Chapter-4 Dot NET Framework

Introduction to .NET Framework

- **.NET** is a software framework which is designed and developed by Microsoft.
- In easy words, it is a virtual machine for compiling and executing programs written in different languages like C#, VB.Net etc.
- There is a variety of programming languages available on the .Net platform, VB.Net and C# being the most common and is used to build applications for Windows, phone, web etc. It provides a lot of functionalities and also supports industry standards.
- Visual Studio is the development tool which 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
- Windows 8, 8.1 or 10 do not provide a pre-installed version 3.5 or later of .NETFramework. 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
- .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)

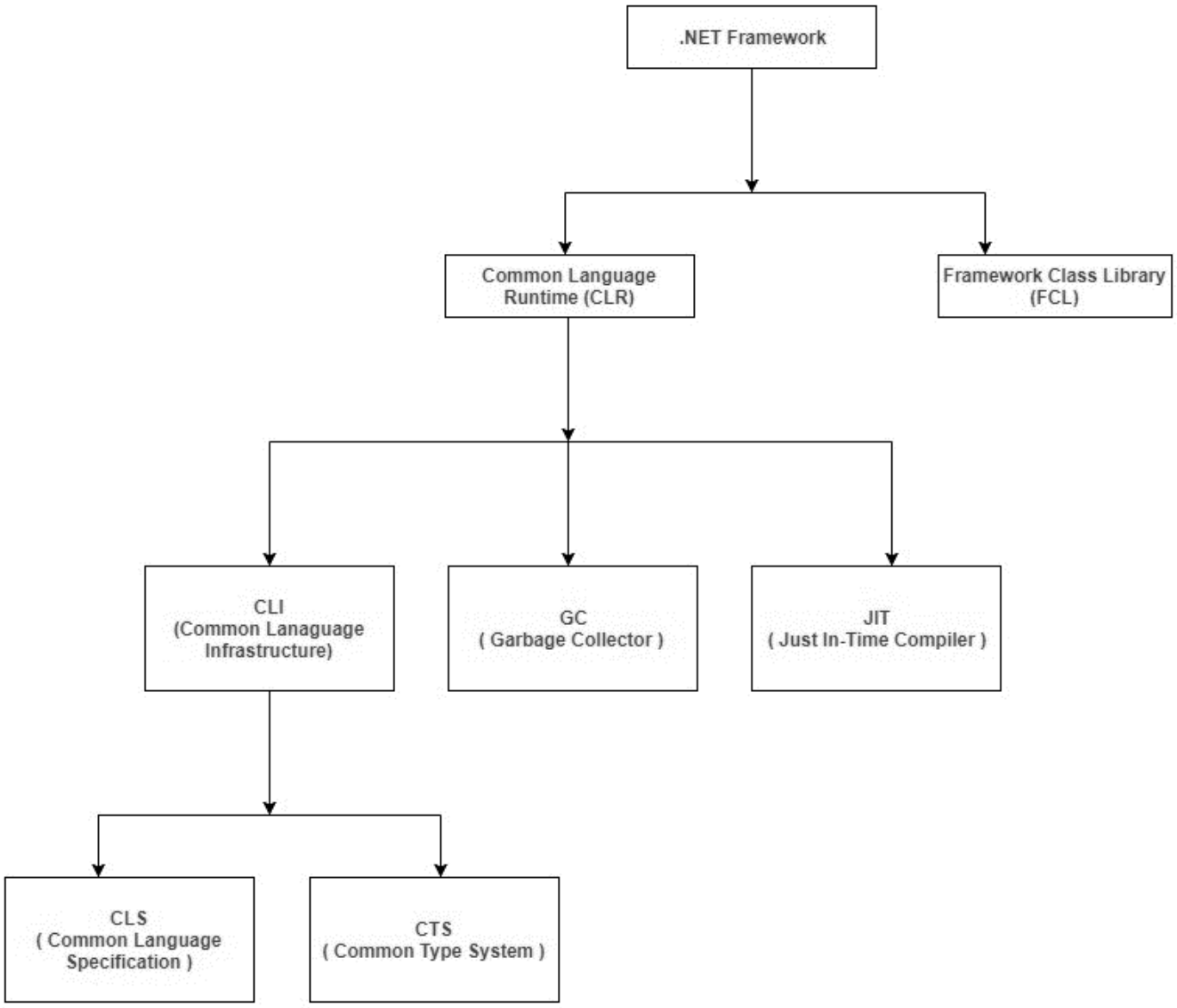


Fig: Component / Block-Diagram of component of Dot net Framework:

