Chapter 1: The .NET Framework

➢ **Introduction to .NET framework :**

.Net Framework is a software development platform developed by Microsoft for building and running Windows applications. The .Net framework consists of developer tools, programming languages, and libraries to build desktop and web applications. It is also used to build websites, web services, and games.

The .Net framework was meant to create applications, which would run on the Windows Platform. The first version of the .Net framework was released in the year 2002. The version was called .Net framework 1.0. The Microsoft .Net framework has come a long way since then, and the current version is .Net Framework 4.7.2.

The Microsoft .Net framework can be used to create both - **Form-based** and **Web-based** applications. Web services can also be developed using the .Net framework.

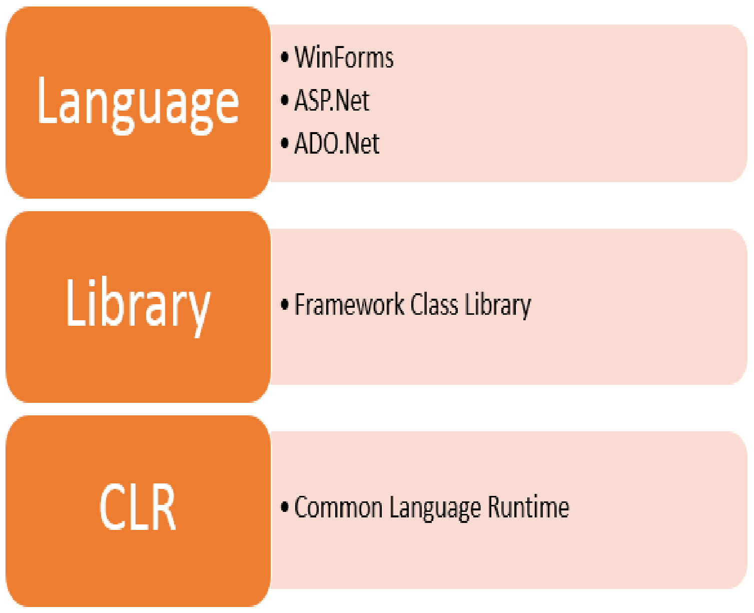
The framework also supports various programming languages such as Visual Basic and C#. So developers can choose and select the language to develop the required application.

- .Net Framework Architecture
- .NET Components
- .Net Framework Design Principle

.Net Framework Architecture

.Net Framework Architecture is a programming model for the .Net platform that provides an execution environment and integration with various programming languages for simple development and deployment of various Windows and desktop applications. It consists of class libraries and reusable components.

The basic architecture of the .Net framework is as shown below.



.Net Framework Architecture Diagram

.NET Components

The architecture of .Net framework is based on the following key components;

1. Common Language Runtime

The "Common Language Infrastructure" or CLI is a platform in .Net architecture on which the .Net programs are executed.

The CLI has the following key features:

- **Exception Handling -** Exceptions are errors which occur when the application is executed.

  Examples of exceptions are:

