MSNet With C# Interview Questions

30 November 2020 09:53 AM

Q1.what is Dll hell in .Net?

->

Dll is a newer version that is not backward compatible with App A. So, **DLL HELL** is the problem where one application will install a newer version of a shared component that is not backward compatible with the version already on the machine, causing other existing applications that rely on the shared component to break.

Q2.Difference between Ref and Out keywords ?

->

REF KEYWORD	OUT KEYWORD
It is necessary the parameters should initialize before it pass to ref.	It is not necessary to initialize parameters before it pass to out.
It is not necessary to initialize the value of a parameter before returning to the calling method.	It is necessary to initialize the value of a parameter before returning to the calling method.
The passing of value through ref parameter is useful when the called method also need to change the value of passed parameter.	The declaring of parameter through out parameter is useful when a method return multiple values.
When ref keyword is used the data may pass in bi-directional.	When out keyword is used the data only passed in unidirectional.

Q3.What is Generic ?

->

Generics allow you to define the specification of the data type of programming elements in a class or a method, until it is actually **used** in the program. In other words, **generics** allow you to write a class or method that can work with any data type.

Q4.whats is Delegates ?

-> #it's a type safe function pointer.

-A delegates holds the ref of a method and then calls the method for execution.

- we can methods by creating instance of class and if it's static we directly call by class name and another way is that we can method by delegates by holding ref of method

Q5.CLR in dot net framework?

->

The Common Language Runtime (**CLR**), the virtual machine component of Microsoft . **NET Framework**, manages the execution of . **NET** programs. ... The **CLR** provides additional services including memory management, type safety, exception handling, garbage collection, security and thread management.

OR

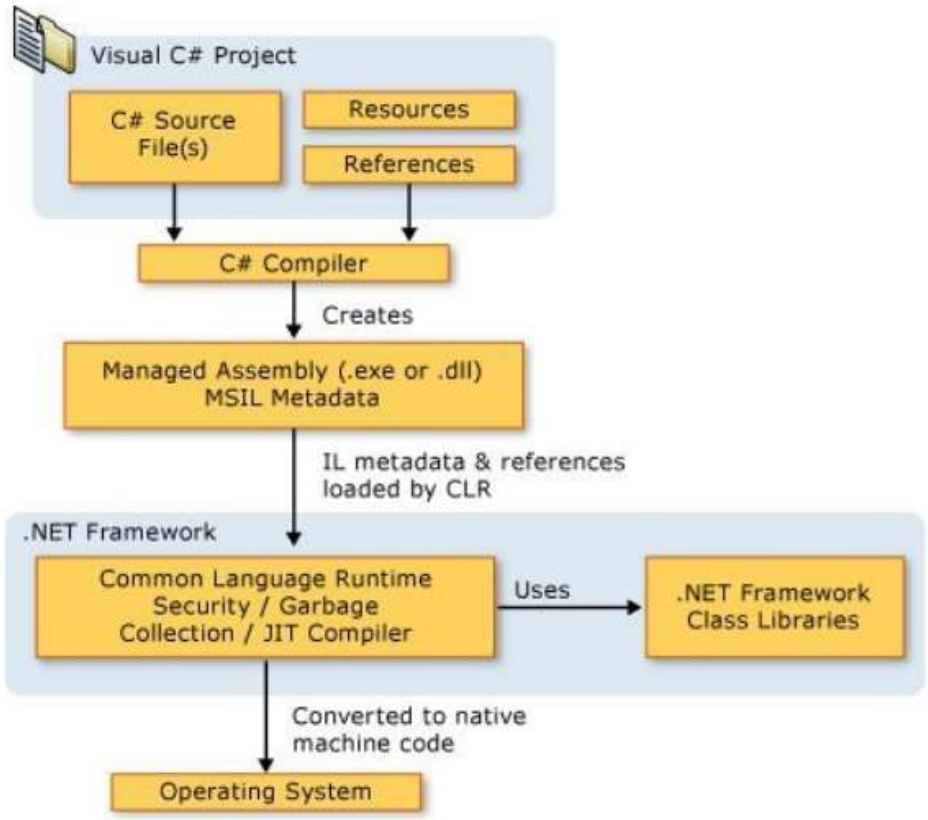
.NET CLR is an execution engine and is the integral part of .NET Framework that is responsible for executing and managing the lifecycle of an executable. CLR stands for Common Language Runtime and also knows as the runtime. CLR's responsibility is to mange any code written in .NET languages such as C#, VB.NET, or F#. The code that targets CLR is also knows as the "managed code".

CLR is the common runtime for all .NET languages. Each language that supports .NET must follow a common standard and must emit and attach metadata with every binary, or portable executable (PE). The metadata includes the types, objects, members, and references. The runtime uses metadata to understand types, load types, allocate memory and manages memory lifetime, resolve method invocations, generate native code, enforce security, and set run-time context boundaries.

Here is a list of CLR functions:

- Automatic memory management with Garbage collection (GC) including releasing unused objects, managing pool of threads, and reserve locations.
- Cross language interoperability
- Structured exception handling
- Code access security
- Thread execution, thread pooling, and context
- Debugging
- Verification and compliance

The following diagram shows the execution model of CLR.



