Chapter-One

INTRODUCTION TO C# AND .NET FRAMEWORK

Introduction to C# Language

- C# is a high level general purpose, Object Oriented programming language which was developed by Anders Hijlsberg.
- It can be used to develop desktop application, web Application and Mobile Application.
- C# is a hybrid of C and C++ and it is a Microsoft programming language developed to compute with Sun's Java Programming language.
- C# is an object oriented language used with XML based web services on the .Net platform and designed for improving productivity in the development of web application.
- C# is an elegant and type-safe object oriented programming language that enables developers to build a variety of secure and robust applications that run on the .Net framework.
- It is also used to developed windows client applications, XML web services, distributed components, client-server applications, database applications and much more.
- All variables and methods including the Main method, the application's entry point are encapsulated within class definition.

Why use C#?

- It is modern general purpose programming language.
- It is object oriented.
- It is easy to learn.
- It is a structured language.
- It produces efficient programs
- It can be compiled on a variety of computer platforms.
- It is part of .Net Framework

Features of C#

1. **Object Orientation**
   - C# is a rich implementation of the object orientation praradigm, which includes encapsulation, abstraction, inheritance and polymorphism.

- ☞ The distinctive features of C# from an object oriented perspective are:
  - ❖ **Unified type system**
  - ❖ **Classes and interfaces**
  - ❖ **Properties, method and events**
- ☞ While C# is primarily an object oriented language, it also borrows from the functional programming paradigm. Specifically:
  - ❖ Function can be treated as value
  - ❖ C# supports patterns for purity

2. Type Safety

☞ C# is type safe object oriented programming language unlike C language. It means that instances of types can interact only through protocols they define, thereby ensuring each type's internal consistency. For example,

int x="21"; in C language no error
In C# language it is an error.

3. Memory Management

☞ C# relies on the runtime to perform automatic memory management. The Common language Runtime has a garbage collector that executes as part of your program, reclaiming memory for objects that are no longer referenced. This frees programmers from explicitly de-allocating the memory for an objects, eliminating the problem of incorrect pointers encountered in language such as C++.

4. Platform Support

☞ C# was used almost entirely for writing code to run on windows platforms. Recently, Microsoft and other companies have invested in other platforms, including Linux, mac OS, and Android.

☞ It allows cross platform C# development for mobile application and portable Class libraries are becoming increasingly widespread.

☞ Microsoft's ASP.NET Core is a cross-platform lightweight web hosting framework that can run either on the .NET framework or on .NET core an open source cross-platform runtime.

.NET Framework.

.Net is software component developed by Microsoft which runs under operating system. .Net framework is platform which is released in the year 2002. Later on several improvement has been taken place for the stronger and efficient platform. The .Net framework is a revolutionary platform that helps you to write the following types of applications:

- Windows applications
- Web applications
- Web services

The .Net framework applications are multi-platform applications. The framework has been designed in such a way that it can be used from any of the following languages: C#, C++, Visual Basic, Jscript, COBOL, etc. All these languages can access the framework as well as communicate with each other. .net is called .net framework because it support multiple language, technology, and database. In dot net, we have nearly 60+ languages, only 9 languages were developed by Microsoft, other languages were developed by third party. In each we have multiple technologies such as ASP.NET, ADO.NET etc and MS-Access, SQL server for database.

Net Architecture:

NET is tiered, modular, and hierarchal. Each tier of the .NET Framework is a layer of abstraction. .NET languages are the top tier and the most abstracted level. The common language runtime is the bottom tier, the least abstracted, and closest to the native environment. This is important since the common language runtime works closely with the operating environment to manage .NET applications. The .NET Framework is partitioned into modules, each with its own distinct responsibility. Finally, since higher tiers request services only from the lower tiers, .NET is hierarchal. The architectural layout of the .NET Framework is illustrated in Figure below.