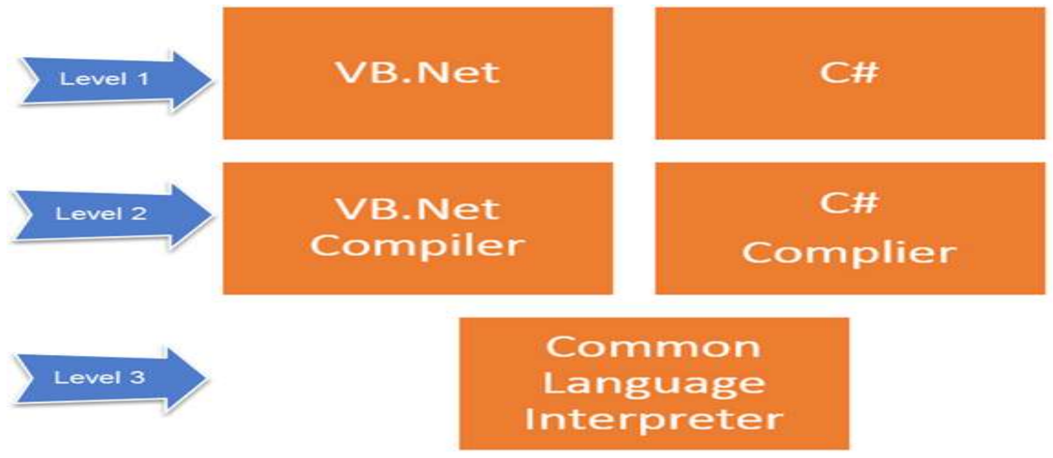
  - If an application tries to open a file on the local machine, but the file is not present.
  - If the application tries to fetch some records from a database, but the connection to the database is not valid.
- **Garbage Collection -** Garbage collection is the process of removing unwanted resources when they are no longer required.

  Examples of garbage collection are

  - A File handle which is no longer required. If the application has finished all operations on a file, then the file handle may no longer be required.
  - The database connection is no longer required. If the application has finished all operations on a database, then the database connection may no longer be required.
- Working with Various programming languages –

As noted in an earlier section, a developer can develop an application in a variety of .Net programming languages.

1. **Language -** The first level is the programming language itself, the most common ones are VB.Net and C#.
2. **Compiler –** There is a compiler which will be separate for each programming language. So underlying the VB.Net language, there will be a separate VB.Net compiler. Similarly, for C#, you will have another compiler.
3. **Common Language Interpreter** – This is the final layer in .Net which would be used to run a .net program developed in any programming language. So the subsequent compiler will send the program to the CLI layer to run the .Net application.



2. Class Library

The .NET Framework includes a set of standard class libraries. A class library is a collection of methods and functions that can be used for the core purpose.

For example, there is a class library with methods to handle all file-level operations. So there is a method which can be used to read the text from a file. Similarly, there is a method to write text to a file.

Most of the methods are split into either the System.* or Microsoft.* namespaces. (The asterisk * just means a reference to all of the methods that fall under the System or Microsoft namespace)

A namespace is a logical separation of methods. We will learn these namespaces more in detail in the subsequent chapters.

3. Languages

The types of applications that can be built in the .Net framework is classified broadly into the following categories.

- **WinForms** – This is used for developing Forms-based applications, which would run on an end user machine. Notepad is an example of a client-based application.
- **ASP.Net** – This is used for developing web-based applications, which are made to run on any browser such as Internet Explorer, Chrome or Firefox.
    - The Web application would be processed on a server, which would have Internet Information Services Installed.
    - Internet Information Services or IIS is a Microsoft component which is used to execute an Asp.Net application.
    - The result of the execution is then sent to the client machines, and the output is shown in the browser.
- **ADO.Net** – This technology is used to develop applications to interact with Databases such as Oracle or Microsoft SQL Server.

Microsoft always ensures that .Net frameworks are in compliance with all the supported Windows operating systems.

.Net Framework Design Principle

Now in this .Net Architecture tutorial, we will learn the design priciples of .Net framework. The following design principles of the .Net framework is what makes it very relevant to create .Net based applications.

1. **Interoperability** - The .Net framework provides a lot of backward support. Suppose if you had an application built on an older version of the .Net framework, say 2.0. And if you tried to run the same application on a machine which had the higher version of the .Net framework, say 3.5. The application would still work. This is because with every release, Microsoft ensures that older framework versions gel well with the latest version.
2. **Portability**- Applications built on the .Net framework can be made to work on any Windows platform. And now in recent times, Microsoft is also envisioning to make Microsoft products work on other platforms, such as iOS and Linux.
3. **Security** - The .NET Framework has a good security mechanism. The inbuilt security mechanism helps in both validation and verification of applications. Every application can explicitly define their security mechanism. Each security mechanism is used to grant the user access to the code or to the running program.

