beginning of the rendering stage.

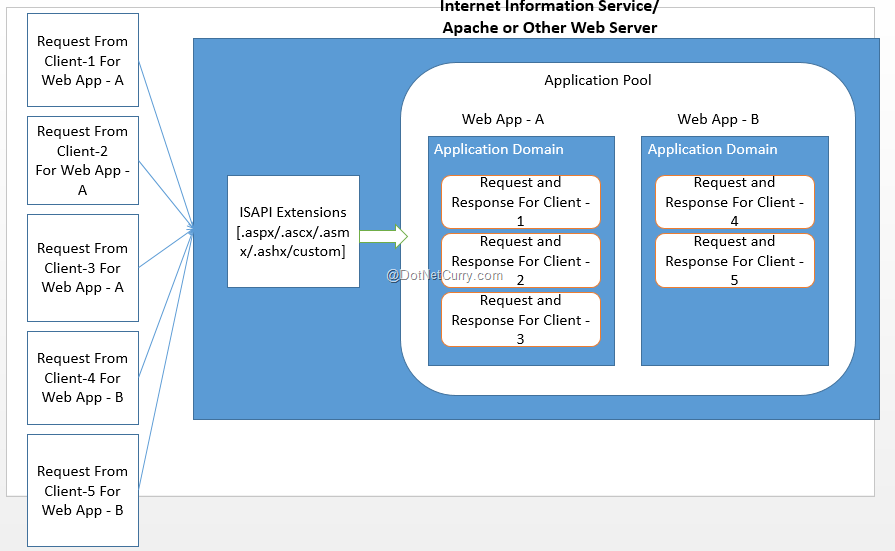
**PreRender** – If you need to make any final updates to the contents of the controls or the page, then use this event. It first fires for the page and then for all the controls.

**PreRenderComplete**\* - Is called to explicitly state that the PreRender phase is completed.

**SaveStateComplete**\* - In this event, the current state of the control is completely saved to the ViewState.

**Unload**– This event is typically used for closing files and database connections. At times, it is also used for logging some wrap-up tasks.

The events marked with \* have been introduced in ASP.NET 2.0. Here’s an overview of the Application Lifecycle architecture.



To understand the life cycle in detail with diagrams, refer to our article [**ASP.NET Application and Page Life Cycle Overview**](https://www.dotnetcurry.com/aspnet/1263/aspnet-application-page-life-cycle-interview-question)

**You have created an ASP.NET Application. How will you run it?**

With ASP.NET 2.0, Visual Studio comes with an inbuilt ASP.NET Development Server to test your pages. It functions as a local Web server. The only limitation is that remote machines cannot access pages running on this local server. The second option is to deploy a Web application to a computer running IIS version 5 or 6 or 7.

ASP.NET Core applications are host agnostic i.e they can be run on IIS or even standalone.

**Explain the AutoPostBack feature in ASP.NET?**

AutoPostBack allows a control to automatically postback when an event is fired. For eg: If we have a Button control and want the event to be posted to the server for processing, we can set AutoPostBack = True on the button.

**How do you disable AutoPostBack?**

Hence the AutoPostBack can be disabled on an ASP.NET page by disabling AutoPostBack on all the controls of a page. AutoPostBack is caused by a control on the page.

**What are the different code models available in ASP.NET 2.0?**

There are 2 code models available in ASP.NET 2.0. One is the single-file page and the other one is the code behind page.

**Which base class does the web form inherit from?**

Page class in the System.Web.UI namespace.

**Which are the new special folders that are introduced in ASP.NET 2.0?**

There are seven new folders introduced in ASP.NET 2.0 :

\App\_Browsers folder – Holds browser definitions(.brower) files which identify the browser and their capabilities.

\App\_Code folder – Contains source code (.cs, .vb) files which are automatically compiled when placed in this folder. Additionally placing web service files generates a proxy class(out of .wsdl) and a typed dataset (out of .xsd).

\App\_Data folder – Contains data store files like .mdf (Sql Express files), .mdb, XML files etc. This folder also stores the local db to maintain membership and role information.

\App\_GlobalResources folder – Contains assembly resource files (.resx) which when placed in this folder are compiled automatically. In earlier versions, we were required to manually use the resgen.exe tool to compile resource files. These files can be accessed globally in the application.

\App\_LocalResources folder – Contains assembly resource files (.resx) which can be used by a specific page or control.

\App\_Themes folder – This folder contains .css and .skin files that define the appearance of web pages and controls.

\App\_WebReferences folder – Replaces the previously used Web References folder. This folder contains the .disco, .wsdl, .xsd files that get generated when accessing remote web services.

**Explain the ViewState in ASP.NET?**

Http is a stateless protocol. Hence the state of controls is not saved between postbacks. Viewstate is the means of storing the state of server side controls between postbacks. The information is stored in HTML hidden fields. In other words, it is a snapshot of the contents of a page.

You can disable viewstate by a control by setting the EnableViewState property to false.

**What does the EnableViewState property signify?**

EnableViewState saves the state of an object in a page between postbacks. Objects are saved in a Base64 encoded string. If you do not need to store the page, turn it off as it adds to the page size.

*There is an excellent*[*article*](http://www.eggheadcafe.com/articles/20060208.asp)*by Peter Bromberg to understand Viewstate in depth.*

**Explain the ASP.NET Page Directives?**

Page directives configure the runtime environment that will execute the page. The complete list of directives is as follows:

@ Assembly - Links an assembly to the current page or user control declaratively.

@ Control - Defines control-specific attributes used by the ASP.NET page parser and compiler and can be included only in .ascx files (user controls).

@ Implements - Indicates that a page or user control implements a specified .NET Framework interface declaratively.

@ Import - Imports a namespace into a page or user control explicitly.

@ Master - Identifies a page as a master page and defines attributes used by the ASP.NET page parser and compiler and can be included only in .master files.

@ MasterType - Defines the class or virtual path used to type the Master property of a page.

@ OutputCache - Controls the output caching policies of a page or user control declaratively.

@ Page - Defines page-specific attributes used by the ASP.NET page parser and compiler and can be included only in .aspx files.

@ PreviousPageType - Creates a strongly typed reference to the source page from the target of a cross-page posting.

@ Reference - Links a page, user control, or COM control to the current page or user control declaratively.

@ Register - Associates aliases with namespaces and classes, which allow user controls and custom server controls to be rendered when included in a requested page or user control.

This list has been taken from [here](http://msdn2.microsoft.com/en-us/library/t8syafc7(vs.80).aspx).

**Explain the Validation Controls used in ASP.NET 2.0?**

Validation controls allows you to validate a control against a set of rules. There are 6 different validation controls used in ASP.NET 2.0.

RequiredFieldValidator – Checks if the control is not empty when the form is submitted.

CompareValidator – Compares the value of one control to another using a comparison operator (equal, less than, greater than etc).

RangeValidator – Checks whether a value falls within a given range of number, date or string.

RegularExpressionValidator – Confirms that the value of a control matches a pattern defined by a regular expression. Eg: Email validation.

CustomValidator – Calls your own custom validation logic to perform validations that cannot be handled by the built in validators.

ValidationSummary – Show a summary of errors raised by each control on the page on a specific spot or in a message box.

**How do you identify that the page is post back?**

By checking the IsPostBack property. If IsPostBack is True, the page has been posted back.

**What are Master Pages?**

Master pages is a template that is used to create web pages with a consistent layout throughout your application. Master Pages contains content placeholders to hold page specific content. When a page is requested, the contents of a Master page are merged with the content page, thereby giving a consistent layout.

**How is a Master Page different from an ASP.NET page?**

The MasterPage has a @Master top directive and contains ContentPlaceHolder server controls. It is quiet similar to an ASP.NET page.

**How do you attach an existing page to a Master page?**

By using the MasterPageFile attribute in the @Page directive and removing some markup.

|  |
| --- |
| <%@ Page Language="C#" MasterPageFile="~/Template.master" AutoEventWireup="true"  CodeFile="About.aspx.cs" Inherits="UI.About"  Title="Your Page Title" %> |

**How do you set the title of an ASP.NET page that is attached to a Master Page?**

By using the Title property of the @Page directive in the content page. Eg:

|  |
| --- |
| <@Page MasterPageFile="Sample.master" Title="I hold content" %> |

**What is a nested master page? How do you create them?**

A Nested master page is a master page associated with another master page. To create a nested master page, set the MasterPageFile attribute of the @Master directive to the name of the .master file of the base master page.

**What are Themes?**

Themes are a collection of CSS files, .skin files, and images. They are text based style definitions and are very similar to CSS, in that they provide a common look and feel throughout the website.

**What are skins?**

A theme contains one or more skin files. A skin is simply a text file with a .skin extension and contains definition of styles applied to server controls in an ASP.NET page. For eg:

<asp:button runat="server" BackColor="blue" BorderColor="Gray" Font-Bold ="true"ForeColor="white"/>

Defines a skin that will be applied to all buttons throughout to give it a consistent look and feel.

**What is the difference between Skins and Css files?**

Css is applied to HTML controls whereas skins are applied to server controls.

**What is a User Control?**

User controls are reusable controls, similar to web pages. They cannot be accessed directly.

**Explain briefly the steps in creating a user control?**

* Create a file with .ascx extension and place the @Control directive at top of the page.
* Included the user control in a Web Forms page using a @Register directive

**What is a Custom Control?**

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.

**What are the differences between user and custom controls?**

User controls are easier to create in comparison to custom controls, however user controls can be less convenient to use in advanced scenarios.

User controls have limited support for consumers who use a visual design tool whereas custom controls have full visual design tool support for consumers.

A separate copy of the user control is required in each application that uses it whereas only a single copy of the custom control is required, in the global assembly cache, which makes maintenance easier.

A user control cannot be added to the Toolbox in Visual Studio whereas custom controls can be added to the Toolbox in Visual Studio.

User controls are good for static layout whereas custom controls are good for dynamic layout.

**Where do you store your connection string information?**

The connection string can be stored in configuration files (web.config).

**What is the difference between ‘Web.config’ and ‘Machine.config’?**

Web.config files are used to apply configuration settings to a particular web application whereasmachine.config file is used to apply configuration settings for all the websites on a web server.

Web.config files are located in the application's root directory or inside a folder situated in a lower hierarchy. The machine.config is located in the Windows directory Microsoft.Net\Framework\*Version*\CONFIG.

There can be multiple web.config files in an application nested at different hierarchies. However there can be only one machine.config file on a web server.

**What is the difference between Server.Transfer and Response.Redirect?**

Response.Redirect involves a roundtrip to the server whereas Server.Transfer conserves server resources by avoiding the roundtrip. It just changes the focus of the webserver to a different page and transfers the page processing to a different page.

Response.Redirect can be used for both .aspx and html pages whereas Server.Transfer can be used only for .aspx pages.

Response.Redirect can be used to redirect a user to an external websites. Server.Transfer can be used only on sites running on the same server. You cannot use Server.Transfer to redirect the user to a page running on a different server.

Response.Redirect changes the url in the browser. So they can be bookmarked. Whereas Server.Transfer retains the original url in the browser. It just replaces the contents of the previous page with the new one.

**What method do you use to explicitly kill a users session?**

Session.Abandon().

**What is a webservice?**

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.

Check out some more ASP.NET Interview Questions and Answers for Freshers and Experienced Developers at [**ASP.NET 2.0 Interview Questions - Intermediate Level**](https://www.dotnetcurry.com/dotnetinterview/63/aspnet-interview-questions-answers-for-experienced)

**Note:** I have tried to mention the contributor wherever possible. If you would like to contribute, kindly use the [Contact form](https://www.dotnetcurry.com/Contact.aspx). If you think that a credit for a contribution is missing somewhere, kindly use the same contact form and I will do the needful.

Check out some other .NET, ASP.NET and C# interview questions and answers for developers over here:

[General .NET Interview Questions](https://www.dotnetcurry.com/ShowArticle.aspx?ID=70)

[.NET Framework Interview Questions](https://www.dotnetcurry.com/ShowArticle.aspx?ID=64)

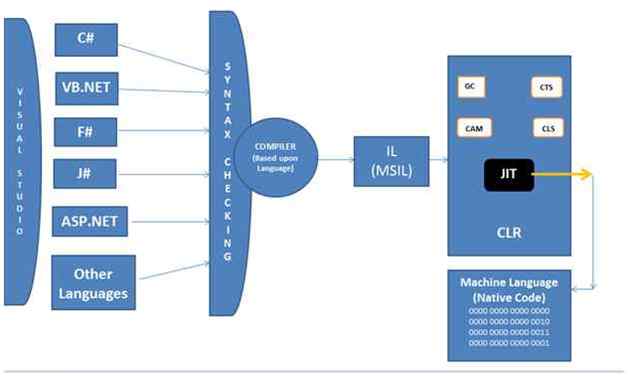
[ASP.NET 2.0 Interview Questions - Intermediate Level](https://www.dotnetcurry.com/dotnetinterview/63/aspnet-interview-questions-answers-for-experienced)

This article has been editorially reviewed by [Suprotim Agarwal.](https://www.dotnetcurry.com/author/suprotim-agarwal)

## **Question 1. What is the .NET Framework?**

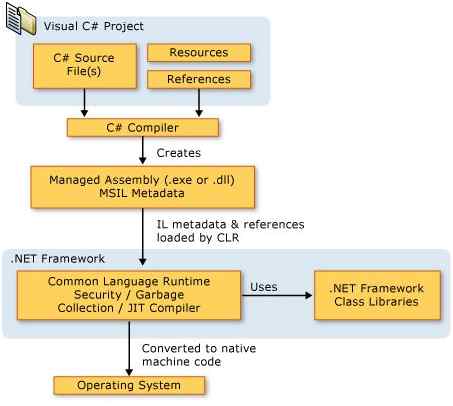
**Answer:**The .NET is a Framework, which is a collection of classes of reusable libraries given by Microsoft to be used in other .NET applications and to develop, build and deploy many types of applications on the Windows platform including the following:

* Console Applications
* Windows Forms Applications
* Windows Presentation Foundation (WPF) Applications
* Web Applications
* Web Services
* Windows Services
* Services-oriented applications using Windows Communications Foundation (WCF)
* Workflow-enabled applications using Windows Workflow Foundation(WF)

That primarily runs on the Microsoft Windows Operating System.  
  