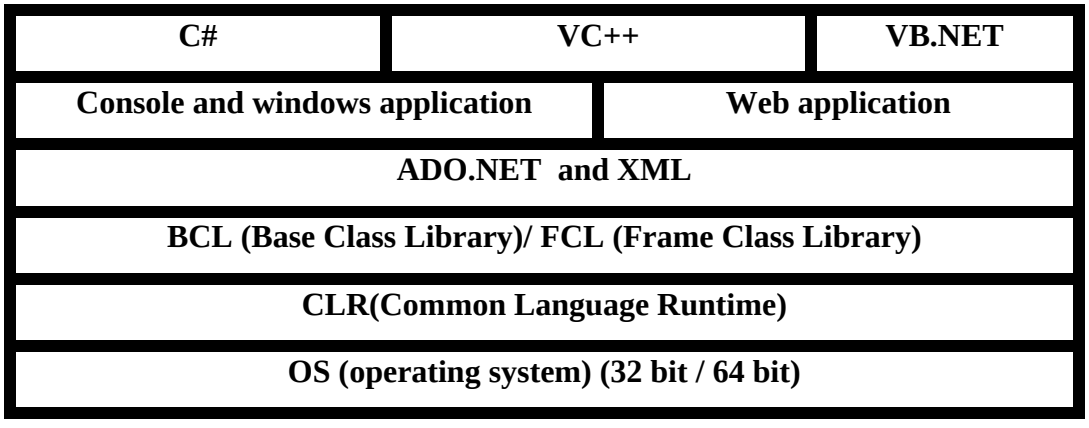


Figure 1.1 : .Net Architecture

NET Framework is a managed environment. The common language runtime monitors the execution of .NET applications and provides essential services. It manages memory, handles exceptions, ensures that applications are well-behaved, and much more.

Language interoperability is one goal of .NET. .NET languages share a common runtime (the common language runtime, a common class library), the Framework Class Library (FCL), a common component model, and common types. In .NET, the programming language is a lifestyle choice. Except for subtle differences, C#, VB.NET, or JScript.NET offer a similar experience.

.NET abstracts lower-level services, while retaining most of their flexibility. This is important to C-based programmers, who shudder at the limitations presented in Visual Basic 6 and earlier.

Component of .net framework

The main two components of .NET Framework are:

- Common Language Runtime (CLR)
- Base Class Library (BCL)/ Framework Class Library(FCL)

Common Language Runtime (CLR)

The CLR is the execution engine for .NET applications and serves as the interface between .NET application and the operating system. It provides an environment to run all the .Net Programs. The code which runs under the CLR is called as **Managed Code**. Programmers need not to worry on managing the memory if the programs are running under the CLR as it provides memory management and thread management. The CLR provides many services such as:

- Load and execute code.
- Manages memory and object
- Convert intermediate language into native code.
- Exception handling.

Common Language Specification

Common Language Specification (CLS) is a set of basic language features that .Net Languages needed to develop Applications and Services, which are compatible with the .Net Framework. When there is a situation to communicate Objects written in different .Net Complaint languages, those objects must expose the features that are common to all the languages. Common Language Specification (CLS) ensures complete interoperability among applications, regardless of the language used to create the application.

Common language specification provides syntactical rule to programming language to develop and create application. .Net has several languages such as C#, VB.net, VC++ etc. Here each and every language will follow its own syntax. One language cannot understand the other language

syntax. For example in C# statement are always end with semicolon (;) terminator but in VB.net it should not end with semicolon (;) . But CLR can understand all the language syntax because it follows its own CLS. CLR will have its own language specification.

So that whatever might be language each and every language will be converted into the intermediate language by with respect to complier. When program written in C# language, CSC compiler covert the C# code into intermediate language which is common to all language and that intermediate language will be understand by CLR (because CLR will have its own specification.)

Common Type System

Common Type System (CTS) describes a set of types that can use different .Net languages have in common, which ensure that objects written in different languages can interact with each other For Communicating between programs written in any .NET complaint language, the types have to be compatible on the basic level. These types can be Value Types or Reference Types . The Value Types are passed by values and stored in the stack. The Reference Types are passed by references and stored in the heap. Common Type System (CTS) provides base set of Types which is responsible for cross language integration. The Common Language Runtime (CLR) can load and execute the source code written in any .Net language, only if the type is described in the Common Type System (CTS)

It deals with data types, here .NET have several language , each and every language have its own data type . One language data type cannot be understandable by the other language data type. But CLR can understand all the language data types because CLR has its own data type. For example in C# integer variable represented using keyword "int" but in VB.NET it is represented by keyword "Integer". so that compiler of both language convert it into "INT32" which is recognized by CLR to represent integer variable. Hence CTS provide data type rules and regulation to the languages.