Architecture of .NET Framework

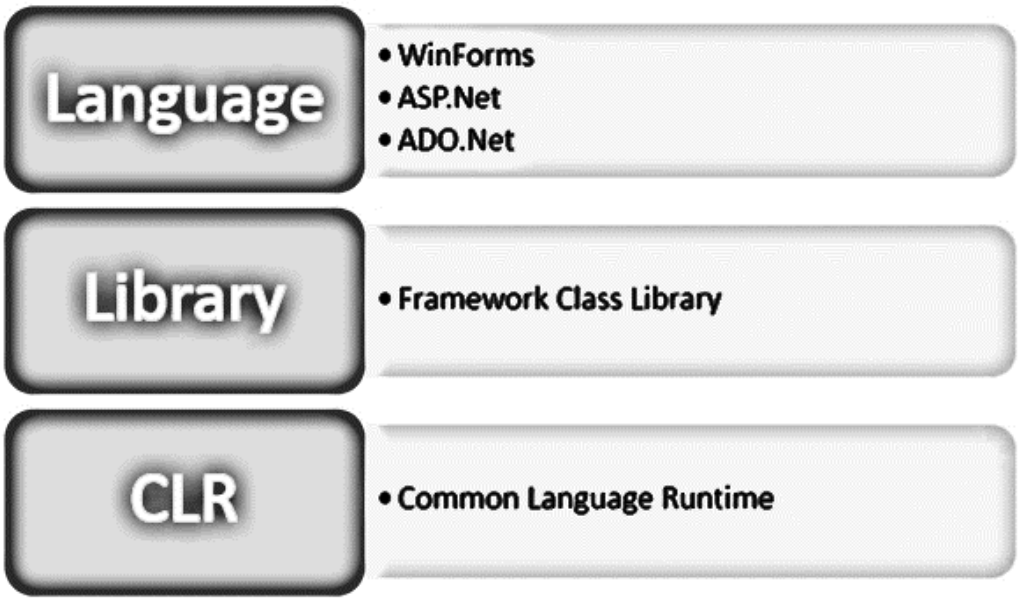


Figure: Architectural diagram of .net framework

C# Features

C# is object oriented programming language. It provides a lot of **features** that are given below.

1. Simple
2. Modern programming language
3. Object oriented
4. Type safe
5. Interoperability
6. Scalable and Updateable
7. Component oriented
8. Structured programming language
9. Rich Library
10. Fast speed

1) Simple

C# is a simple language in the sense that it provides structured approach (to break the problem into parts), rich set of library functions, data types etc.

2) Modern Programming Language

C# programming is based upon the current trend and it is very powerful and simple for building scalable, interoperable and robust applications.

C# supports number of modem features, such as:

- Automatic Garbage Collection
- Error handling features
- Modern debugging features
- Robust Security features

3) Object Oriented

In C#, everything is an object. There are no more global functions, variable and constants.

It supports all three object oriented features:

- Encapsulation
- Inheritance
- Polymorphism

4) Type Safe

Type safety promotes robust programming. Some examples of type safety are:

- All objects and arrays are initialized by zero dynamically
- An error message will be produced, on use of any uninitialized variable
- Automatic checking of array (out of bound and etc.)

5) Interoperability

C# provides support for using COM (component object model) objects, no matter what language was used to author them. C# also supports a special feature that enables a program to call out any native API.

6) Scalable and Updateable

C# is automatic scalable and updateable programming language. For updating our application we delete the old files and update them with new ones.

7) Component Oriented

C# is component oriented programming language. It is the predominant software development methodology used to develop more robust and highly scalable applications.

8) Structured Programming Language

C# is a structured programming language in the sense that we can break the program into parts using functions. So, it is easy to understand and modify.

9) Rich Library

C# provides a lot of inbuilt functions that makes the development fast.