* [.NET Framework Overview](http://www.c-sharpcorner.com/UploadFile/cb1429/net-framework-overview/)

## **Question 2. What is CLR?**

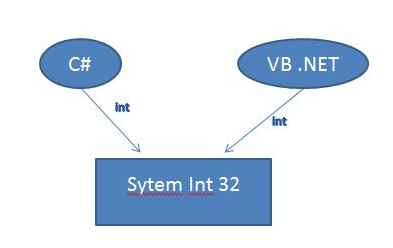
**Answer:**The CLR stands for Common Language Runtime and it is an Execution Environment. It works as a layer between Operating Systems and the applications written in .NET languages that conforms to the Common Language Specification (CLS). The main function of Common Language Runtime (CLR) is to convert the Managed Code into native code and then execute the program. The Managed Code compiled only when it is needed, that is it converts the appropriate instructions when each function is called. The Common Language Runtime (CLR)’s just in time (JIT) compilation converts Intermediate Language (MSIL) to native code on demand at application run time.  
  
When a .NET application is executed at that time the control will go to Operating System, then Operating System create a process to load **CLR.**  
The program used by the operating system for loading CLR is called runtime host, which are different depending upon the type of application that is desktop or web based application i.e.  
  
The runtime host for **desktop applications** is API function called **CorbinToRuntime**.  
  
The runtime host for **web based**applications is ASP.NET worker process **(aspnet-wp.exe)**.  
  
  
  
CLR runtime engine comes with set of services, which are classified as follows  
  
**CLR services**

* Assembly Resolver
* Assembly Loader
* Type Checker
* COM marshalled
* Debug Manager
* Thread Support
* IL to Native compiler
* Exception Manager
* Garbage Collector

To know more about them follow the link:

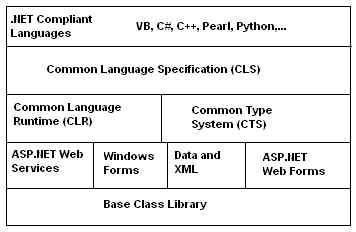
* [Introduction to CLR in .NET framework](http://www.c-sharpcorner.com/Blogs/10261/introduction-to-clr-in-net-framework.aspx)

## **Question 3. What is CTS?**

**Answer:**The Common Type System (CTS) standardizes the data types of all programming languages using .NET under the umbrella of .NET to a common data type for easy and smooth communication among these .NET languages.  
  
  
  
To implement or see how CTS is converting the data type to a common data type, for example, when we declare an int type data type in C# and VB.NET, then they are converted to int32. In other words, now both will have a common data type that provides flexible communication between these two languages.  
  
For more details follow the link:

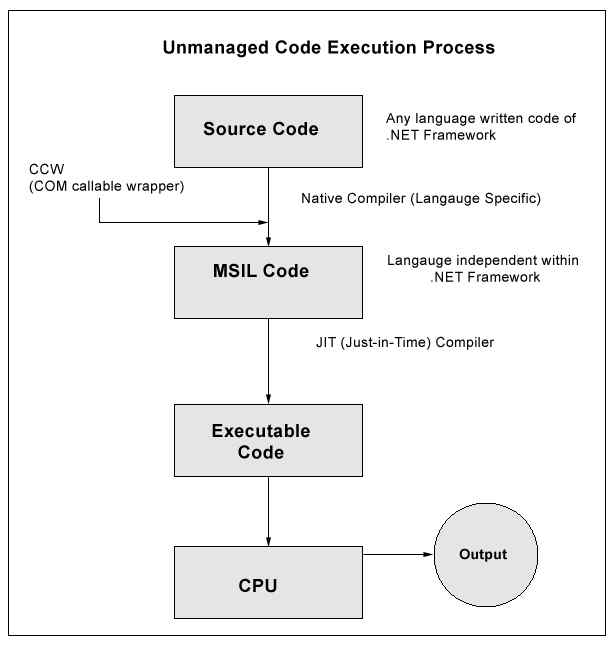
* [Common Type System (CTS) in .NET](http://www.c-sharpcorner.com/UploadFile/cb1429/cts-common-type-system-in-net/)

## **Question 4. What is CLS?**

**Answer**: One of the important goals of .NET Framework is to support Multiple Languages. This is achieved by CLS. For multiple languages to interoperate, it is necessary that they should go on in common in certain features such as Types that are used. For example, every language has its own size and range for different data types. Thus CLS is the agreement among language designers and class library designers concerning these usage conventions.  
  
  
  
Follow the link for more details:

* [What does .NET really mean? – Understanding .NET](http://www.c-sharpcorner.com/UploadFile/saragana/DotNETreallymeanUnderstandingNET11292005005026AM/DotNETreallymeanUnderstandingNET.aspx)

## **Question 5. What is managed code?**

**Answer:** The resource, which is within your application domain is, managed code. The resources that are within domain are faster.  
  
The code, which is developed in .NET framework, is known as managed code. This code is directly executed by CLR with help of managed code execution. Any language that is written in .NET Framework is managed code.  
  
Managed code uses CLR which in turn looks after your applications by managing memory, handling security, allowing cross - language debugging, and so on.  
  
  
  
For more details follow the link:

* [Managed code and unmanaged code in .NET](http://www.c-sharpcorner.com/uploadfile/puranindia/managed-code-and-unmanaged-code-in-net/)

## **Question 6. What is MSIL?**

**Answer:**When we compile our .NET code then it is not directly converted to native/binary code; it is first converted into intermediate code known as MSIL code which is then interpreted by the CLR. MSIL is independent of hardware and the operating system. Cross language relationships are possible since MSIL is the same for all .NET languages. MSIL is further converted into native code.  
  
MSIL  
  
For more details follow the link:

* [.Net Architecture and .Net Framework Basics](http://www.c-sharpcorner.com/uploadfile/09f663/net-architecture-and-net-framework-basics/)

## **Question 7. What is JIT?**

**Answer**: A Web Service or Web Forms file must be compiled to run within the CLR. Compilation can be implicit or explicit. Although you could explicitly call the appropriate compiler to compile your Web Service or Web Forms files, it is easier to allow the file to be complied implicitly. Implicit compilation occurs when you request the .asmx via HTTP-SOAP, HTTP-GET, or HTTP-POST. The parser (xsp.exe) determines whether a current version of the assembly resides in memory or in the disk. If it cannot use an existing version, the parser makes the appropriate call to the respective compiler (as you designated in the **Class** property of the .asmx page).  
  
When the Web Service (or Web Forms page) is implicitly compiled, it is actually compiled twice. On the first pass, it is compiled into IL. On the second pass, the Web Service (now an assembly in IL) is compiled into machine language. This process is called Just-In-Time JIT compilation because it does not occur until the assembly is on the target machine.  
  
**JIT Types:**  
  
  
  
Follow the link for more details:

* [JIT (Just-In-Time) Compiler](http://www.c-sharpcorner.com/UploadFile/nipuntomar/jit-just-in-time-compiler/)

## **Question 8. What is portable executable (PE)?**

**Answer:**Every .NET program first compiles with an appropriate compiler like if we write a program in C# language then it gets compiled by C# compiler (i.e. csc.exe).  
  
In .NET framework every program executes (communicate) in an operating system by using CLR (Common Language Runtime).  
  
  
  
Managed module is standard windows Portable Executable (PE) file which contains the following parts.

* **PE Header**  
    
  It is similar to common object file format.
* **CLR Header**  
    
  This contains CLR version required to run this managed module, location & metadata. This also contains entry point of function i.e. the address of entry point of function.
* **Metadata**  
    
  This contains table information means variable with its data types and default values, functions / methods which are declared & defined in our program.

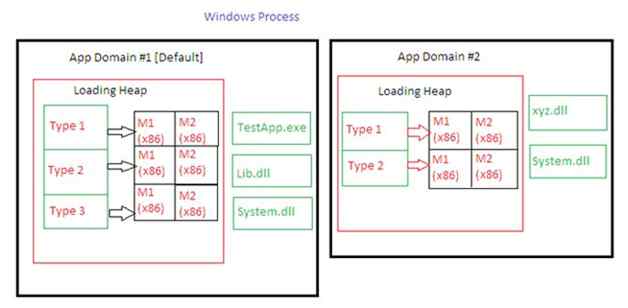
Follow the link for more details:

* [C# Program Compilation Steps](http://www.c-sharpcorner.com/UploadFile/a8024d/C-Sharp-program-compliation-steps/)

## **Question 9. What is an application domain?**

**Answer:** An Application Domain is a logical container for a set of assemblies in which an executable is hosted. As you have seen, a single process may contain multiple Application Domains, each of which is hosting a .NET executable. The first appdomain created when the CLR is initialized is called the default AppDomain and this default one is destroyed when the Windows process is terminated.

* An AppDomain can be independently secured.
* An AppDomain can be unloaded.
* Independently configured.
* No mutual intervention by multiple appdomains.
* Performance

**  
  
How does an AppDomain get created**  
  
The AppDomain class is used to create and terminate Application Domains, load and unload assemblies and types and enumerates assemblies and threads in a domain. The following table shows some useful methods of the AppDomain class:

|  |  |
| --- | --- |
| Methods | Description |
| CreateDomain() | It allows us to create a new Application Domain. |
| CreateInstance() | Creates an instance of type in an external assembly. |
| ExecuteAssembly() | It executes an \*.exe assembly in the Application Domain. |
| Load() | This method dynamically loads an assembly into the current app domain. |
| UnLoad() | It allows us to unload a specified AppDomain within a given process. |
| GetCurrentThread() | Returns the ID of the active thread in the current Application Domain. |

In addition, the AppDomain class also defined as a set of properties that can be useful when you wish to monitor the activity of a given Application Domain.

|  |  |
| --- | --- |
| Properties | Description |
| CurrentDomain | Gets the Application Domain for the currently executing thread. |
| FriendlyName | Gets the friendly name of the current Application Domain. |
| SetupInformation | Get the configuration details for a given Application Domain. |
| BaseDirectory | Gets the directory path that the assembly resolver uses to probe for assemblies. |

Follow the link for more details:

* [.NET Application Domain Internals](http://www.c-sharpcorner.com/UploadFile/ajyadav123/net-application-domain-internal/)

## **Question 10. What is an assembly?**

**Answer:** An Assembly is a basic building block of .NET Framework applications. It is basically compiled code that can be executed by the CLR. An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality. An Assembly can be a DLL or exe depending upon the project that we choose.  
  
Assemblies are basically the following two types:

1. Private Assembly
2. Shared Assembly

Follow the link for more details:

* [What is An Assembly](http://www.c-sharpcorner.com/UploadFile/78607b/what-is-assembly/)

## **Question 11. What are the contents of assembly?**

**Answer:**Assembly

* An Assembly is a basic unit of application deployment and versioning.
* An Assembly is also called the building block of a .NET application.
* An Assembly is either a .exe or .dll file.

An Assembly structure consists of the following parts:

* Assembly manifest (name, language and version).
* CIL code (logic part).
* Type information (Datatype).
* Resources.

Follow the link for more details:

* [Assemblies in C# : Part 1](http://www.c-sharpcorner.com/UploadFile/736bf5/assembly-in-C-Sharp-part-1/)

## **Question 12. What are the different types of assembly?**

**Answer**: An Assembly contains metadata and manifest information. The reason for the emergence of assembly concept was to overcome the common "**DLL Hell**" problem in COM. The assembly contains all the code, resources, metadata and even version information. Metadata contains the details of every "type" inside the assembly. In addition to metadata, assemblies also have a special file called Manifest. It contains information about the current version of the assembly, culture information, public key token if available and other related information.  
  