Q6.What is CLI?

->

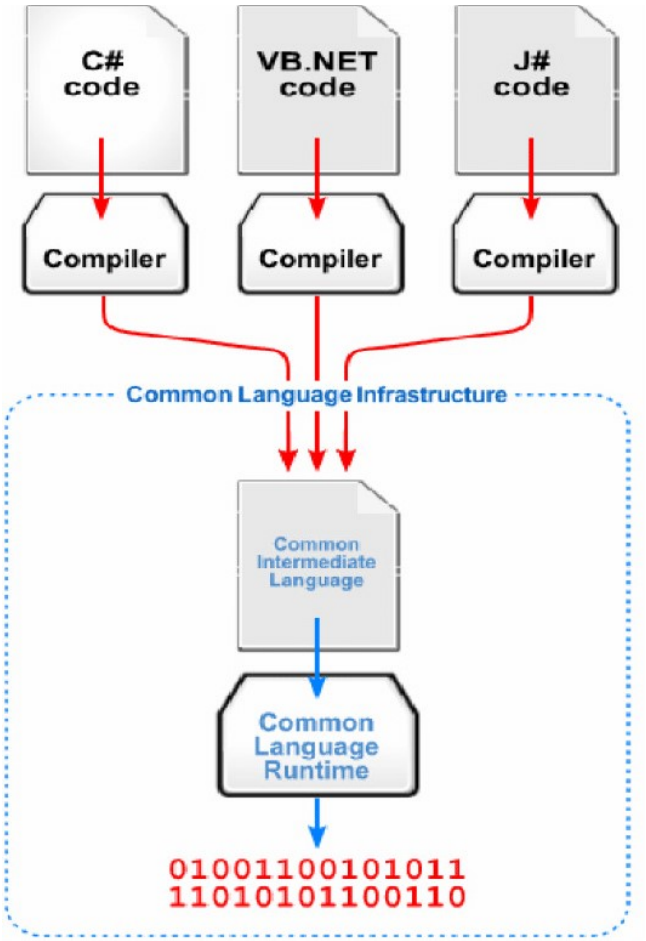
Common Language Infrastructure (CLI)

#

The CLR code uses a common type system (CTS) that is based on a common language infrastructure (CLI).

CLI is a specification developed by Microsoft that describes the executable code and runtime environment. In simple terms this allows us to use various high-level programming languages on various machines without rewriting the code.

CLI is divided into four main components.



Common intermediate Language(CIL)

DotNet compatible language compiles to a second platform-neutral language called Common Intermediate language(CIL).

Common type system (CTS)

CTS defines some basic data types and every language that is designed for use with .NET framework should be able to match its data types to these defined basic data types. So when various languages are designed following CTS, they will be able to communicate with each other and this is nothing but cross-language interoperability or communication.

Common Language Specification (CLS)

CLS is a set of specifications that must be met by every language to be considered as .NET compliant. It is a subset of CTS types and a set of rules. Example: Elimination of pointers and multiple inheritance.

Metadata

Metadata gives information about all the classes and the class members defined in the assembly.

Virtual Execution System (VES)

VES loads and runs the programs that are compatible with the CLI using metadata. To make it clear, CLI is a set of specifications for a virtual operating system that is nothing but a Common Language Runtime (CLR).

C# | Namespaces

Namespaces are used to organize the classes. It helps to control the scope of methods and classes in larger **.Net** programming projects. In simpler words you can say that it provides a way to keep one set of names(like class names) different from other sets of names. The biggest advantage of using namespace is that the class names which are declared in one namespace will not clash with the same class names declared in another namespace. It is also referred as **named group of classes** having common features. The members of a namespace can be **namespaces, interfaces, structures, and delegates.**

Q7.What Is the difference between system.console.writeline and console.writeline in c#?

->
The difference is that System.Console.WriteLine explicitly qualifies the namespace System whereas Console.WriteLine does not do so.
Console.WriteLine is only valid if the using directive has been used to qualify the System namespace either in the same file but outside the namespace or inside any of the namespaces that contains the class where the call to the method is being made.
System.Console.WriteLine is only needed if the System namespace was not qualified using the using directive for that namespace or if there is another namespace that has been qualified that contains either a static class called Console containing method called WriteLine or a non-static class containing a static method called WriteLine.
If the System namespace has already been qualified using the using directive and there is no other qualified namespace that contains a Console.WriteLine, there is no difference in the code that is generated by the compiler.

Console Class in C#

A console is an operating system window through which a user can communicate with the operating system or we can say a console is an application in which we can give text as an input from the keyboard and get the text as an output from the computer end.

.NET FrameWork Class Library(FCL)

The Framework Class Library or **FCL** provides the system functionality in the . NET Framework as it has various classes, data types, interfaces, etc. to perform multiple functions and build different types of applications such as desktop applications, web applications, mobile applications, etc.

ThreadPool Class

Provides a pool of threads that can be used to execute tasks, post work items, process asynchronous I/O, wait on behalf of other threads, and process timers.