4. **Memory management** - The Common Language runtime does all the work or memory management. The .Net framework has all the capability to see those resources, which are not used by a running program. It would then release those resources accordingly. This is done via a program called the "Garbage Collector" which runs as part of the .Net framework.

   The garbage collector runs at regular intervals and keeps on checking which system resources are not utilized, and frees them accordingly.

5. **Simplified deployment** - The .Net framework also have tools, which can be used to package applications built on the .Net framework. These packages can then be distributed to client machines. The packages would then automatically install the application.

➢ The origin of .NET Technology:

.NET is a software framework that is designed and developed by Microsoft. The first version of the .Net framework was 1.0 which came in the year 2002. In easy words, it is a virtual machine for compiling and executing programs written in different languages like C#, VB.Net, etc.
It is used to develop Form-based applications, Web-based applications, and Web services. There is a variety of programming languages available on the .Net platform, VB.Net and C# being the most common ones. It is used to build applications for Windows, phone, web, etc. It provides a lot of functionalities and also supports industry standards.
.NET Framework supports more than 60 programming languages in which 11 programming languages are designed and developed by Microsoft. The remaining Non-Microsoft Languages which are supported by .NET Framework but not designed and developed by Microsoft.

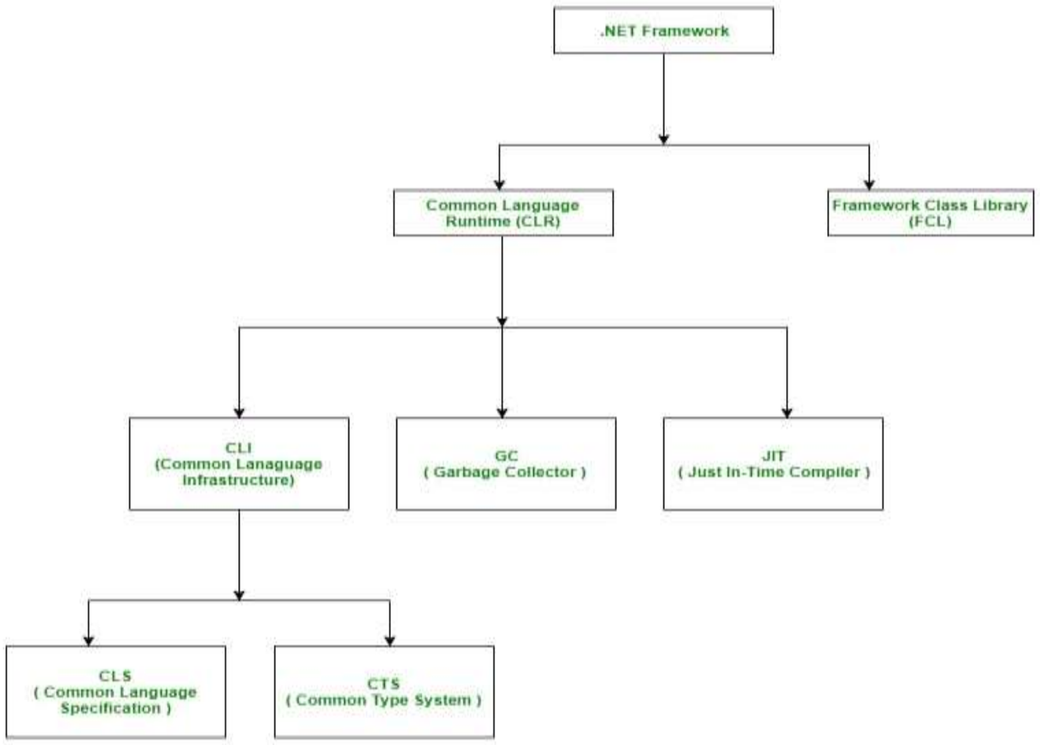
11 Programming Languages which are designed and developed by Microsoft are:

- C#.NET
- VB.NET
- C++.NET
- J#.NET
- F#.NET
- JSCRIPT.NET
- WINDOWS POWERSHELL
- IRON RUBY
- IRON PYTHON
- C OMEGA
- ASML(Abstract State Machine Language)

Main Components of .NET Framework

- **Common Language Runtime(CLR):** CLR is the basic and Virtual Machine component of the `.NET` Framework. It is the run-time environment in the .NET Framework that runs the codes and helps in making the development process easier by providing the various services such as remoting, thread management, type-safety, memory management, robustness, etc.. Basically, it is responsible for managing the execution of `.NET` programs regardless of any .NET programming language. It also helps in the management of code, as code that targets the runtime is known as the Managed Code and code doesn't target to runtime is known as Unmanaged code.
- **Framework Class Library(FCL):** It is the collection of reusable, object-oriented class libraries and methods, etc that can be integrated with CLR. Also called the Assemblies. It is just like the header files in C/C++ and packages in the java. Installing `.NET` framework

- basically is the installation of CLR and FCL into the system. Below is the overview of .NET Framework


.NET Framework
Common Language Runtime (CLR)
Framework Class Library (FCL)
CLI (Common Lanaguage Infrastructure)
GC (Garbage Collector)
JIT (Just In-Time Compiler)
CLS (Common Language Specification)
CTS (Common Type System)

Is .NET application platform dependent or platform independent?

The combination of *Operating System Architecture and CPU Architecture* is known as the platform. Platform dependent means the programming language code will run only on a particular Operating System. A *.NET application is platform-dependent* because of the .NET framework which is only able to run on the Windows-based operating system. The .Net application is platform-independent also because of the *Mono framework*. Using the Mono framework the .Net application can run on any Operating System including windows. Mono framework is a third party software developed by the Novell company which is now a part of Micro Focus Company. It is a paid framework.

Important Points:

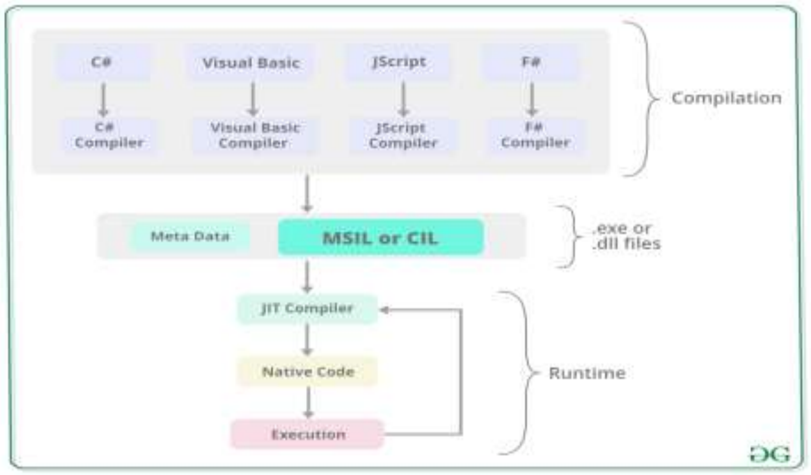
- Visual Studio is the development tool that is used to design and develop the .NET applications. For using Visual Studio, the user has to first install the .NET framework on the system.
- In the older version of Windows OS like XP SP1, SP2 or SP3, the .NET the framework was integrated with installation media.
- Windows 8, 8.1 or 10 do not provide a pre-installed version 3.5 or later of .NET Framework. Therefore, a version higher than 3.5 must be installed either from a Windows installation media or from the Internet on demand. Windows update will give recommendations to install the .NET framework.