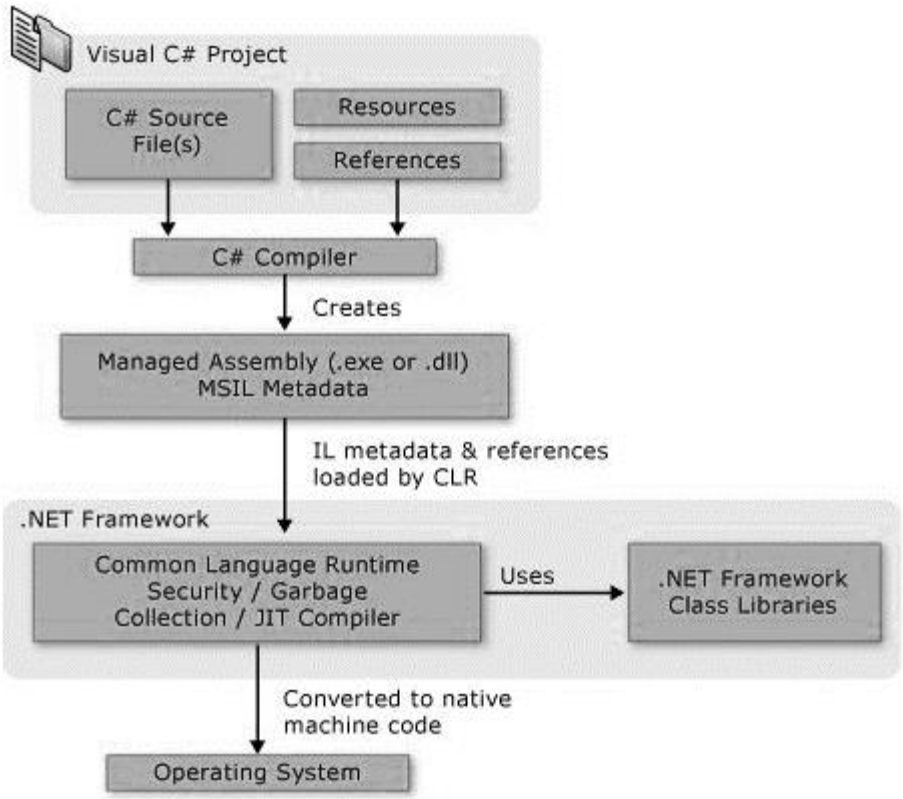
10) Fast Speed

The compilation and execution time of C# language is fast.

Common Language Runtime (CLR)

- The heart of .NET Framework is a run-time environment called the common language runtime (CLR) which is responsible for managing the execution of .NET programs regardless of any .NET programming language.
- The CLR manages the life cycle and executes .NET applications (code).
- It also provides services that make the development process easier.
- It is a program execution engine that loads and executes the program.
- It converts the program into native code.
- It acts as an interface between the framework and operating system.
- It is the run-time environment in the .NET Framework that runs the codes and in making the development process easier by providing the various services helps such as exception handling, remoting, thread management, memory management, and garbage collection. Moreover, it provides security, type-safety, interoperability, and portability.
- 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.

Let's try to look at the process of a C# application in this below diagram.



As you can see from the above diagram, the CLR provides several services.

Functions of .NET CLR

- Convert code into CLI
- Exception handling
- Type safety
- Memory management (using the Garbage Collector)
- Security
- Improved performance
- Language independency
- Platform independency
- Architecture independency

Components of .NET CLR

The key components of CLR includes the following:

- Class Loader - Used to load all classes at run time.

- MSIL (Microsoft Intermediate Language) code to Native code - The Just in Time (JTI) compiler will convert MSIL code into native code.
- Code Manager - It manages the code at run time.
- Garbage Collector - It manages the memory. Collect all unused objects and deallocate them to reduce memory.
- Thread Support - It supports multithreading of our application.
- Exception Handler - It handles exceptions at run time.

Benefits of .NET CLR

The runtime provides the following benefits:

- ❖ Performance improvements.
- ❖ The ability to easily use components developed in other languages.
- ❖ Extensible types provided by a class library.
- ❖ Language features such as inheritance, interfaces, and overloading for object-oriented programming.
- ❖ Support for explicit free threading that allows creation of multithreaded, scalable applications.
- ❖ Support for structured exception handling.
- ❖ Support for custom attributes.
- ❖ Garbage collection.
- ❖ Use of delegates instead of function pointers for increased type safety and security.

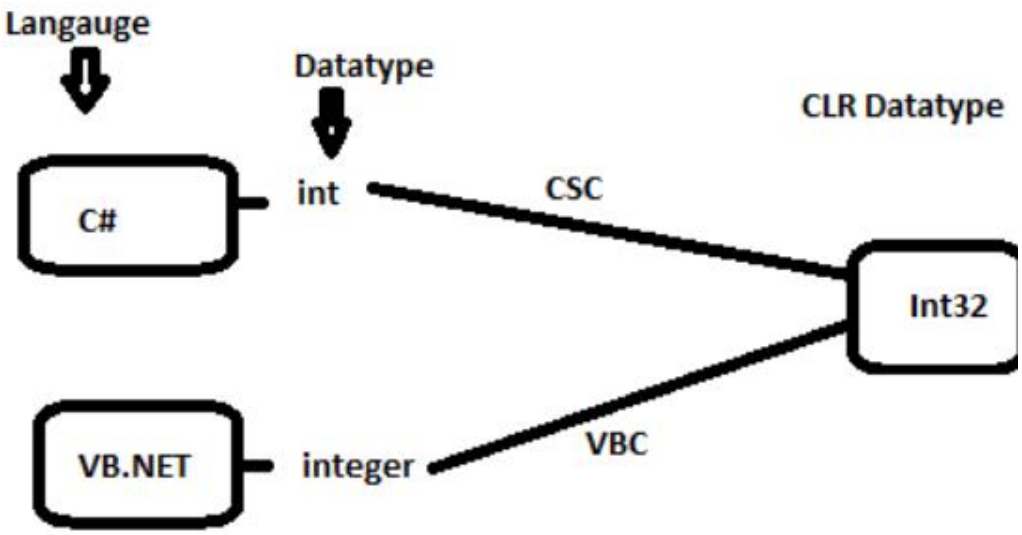
Common Type System (CTS)

- ❖ Common Type System (CTS) describes the datatypes that can be used by managed code.
- ❖ CTS defines how these types are declared, used and managed in the runtime.
- ❖ It facilitates cross-language integration, type safety, and high-performance code execution.