There are in all 3 different types of assemblies:

1. Private Assembly
2. Shared or Strong named assembly
3. Satellite assembly

Follow the link for more details:

* [Assembly in .NET: Part 1](http://www.c-sharpcorner.com/uploadfile/851045/assembly-in-net-part-1/)

## **Question 13. What is a dynamic assembly?**

**Answer:**Technically, the act of loading external assemblies on demand is known as Dynamic Loading. Using the Assembly class, we can dynamically load both private and shared assemblies from the local location to a remote location as well as, explore its properties.  
  
To illustrate dynamic loading, we are creating a console based application that loads an external TestLib.dll assembly. During the execution, the application asks the user to specify the dynamic loading assembly name and that reference is passed to the helper method that is responsible for loading the assembly.  
  
Follow the link for more details:

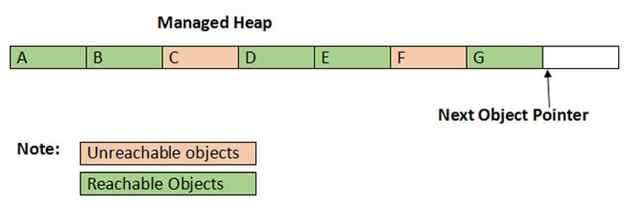
* [Using Reflection with C# .NET](http://www.c-sharpcorner.com/UploadFile/84c85b/using-reflection-with-C-Sharp-net/)

## **Question 14. What is GAC?**

**Answer:** The GAC is a shared location of computer where we can put an assembly so that it will be accessible from many locations, I mean it is accessible from another project or application. It's always a good practice to provide a strong name to a public assembly, I mean the assembly to be registered in the GAC,  otherwise the DLL hell problem may occur.  
  
**Problems that occurred**  
I have seen DLLs added to the GAC that you can't remove - very frustrating. I have seen registered DLLs into the cache - verified everything is there ok using ILDASM only to find the DLLs are no longer in the GAC.  
  
**Strongly naming the assembly**  
When doing this make sure you get the directory slashes \\ correct within the assembly file (assembly.cs). - if not, you will get errors whilst the code is looking for the .snk file. If you get errors which leave you scratching your head - best bet is to remove the .snk file and start over.  
  
**Project References**  
Also be careful and watch where you build projects as the referenced DLLs can easily be built to the development instead of the release folder - sometimes even when you specify the release folder. This can be very, very frustrating.  
  
**Conclusion**  
My conclusion on using the GAC was only use it if you really need to as it isn't the 'end of DLL hell' as first thought. Also only use it if you are using a DLL that is shared by other projects. Don't put it in the GAC if you don't have to.  
  
For more details follow the link:

* [Global Assembly Cache(GAC) Hell](http://www.c-sharpcorner.com/UploadFile/gsuttie/GACHell09092005073302AM/GACHell.aspx)

## **Question 15. What is a garbage collector?**

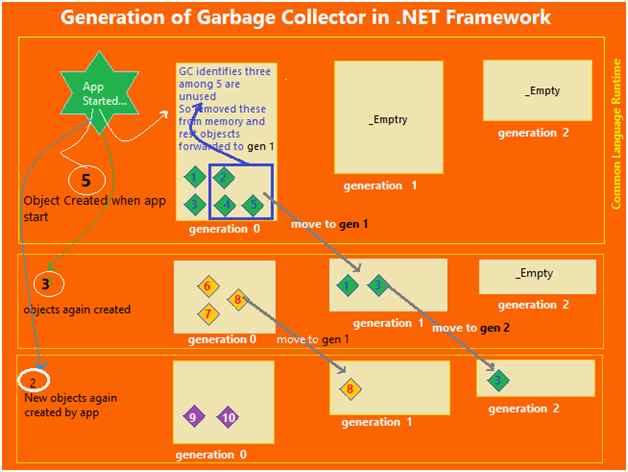
**Answer:**The Garbage Collector (GC) is the part of the .NET Framework that allocates and releases memory for your .NET applications. The Common Language Runtime (CLR) manages allocation and deallocation of a managed object in memory. C# programmers never do this directly, there is no delete keyword in the C# language. It relies on the garbage collector.  
  
**Example**  
Assume the managed heap contains a set of objects named A, B, C, D, E, F and G. During garbage collection, these objects are examined for active roots. After the graph has been constructed, unreachable objects (that we will assume are objects C and F) are marked as garbage in reddish color in the following diagram.  
  
  
  
For more details follow the link:

* [.Net Memory Management](http://www.c-sharpcorner.com/UploadFile/26b237/net-memory-management/)

## **Question 16. What are generations and how are they used by the garbage collector?**

**Answer**: Basically the generation of Garbage Collection (GC) shows the life of objects, it means it defines how long an object will stay in the memory. It's categorized into the following three generations:

* Generation 0
* Generation 1
* Generation 2

  
  
For more details follow the link:

* [Monitoring the Activities of Garbage Collection in .NET Using CLR Profiler](http://www.c-sharpcorner.com/UploadFile/7ca517/monitoring-the-activities-of-garbage-collection-in-net-usin/)

<https://www.c-sharpcorner.com/UploadFile/puranindia/interview-question-on-net-framework-or-clr/>

**Q) .net vs java**

|  |  |  |
| --- | --- | --- |
| **Feature** | **MS .Net** | **Java/JEE** |
| Operating System | Windows | Multiple OS |
| Common Language Runtime | Common Language Runtime | Java Virtual Machine |
| XML | System XML | Java API for XML Processing |
| Naming | Java Naming and Directory Interface | Active Directory Service Interfaces |
| HTTP Engine | Internet Information Services | Application Servers from various vendors |
| Server Components | .Net, COM+Services | Enterprise Java Beans |

**Q1) What is the relation between Classes and Objects?**

Class applies to a type or model of a GROUP of itedot-ms, objects, or concepts.  
Object applies to a specific material item or concept, a group of which may comprise a class.

**Q2) What is an IL?**

(IL) Intermediate Language is also known as MSIL is MicroSoft Intermediate Language or Common Language Runtime (CLR) . When we compile .Net applications, its complied to MSIL, which is not machine read language. Hence Common Language Runtime (CLR) with JustIn Time Complier (JIT) , converts this MSIL to native code (binary code) , which is machine language.

**Q3) What are the defining traits of an object-oriented language?**

The defining traits of an object-oriented language are:

* Inheritance
* Abstraction
* Encapsulation
* Polymorphism

**1. Inheritance**: The main class or the root class is called as a Base Class. Any class which is expected to have ALL properties of the base class along with its own is called as a Derived class. The process of deriving such a class is Derived class.

**2. Abstraction**: Abstraction is creating models or classes of some broad concept. Abstraction can be achieved through Inheritance or even Composition.

**3. Encapsulation:** Encapsulation is a collection of functions of a class and object. The “Food” class is an encapsulated form. It is achieved by specifying which class can use which members (private, public, protected) of an object.

**4. Polymorphism**: Polymorphism means existing in different forms. Inheritance is an example of Polymorphism. A base class exists in different forms as derived classes. Operator overloading is an example of Polymorphism in which an operator can be applied in different situations.

**Q4) What is the concept of DISPOSE method?**

DISPOSE method belongs to IDisposable interface. It is used to free unmanaged resources like files, network connection etc. It manages and handles this by an instance of the class that implements this interface. Dispose methods must be called explicitly and hence the any object using IDisposable must also implement finalizer to free resources in situations wherein Dispose is not called. Multiple calls to dispose method must be ignored when called once. The objects disposable methods must be called in the order of containment.

**Q5) What is a CLR?**

The Common Language Runtime (CLR) is a core component of .NET framework. It is Microsoft’s implementation of the Common Language Infrastructure (CLI) standard, which defines an execution environment for program code. In the CLR, code is expressed in a form of bytecode called the Common Intermediate Language (CIL) . Developers using the CLR write code in a language such as C# or VB.NET. At compile time, the .NET compiler converts such code into CIL code. At runtime, the CLR’s just-in-time compiler converts the CIL code into code native to the operating system. Alternatively, the CIL code can be compiled to native code in a separate step prior to runtime by using the Native Image Generator (NGEN) . This speeds up all later runs of the software as the CIL-to-native compilation is no longer necessary.

During the execution of the program, the Common Language Runtime (CLR) manages memory, Thread execution, Garbage Collection (GC) , Exception Handling, Common Type System (CTS) , code safety verifications, and other system services. The Common Language Runtime (CLR) environment is also referred to as a managed environment, because during the execution of a program it also controls the interaction with the Operating System.

**Q6) What is CTS?**

CTS stands for Common Type System. The CTS makes available a common set of data types so that compiled code of one language could easily interoperate with compiled code of another language by understanding each others’ data types. If two languages (c# or vb.net or j# or vc++) want to communicate with each other, they have to convert into some common type (i.e. in COMMON LANGUAGE RUNTIME) . In C# we use int which is converted to Int32 of CLR to communicate with vb.net which uses Integer or vice versa.

**Q7) What is a CLS (Common Language Specification)?**

CLS is a specification that defines the rules to support language integration. This is done in such a way, that programs written in any language (.NET compliant) can communicate with one another. This also can take full advantage of inheritance, polymorphism, exceptions, and other features. This is a subset of the CTS, which all .NET languages are expected to support.

**Q8) Difference between Abstract Classes and Interfaces.**

Following are the differences between Abstract Classes and Interfaces:

1. When a derived class is inherited from an Abstract class, no other class can be extended then. Interface can be used in any scenario.
2. Abstract class contains abstract method, i.e. actual implementation logic. On the other hand, interfaces have no implementation logic.
3. Every method in an interface must be abstract. This is not necessary in case of abstract classes.

**Q9) What is an Assembly?**

In the .NET framework, an assembly is a partially compiled code library for use in deployment, versioning and security. There are two types: process assemblies (EXE) and library assemblies (DLL). A process assembly represents a process which will use classes defined in library assemblies. .NET assemblies contain code in CIL, which is usually generated from a CLI language, and then compiled into machine language at runtime by the CLR just-in-time compiler.  
An assembly can consist of one or more files. Code files are called modules. An assembly can contain more than one code module and since it is possible to use different languages to create code modules it is technically possible to use several different languages to create an assembly. Visual Studio however does not support using different languages in one assembly.

[Check Out .Net Tutorials](https://mindmajix.com/net)

**Q10) What are the various objects in Dataset?**

The DataSet class exists in the System.Data namespace.  
The Classes contained in the DataSet class are:

1. DataTable
2. DataColumn
3. DataRow
4. Constraint
5. DataRelation

**Q11) What is a NameSpace?**

Namespace is a group of classes, structures, interfaces, enumerations, and delegates, organized in a logical hierarchy by function, that enable you to access the core functionality you need in your applications.

Namespaces are the way that .NET avoids name clashes between classes. A Namespace is no more than a grouping of data types, but it has the effect that the names of all data types within a namespace automatically get prefixed with the name of the namespace. It is also possible to nest namespaces within each other.

**Q12) What is the difference between NameSpace and Assembly?**

A Namespace is a logical naming scheme for types in which a simple type name, such as MyType, is preceded with a dot-separated hierarchical name. Such a naming scheme is completely under control of the developer. The .NET Framework uses a hierarchical naming scheme for grouping types into logical categories of related functionality, such as the ASP.NET application framework, or remoting functionality. Design tools can make use of namespaces to make it easier for developers to browse and reference types in their code.

The concept of a namespace is not related to that of an Assembly. A single assembly may contain types whose hierarchical names have different namespace roots, and a logical namespace root may span multiple assemblies. In the .NET Framework, a Namespace is a logical design-time naming convenience, whereas an Assembly establishes the name scope for types at run time.

# .Net (dot-net) interview questions for Fresher and experienced. ->

##### **What is .NET?**

-->  
.NET is essentially a framework for software development. It is similar in nature to any other software development framework (J2EE etc) in that it provides a set of runtime containers/capabilities, and a rich set of pre-built functionality in the form of class libraries and APIs  
The .NET Framework is an environment for building, deploying, and running Web Services and other applications. It consists of three main parts: the Common Language Runtime, the Framework classes, and ASP.NET.

##### **How many languages .NET is supporting now?**

-->  
When .NET was introduced it came with several languages. VB.NET, C#, COBOL and Perl, etc. .net Supports More than 44 languages.

##### **How is .NET able to support multiple languages?**

-->  
A language should comply with the Common Language Runtime standard to become a .NET language. In .NET, code is compiled to Microsoft Intermediate Language (MSIL for short). This is called as Managed Code. This Managed code is run in .NET environment. So after compilation to this IL the language is not a barrier. A code can call or use a function written in another language.

##### **What is an application server?**

-->  
As defined in Wikipedia, an application server is a software engine that delivers applications to client computers or devices. The application server runs your server code. Some well known application servers are IIS (Microsoft), WebLogic Server (BEA), JBoss (Red Hat), WebSphere (IBM).

##### **What is inheritance?**

-->  
Inheritance represents the relationship between two classes where one type derives functionality from a second type and then extends it by adding new methods, properties, events, fields and constants. C# support two types of inheritance:  
- Implementation inheritance  
- Interface inheritance

##### **What is implementation and interface inheritance?**

-->  
When a class (type) is derived from another class(type) such that it inherits all the members of the base type it is Implementation Inheritance.  
When a type (class or a struct) inherits only the signatures of the functions from another type it is Interface Inheritance.  
In general Classes can be derived from another class, hence support Implementation inheritance. At the same time Classes can also be derived from one or more interfaces. Hence they support Interface inheritance.