.Net Framework Class Library (FCL)

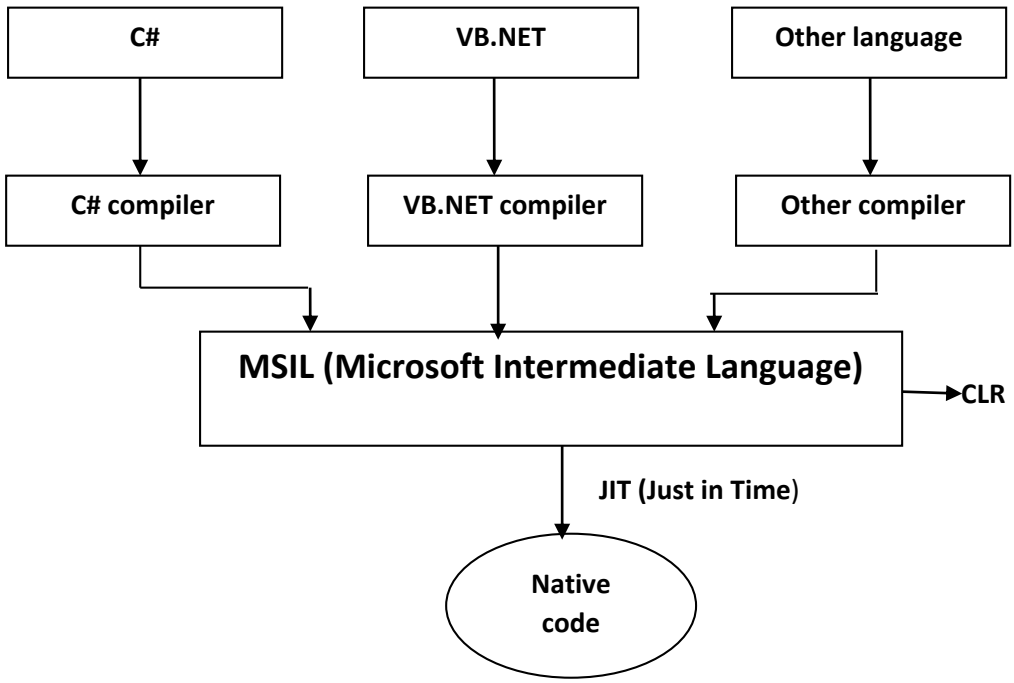
The .Net Framework class library (FCL) provides the core functionality of .Net Framework architecture. The .Net Framework Class Library (FCL) includes a huge collection of reusable classes, interfaces, and value types that expedite and optimize the development process and provide access to system functionality.

The .Net Framework class library (FCL) organized in a hierarchical tree structure and it is divided into Namespaces and assembly which is also called user defined class library. Namespaces is a logical grouping of types for the purpose of identification. Framework class library (FCL) provides the consistent base types that are used across all .NET enabled languages. The Classes are accessed by namespaces, which reside within Assemblies. User defined library is called assembly where

user is or programmer is going to create class and method. The assembly consist of .dll or .exe file. The .dll (dynamic linking library) is used reusability purpose and cannot open because it cannot contain the main function. The .exe(executable file) is used for output purpose and can be easily opened because it contains main method.

The System Namespace is the root for types in the .NET Framework. The .Net Framework class library (FCL) classes are managed classes that provide access to System Services. The .Net Framework class library (FCL) classes are object oriented and easy to use in program developments. Moreover, third-party components can integrate with the classes in the .NET Framework.

.Net Execution model



The execution of .net model consist of two phase.

- First program written in .net supporting language convert into Microsoft intermediate language (MSIL) using respective compiler.
- Second MSIL code can be converted through CLR into native code using JIT compiler.

MSIL (Microsoft Intermediate Language)

It is language independent code. When you compile code that uses the .NET Framework library, you don't immediately create operating system - specific native code. Instead, you compile your code into Microsoft Intermediate Language (MSIL) code. The MSIL code is not specific to any operating system or to any language.

JIT (Just-in-Time)

Just - in - Time (JIT) compiler, this compiles MSIL into native code that is specific to the OS and machine architecture being targeted. Only at this point can the OS execute the application. The just - in - time part of the name reflects the fact that MSIL code is only compiled as, and when, it is needed.

