The .NET Framework

# Definition

.NET Framework is a software framework developed by Microsoft that runs primarily on top of Operating Systems (OS) like Windows, Linux, Mac os etc. Programmers, instead of writing code directly for the OS using programming languages (like C / C++), will now write code for the .NET framework using PL's like C#.NET or VB.NET.

The software framework consists of

* A set of compilers (C# compiler, VB.NET compiler, VC++.NET compiler, J# compiler etc.),
* The Just In Time (JIT) compiler,
* The Common Language Runtime (CLR),
* The MSIL (Microsoft Intermediate Language) formats.
* Framework service components like Garbage Collector (GC), Process Optimizer etc.

Writing code for the .NET Framework using languages like C# and getting the output as MSIL, and then further compiling it to the native code (called as 2-STEP compilation) has ONE BIG Advantage - Platform Independence.

# 2. The 2-STEP Compilation Process

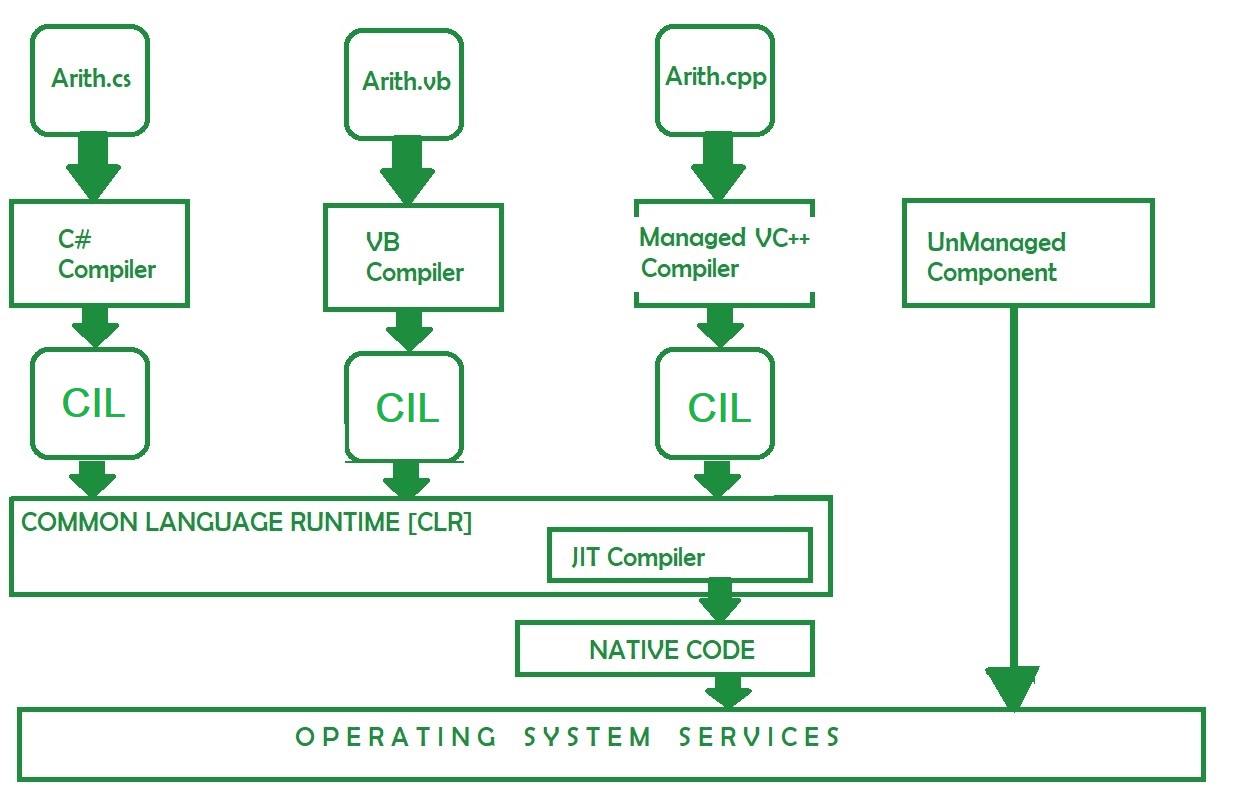


Fig.1 The .NET 2-Step compilation process.