##### **What is inheritance hierarchy?**

-->  
The class which derives functionality from a base class is called a derived class. A derived class can also act as a base class for another class. Thus it is possible to create a tree-like structure that illustrates the relationship between all related classes. This structure is known as the inheritance hierarchy.

##### **How do you prevent a class from being inherited?**

-->  
In VB.NET you use the NotInheritable modifier to prevent programmers from using the class as a base class. In C#, use the sealed keyword.

##### **Define Overriding?**

-->  
Overriding is a concept where a method in a derived class uses the same name, return type, and arguments as a method in its base class. In other words, if the derived class contains its own implementation of the method rather than using the method in the base class, the process is called overriding.

##### **Can you use multiple inheritance in .NET?**

-->  
.NET supports only single inheritance. However the purpose is accomplished using multiple interfaces.

##### **What is an Interface?**

-->  
An interface is a standard or contract that contains only the signatures of methods or events. The implementation is done in the class that inherits from this interface. Interfaces are primarily used to set a common standard or contract.

##### **What is business logic?**

-->  
It is the functionality which handles the exchange of information between database and a user interface.

##### **What is a component?**

-->  
Component is a group of logically related classes and methods. A component is a class that implements the IComponent interface or uses a class that implements IComponent interface.

##### **What is a control?**

-->  
A control is a component that provides user-interface (UI) capabilities.

##### **What are design patterns?**

-->  
Design patterns are common solutions to common design problems.

##### **What is a connection pool?**

-->  
A connection pool is a ‘collection of connections’ which are shared between the clients requesting one. Once the connection is closed, it returns back to the pool. This allows the connections to be reused.

##### **What is a flat file?**

-->  
A flat file is the name given to text, which can be read or written only sequentially.

##### **What are functional and non-functional requirements?**

-->  
Functional requirements defines the behavior of a system whereas non-functional requirements specify how the system should behave; in other words they specify the quality requirements and judge the behavior of a system.E.g.  
  
Functional - Display a chart which shows the maximum number of Customer in a region.  
Non-functional – The data presented in the chart must be updated every 60 minutes.

##### **What is the global assembly cache (GAC)?**

-->  
GAC is a machine-wide cache of assemblies that allows .NET applications to share libraries. GAC solves some of the problems associated with dll’s (DLL Hell).

##### **What is Boxing/Unboxing?**

-->  
  
Boxing is used to convert value types to object.  
E.g. int x=1;  
object obj=x ;  
Unboxing is used to convert the object back to the value type.  
E.g. int y=(int)obj;  
Boxing/unboxing is quiet an expensive operation.

##### **What is globalization?**

-->  
Globalization is the process of customizing applications that support multiple cultures and regions.

##### **What is localization?**

-->  
Localization is the process of customizing applications that support a given culture and regions.

##### **What is MIME?**

-->  
The definition of MIME or Multipurpose Internet Mail Extensions as stated in MSDN is “MIME is a standard that can be used to include content of various types in a single message. MIME extends the Simple Mail Transfer Protocol (SMTP) format of mail messages to include multiple content, both textual and non-textual. Parts of the message may be images, audio, or text in different character sets. The MIME standard derives from RFCs such as 2821 and 2822”.

##### **How ASP .NET different from ASP?**

-->  
Scripting is separated from the HTML, Code is compiled as a DLL, these DLLs can be executed on the server.

##### **What is smart navigation?**

-->  
The cursor position is maintained when the page gets refreshed due to the server side validation and the page gets refreshed.

##### **What is view state?**

-->  
The web is stateless. But in ASP.NET, the state of a page is maintained in the in the page itself automatically. How? The values are encrypted and saved in hidden controls. this is done automatically by the ASP.NET. This can be switched off / on for a single control

##### **How do you validate the controls in an ASP .NET page?**

-->  
Using special validation controls that are meant for this. We have Range Validator, Email Validator.

##### **Can the validation be done in the server side? Or this can be done only in the Client side?**

-->  
Client side is done by default. Server side validation is also possible. We can switch off the client side and server side can be done.

##### **How to manage pagination in a page?**

-->  
Using pagination option in DataGrid control. We have to set the number of records for a page, then it takes care of pagination by itself.

##### **What is ADO .NET and what is difference between ADO and ADO.NET?**

-->  
ADO.NET is stateless mechanism. I can treat the ADO.Net as a separate in-memory database where in I can use relationships between the tables and select insert and updates to the database. I can update the actual database as a batch.

##### **What is a Manifest?**

-->  
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 (Portable Executable) file (an .exe or .dll) with Microsoft intermediate language (MSIL) code or in a standalone PE (Portable Executable) file that contains only assembly manifest information. The following table shows the information contained in the assembly manifest. The first four items the assembly name, version number, culture, and strong name information make up the assembly's identity.  
Assembly name: A text string specifying the assembly's name.  
Version number: A major and minor version number, and a revision and build number. The common language runtime uses these numbers to enforce version policy.  
Culture: Information on the culture or language the assembly supports. This information should be used only to designate an assembly as a satellite assembly containing culture- or language-specific information. (An assembly with culture information is automatically assumed to be a satellite assembly.) Strong name information: The public key from the publisher if the assembly has been given a strong name. List of all files in the assembly:  
A hash of each file contained in the assembly and a file name. Note that all files that make up the assembly must be in the same directory as the file containing the assembly manifest.  
Type reference information: Information used by the runtime to map a type reference to the file that contains its declaration and implementation. This is used for types that are exported from the assembly.  
Information on referenced assemblies: A list of other assemblies that are statically referenced by the assembly. Each reference includes the dependent assembly's name, assembly metadata (version, culture, operating system, and so on), and public key, if the assembly is strong named.

##### **What is the difference between "using System.Data;" and directly adding the reference from "Add References Dialog Box"?**

-->  
When u compile a program using command line, u add the references using /r switch. When you compile a program using Visual Studio, it adds those references to our assembly, which are added using "Add Reference" dialog box. While "using" statement facilitates us to use classes without using their fully qualified names.  
For example: if u have added a reference to "System.Data.SqlClient" using "Add Reference" dialog box then u can use SqlConnection class like this:  
System.Data.SqlClient.SqlConnection  
But if u add a "using System.Data.SqlClient" statement at the start of ur code then u can directly use SqlConnection class.On the other hand if u add a reference using "using System.Data.SqlClient" statement, but don't add it using "Add Reference" dialog box, Visual Studio will give error message while we compile the program.

##### **What is a Metadata?**

-->  
Metadata is information about a PE. In COM, metadata is communicated through non-standardized type libraries.  
In .NET, this data is contained in the header portion of a COFF-compliant PE and follows certain guidelines; it contains information such as the assembly’s name, version, language (spoken, not computer , culture), what external types are referenced, what internal types are exposed, methods, properties, classes, and much more.  
The CLR uses metadata for a number of specific purposes. Security is managed through a public key in the PE’s header.  
Information about classes, modules, and so forth allows the CLR to know in advance what structures are necessary. The class loader component of the CLR uses metadata to locate specific classes within assemblies, either locally or across networks.  
Just-in-time (JIT) compilers use the metadata to turn IL into executable code.  
Other programs take advantage of metadata as well.  
A common example is placing a Microsoft Word document on a Windows 2000 desktop. If the document file has completed comments, author, title, or other Properties metadata, the text is displayed as a tool tip when a user hovers the mouse over the document on the desktop. You can use the Ildasm.exe utility to view the metadata in a PE. Literally, this tool is an IL disassemble.

##### **What is "Common Type System" (CTS)?**

-->  
CTS defines all of the basic types that can be used in the .NET Framework and the operations performed on those type.All this time we have been talking about language interoperability, and .NET Class Framework. None of this is possible without all the language sharing the same data types. What this means is that an int should mean the same in VB, VC++, C# and all other .NET compliant languages. This is achieved through introduction of Common Type System (CTS).

##### **What is "Common Language Specification" (CLS)?**

-->  
CLS is the collection of the rules and constraints that every language (that seeks to achieve .NET compatibility) must follow. It is a subsection of CTS and it specifies how it shares and extends one another libraries.

##### **What is "Common Language Runtime" (CLR)?**

-->  
CLR is .NET equivalent of Java Virtual Machine (JVM). It is the runtime that converts a MSIL code into the host machine language code, which is then executed appropriately. The CLR is the execution engine for .NET Framework applications. It provides a number of services, including:  
- Code management (loading and execution)  
- Application memory isolation  
- Verification of type safety  
- Conversion of IL to native code.  
- Access to metadata (enhanced type information)  
- Managing memory for managed objects  
- Enforcement of code access security  
- Exception handling, including cross-language exceptions  
- Interoperation between managed code, COM objects, and pre-existing DLL's (unmanaged code and data)  
- Automation of object layout  
- Support for developer services (profiling, debugging, and so on).

##### **What is the difference between a namespace and assembly name?**

-->  
A namespace is a logical naming scheme for types in which a simple type name, such as MyType, is preceded with a dot-separated hierarchical name. Such a naming scheme is completely under control of the developer. For example, types MyOffice.FileAccess.A and MyOffice.FileAccess.B might be logically expected to have functionally related to file access. The .NET Framework uses a hierarchical naming scheme for grouping types into logical categories of related functionality, such as the ASP.NET application framework, or remoting functionality. Design tools can make use of namespaces to make it easier for developers to browse and reference types in their code. The concept of a namespace is not related to that of an assembly. A single assembly may contain types whose hierarchical names have different namespace roots, and a logical namespace root may span multiple assemblies. In the .NET Framework, a namespace is a logical design-time naming convenience, whereas an assembly establishes the name scope for types at run time.

##### **What’s a Windows process?**

-->  
It’s an application that’s running and had been allocated memory.

##### **What’s typical about a Windows process in regards to memory allocation?**

-->  
Each process is allocated its own block of available RAM space, no process can access another process’ code or data. If the process crashes, it dies alone without taking the entire OS or a bunch of other applications down.

##### **Explain what relationship is between a Process, Application Domain, and Application?**

-->  
Each process is allocated its own block of available RAM space, no process can access another process’ code or data. If the process crashes, it dies alone without taking the entire OS or a bunch of other applications down. A process is an instance of a running application. An application is an executable on the hard drive or network. There can be numerous processes launched of the same application (5 copies of Word running), but 1 process can run just 1 application.

##### **What are possible implementations of distributed applications in .NET?**

-->  
.NET Remoting and ASP.NET Web Services. If we talk about the Framework Class Library, noteworthy classes are in System.Runtime.Remoting and System.Web.Services.

##### **What are the consideration in deciding to use .NET Remoting or ASP.NET Web Services?**

-->  
Remoting is a more efficient communication exchange when you can control both ends of the application involved in the communication process. Web Services provide an open-protocol-based exchange of information. Web Services are best when you need to communicate with an external organization or another (non-.NET) technology.

##### **Choosing between HTTP and TCP for protocols and Binary and SOAP for formatters, what are the trade-offs?**

-->  
Binary over TCP is the most effiecient, SOAP over HTTP is the most interoperable.

##### **What’s SingleCall activation mode used for?**

-->  
If the server object is instantiated for responding to just one single request, the request should be made in SingleCall mode.

##### **What’s Singleton activation mode?**

-->  
A single object is instantiated regardless of the number of clients accessing it. Lifetime of this object is determined by lifetime lease.

##### **How do you trigger the Paint event in System.Drawing?**

-->  
Invalidate the current form, the OS will take care of repainting. The Update method forces the repaint.

##### **With these events, why wouldn’t Microsoft combine Invalidate and Paint, so that you wouldn’t have to tell it to repaint, and then to force it to repaint?**

-->  
Painting is the slowest thing the OS does, so usually telling it to repaint, but not forcing it allows for the process to take place in the background.

##### **How can you assign an RGB color to a System.Drawing.Color object?**

-->  
Call the static method FromArgb of this class and pass it the RGB values.

##### **What class does Icon derive from? Isn’t it just a Bitmap with a wrapper name around it?**

-->  
No, Icon lives in System.Drawing namespace. It’s not a Bitmap by default, and is treated separately by .NET. However, you can use ToBitmap method to get a valid Bitmap object from a valid Icon object. file  
By Jilani Shaikh

<https://www.blendinfotech.com/dotnet-framework-interview-questions-and-answers>

**What is Managed Code?**

The code that can be compiled into an intermediate language so we cannot run it directly on the computer as of native code.  Managed code runs in the Common Language Runtime.

Key features of managed code are:

* Managed code increases the performance gain of the application as well as it doesn`t require extra security feature while using unmanaged code we require extra security to increase the performance of the application
* It includes the garbage collection and scalability feature which is the lifetime control of object
* It makes the deployment easy and improves the version facilities at the end of dll.
* It provides Built-in security by using code access security and avoiding buffer overruns.