JIT are of three types:

1. Pre JIT
2. Econo JIT
3. Normal JIT

Pre JIT

It converts all the code in executable code and it is slow

Econo JIT

It will convert the called executable code only. But it will convert code every time when a code is called again.

Normal JIT

It will only convert the called code and will store in cache so that it will not require converting code again. Normal JIT is fast.

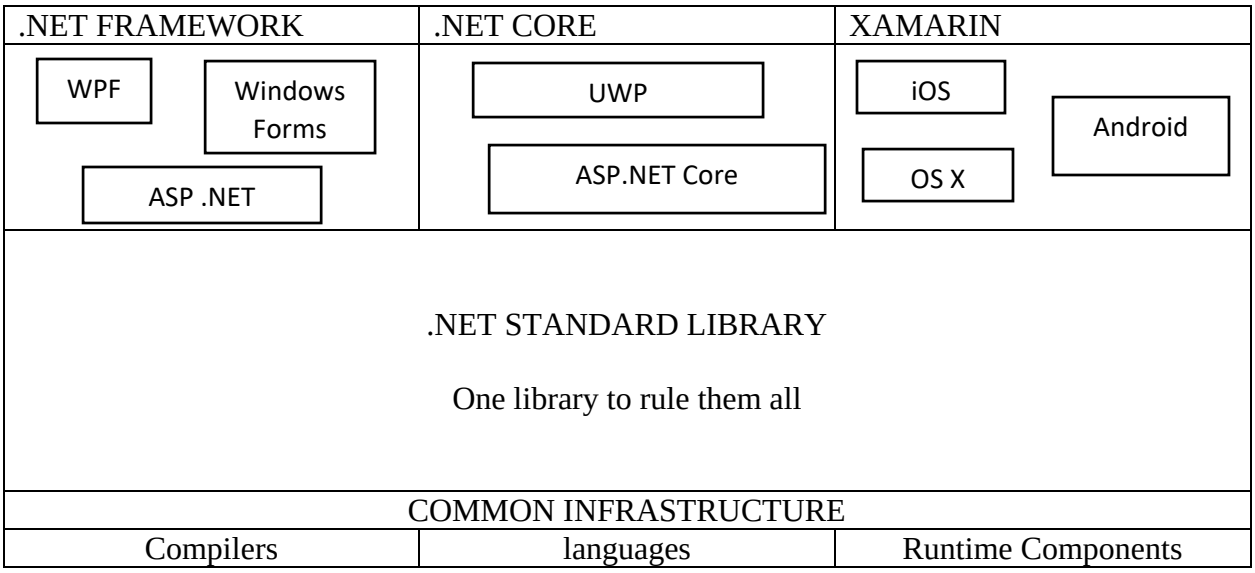
.Net Standard 2.0

.Net standard is a specification which dictates what the Base Class libraries of different .Net platform should implement to unify the Base Class libraries of different .Net Platforms. Here platform means full .Net Framework, .Net Core, Xamarin, Silverlight, Xbox etc. This also enables code sharing between applications that runs on these different platforms. For example, a library or a component that is developed on top of a platform that implements specific .Net standards version can be shared by all the applications that runs on any of the platforms that implements same .Net Standard version.

.Net standard is not a Framework, it's merely a specification describing a minimum baseline of functionality (types and members), Which guarantees compatibility with a certain set of

Framework. The concept is similar to C# interfaces: .Net standard is like an interface that concrete types (Framework) can implement.

.Net standard has solved all this in a different way, it provided an API specification which all platforms should implement to remain .Net Standard complaint. This has unified the base class libraries of different .Net platforms and has paved way to share libraries and also brought the BCL evolution centralized as seen below.



.NET Standard 2.0 is a new version that significantly increases the number of API's compared to the previous version (1.6.1). In fact, the API surface has more than doubled with .NET Standard 2.0

The .NET Standard 2.0 is supported by the following .NET implementations:

- .NET Core 2.0 or latter
- .NET Framework 4.6.1 or latter
- Mono 5.4 or later
- Xamarin.iOS 10.14 or later
- Xamarin.Mac 3.8 or later
- Xamarin.Android 8.0 or later
- Universal Windows Platform 10.0.16299 or later

What's new in the .NET Standard 2.0

- The .NET Standard 2.0 includes the following new features

    - A vastly expanded set of APIs
    - Support for .NET Framework libraries
    - Support for Visual Basic

- Tooling support for .NET Standard libraries.