# 3. Platform Independence

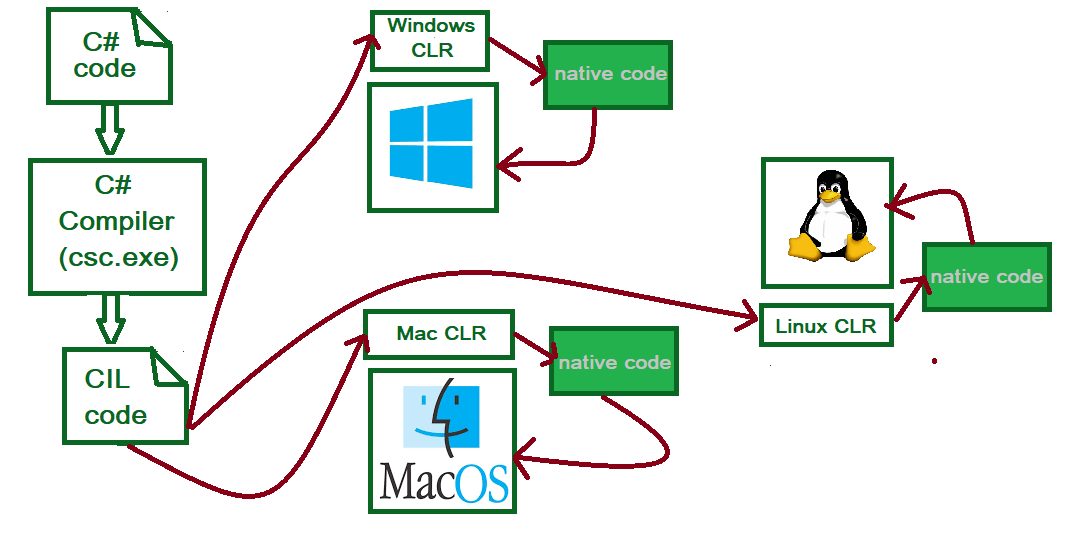


Fig.2 The Platform Independence

# 4. Components of the .NET Framework

## 4.1 C# Compiler

A compiler following the C# syntax and semantics which will convert C# source (\*.cs) files into any one of these - Executables (exe), Libraries (dll) or Netmodules (netmodule).

For more details - Source file Compiling options

## 4.2 Common Language Runtime (CLR)

The Common Language Runtime (CLR), the [virtual machine](https://en.wikipedia.org/wiki/Virtual_machine) component of [Microsoft](https://en.wikipedia.org/wiki/Microsoft) [.NET framework](https://en.wikipedia.org/wiki/.NET_framework), manages the execution of .NET programs. [Just-in-time compilation](https://en.wikipedia.org/wiki/Just-in-time_compilation) converts the [Managed code](https://en.wikipedia.org/wiki/Managed_code) (compiled MSIL code), into [machine instructions](https://en.wikipedia.org/wiki/Machine_instructions) which are then executed on the [CPU](https://en.wikipedia.org/wiki/CPU) of the computer.

The CLR provides additional services including [memory management](https://en.wikipedia.org/wiki/Memory_management), [type safety](https://en.wikipedia.org/wiki/Type_safety), [exception handling](https://en.wikipedia.org/wiki/Exception_handling), [garbage collection](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)), security and [thread management](https://en.wikipedia.org/wiki/Thread_management). All programs written for the .NET framework, regardless of [programming language](https://en.wikipedia.org/wiki/Programming_language), are executed by the CLR. All versions of the .NET framework include CLR.

CLR implements the [Virtual Execution System](https://en.wikipedia.org/wiki/Virtual_Execution_System) (VES) as defined in the [Common Language Infrastructure](https://en.wikipedia.org/wiki/Common_Language_Infrastructure) (CLI) standard, initially developed by Microsoft itself. A public standard defines the Common Language Infrastructure specification.

With Microsoft's move to [.NET Core](https://en.wikipedia.org/wiki/.NET_Core), the CLI VES implementation is known as [**CoreCLR**](https://en.wikipedia.org/wiki/CoreCLR) instead of CLR.

**Note**: CLR is equivalent to Java Virtual Machine (JVM).

(Source: Wikipedia)

## 4.3 JIT Compiler

Just In Time Compilers (JIT compilers) converts the MSIL into the Native code using the CLR for that operating system.

## 4.4 CIL (Common Intermediate Language)

Common Intermediate Language (CIL), formerly called Microsoft Intermediate Language (MSIL) or Intermediate Language (IL), is the [intermediate language](https://en.wikipedia.org/wiki/Intermediate_language) binary instruction set defined within the [Common Language Infrastructure](https://en.wikipedia.org/wiki/Common_Language_Infrastructure) (CLI) specification. CIL instructions are executed by a CLI-compatible runtime environment such as the CLR. Languages which target the CLI compile to CIL. CIL is [object-oriented](https://en.wikipedia.org/wiki/Object-oriented), [stack-based](https://en.wikipedia.org/wiki/Stack_machine) [bytecode](https://en.wikipedia.org/wiki/Bytecode). Runtimes typically [just-in-time](https://en.wikipedia.org/wiki/Just-in-time_compilation) compile CIL instructions into [native code](https://en.wikipedia.org/wiki/Native_code).

(Source: Wikipedia)

# 5. Features of the .NET Framework.

## 5.1 Platform Independence and Reusability

A code component written for .NET framework and compiled as an Assembly can be executed in any OS (Windows, Linux, Mac-OS etc). The programmer need not re-write the whole logic again in a different language just because the OS is different. This way the component so developed is a re-usable component.

## 5.2 Language interoperability

Code written in one .NET language and compiled into a package / library can be used in source code written in another .NET language. This is language interoperability and this is possible because of Common Type System (CTS).

