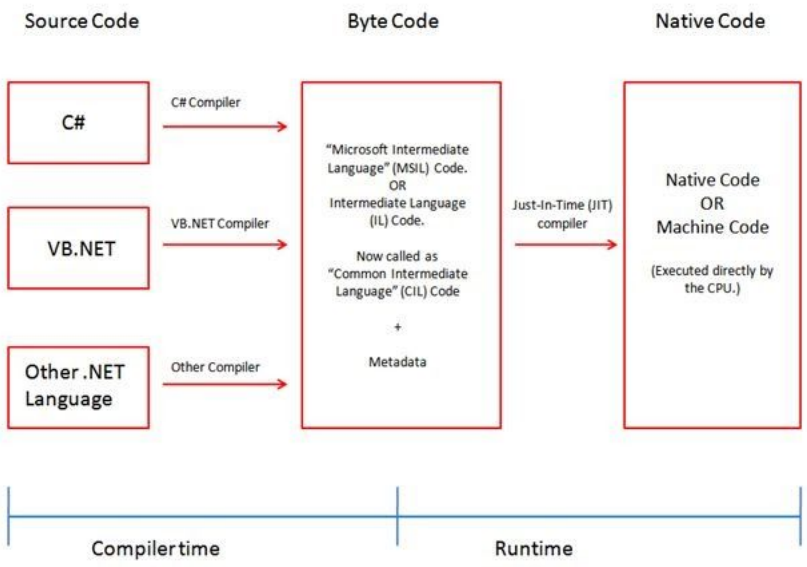
.NET Framework Introduction Assignment

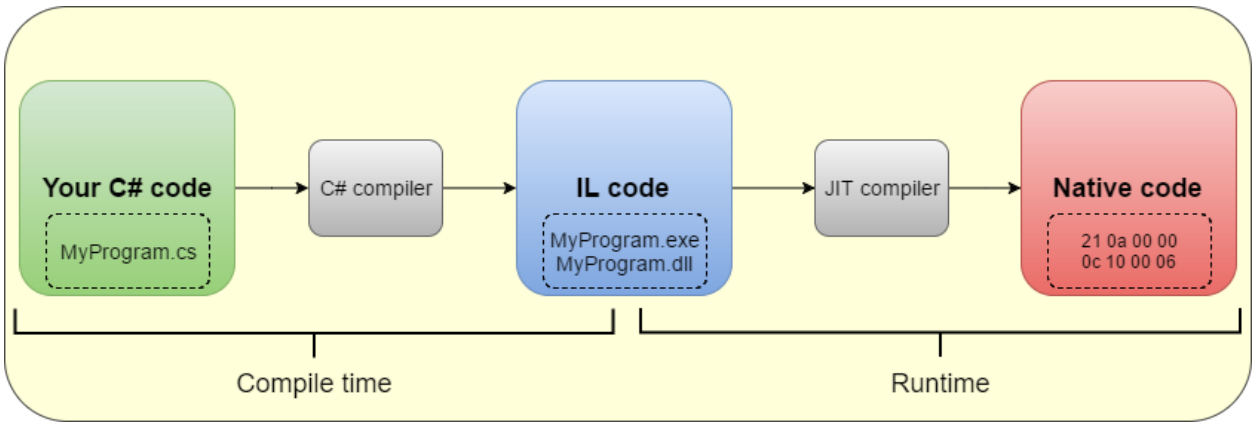
By Praveen Kumar Dande

1) Unmanaged Code is directly executed by OS but Managed Code undergoes a certain steps to get executed.



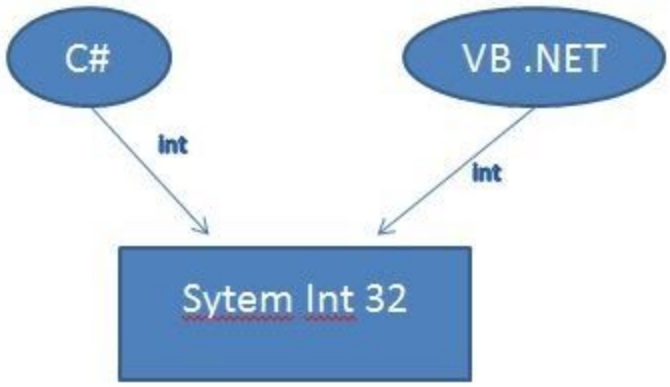
When any application is written targeting .net framework, the code will get compiled with respective compiler (say csc compiler OR vb compiler), based on whether we are using C# as a language or VB.