Managed code runs entirely "inside the sandbox," meaning that it cannot make calls outside of the .NET Framework.  To perform more safely and efficiently we use managed code to get the maximum benefit from the features of the .NET Framework.

Code in VB.NET and Cis the managed code. If you're working with those applications, you are making managed code. We can use managed code using c++ according to the choice of the user.

**What is MSIL?**

A .NET programming language like C#, VB.NET etc., not directly compiled into executable code; instead, they compile into an intermediate code called Microsoft Intermediate Language (MSIL). Since it automatically converts to MSI, so the programmer doesn’t need to be worried about that.

MSIL is similar to Java Byte code. A Java program is compiled into Java Byte code (the .class file) by a Java compiler, the class file is then sent to JVM which converts it into the host machine language.

Along with the Microsoft Intermediate Language MSIL the compiler also produces the MetaData which are contained in the portable executable PE file when the compiler produces MSIL it provides some instruction like  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

**Advantages -**

* MSIL provides language interoperability as code is written in any .net language is compiled into MSIL.
* Same performance for all .net languages
* It provides support to different runtime environments
* JIT compiler in CLR converts MSIL code into native machine code which is executed by OS

**What is JIT?**

JIT (**JUST-IN-TIME**) is a Compiler before MSIL be executed, it first converted to native code by .NET framework Just In Time(JIT) compiler, which runs on the same System on which the JIT Compiler is executing.

Just-in-time (JIT) is a term that is used to describe any action like compilation or object activation that only happens when it is necessary. This term is associated mostly with software compilation. It is designed because it has features to be accessed on multiple platforms and it also has high speed.

In Microsoft .NET there are three types of JIT (Just-In-Time) compilers :

* **Pre-JIT Compiler:** It compiles complete source code into native code in a single compilation cycle. It converts the .NET code from platform independent state to platform-specific stage.
* **Econo JIT Compiler:** It compiles only those methods that call at runtime and once these compiles methods are not in use are removed. Econo JIT spends less time compiling so startup latency is lower for interactive applications.
* **Normal JIT Compiler:** It compiles only those methods that are called at runtime, and after the execution of these methods is stored in memory, it is commonly referred to as “**Jitted**”. Methods that are compiled once are not compiled again.

**What is Portable Executable?**

All windows executable files and assemblies follow the Portable executable file formats.

.NET PE file format consists of four parts:

1. **PE/COFF Headers:** it contains information regarding exe or dll files.
2. **CLR Header:** contains information about CLR, memory management and CLR other related stuff.
3. **CLR Data:** Contains metadata of ddls and MSIL code generated by compilers.
4. **Native image section:** contains native image section like .data, .rdata, .rsrc, .text etc

All .NET assemblies are built over the PE file formats, that is used for all exes and dlls, which itself are built over the MS-DOS executable file formats.

Most of PE files are split up into separate sections, where each section stores different types of data. Like: ***.***text stores all executable code, .rsrc store unmanaged resources, .debug stores debugging information, .data section stores the global variables..etc.,.

A PE file has headers and sections that tell the dynamic linker about how to map file into memory. The task of the dynamic linker is to map each section of the memory and assign the correct permissions to the resulting regions, according to the instructions found in the headers

In a .NET framework executable, the PE code section is having a stub that invokes the CLR virtual machine startup entry, \_CorExeMain or \_CorDllMain in mscoree.dll, which is much the same as in Visual Basic executables.

The development on this platform intends to be binary compatible with Microsoft .NET, it uses the same PE format as the Microsoft implementation.

**What is Assembly?**

Microsoft .NET Framework application is a primary unit for deployment in Asp.Net.It provides all required execution information to common language runtime so It is called as the building block of an application.

In .NET, there are two kinds of assemblies,

1. Single file
2. Multi file.

An assembly that contains all information like IL, Metadata, and Manifest  in a single package is called as single file assembly. Most of the assemblies in .NET are build up as a single file assemblies.

Multiple file assemblies contain the multiple .NET binaries which are generated for bigger applications. In this, there will always be one assembly which will contain a manifest and while others will have IL and Metadata instructions.

Assembly has the code that the common language runtime(CLR) executes. Microsoft intermediate language (MSIL) code will not be executed the portable executable (PE) file if it does not have an associated assembly manifest.

It resides in each and every type’s identity which includes the name of the assembly. A type which is called MyType is loaded in the scope of one assembly and it is not the same as the type which is called MyType which is loaded in the scope of another assembly.

Assemblies can be of static or dynamic type. Static assemblies are those which include .NET Framework types (interfaces and classes), also if it is having the resources for the assembly (bitmaps, JPEG files, resource files, and so on). it is stored on disk in portable executable (PE) files. We can also create dynamic assemblies using a .NET framework, which can run directly from memory and will not save in the disk before execution. You can save dynamic assemblies to disk after their execution.

There are two kinds of assemblies in .NET;

1. Private
2. Shared

**Private assemblies** are very simple as they can be easily called upon each time from the assembly folder.

**Shared assemblies** known as a strongly named assembly are copied at a single location (usually GAC). For applications that are using the same assemblies, the same copy of shared assemblies are called from its original location, and hence they are not copied to each applications private folder. Each shared assembly has the four part name which includes its face name, version, public key token and culture information.

**What is GAC?**

GAC stands for Global Assembly Cache.

* It reserves an area for .NET Application to store the assemblies in the memory running on a certain machine.
* It helps in sharing the assemblies to all .NET application.
* To install in GAC, assemblies must have a strong name and must be publicly shared.
* by using Graphical User interface and the command line tool we can work with a cache by installing the CLR which has the global assembly cache and the one with windows shell extensions this is called as Global Assembly Cache tool (Gacutil.exe).
* The .NET Framework provides two tools for working with cache

When we want to share the assembly to the whole application then we use GAC which is stored in the folder of windows directory The concept of GAC is the result of the .NET architecture whose design addresses the issue of "DLL hell" that existed in COM (Component Object Model). In .Net there is no need to register the assembly before taking it in use. We can easily access it globally since it will not conflict by identifying its name, version, architecture, culture and public key

It has a certain feature of a shared library where different users can use the file which is stored in a common folder. In .NET 4.0, its default location is: %windir%\Microsoft.NET\assembly

There are two tools related to GAC are:

1. **GAC Tool (gacutil.exe):** It is used to check for assembly existence, to register the shared assembly and to view and manipulate GAC contents.
2. **Assembly Cache Viewer (shfusion.dll):** It helps in displaying the details like version, culture etc. with the assemblies which are associated with it in the cache.

**How it possible .net support multiple languages?**

The .NET application is said to be multilingual because we can deploy it in different language, the language it supports are visual basics, .NET, C#, J# which compiled to a command Intermediate language the Java byte code which is developed by Microsoft is a low-level language with simple syntax which is easily be translated into native machine code

* **CLR:**

CLR helps .NET framework to be multilingual, the code running under CLR is termed as managed code, the main task of CLR is to convert compiled code into native code, it has different compilers some of them are

VB .NET, C#, C++, JScript or any third party compiler such as COBOL.

It simply means that you can create pages in different languages (like C#, VB .NET, J# etc.) and once all of these pages are compiled they all can be used in a single application

* **CTS:**

CTS is Common Type System it is used when we are using the managing types, it is basically used to handle the data types. .NET have many languages and each language is having many other data types one language data types cannot be used in other languages, so in that case, we use the CTS

**Example:** we have the application in C# which is having the data type of int and there is another data type of integer in VB.NET, them CTS helps in converting the integer into the int32 structure.

* **CLS:**

It is Common Language Specification, it has developed certain rules which everyone used to follow like if we have created the code for the multiple inheritances in C++ and if use that code in c# and it will not compile since multiple inheritance is not allowed in C# also it makes C# a case sensitive members of the same class cannot have the same name like Add() and add() they consider as different function.

**What is .net tracing?**

When we want to record the unusual activity during runtime, .net takes the help of the trace by following the page execution path which displays the information at run time

There are 2 levels of tracing used in .NET

1. Page Level Tracing
2. Application level Tracing

**Page Level Tracing:** it is for managing the trace of the page whether it is enabled or disabled, if it is enabled then a page which is requested, it appends to the page a series of tables containing the details of page request during execution

In this there are 2 Trace Modes:

* **SortByCategory:** In this, it sort trace message according to the categories
* **SortByTime:** In this, it sorts the message according to the order in which they are processed.

**Application Level tracing:** In this, we can set the trace for the whole application at once instead of setting for individual page, it is useful when we want to trace for the whole application instead of enabling and disabling for an individual page

We can trace the information according to the request limit we have to specify the default number is 10, as soon as it reaches the request limit it stops tracing.

**What is Garbage collector?**

When an object is created in C#, CLR (normal language runtime) assigns memory for that object from the heap. This procedure is repeated for every newly created variable/object, however, there is a drawback to everything, Memory isn't unrestricted and we have to clean some utilized space so as to prepare for new items. Here, comes the garbage collection is a picture. Garbage Collector manages distribution & recovering of memory. GC makes an outing to the heap and gathers all objects that are no longer utilized by the application and free up the memory.

The Garbage Collection is a significant technique in the .Net framework to free the unused managed code data in the memory. In the Common Language Runtime (CLR), the Garbage Collector acts as an automatic memory manager. It gives the following advantages:

* Empowers you to build up your application without liberating memory.
* Assigns objects to the managed heap efficiently.
* Recovers objects that are never again being utilized, clears their memory and keeps the memory accessible for future allotments.
* Gives memory security by ensuring that an object can't utilize the content of another object.

**How does garbage collection work?**

Certain Garbage Collection ought to be dealt with by the .Net structure. At the point when the object is created then it will be put in the Generation 0. The Garbage Collection utilizes an algorithm which checks the object’s in the age by identifying its current generation number. Once the object’s lifetime is over then it will be expelled from the memory. The two sorts of items. One is Live Objects and Dead Objects. The Garbage Collection algorithm gathers every single unused object that are dead objects in the age. In the event that the live objects running for long time, at that point dependent on that lifetime it will be moved to the next generation.

The object cleaning in the age won't happen precisely after the existence time over of the specific object. At some particular time Garbage Collection starts acting on it & implement its sweeping algorithm to free up the spaces.

**When GC got triggered?**

There are no particular timings for Garbage Collector to get activated. Garbage Collection happens when one of the accompanying conditions is valid:

* The system has low physical memory. This is recognized by either the low memory warning from the operating system or low memory demonstrated by the host. At the point when an allocation of virtual memory is requested, the virtual memory manager needs to locate a single free block that is enormous enough to fulfill that portion demand. Regardless of whether you have 2 GB of free space, the portion that requires 2 GB will be ineffective except if the majority of that free space is in a single address block.
* The memory that is utilized by allocated objects on the managed heap exceeds an acceptable limit. This limit is continuously balanced as the process runs.
* The GC.Collect method is called. In practically all cases, you don't need to call this technique, in light of the fact that the Garbage Collector runs continuously. This technique is principally utilized for extraordinary circumstances and testing.

**Explain ASP .Net application page life cycle?**

a) User sends a solicitation to the IIS

* IIS first checks which ISAPI expansion can serve this solicitation. Contingent upon document augmentation the solicitation is handled.

b) An occasion of the ApplicationManager class is made, which is the Application Domain that the solicitation is handled in.

* As we as a whole know, the application space makes separation between two web applications facilitated on similar IIS. So on the off chance that there is an issue in one application space, it doesn't influence the other application area.

c) In the application space, an occasion of the HostingEnvironment class for example the 'HttpRuntime' object is made, which gives access to data about the application, for example, the name of the organizer where the application is put away.

d) Once the facilitating condition is made, the essential center ASP.NET items like 'HttpContext', 'HttpRequest' and 'HttpResponse' objects are made.

e) The application is begun by making a case of the HttpApplication class. On the off chance that the application has a Global.asax document, ASP.NET rather makes an occasion of the Global.asax class that is gotten from the HttpApplication class.

* HttpApplication cases may be reused for numerous solicitations.

f) Once 'HttpApplication' is made, it begins preparing demands. It experiences following 4 stages:

* **(M: HttpModule):** Client solicitation handling begins. Before the ASP.NET motor proceeds to make the ASP.NET HttpModule radiate occasions which can be utilized to infuse modified rationale. There are 6 significant occasions that you can use before your page item is made:
  + BeginRequest
  + AuthenticateRequest
  + AuthorizeRequest
  + ResolveRequestCache
  + AcquireRequestState
  + PreRequestHandlerExecute.
* **(H: 'HttpHandler'):** Once the over 6 occasions are terminated, ASP.NET motor will summon ProcessRequest occasion in the event that you have executed HttpHandler in your task.
* **(P: ASP.NET page):** Once the HttpHandler rationale executes, the ASP.NET page item is made. While the ASP.NET page item is made, numerous occasions are terminated which can assist us with writing our custom rationale inside those page occasions.
* **(M: HttpModule):** Once the page article is executed and emptied from memory, HttpModule gives post page execution occasions which can be utilized to infuse custom post-handling rationale. There are 4 significant post-handling occasions PostRequestHandlerExecute, ReleaserequestState, UpdateRequestCache and EndRequest.