## 5.3 Memory Management using GC

An actively running .NET application (running as a process in the underlying OS), allocates memory for newly created objects in the managed heap area. However, unlike other languages (say C++), the memory de-allocation is done using what is known as the Garbage Collection process, by the Garbage Collector (GC). The GC runs in the background of the .NET process as a 'thread'.

## 5.4 Framework Class Library (FCL) Support

FCL is a set of classes that provides the User interface, Data access, Database connectivity, Cryptography, Web application development, Numeric algorithms, and Network communications. Programmers produce software by combining their source code with .NET Framework and other libraries. Fig.3 shows a part of a huge FCL with their classes depicted.

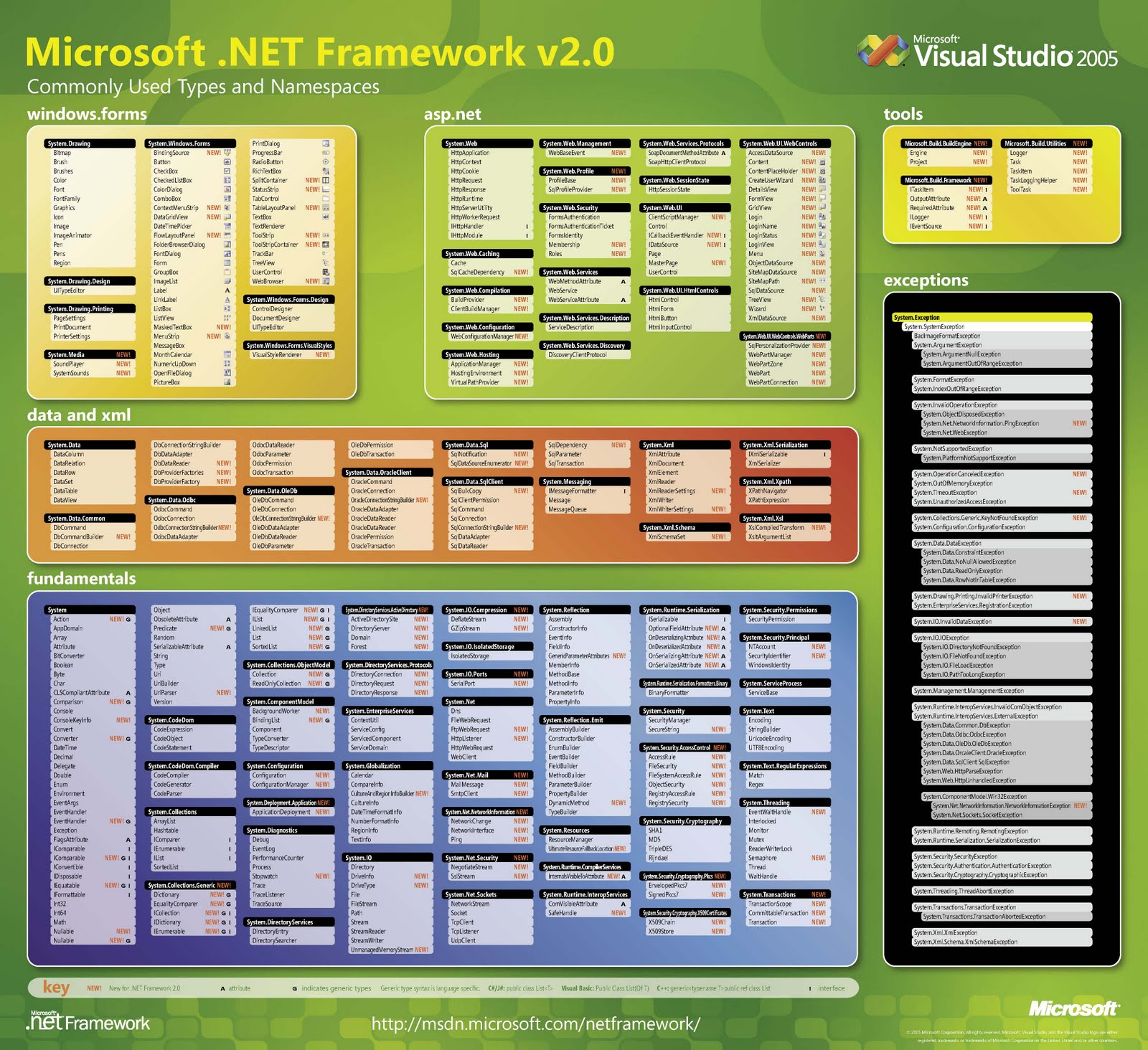


Fig.3. Part of the Framework Class Library (FCL).

## 5.5 Common Type System (CTS)

The Common Type System defines how types are declared, used, and managed in the common language runtime, and is also an important part of the runtime's support for cross-language integration. The common type system performs the following functions:

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

## 5.6 Managed Code

Computer code written to target (run on top of) the .NET Framework is called "Managed code". Any code that doesn’t run top of the CLR will not be able to leverage the advantages offered by the CLR and hence will be called as Un-Managed code. They have to be dealt separately outside the ambit of the CLR by methods like COM Callable Wrapper (CCW) and COM-Interop.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*