OR we can also understand like,

- ❖ CTS deals with the data type.
- ❖ So here we have several languages and each and every language has its own data type and one language data type cannot be understandable by other languages but .NET Framework language can understand all the data types.

For example: C# has an **int** data type and VB.NET has **Integer** data type. Hence a variable declared as an **int** in C# and **Integer** in VB.NET, finally after compilation, uses the same structure **Int32** from CTS.



- For Communicating between programs written in any .NET complaint language, the types have to be compatible on the basic level.

The common type system (CTS) performs the following functions:

- Establishes a framework that helps enable cross-language integration, type safety, and high-performance code execution.
- Provides an object-oriented model that supports the complete implementation of many programming languages.
- Defines rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.
- Provides a library that contains the primitive data types (such as Boolean, Byte, Char, Int32, and UInt64) used in application development.

All types in .NET are either value types or reference types.

- When variables are passed by values as parameters, then no change can be made to them when retrieved.

- When variables are passed by reference, their address is passed due to which a change in their values can be reflected back in the calling function.

The common type system in .NET supports the following five categories of types:

- Classes
- Structures
- Enumerations
- Interfaces
- Delegates

Common Language Specification (CLS)

- CLS stands for Common Language Specification and it is a subset of CTS.
- CLS defines a minimum set of features that must be supported by all languages that target CLR.
- CTS and CLS are parts of .NET CLR and are responsible for type safety within the code. Both allow cross-language communication and type safety.
- It defines a set of rules and restrictions that every language must follow which runs under the .NET framework. The languages which follow these set of rules are said to be CLS Compliant. In simple words, CLS enables cross-language integration or Interoperability.

For Example

If we talk about C# and VB.NET then, in C# every statement must have to end with a semicolon. It is also called a statement Terminator, but in VB.NET each statement should not end with a semicolon (;).

So these syntax rules which you have to follow from language to language differ but CLR can understand all the language Syntax because in .NET each language is converted into MSIL(Microsoft Intermediate Language) code after compilation and the MSIL code is language specification of CLR.

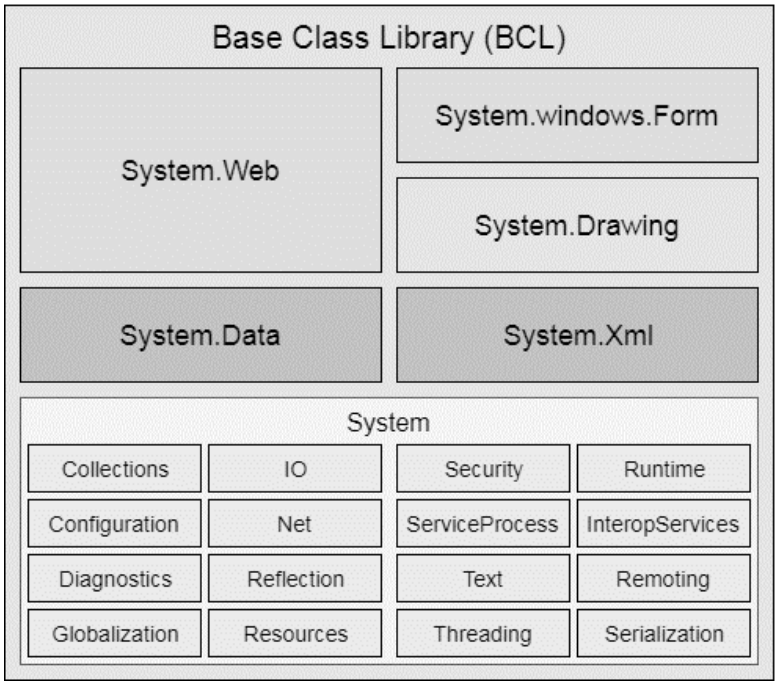
- Sometimes there is a need for code written in one language to access another language, but programming languages that target CLR are different from one other.