➢ **Microsoft Intermediate Language (MSIL):**

The Microsoft Intermediate Language (MSIL), also known as the Common Intermediate Language (CIL) is a set of instructions that are platform independent and are generated by the language-specific compiler from the source code. The MSIL is platform independent and consequently, it can be executed on any of the Common Language Infrastructure supported environments such as the Windows .NET runtime.

The MSIL is converted into a particular computer environment specific machine code by the JIT compiler. This is done before the MSIL can be executed. Also, the MSIL is converted into the machine code on a requirement basis i.e. the JIT compiler compiles the MSIL as required rather than the whole of it.

Execution process in Common Language Runtime (CLR): The execution process that includes the creation of the MSIL and the conversion of the MSIL into machine code by the JIT compiler is given as follows:

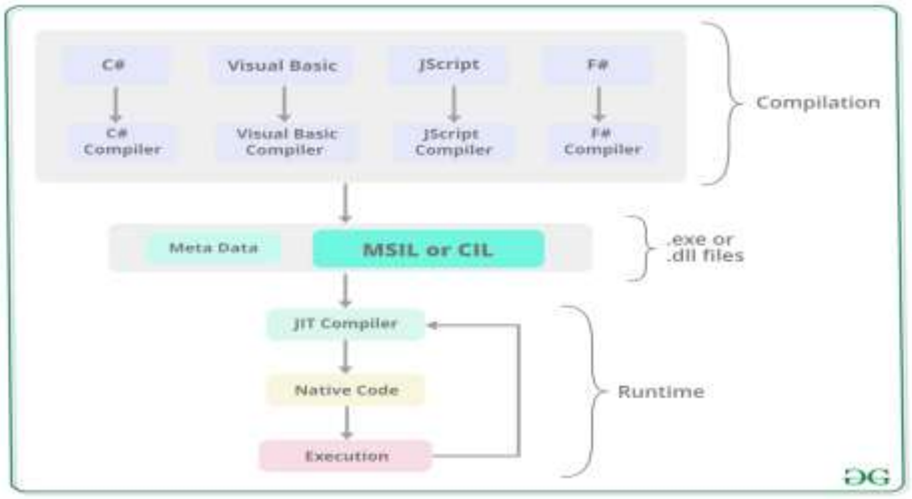


- The source code is converted into the MSIL by a language-specific compiler in the compile time of the CLR. Also, along with the MSIL, metadata is also produced in the compilation. The metadata contains information such as the definition and signature of the types in the code, runtime information, etc.
- A Common Language Infrastructure (CLI) assembly is created by assembling the MSIL. This assembly is basically a compiled code library that is used for security, deployment, versioning, etc. and it is of two types i.e. process assembly (EXE) and library assembly (DLL).
- The JIT compiler then converts the Microsoft Intermediate Language(MSIL) into the machine code that is specific to the computer environment that the JIT compiler runs on. The MSIL is converted into the machine code on a requirement basis i.e. the JIT compiler compiles the MSIL as required rather than the whole of it.
- The machine code obtained using the JIT compiler is then executed by the processor of the computer.

➢ **Just-In-Time compiler(JIT):**

Just-In-Time compiler(JIT) is a part of **Common Language Runtime (CLR)** in *.NET* which is responsible for managing the execution of *.NET* programs regardless of any *.NET* programming language. A language-specific compiler converts the source code to the intermediate language. This intermediate language is then converted into the machine code by the Just-In-Time (JIT) compiler. This machine code is specific to the computer environment that the JIT compiler runs on.