**Explain asp .net page life cycle events?**

* **PreInit:**This event is raised after the start stage is complete and before the initialization stage.
* **Init:** This event occurs after all controls have been initialized. We can use this event to read or initialize control properties.
* **InitComplete:** This event occurs at the end of the page's initialization stage. We can use this event to make changes to view state that we want to make sure are persisted after the next postback.
* **PreLoad:** This event occurs before the postback data is loaded in the controls.
* **Load:** This event is raised for the page first time and then recursively for all child controls.
* **Control events:** This event is used to handle specific control events such as Button control' Click event.
* **LoadComplete:** This event occurs at the end of the event-handling stage. We can use this event for tasks that require all other controls on the page be loaded.
* **PreRender:** This event occurs after the page object has created all controls that are required in order to render the page.
* **PreRenderComplete**: This event occurs after each data bound control whose DataSourceID property is set calls its DataBind method.
* **SaveStateComplete**: It is raised after view state and control state have been saved for the page and for all controls.
* **Render:** This is not an event; instead, at this stage of processing, the Page object calls this method on each control.
* **Unload:**This event raised for each control and then for the page.

**What are the client side and server side state management technique supported in .net?**

State Management is to protect state control in an application in light of the fact that ASP.NET web applications are stateless. Another instance of the Web page class is made each time the page is posted to the server. When a client enters data into a web application, that data would be lost in the round excursion from the program (MSDN).

In a solitary line, State Management keeps up and stores the data of any client till the finish of the client session. There are two types of state management techniques available:

* Client Side
  + Storing data on the client in various ways.
  + Client-Side state management techniques are as follows:
  + View state (used by default by ASP .NET)
  + Control state
  + Hidden fields
  + Cookies
  + Query string
* Server Side
  + Storing data in memory on the server
  + Server-Side state management techniques are as follows:
    - Application state
    - Session state

**When to choose client side and server side state management?**

**Client-Side:**

* For better scalability by saving user’s data at client-side(browser)
* To support multiple servers, so, when a request is diverted to another server no need to worry about the state.
* To empower the server-side states to be capable to help numerous servers you'll need to utilize techniques like a sticky connection for load balancing.

**Server Side**

* Pick server-side for better security and to decrease data transfer capacity and website page's size.
* Server side state management is more secure. The state is saved on the server also, consequently isn't conveyed to the client.
* Secret state information shouldn't be utilized with client-side state management
* Server-side diminishes the traffic to and from the user since information isn't sent to the browser and, it's saved on the server.
* Client-side state management technique increases bandwidth usage & therefore application becomes less responsive and you'll experience performance issues.

**What is control state?**

ASP.NET page has by default EnableViewState property true. This makes ASP.NET consequently monitor the condition of the considerable number of controls present on the page. Presently on the off chance that I am dealing with a web page where I realize every one of the controls will be dynamically bound to certain information on each postback, I don't need to monitor the old data for the controls. So I may choose to turn it off. Presently this is a decent decision however things could get chaotic if a similar page contains a custom control and that custom control is utilizing ViewState to deal with the information inside the control.

So, ASP.NET already provides a technique to handle such scenarios and it's called ControlState.

At whatever point we build up a custom control and need to safeguard some data, we can utilize view state yet assume view state is disabled explicitly by the client, the control won't work as expected. For expected outcomes for the control we need to utilize Control State property. The Control state is isolated from the View state.

The most effective method to utilize control state property, override the OnInit() function for the control and include a call for the Page.RegisterRequiresControlState() function with the instance of the control to register. At that point override LoadControlState and SaveControlState so as to save the required state data.

**What is hidden field and how to implement?**

Hidden field is a control given by ASP.NET which is utilized to store limited quantities of information on the client. It stores one value for a variable and it is an ideal way when the value of that variable is changing regularly. Hidden field control isn't rendered to the browser and is invisible too. A hidden field goes with each request like a standard control's worth.

Let’s see an example of how to use this hidden field. This would increase the value by 1 on every button click.

**The source of the hidden field control is:**

<asp:HiddenField ID="HiddenField1" runat="server"  />

**In the code-behind page:**

Protected void Page\_Load(object sender, EventArgs e)

{

   if (HiddenField1.Value != NULL)

   {

int val = Convert.ToInt32(HiddenField1.Value) + 1;

    HiddenField1.Value = val.ToString();

Label.Text = val.ToString();

   }

}

protected void Button1\_Click(object sender, EventArgs e)

{

  //No Action Button Click

}

**What is view state and how to implement?**

View state is another client side state management technique in ASP.NET to store client's information, i.e., in some cases the client needs to protect information briefly after a postback, at that point the view state is the favored route for doing it. It stores information in the produced HTML utilizing the hidden field, not on the server.

View State gives page level state management i.e., as long as the client is on the present page, the state is accessible and the client sidetracks to the following page and the present page state is lost. View State can store any sort of information since it is item type yet it is best not to store complex information because of the requirement for serialization and deserilization on each postback. View state is on by default for all server-side controls of ASP.NET with a property EnableviewState set to true.

Let us see with an example how ViewState is utilized & implemented. In the below example we try to save the count of postbacks on each button click.

**What is cookies?**

It’s a tiny file that is being created by the client's browser and also get stored on the hard disk of that client. It doesn't utilize server memory. For the most part, a cookie is utilized to identify users.

A cookie stores client data. At whatever point a client makes a request for a page the first run through, the server creates a cookie and sends it to the client alongside the mentioned page and the client browser gets that cookie and stores it on the client’s either permanently or temporarily. Whenever the client makes a request for a similar site, either the equivalent or another page, the browser checks the presence of the cookie for that site in the folder. In the event that the cookie exists, it sends that request with the same cookie, else that request is treated as a new request. Let see a sample code to understand how it works:

int postbacks = 0;

if (Request.Cookies["number"] != null)

{

postbacks = Convert.ToInt32(Request.Cookies["number"].Value) + 1;

}

// Generating Response

else

{

postbacks = 1;

}

Response.Cookies["number"].Value = postbacks.ToString();

Result.Text = Response.Cookies["number"].Value;

**What are the different type of cookies?**

There are two kinds of cookies created by the client’s browser, which are as follows:

* **Persistence Cookie:** Cookies which has an expiry date time is known as persistence cookies. Persistence cookies are permanently stored until the expiry date time which was set. Let us see how to create persistence cookies. There are two ways to create them:

**Link:**<https://www.codeproject.com/Articles/492397/State-Management-in-ASP-NET-Introduction>

* **Non-Persistence Cookie:** Non-Persistence cookies are not for all time put away on the client customer. It keeps up client data as long as the user accesses the same browser. At the point when the client shuts the browser, the cookie will be disposed of. Non Persistence cookies are helpful for open/public PCs. Let us see how to create non-persistence cookies. There are two ways to create them:

**What is application state?**

Application state is a server-side state the board procedure. The data put away in the application state is basic for all clients of that specific ASP.NET application and can be accessed from anyplace in the application. It is likewise called the application-level state of the management. Information put away in the application ought to be of little size.

Let’s see how to set & get a value in the application state object:

**Following are the Application events on ASP.NET:**

There are three events in ASP.NET. The application occasion is written in Global.asax. This document isn't made, of course, it is made expressly by the engineer in the root directory. An application can make more than one Global.asax record yet just the root one is perused by ASP.NET.

* **Application\_start:** The Application\_Start event is raised when an application domain begins.
* **Application\_Error:** It is raised when an unhandled exception happens, and we can deal with the exception of this event.
* **Application\_End:** The Application\_End event is raised just before an application domain ends as a result of any reason, may IIS server restarting or rolling out certain improvements in an application cycle.

**What is session state?**

Session Management is a solid system to look after state. By and large session is utilized to store client's data as well as remarkably recognize a client (or state program or browser). The server keeps up the condition of client data by utilizing a session ID. At the point when clients make a request without a session ID, ASP.NET makes a session ID and sends it with each request and response to the same user/client.

**Following are the Session Events in ASP.NET:**

To deal with a session, ASP.NET gives two events: **session\_start** and **session\_end** that is written in Global.asax in the root directory of the project.

* **Session\_Start:** The Session\_start occasion is raised each time another client makes a request without a session ID, i.e., the new program gets to the application, at that point a session\_start occasion raised. We should see Global.asax document.
* **Session\_End:** The Session\_End event is raised when the session closes either in view of a time out or explicitly by utilizing Session.Abandon(). The Session\_End event is brought distinctly up for the situation of *In proc mode*, not in the *state server* and *SQL Server modes*.

**What is session state mode available in asp.net?**

There are four session storage mechanisms provided by ASP.NET:

* **In Proc mode**

In proc mode is the default mode given by ASP.NET. In this mode, session values are put away in the web server's memory (in IIS). On the off chance that there are more than one IIS servers, at that point session values are put away in every server independently on which request has been made. Since the session values are put away in the server, at whatever point server is restarted the session esteems will be lost.

* **State Server mode**

This mode could store session in the webserver however out of the application pool. Yet, as a rule if this mode is utilized there will be a different server for putting away sessions, i.e., state server. The advantage is that when IIS restarts the session is accessible. It stores sessions in a different Windows service. For State server session mode, we need to design it unequivocally in the web config record and begin the aspnet\_state service.

* **SQL Server mode**

Session is put away in a SQL Server database. This sort of session mode is likewise isolated from IIS, i.e., session is accessible even subsequent to restarting the IIS server. This mode is profoundly secure and dependable yet in addition has a burden that there is overhead from serialization and deserialization of session data. This mode ought to be utilized when reliability is of higher importance than performance of application.

* **Custom mode**

By and large we ought to lean toward in proc state server mode or SQL Server mode yet on the off chance that you have to store session information utilizing other than these procedures, at that point ASP.NET gives a custom session mode. Thusly we need to keep up everything customized even producing session ID, information store, and furthermore security.

**What is the difference between Session.Abondon() and Session.Clear()?**

* **Session.Clear():**
  + Clears all the session values but doesn't destroy the Session.
  + Removes only the values (content) from the Object.
  + The session with the same key is still alive.
  + Use Session.Clear(): if you don’t want to kill the session but just wants to clear the session-specific data and reinitialize the session.
  + It will not raise the Session\_OnEnd event.
* **Abondon():**
  + Destroys the whole session object & release all the resources
  + If you do not call the Abandon method explicitly, the server destroys the objects when the session times out.
  + It has no return value and no parameters
  + Abandon raises the Session\_OnEnd event request.
  + If a request is made that includes the session identifier for an expired or abandoned session, a new session is started using the same session identifier

**What is asp.net web parts?**

ASP.NET Web Parts is a set of controls for creating Websites that enable users to update the content, behavior and appearance of Web pages directly from a browser. The modifications can be applied to all or individual end-users. When users modify pages or controls, those settings can be saved to retain a user's personal setting preferences across future browser sessions. This feature is called personalization. These Web Parts capabilities mean that developers can authorize end-users to personalize any Web application dynamically, without intervention of any developer or administrator. Different Web Parts can share their data with each other.

**Web Parts Modes:**

* **Normal mode:** End users cannot edit & move sections of the page.
* **Edit Mode:** End users can edit Web Parts & custom properties on the page.
* **Design Mode:** End users can reorder the pages Web Parts in a WebPartZone.
* **Catalog Mode:** New Web Parts can be added by the end-user in any WebPartZone on the page.

**What is master page?**

Utilizing master pages, we can create the common UI components for all the site pages and make a consistent look and feel for the entire site.

* A Master page offers a format/template for at least one or more web forms.
* It characterizes placeholders for the content, which can be superseded by the content pages.
* You would then be able to make individual content pages that contain the content you need to show. At the point when clients demand the content pages, they converge with the master page to create yield that consolidates the format of the master page with the content from the content page
* A Master page has .master as a file extension
* Any Master page is distinguished by an exceptional @Master directive ***<%@Master Language="C#" %>***
* It can have from 1 to many ContentPlaceHolder i.e. can hold multiple content pages

**How Content page attached with master page?**

* **Page Level**: By using the page directive on each content page to bind it to a master page
* **Application Level**: By making a setting in the pages element of the application's Web.config file. All the specified ASP.NET pages (.aspx files) in the application will be automatically bind to a master page.
* **Folder Level:** This strategy is used to bind at the application level, except that you make the setting in a Web.config file in one folder only. Here, master-page bindings are applied to all the ASP.NET pages in that folder.