For Example C# is case sensitive but VB is not. This can cause a problem when we access code written in one language from another language. So .Net has come up with CLS; i.e., Common Language Specification

The Base Class Library, the Dot Net Class Library, Intermediate Language

Base Class Library (BCL):

- It is a library of classes, interfaces, and value types.
- This library system functionality and is the foundation of .NET Framework applications, components, and controls are built.
- The .NET base class library exists in order to encapsulate a huge number of common functions and makes them easily accessible to the developer.
- It serves as the main point of interaction between developer and runtime.
- BCL includes the classes in namespaces like System, System.Diagnostics, System.Globalization, System.Resources, System.Text , System.Runtime.Serialization and System.Data etc. and provides the functionality like ADO.NET, XML, Threading, IO, Security, Diagnostics, Resources, Globalization, collections etc.
- The BCL contains all of the built-in types, arrays, exceptions, math libraries, basic File I/O, security, collections, reflection, networking, string manipulation, threading, and more.
- Any namespace that start with System is a part of the BCL.
- The Base Class Library (BCL) is the core set of classes that serve as the basic API of the Common Language Runtime
- The .Net base class library provides namespaces which are frequently used, some of them are as mentioned in figure below.



Framework Class Library / .net class library (FCL)

- Framework class library contains thousands of classes used to build different types of applications and provides all the basic functionalities and services that application needs. For e.g.

  - Desktop application,
  - Web application (ASP.Net, MVC, WCF),
  - Mobile application,
  - Xbox application,
  - Windows services etc.

Thousands of classes in FCL supports the following functions.

- ✓ Base and user-defined data types
- ✓ Support for exceptions handling
- ✓ input/output and stream operations
- ✓ Communications with the underlying system
- ✓ Access to data
- ✓ Ability to create Windows-based GUI applications
- ✓ Ability to create web-client and server applications
- ✓ Support for creating web services

- The Framework Class Library (FCL) is a superset of the BCL classes and refers to the entire class library that ships with .NET Framework.

- It also includes an expanded set of libraries, including Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, and Windows Communication Foundation among others.

Intermediate Language or MSIL/ Common Intermediate Language (CIL):

- MSIL stands for Microsoft Intermediate Language.
- Also called Intermediate Language (IL) or Common Intermediate Language (CIL).
- During the compile time, the compiler convert the source code into Microsoft Intermediate Language (MSIL).

- Microsoft Intermediate Language (MSIL) is a CPU-independent set of instructions that can be efficiently converted to the native code.
- During the runtime the Common Language Runtime (CLR)'s Just in Time (JIT) compiler converts the Microsoft Intermediate Language (MSIL) code into native code to the Operating System.
- When a compiler produces Microsoft Intermediate Language (MSIL), it also produces Metadata.
- The Microsoft Intermediate Language (MSIL) and Metadata are contained in a portable executable (PE) file.
- Microsoft Intermediate Language (MSIL) includes instructions for loading, storing, initializing, and calling methods on objects, as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling, and other operations.

Portable Executable (PE) File Format

- The Portable Executable (PE) format is a file format for executables, object code, and DLLs, used in 32-bit and 64-bit versions of Windows operating systems.
- The PE file format was defined to provide the best way for the Windows Operating System to execute code and also to store the essential data which is needed to run a program.
- Portable Executable File Format is derived from the Microsoft Common Object File Format (COFF).

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

- JIT provides support for multiple platforms.
- Compilation or object activation only at the time when it becomes necessary is JIT

How?

- Before the MSIL or CIL can be executed, JIT compiler converts the Microsoft Intermediate Language (MSIL) or Common Intermediate Language (CIL) into the machine code.
- The MSIL is converted into machine code (Portable Executable) 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