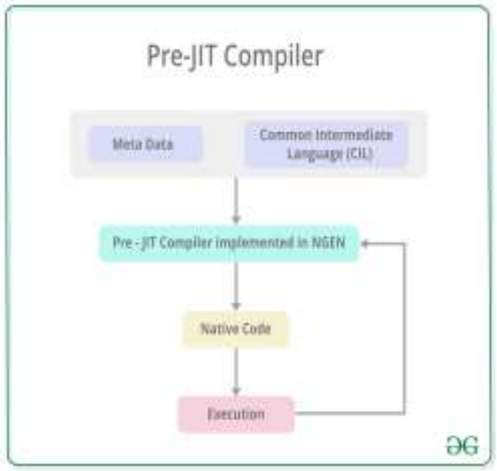
- **Working of JIT Compiler:** The JIT compiler is required to speed up the code execution and provide support for multiple platforms. Its working is given as follows:



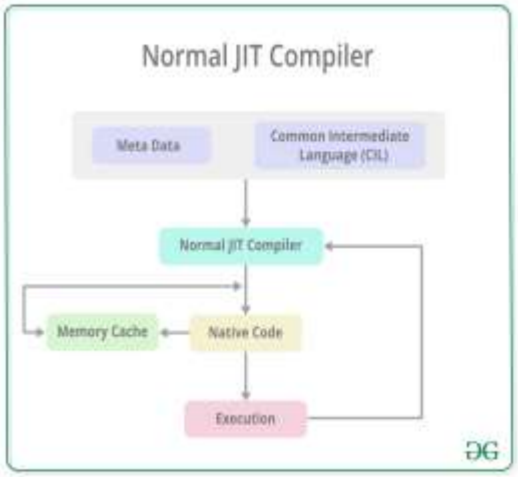
The JIT compiler converts the Microsoft Intermediate Language(MSIL) or Common Intermediate Language(CIL) into the machine code. This is done before the MSIL or CIL can be executed. The MSIL is converted into machine code on a requirement basis i.e. the JIT compiler compiles the MSIL or CIL as required rather than the whole of it. The compiled MSIL or CIL is stored so that it is available for subsequent calls if required.

Types of Just-In-Time Compiler: There are **3** types of JIT compilers which are as follows:

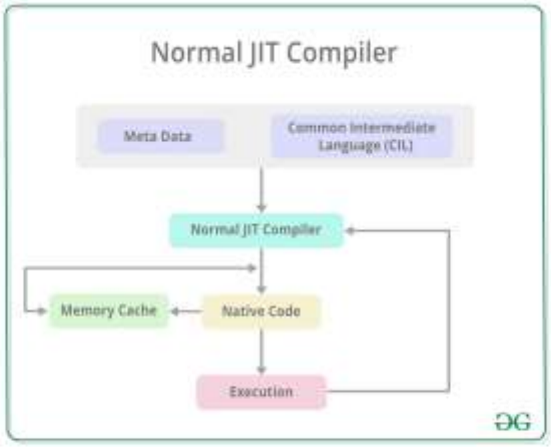
- **Pre-JIT Compiler:** All the source code is compiled into the machine code at the same time in a single compilation cycle using the Pre-JIT Compiler. This compilation process is performed at application deployment time. And this compiler is always implemented in the ***Ngen.exe (Native Image Generator)***.


Pre-JIT Compiler
Meta Data
Common Intermediate Language (CIL)
Pre - JIT Compiler implemented in NGEN
Native Code
Execution

- **Normal JIT Compiler:** The source code methods that are required at run-time are compiled into machine code the first time they are called by the Normal JIT Compiler. After that, they are stored in the cache and used whenever they are called again.



- **Econo JIT Compiler:** The source code methods that are required at run-time are compiled into machine code by the Econo JIT Compiler. After these methods are not required anymore, they are removed. This JIT compiler is obsolete starting from dotnet 2.0



Advantages of JIT Compiler:

- The JIT compiler requires less memory usage as only the methods that are required at run-time are compiled into machine code by the JIT Compiler.
- Page faults are reduced by using the JIT compiler as the methods required together are most probably in the same memory page.
- Code optimization based on statistical analysis can be performed by the JIT compiler while the code is running.

Disadvantages of JIT compiler:

- The JIT compiler requires more startup time while the application is executed initially.
- The cache memory is heavily used by the JIT compiler to store the source code methods that are required at run-time.