**Difference between machine.config, web.config and global.asax?**

* **config:**
  + This is automatically installed when you install Visual Studio. Net.
  + This is also called a machine level configuration file.
  + Only one Machine.config file exists on a server.
  + This file is at the highest level in the configuration hierarchy.The settings of Machine.config file are applied to the whole asp.net application.
* **config:**
  + This is automatically created when you create an ASP.Net web application project.
  + This is also called the application-level configuration file.
  + This file inherits the setting from the Machine.config.
  + Web. Config is used for asp.net web projects/web services.
  + The settings made in the Web.config file are applied to that particular web application only.
* **asax:**
  + Used to handle application and session-level events and objects for an ASP.NET web site running on an IIS Web Server.
  + Code to handle application events (such as the start and end of an application) resides in Global.asax.
  + Such event code cannot reside in the ASP.NET page or web service code itself, since during the start or end of the application, its code has not yet been loaded (or unloaded).
  + It is also used to declare data that is available across different application requests or across different browser sessions. This process is known as application and session state management

**What is the difference between server.Transfer() and Response.Redirect()?**

| **Response.Redirect** | **Server.Transfer** |
| --- | --- |
| Browser URL changes | URL does not change |
| Redirection happens through browser | Redirection happens through a server |
| Http Request à Browser àWebserver àBrowser | Http Request à Webserver àBrowser |
| Round trip | No round trip |
| Cross-server redirection is possible only by using ‘Response.Redirect’ i.e. it allows redirection to internal as well as external websites and website pages. | From [www.google.com](http://www.google.com/) à [www.xyz.com](http://www.xyz.com/) not possible. This redirection not possible because both are on different servers. |
| For both .aspx & .html | Only for .aspx |
| It’s a client process | It’s a server process |

**Hyperlink vs link button in asp.net?**

* A Hyperlink just redirects to a given URL identified by "NavigateURL" property.
* Link Button does not have Navigate URL instead it has “Postback URL” property
* LinkButton causes a postback to the same page but it doesn't redirect to a given URL
* HyperLink doesn't have the OnClick event. Link Button has Onclick event
* if you need to do any operation with data that's on the page you will have to use a LinkButton (or a Button), but if you just need to redirect the user to somewhere else go for an HyperLink

**User control vs Custom control?**

| **Custom Controls** | **User Controls** |
| --- | --- |
| Compiled into their own assembly(.dll) | Compiled into the web application assembly |
| Therefore, these can be added to the toolbox and can be used like any other existing controls. | Therefore, these cannot be added to the toolbox |
| Drag n drop from toolbox | Drag n drop from solution explorer. .ascx file is created for each project in the solution |
| Complex to create. | Easier to create just like a web form. |
| They don’t have a designer. So, everything from a declaration of controls to rendering them has to be written through code. | They have designer associated with them, which makes it easier to create them. |
| Multiple copies of custom controls are not required in the projects. | Every web app project requires its own copy of user control |

**From the content page, how can you reference a control on the master page?**

**1. Using the FindControl() method**

* using System;
* using System.Collections.Generic;
* using System.Linq;
* using System.Web;
* using System.Web.UI;
* using System.Web.UI.WebControls;
* public partial class FindControl : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

     //Define Label Control and Fetch the control of MASTER PAGE

     Label lbl = (Label)Page.Master.FindControl("lblFirstName");

     //Set the value to CONTENT PAGE label control.

     lblFindControlUserName.Text = "Value Received in Content Page : "+lbl.Text;

}

}

**2. Without using the FindControl() method**

Yes, by casting the Master to your MasterPage as shown in the below code sample:

protected void Page\_Load(object sender, EventArgs e)

{

    MyMasterPage MP = this.Master;

    MP.MyTextBox.Text = "Text Box Found";

}

**What is .net framework?**

It is a platform which is developed by Microsoft for the software development. In the current scenario, the version used by the programmer is 4.7.1.

It is very effective when we want to create - Form-based and Web-based applications. Web services using the development tool, .Net framework.

It is multilingual in terms of a programming language. It uses C# or Visual Basic to develop the application, in this user can make its own choice by choosing its own language to develop the application

**.Net Framework Architecture**:

**1. Language** :

* **WinForms** – This is used for developing Windows Forms applications, which runs on an end user machine. Example: Notepad.
* **ASP.Net**– This is used for developing web applications which runs on browsers such as Internet Explorer, Chrome or Firefox
* **ADO.Net** – This is used to develop applications that interact with Databases such as Oracle or Microsoft SQL Server

**2. Class Library:**.Net Framework included a number of class libraries which contain the method and function which helps in handling the file level operation. For example, a class library that has methods to handle all file-level operations like a method which can be used to read the text from a file

**3. Common Language Runtime:**Common Language Infrastructure has the following key features:

* **Exception Handling:**Exceptions are errors which occur when the application is executed.like if you are opening the file from the local which is not present at the local then it will give an exception
* **Garbage Collection:**when we wanted to remove the unwanted resources from the code which is no longer in use can be done by the garbage collector.

Like as the database connection in the application which is no longer in use when compilation stops

**What is CLR?**

**1. Common Language Runtime**. It works as an interface between an Operating Systems and the applications which are developed using the .NET framework. The main objective of CLR is to execute the program by converting the managed code to native code. CLR 's Just In Time (JIT) compilation converts Intermediate Language (MSIL) to native code at application run time.

When .Net application is executed, then the control goes to Operating System, which creates a process to load CLR

**2. CLR services**

* Assembly Resolver
* Assembly Loader
* Type Checker
* COM marshaller
* Debug Manager
* Thread Support
* IL to Native compiler
* Exception Manager
* Garbage Collector

**3. Assembly Resolve**

It will send the request to the assembly loader by identifying assembly whether it is private or shared assembly.

**4. Assembly Loader**

According to the assembly resolver instruction, the assembly loader loads the instruction into the application.

**5. Type Checker**

To provide the safety checker helps in verifying the types which are used in the application with CTS or CLS

**6. COM marshaller**

It helps in getting communicating with the COM components which supports the COM interoperability.

**7. Debug Manager**

Using this we can check the code by line by line according to that we can make the changes in the code without terminating the application execution.

**8. Thread Support**

It manages more than one execution path in the application process, this provides multithreading support.

**9. IL to Native compiler**

Just In Time (JIT) compiler is used to convert the IL code to native code.

**10. Exception Manager**

It will handle exceptions thrown by application by executing catch block provided by exception, if there is no catch block, it will terminate the application.

**11. Garbage Collector**

when we wanted to remove the unwanted resources from the code which is no longer in use can be done by the garbage collector.

Like as the database connection in the application which is no longer in use when compilation stops.

**What is CLS?**

CLS stands for**Common Language Specification** as the name suggests it set of certain feature which is very helpful for library and compiler writers if any other language that supports CLS can be used fully in each other's language, thus we can say that CLS is a subset of the common type system.

It is actually a set of restrictions on the CTS. It not only defines the types allowed in external calls, but also the rules for using them, depending on the goal of the user.

CLS is basically a subset of the entire set of features supported by CLR. With CLS, we can call virtual methods or can overload methods and not include things such as unsigned types.

It defines a common level of language functionality. CLR is the set of rules that a language compiler must follow while creating a .NET application at run in CLR. Anyone who wants to write a .NET·-compliant compiler needs simply to adhere to these rules and that's it.CLS is a set of rules through which we can exchange information over a single platform. The beauty of this is that the restriction to use features only applies to public and protected members of classes public classes. Within the private methods of your classes, any non-CLS code can be written, because the code in other assemblies cannot access this part of your code

**What is CTS?**

The **Common Type System** defines how types are declared, used, and managed in the common language runtime, and is also an important part of the runtime's support for cross-language integration.

Here we have several languages and each and every language has its own data type and 1 language data type cannot be understandable by other languages.

CTS is a specification created by Microsoft and included in the European Computer Manufacturer‘s Association standard. It has certain standard features for implementing the .NET framework.

There are different types which CTS Support:

1. **Value Types**: It directly contains the data and instances are either allocated on the stack or allocated inline in a structure. Value types can be built-in, user-defined or enumerations types.
2. **Reference Types**: Value’s memory address reference are stored and are allocated on the heap. This can be any of the pointer types, interface types or self-describing types.

Operations on variables of a value type do not affect any other variable, whereas, operations on variables of a reference type can affect the same object referred to by another variable.

**What is asp.net?**

Started in the 2000s, ASP.net is a widely used web application framework which runs on Windows. It allows the development of applications, dynamic website, and web services. The biggest benefit of designing websites using ASP.net is the low cost and high speed. It also Another great benefit is its vast language support. ASP.net doesn’t need to be installed or configured separately unlike other platforms since it is built into the Windows environment.

Developers are widely using the framework since it allows building websites that are much faster and dynamic. The reason is the codes are compiled rather than interpreted. This means that the code is converted into object code which can be executed repeatedly by the .Net platform. The compilation process takes little time and happens only once unlike interpretation in which the code is read and interpreted every time it is being executed.

ASP.net is written in OOP languages, i.e., Object Oriented Programming languages, such as VB.net and C#. This is the reason it is fast, easy to use and totally reliable.

**What is .net?**

.Net is a framework that offers the most unique programming guidelines to develop not only web solutions but also a range of applications. It is compatible with many programming languages such as F#, C++, VB.net, and C#.  It has a very easy to use Integrated Development Environment (IDE) where you can write your code. It also supports editing of the code, designing interface, performance analysis, debugging and server management. Apps created using .Net can be run on Windows, LINUX, and Mac OS X.

It has a huge collection of prewritten codes which are known as predefined class libraries. It’s the work of other developers that can easily be used by adding those code into your programs. .Net libraries also have pre-written codes for database connectivity, database access, encryption and security.

.Net framework is widely used by big MNCs and IT companies for its varied advantages. Some loyalists are HCL, TCS, Dell, Epic Systems, Accenture, Quicken Loans, etc.

**What is the difference between c# and VB .net?**

Both VB.net and C# use Object Oriented Programming Language and run on the .Net framework. These High-Level Languages are also referred to as Common Language Infrastructure (CLI) languages which means their code need not be translated while getting executed on a different platform.

Although they have many things in common, their main differences are as follows.

#### **C#**

* C# comes with all the features of Python, Java, C++, and other languages since it is evolved from C.
* To declare variables, the users must define their own variables or use the built-in types.
* By using keywords in C#, unmanaged resources can be released.
* Optional parameter is not supported in C#.
* C# allows new implementation without overriding the base class member. The base class member can be used in a derived class just with the keyword ‘new.

#### **VB.net**

* VB.net is user-friendly since it uses mnemonics and is much similar to the English language. It can be easily learned.
* To declare variables in VB.net, classes, Private, Public, Protected, Friend, and Static must be used.
* The feature of releasing unmanaged resources is not available in VB.net.
* An optional parameter is supported in VB.net.
* In VB.net, a new implementation of base class without overriding the base class member is allowed. The base class member can be used in a derived class just with the keyword ‘Shadows’.

**What is the difference between unmanaged and managed code?**

**Managed Code:** Managed Code is the code which isn’t run by the operating system directly. It is implemented in the run time environment. There are many benefits of Managed runtime such as type checking, garbage collection, bound checking, exception handling, etc. without the need for interference from the programmer.

The compiler compiles the code in the .Net framework and translates into an intermediate language, MSIL. It is then translated to executable code.



**Unmanaged Code:** Unmanaged code is not translated to executable code. It directly runs on the operating system. On every platform an unmanaged code runs, it is compiled separately. The reason is that it depends on the architecture of the system. Since it is compiled into a to native code, it can lead to many problems such as pointer override, buffer overflow, memory leak, etc. Unmanaged code provides direct access to the hardware and is not 100% secure to create applications.



**What is caching in asp.net?**

aching is a technique which is used to store data that is frequently used in memory. The runtime of ASP.net has a key-value map which is called the cache. It reduces the execution time when the next time the same information is requested. The code will not need to be generated since the pages can directly be retrieved from the memory. Caching is very important to transactions related to data because of its quick response time.

The data will be unavailable in certain situations:

* When the process doesn’t take place for any reason
* The memory is released from the application
* The lifetime of the data is expired

The items inside the cache can be accessed using an indexer. It can also be used to control and link the objects to each other.

* **Output Caching:** When the page is requested again, a cached copy is sent.
* **Data Caching:** Refills the cache when it is expired.
* **Object Caching:** Caches the objects.

**What is .net framework?**

.Net framework basically is a platform for building multiple applications on windows. It has various inbuilt functionalities available in various forms like class, library, and APIs which are used in building, deploying and running the web services and N no of other applications. It even supports various languages such as C#, COBOL, Perl, VB.Net, etc.

This framework actually supports a programming model that is object-oriented.

It is also an open-source, server-side, web application framework designed for web development resulting in web pages which are dynamics by nature.

This framework came into existence through Microsoft to allow programmers to build web sites, web applications and web services that are dynamic by nature.

In nutshell .Net framework is meant for creating applications, which would run on the Windows as well as web Platform. The .Net framework’s 1st version was released in the year 2002.

**What are the important component in .net?**

4 main components of .Net Framework are mentioned below:

* CLR(Common Language Runtime)
* CTS(Common type system)
* BCL(Base Class Library)
* CLS(Common Language Specification)

**1. CTS stands for Common Type System**

It has a lot of guidelines which state how an information type ought to be proclaimed, characterized and utilized in the program. It portrays the information types that are to be utilized in the application.

We can plan our own classes and qualities by following the principles that are available in the CTS. The principles are made with the goal that the information type pronounced utilizing a programming language is callable by an application that is created utilizing an alternate language.