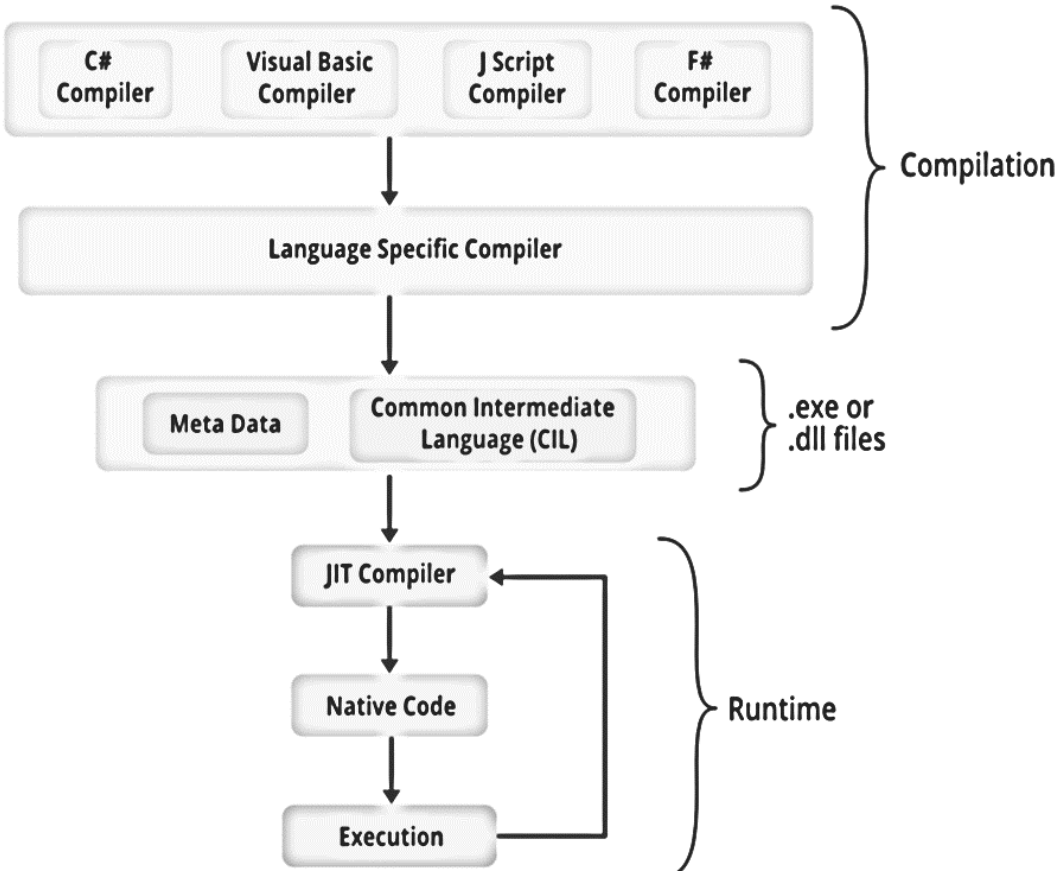


Figure: Working Mechanism of JIT Compiler

Types of Just-In-Time Compiler:

- **3** types

1. Pre-JIT Compiler

- It compiles all the source code into the machine code at the same time in a single compilation cycle.
- This compilation process is performed at application deployment time.

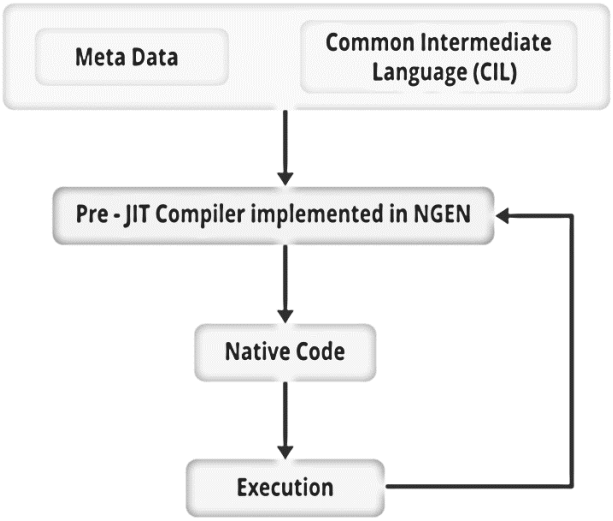


Figure: Working mechanism of Pre-JIT compiler

2. Normal JIT Compiler

- NJIT compiles the methods that are required or called during run-time are compiled into machine code at the instant of their first call.
- After that, compiled code are stored in the cache and used whenever they are called again.

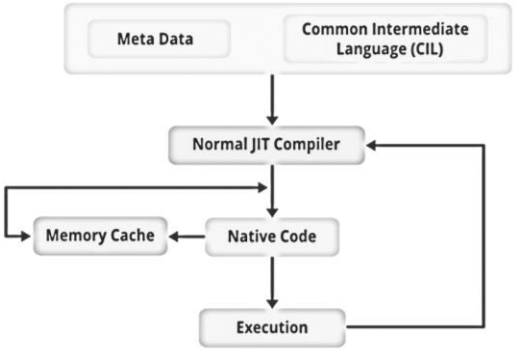


Figure: how normal JIT compiler Works?

3. Econo JIT Compiler

- ❖ It compiles the methods that are required at run-time into machine code.
- ❖ After these methods are not required anymore, they are removed.

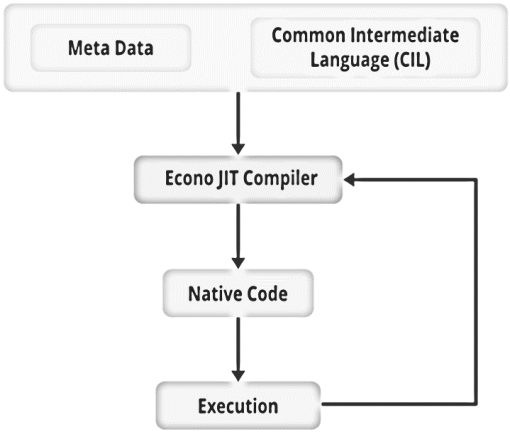


Figure: working of Econo JIT

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.