After compilation of the code, what we get is an **IL Assembly** (Intermediate Language Assembly) later which will be converted into **IL Code Or Byte Code** (Intermediate Language Code). The .exe or .dll file what we get contains IL Code, later which will converted in to native code. The Process of Converting IL Code to native code will be done with the help of **JIT compiler(Just-In-Time Compiler). JIT** will be present at **CLR**(Common Language Runtime) level.



2)
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.

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.



Functions of the Common Type System

- To establish a framework that helps enable cross-language integration, type safety, and high performance code execution.
- To provide an object-oriented model that supports the complete implementation of many programming languages.
- To define rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.
- The CTS also defines the rules that ensures that the data types of objects written in various languages are able to interact with each other.
- The CTS also specifies the rules for type visibility and access to the members of a type, i.e. the CTS establishes the rules by which assemblies form scope for a type, and the Common Language Runtime enforces the visibility rules.
- The CTS defines the rules governing type inheritance, virtual methods and object lifetime.
- Languages supported by .NET can implement all or some common data types…
- it is used to communicate with other languages

The specification for the CTS is contained in Ecma standard 335, "Common Language Infrastructure (CLI) Partitions I to VI.

3)

Garbage Collector:

It is used to provide the Automatic Memory Management feature. If there was no garbage collector, programmers would have to write the memory management codes which will be a kind of overhead on programmers.

JIT(Just In Time Compiler):

It is responsible for converting the CIL(Common Intermediate Language) into machine code or native code using the Common Language Runtime environment.

Exception Handler

It handles the exception at runtime to avoid application failure.

4)

Library:

- Libraries contain code that can be used in many programs.
- They can be two types static libraries(.lib) and dynamic libraries.(.dll)
- (.lib) library code need to be called at the compilation while (.dll) is called at runtime.

EXE

- An exe always runs in its own address space i.e., It is a separate process. It has its own entry point
- The purpose of an EXE is to launch a separate application of its own
- Exe cannot be shared with the other program application.

DLL.

- A dll always needs a host exe to run. i.e., it can never run in its own address space. It doesn have a entry point.
- The purpose of a DLL is to have a collection of methods/classes which can be re-used from some other application.
- DLL binding occurs at run-time. That is why its called "Dynamic Link" library.

5)The common language runtime (CLR) supports a security model called code access security for managed code. In this model, permissions are granted to assemblies based on the identity of the code.

The code access security mechanism supported by the CLR is based on the assumption that the runtime can host both fully trusted and partially trusted code. The resources that are protected by CLR code access security are typically wrapped by managed application programming interfaces that require the corresponding permission before allowing access to the resource. The demand for the permission is satisfied only if all the callers (at the assembly level) in the call stack have the corresponding resource permission.

The security policy that determines the permissions granted to assemblies is defined in three different places:
Machine policy: This is the policy in effect for all managed code running in the machine on which SQL Server is installed.
User policy: This is the policy in effect for managed code hosted by a process. For SQL Server, the user policy is specific to the Windows account on which the SQL Server service is running.
Host policy: This is the policy set up by the host of the CLR (in this case, SQL Server) that is in effect for managed code running in that host.

Type Safety:
Type safety in .NET has been introduced to prevent the objects of one type from peeking into the memory assigned for the other object.
In simple terms, it implies that the CLR and the Language Compiler has to ensure that a variable / object - which belongs to a specific datatype (ValueType or ReferenceType) shall have the said attributes and behaviors, as defined by that type.

For example, if you have declared a variable as an integer, it cannot be assigned any value which is not an integer (by implicit conversion, or explicit conversion).