**2. CLR stands for Common Language Runtime.**

It is one of the most significant parts of the .Net structure. It gives building squares to numerous applications.

An application assembled utilizing C# gets incorporated by its own compiler and is changed over into an Intermediate language. This is then focused on CLR. CLR does different tasks like memory the executives, Security checks, congregations to be stacked and string the board. It gives a protected execution condition to applications.

**3. CLS stands for Common Language Specification**

With the rules mentioned under CLS, the developers are made to use the components that are inter-language compatible. They are reusable across all the .Net compliant languages.

**4. BCL stands for Base Class Library**

BCL otherwise called the Class library (CL). BCL is a subset of Framework class library (FCL). Class library is the accumulation of reusable sorts that are firmly incorporated with CLR. Base Class library gives classes and types that are useful in performing an everyday task, for example, managing string and crude sorts, database association, IO activities.

**What is JIT?**

JIT is abbreviated as Just In Time. It acts as a compiler that helps in converting the Intermediate Language to a Native code. The code is generally available in Native language during the execution step. Native code is actually nothing special but the hardware specifications that can be easily understood by the CPU.

The native code even is stored so that it is accessible for subsequent calls by the end-users.

In short, it converts the MSIL code within an assembly to native code that can be understood by the CPU architecture of the target machine to run a .NET application.. It also cross-checks the values that are passed to parameters of any method.

It helps in managing the execution of .NET programs regardless of any type of .NET programming language. The first step is a language-specific compiler converting the source code to the intermediate language. Then the intermediate language is transformed into the machine code by the JIT compiler.

**What do you mean by managed and unmanaged code?**

The code being managed by the CLR is called Managed code. This code basically runs within the CLR. Thus, it is important to get the .Net framework installed in order to execute the managed code. CLR manages its memory through garbage collection and even uses the other features of .Net like CAS and CTS for proper and perfect management of the code.

**Following steps are followed while executing a Managed code:**

* **Step I:** It involves choosing a language compiler which entirely depends on the language in which the code is written.
* **Step II:** Converting the above-achieved code into an Intermediate Language with the help of its own compiler.
* **Step III:** The Intermediate Language is then targeted to CLR followed by conversion into native code with the help of JIT compiler.
* **Step IV:** The final step involves the execution of the Native code.

On the other hand, Unmanaged code is any code that does not at all depend on CLR for execution which means it is developed by any other language other than .Net framework.

It even uses its own runtime environment for compiling and execution of the data.

Though it is not being run inside the CLR, but the unmanaged code works properly if all the other parameters are correctly mentioned here.

**What is assemble and what are the type of assemble?**

An assembly is a combination of logical units. The Logical units here mean the types and resources which are needed to build and deploy any application using the .Net framework. It is the standard parameter developed with the Microsoft.NET.

They might be executable (.exe) file or available in dynamic link library (DLL) file.

An assembly consists of either single file or multiple files for use. In conditions wherein we have multi-file, there is only one master module containing the manifest where as other assemblies exist as non-manifest modules.

A module in .NET is defined as a subpart of a multi-file. To add on further assembly is actually one of the most interesting and extremely useful areas of .NET architecture along with the feature of reflections and attributes.

In .NET there are basically three types of assemblies:

* private
* shared
* satellite

**what are the different type of assemble?**

* **Private Assembly**

Private assembly functions by coping in application folders separately, where we require to use the assembly’s functionalities;

In this case, the features does not function without copying. In short, it means every time we have to exclusively copy into the BIN folder of each application folder for the functioning of the private assembly.

* **Public Assembly**

Public assembly is also known as Shared assembly does not require to copy separately into all application folders. Only one single copy is required at system level. Like the private assembly, there is no need to copy the assembly into the application folder.

It should be installed in GAC (Global assembly cache).

* **Satellite Assembly**

The language and culture-specific resources for an application are being deployed with the help of satellite assembly.

**What is exe and dll?**

The two Assembly executable modules are EXE and DLL.

Exe in simple words means an executable file that is generated when we build an application. The assemblies are directly loaded when we run an Exe. Meanwhile, an Exe cannot be shared with the other applications. It is a file extension for an executable file format. An executable file can be easily run by a program in Microsoft DOS or Windows through a command or a double click.

DLL stands for Dynamic Link Library. It is a library that has codes that are hidden. The codes are actually encapsulated inside the library. An Application can consist of n number of DLLs which can be shared with the other applications as well. Other applications that need to share this DLL need not worry about the code till the time they are able to call the function on this DLL.

DLL files can be used by more than one program at a time. In fact, they can even be used by multiple programs being executed within the same time. DLLs usually come with the Windows operating system while some of them are added when new programs are being installed.

**What is caching?**

A cache essentially stores the output produced by a page in the memory and this output in the cache will serve us (clients) later on.

Caching acts as a storage tank for the applications to access data temporarily rather than looking for its original location. In other words, it is a memory for applications to be executed which in turn increases the efficiency of the performance of the application along with its speed.

It is actually a good habit for the coder to use caching at the time of building the application. It is one of the most interesting concepts and operations in asp.net.

Depending upon the requirement we can run any application by applying the caching concept.

Caching provides solutions or results to the end users depending on their request. Admin then comes into the picture and needs to recreate the pages.

**Given below are the 3 different types of Caching:**

* Page Caching
* Data Caching
* Fragment Caching

**What is page caching?**

For page caching, we need to specify **@OutputCache** directive at the top of the page as per the following syntax:

<%@ OutputCache Duration=5 VaryByParam="None" %>

As should be obvious, there are two attributes to this. They are:

* **Duration** - The time in seconds of to what extent the output ought to be reserved. After the predefined length has passed, the cached output will be expelled and page content created for the following new request. That yield will again be reserved for 5 seconds and the procedure repeats.
* **VaryByParam** - This characteristic is mandatory and indicates the querystring parameters to differ the reserve.

In the above piece, we have indicated the *VaryByParam* value as *None* which means the page content to be served is the equivalent paying little mind to the parameters went through the querystring.

On the off chance that there are two requests to a similar page with fluctuating querystring parameters, e.g.:

*.../PageCachingByParam.aspx?id=10 and .../PageCachingByParam.aspx?id=11]*

A separate page content is produced for every one of them, the mandate ought to be:

<%@ OutputCache Duration=5 VaryByParam="id" %>

The page content for the two requests will each be stored for the time indicated by the Duration characteristic

To determine numerous parameters, use a semicolon to isolate the parameter names. On the off chance that we indicate the VaryByParam property as \*, the reserved substance is shifted for all parameters went through the querystring.

A few pages create distinctive substances for various programs. In such cases, there is an arrangement to shift the reserved yield for various programs. The @OutputCache order must be changed to:

<%@ OutputCache Duration=5 VaryByParam="id" VaryByCustom="browser" %>

This will change the stored yield for the browser as well as its versions. I.e., IE5, IE 6, Netscape 4, Netscape 6 will all get distinctive cached versions of the output.

**What is data caching?**

ASP.NET additionally supports the caching of information as objects. We can store objects in memory and use them crosswise over different pages in our application. This element is actualized utilizing the Cache class. This cache has a lifetime equal to that of the application. Objects can be put away as name-value pairs in the cache. A string can be embedded into the cache as pursues:

Cache["name"]="Dhruv";

The cached string value can be fetched like this:

if (Cache["name"] != null)

Labe12.Text= Cache["name"].ToString();

To embed objects into the Cache, the Add() or various forms of the Insert() of the Cache class can be utilized. These methods enable us to utilize the more dominant features given by the Cache class. One of the over-burdens of the Insert method is utilized as pursues:

Cache.Insert("Name", strName, new CacheDependency(Server.MapPath("name.txt"), DateTime.Now.AddMinutes(2), TimeSpan.Zero);

The initial two parameters are the key and the object to be embedded. The third parameter is of sort CacheDependency and encourages us to set a reliance on this value to the document named name.txt. So at whatever point this document changes, the value in the cache is expired. We can mention NULL to show no dependency on this. The fourth parameter indicates the time at which the value ought to be deleted from the cache. The last parameter is the sliding expiration parameter which demonstrates the time interim after which the object is to be expelled from the cache after this time.

The cache consequently expels the least utilized things from memory, when framework memory turns out to below. This procedure is called scavenging. We can mention priority for the objects that are added to cache so that those few things are given more importance than others:

Cache.Insert("Name", strName, new CacheDependency(Server.MapPath("name.txt"), DateTime.Now.AddMinutes(2), TimeSpan.Zero, CacheItemPriority.High, null);

The *CacheItemPriority* sets the priority level to an item. The *CacheItemPriority.High* allocates a high-level priority to an item with the goal that this is most drastically averse to be erased from the store.

**What is fragment caching?**

In some cases we should store just parts of a page. For instance, we may have a header for our page which will have similar content for all clients. There may be some content/picture in the header which may change regularly. All things considered, we will need to store this header for a span of a day.

This can be achieved by using user controls (.ascx) in web form. In asp.net for each user control we can set cache durations separately. The arrangement is to put the header content into a user control and afterward indicate that the user control content ought to be cached. This system is called fragment caching.

To indicate that a user control ought to be reserved, we utilize the @OutputCache order simply like we utilized it for the page.

**What is GAC?**

GAC stands for Global Assembly Cache.

* Any computer where the CLR is installed has a global assembly cache.
* Global Assembly Cache stores assemblies designated to be shared by multiple applications.
* It's a method to store DLLs in such a way that it can be accessible globally without worrying about any conflicts.
* Versioning is simple on the grounds that the GAC can hold numerous versions of the same DLLs without any issues.
* Bin folder of the application containing the local copy of DLL isn't required, so a couple of megabytes can be saved on the hard drive.
* Assembly can be installed in GAC by using a developer tool called the Global Assembly Cache tool (Gacutil.exe), provided in the Windows Software Development Kit (SDK).

**Add, Remove and View Assemblies in the GAC**

* To install an assembly in the GAC, use the command as below:

gacutil /i “C:\someFolder\SomeAssembly.dll”

Note: full path must be mentioned so that the tool gets the correct assembly.

* To remove an assembly from the GAC, use the command as below:

gacutil /u SomeAssembly

Note: Only an assembly name is required. This command will remove all versions of the assembly.

* To remove a specific version of an assembly, additional versioning information is required:

gacutil / u SomeAssembly, Version = 1.1.1.1

* To view the contents of the GAC:

gacutil / l

**What is the difference between assemble and namespace?**

**Namespace**

* It is a logical grouping of classes that belongs to the same functionality.
* Example: System.Data & System. Web are namespaces.

**Assembly**

* Act as the building block of .NET Framework applications.
* It is a group of types & resources which work together to create a logical unit of functionality.
* An assembly is a precompiled code that can be executed by the .NET environment.
* It may contain one or more than one Namespaces.
* Any .NET program can consists of one or more assemblies.
* Example: System.Web.dll and System.Data.dll are assemblies.
* In the other way we can say an assembly is a project. Each project within a same solution, gets compiled into its own individual assembly.
* An assembly can have multiple namespaces.

**What is asp.net?**

.NET is a designer stage comprised of programming languages, tools, and libraries for constructing a wide range of applications. ASP.NET implemented on a .NET platform using tools and libraries specifically for designing web apps.

ASP represents Active Server Pages. ASP.Net is a piece of .Net technology and it contains CLR as well. It is an open source server-side innovation that empowers the developers to fabricate amazing web administrations, sites, and web applications. It is a structure created by Microsoft on which we can grow new age sites utilizing web forms (aspx), MVC, HTML, JavaScript, and CSS and so on. It is a successor of Microsoft Active Server Pages (ASP).

These are a few things that ASP.NET adds to the .NET stage:

* Basic framework for handling web request/response
* Template for Website like RAZOR for dynamic web pages
* Libraries for basic web patterns, for example, Model View Controller (MVC)
* Providing an authentication framework that incorporates libraries, a database, and layout pages for taking care of logins, including multifactor and external authentication.

**Out of asp.net which one is stateless?**

In Classic ASP is stateless i.e. no real state management technique is implemented. If you wish to identify the user an additional code needs to be implemented

In contrast to this, .Net provides session & application state management techniques.

ASP.NET has some additional methods which can be used to persist data in addition to the Session variables, which is commonly used in classic ASP. These include ViewState and ControlState. This means that some of the tedious work from ASP such as storing information in hidden variables and then re-populating and updating these values with each post or get is automatically handled by ASP.NET.

**What is MVC?**

MVC is a framework used to make web applications. It is a design model for building the .Net applications. The web application base expands on Model-View-Controller design which isolates the application rationale from UI, and input by the user will be constrained by the Controller.

* **The Model**

The model is responsible for managing the data of the application. It responds to the request from the view and it also responds to instructions from the controller to update itself.

* **The View**

It implies the introduction of information in a specific format, activated by a controller's choice to present the information. It uses the data prepared by the Controller to generate a final presentable response. They are script-based templating frameworks like JSP, ASP, PHP and simple to incorporate with AJAX innovation.

* **The Controller**

The controller is in charge of reacting to the client input and perform interaction on the model objects. The controller gets the input, it approves the input and perform the business task that changes the state of data objects.