Note:

- ✓ Much of the disadvantages of the JIT compiler can be handled using the Ahead-of-time (AOT) compilation.

- ✓ This involves compiling the MSIL into machine code so that runtime compilation is not required and the machine code file can be executed natively.

Managed and unmanaged Code

- ❖ In Microsoft .Net Framework, Manage code is the code that has executed by the CLR environment.
- ❖ Benefits of Managed Code include programmer's convenience and enhanced security.
- ❖ Managed code is designed to be more reliable and robust than unmanaged code, examples are Garbage Collection, Type Safety etc.
- ❖ The Managed Code running in CLR cannot be accessed outside the runtime environment as well as cannot call directly from outside the runtime environment which makes the programs more isolated and at the same time computers are more secure.
- ❖ **Unmanaged Code** is directly executed by the computer's CPU.
- ❖ Unmanaged Code can bypass the .NET Framework and make direct calls to the Operating System. Calling unmanaged code presents a major security risk.
- ❖ Data types, error-handling mechanisms, creation and destruction rules, and design guidelines vary between managed and unmanaged object models.

Garbage Collection

- ❖ The .Net Framework provides a new mechanism for releasing unreferenced objects from the memory (that is we no longer needed that objects in the program), this process is called Garbage Collection (GC).
- ❖ Automatic memory management is made possible by **Garbage Collection in .NET Framework**.
- ❖ When a class object is created at runtime, certain memory space is allocated to it in the heap memory.
- ❖ However, after all the actions related to the object are completed in the program, the memory space allocated to it is a waste as it cannot be used. In this case, garbage collection is very useful as it automatically releases the memory space after it is no longer required.
- ❖ Garbage collection will always work on **Managed Heap** and internally it has an Engine which is known as the **Optimization Engine**.

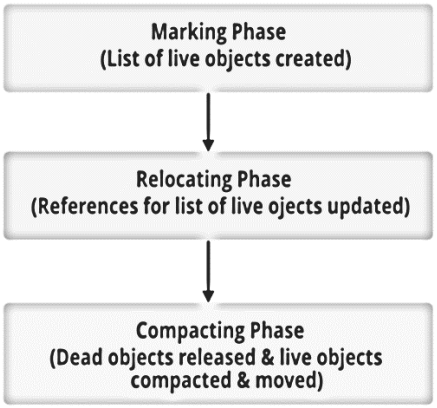
Garbage Collection occurs if at least one of multiple conditions is satisfied.

1. If the system has low physical memory, then garbage collection is necessary.

2. If the memory allocated to various objects in the heap memory exceeds a pre-set threshold, then garbage collection occurs.
3. If the *GC.Collect* method is called, then garbage collection occurs. However, this method is only called under unusual situations as normally garbage collector runs automatically.

Phases in Garbage Collection

❖ Mainly **3** phases in garbage collection.



1. **Marking Phase:**
   - ✓ A list of all the live objects is created and this is done by following the references from all the root objects.
   - ✓ All of the objects that are not on the list of live objects are potentially deleted from the heap memory.
2. **Relocating Phase:**
   - ✓ The references of all the objects that were on the list of all the live objects are updated in the relocating phase so that they point to the new location where the objects will be relocated to in the compacting phase.
3. **Compacting Phase:**
   - ✓ The heap gets compacted in the compacting phase as the space occupied by the dead objects is released and the live objects remaining are moved. All the live objects that remain after the garbage collection are moved towards the older end of the heap memory in their original order.

Benefits of Garbage Collection

❖ Garbage Collection succeeds in allocating objects efficiently on the heap memory using the generations of garbage collection.
❖ Manual freeing of memory is not needed as garbage collection automatically releases the memory space after it is no longer required.

- ❖ Garbage collection handles memory allocation safely so that no objects use the contents of another object mistakenly.
- ❖ The constructors of newly created objects do not have to initialize all the data fields, as garbage collection clears the memory of objects that were previously released.

Application Installation and Assembly

- ❖ C# Assembly is a standard library developed for .NET. Common Language Runtime, CLR, MSIL, Microsoft Intermediate Language, Just In Time Compilers, JIT, Framework Class Library, FCL, Common Language Specification, CLS, Common Type System, CTS, Garbage Collector, GC.
- ❖ Assemblies are the building blocks of .NET Framework applications.
- ❖ They form the fundamental unit of deployment, version control, reuse, activation scoping, and security permissions.
- ❖ An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality.
- ❖ An assembly provides the common language runtime (CLR) with the information it needs to be aware of type implementations.
- ❖ Assemblies can be static or dynamic.
- ❖ There are several ways to create assemblies. You can use development tools, such as Visual Studio, that you have used in the past to create .dll or .exe files

Functions of Assembly:

- ❖ It contains code that the common language runtime executes.
- ❖ It forms a security boundary i.e. An assembly is the unit at which permissions are requested and granted.
- ❖ It forms a type boundary i.e. every type's identity includes the name of the assembly in which it resides.
- ❖ It forms a reference scope boundary.
- ❖ It forms a version boundary.
- ❖ It forms a deployment unit.
- ❖ It is the unit at which side-by-side execution is supported**.**

.NET supports three kind of assemblies:

1. **Private** : requires to copy separately in all application folders
2. **Shared:** not required to copy separately into all application folders and also called public assembly.

3. **Satellite:** used for deploying language and culture-specific resources for an application.

Windows Communication Foundations:

- WCF = **Windows Communication Foundation**
- WCF is a part of the .NET framework
- WCF 4.5 is the most recent version that is now widely used.
- **WCF** deals with communication (in simple terms - sending and receiving data as well as formatting and serialization involved).
- **WCF** is Microsoft's unified programming model for building service-oriented applications. It enables developers to **build secure, reliable, transacted solutions that integrate across platforms** and interoperate with existing investments.
- Windows Communication Foundation (**WCF**) is a framework for creating service oriented applications.
- Windows communication Foundation (**WCF**) is used for connecting different applications and passing the data's between them using endpoints.
- **WCF** is your Backend app (services that involve server connections to acquire data for you to deliver to the Frontend to present)
-

A WCF application consists of three components:

- ✓ WCF service,
- ✓ WCF service host, and
- ✓ WCF service client.

- WCF platform is also known as the Service Model.

Advantages of WCF

- ✓ It is interoperable with respect to other services. This is in sharp contrast to .NET Remoting in which both the client and the service must have .Net.
- ✓ WCF services offer enhanced reliability as well as security in comparison to ASMX (Active Server Methods) web services.

- ✓ Implementing the security model and binding change in WCF do not require a major change in coding. Just a few configuration changes is required to meet the constraints.
- ✓ WCF has built-in logging mechanism whereas in other technologies, it is essential to do the requisite coding.
- ✓ WCF has integrated AJAX and support for JSON (JavaScript object notation).
- ✓ It offers scalability and support for up-coming web service standards.
- ✓ It has a default security mechanism which is extremely robust.

Features of WCF

WCF includes the following set of features

- ❖ Service Orientation
- ❖ Interoperability
- ❖ Multiple Message Patterns
- ❖ Service Metadata
- ❖ Data Contracts
- ❖ Security
- ❖ Multiple Transports and Encodings
- ❖ Reliable and Queued Messages
- ❖ Durable Messages
- ❖ Transactions
- ❖ AJAX and REST Support
- ❖ Extensibility

Windows Presentation Foundation:

- ❖ WPF = **Windows Presentation Foundation**
- ❖ It is a part of the .NET framework.
- ❖ **WPF**, provides a unified framework for building applications and high-fidelity experiences in Windows Vista that blend application UI, documents, and media content.

- **WPF** offers developers 2D and 3D graphics support, hardware-accelerated effects, scalability to different form factors, interactive data visualization, and superior content readability.

- Windows Presentation Foundation (**WPF**) is basically a way of displaying user interface.
- Windows Presentation Foundation is used for designing rich internet applications in the format of xaml.
- With **WPF**, developers can use XAML, the Extensible Application Markup Language, to create custom controls, graphics, 3D images and animations that are not available in traditional HTML implementations.
- **WPF** is your Frontend (presentation: .htm, .xaml & .css)
- **WPF** deals with presentation (UI).
- WPF is more than just a wrapper.
- It contains a mixture of managed and unmanaged code.

Advantages of WPF

- In the earlier GUI frameworks, there was no real separation between how an application looks like and how it behaved. Both GUI and behavior was created in the same language, e.g. C# or VB.Net which would require more effort from the developer to implement both UI and behavior associated with it.

- In WPF, UI elements are designed in XAML while behaviors can be implemented in procedural languages such C# and VB.Net. So it very easy to separate behavior from the designer code.

- With XAML, the programmers can work in parallel with the designers. The separation between a GUI and its behavior can allow us to easily change the look of a control by using styles and templates.

Features of WPF

WPF is a powerful framework to create Windows application. It supports many great features:

Feature	Description

Control inside a Control	Allows to define a control inside another control as a content.
Data binding	Mechanism to display and interact with data between UI elements and data object on user interface.
Media services	Provides an integrated system for building user interfaces with common media elements like images, audio, and video.
Templates	In WPF you can define the look of an element directly with a Template
Animations	Building interactivity and movement on user Interface
Alternative input	Supports multi-touch input on Windows 7 and above.
Direct3D	Allows to display more complex graphics and custom themes

End of chapter

……***……