***.Net Framework***

The .NET Framework is a technology that supports building and running the next generation of apps and XML Web services. The .NET Framework is designed to fulfill the following objectives:

* To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
* To provide a code-execution environment that minimizes software deployment and versioning conflicts.
* To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
* To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
* To make the developer experience consistent across widely varying types of apps, such as Windows-based apps and Web-based apps.
* To build all communication on industry standards to ensure that code based on the .NET Framework integrates with any other code.

The .NET Framework consists of the common language runtime (CLR) and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. Think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that doesn't target the runtime is known as unmanaged code. The class library is a comprehensive, object-oriented collection of reusable types that you use to develop apps ranging from traditional command-line or graphical user interface (GUI) apps to apps based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable ASP.NET apps and XML Web services, both of which are discussed later in this topic.

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code derives functionality. This not only makes the .NET Framework types easy to use but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components integrate seamlessly with classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces for developing your own collection classes. Your collection classes blend seamlessly with the classes in the .NET Framework.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. Use the .NET Framework to develop the following types of apps and services:

* Console apps.
* Windows GUI apps (Windows Forms)
* Windows Presentation Foundation (WPF) apps.
* ASP.NET apps.
* Windows services.
* Service-oriented apps using Windows Communication Foundation (WCF).
* Workflow-enabled apps using Windows Workflow Foundation (WF).

The Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form app, you can use the Web Forms classes.

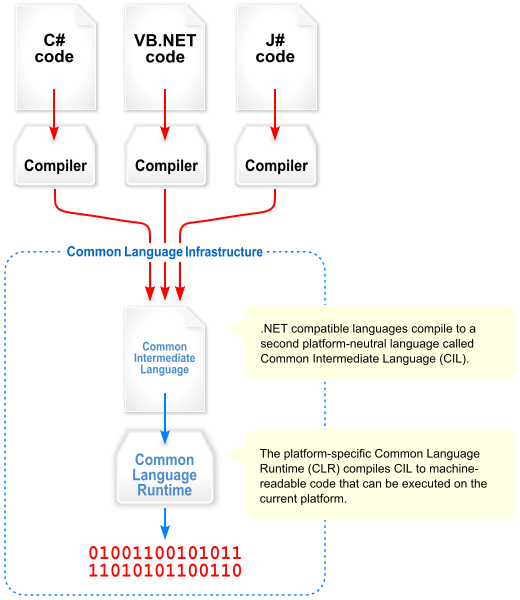
The .NET Framework is evolving to accommodate different platforms such as Windows Phone and Windows Store apps as well as traditional desktop and web apps, and to maximize code reuse. In addition to our regular .NET Framework releases, we release new features out of band (OOB) to improve cross-platform development or to introduce new functionality.

OOB releases for core common language runtime (CLR) components are delivered through the [NuGet Package Manager](http://nuget.codeplex.com/), which is an open source Visual Studio extension. NuGet enables you to browse and add libraries to your .NET Framework projects easily from the Solution Explorer in Visual Studio.

.NET Framework (pronounced dot net) is a [software framework](https://en.wikipedia.org/wiki/Software_framework) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) that runs primarily on [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows). It includes a large [class library](https://en.wikipedia.org/wiki/Class_library) named [Framework Class Library](https://en.wikipedia.org/wiki/Framework_Class_Library) (FCL) and provides [language interoperability](https://en.wikipedia.org/wiki/Language_interoperability) (each language can use code written in other languages) across several [programming languages](https://en.wikipedia.org/wiki/Programming_language). Programs written for .NET Framework execute in a [software](https://en.wikipedia.org/wiki/Software) environment (in contrast to a [hardware](https://en.wikipedia.org/wiki/Computer_hardware) environment) named [Common Language Runtime](https://en.wikipedia.org/wiki/Common_Language_Runtime)(CLR), an [application virtual machine](https://en.wikipedia.org/wiki/Process_virtual_machine) that provides services such as security, [memory management](https://en.wikipedia.org/wiki/Memory_management), and [exception handling](https://en.wikipedia.org/wiki/Exception_handling). (As such, computer code written using .NET Framework is called "[managed code](https://en.wikipedia.org/wiki/Managed_code)".) FCL and CLR together constitute .NET Framework.

FCL provides [user interface](https://en.wikipedia.org/wiki/User_interface), [data access](https://en.wikipedia.org/wiki/Data_access), [database connectivity](https://en.wikipedia.org/wiki/Database_connection), [cryptography](https://en.wikipedia.org/wiki/Cryptography), [web application](https://en.wikipedia.org/wiki/Web_application) development, numeric [algorithms](https://en.wikipedia.org/wiki/Algorithm), and [network communications](https://en.wikipedia.org/wiki/Computer_networking). Programmers produce software by combining their [source code](https://en.wikipedia.org/wiki/Source_code) with .NET Framework and other libraries. The framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) largely for .NET software called [Visual Studio](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio).

.NET Framework led to a family of .NET platforms targeting [mobile computing](https://en.wikipedia.org/wiki/Mobile_computing), [embedded devices](https://en.wikipedia.org/wiki/Embedded_device), alternative [operating systems](https://en.wikipedia.org/wiki/Operating_system), and [web browser plug-ins](https://en.wikipedia.org/wiki/Browser_extension)



The Mono runtime implements the [ECMA Common Language Infrastructure](http://www.ecma-international.org/publications/standards/Ecma-335.htm) (CLI). The Mono runtime implements this virtual machine.  
If you are interested in the technical aspects of the Mono runtime check the [Runtime Documentation](http://www.mono-project.com/docs/advanced/runtime/docs/).

The runtime offers the following services:

* Code Execution
  + Code loading
  + Support for dynamically generating code
  + [On-the-fly marshalling to invoke native methods.](http://www.mono-project.com/docs/advanced/pinvoke/)
  + [COM Interoperability](http://www.mono-project.com/docs/advanced/com-interop/)
* Garbage Collection, using one of:
  + [Precise SGen Garbage Collector](http://www.mono-project.com/docs/advanced/garbage-collector/sgen/)
  + Conservative Boehm Garbage Collector
* Code Generation
  + Just-in-Time compilation, partial and full [Ahead-of-Time modes](http://www.mono-project.com/docs/advanced/aot/)
  + Backend engines:
    - Mono’s own engine
    - [LLVM optimizing compiler backend engine](http://www.mono-project.com/docs/advanced/mono-llvm/)
  + [First-class SIMD datatypes (Mono.Simd)](http://go-mono.com/docs/index.aspx?link=N%3aMono.Simd)
* Exception Handling
  + Software-triggered exceptions
  + Hardware-triggered exceptions like Floating point exceptions, null reference exceptions
* Operating system interface
  + File system IO
  + Networking IO
  + Access to operating system properties and features
  + On Unix systems, [Mono.Posix APIs](http://go-mono.com/docs/index.aspx?link=N%3aMono.Posix)
* Program isolation using Application Domains (AppDomain)
* Thread management:
  + Threadpool for user code
  + Threadpool for networked IO
  + Asynchronous method invocation
* Console access
* Security System
  + [CoreCLR/Transparent Security Sandbox](http://www.mono-project.com/docs/advanced/coreclr-howto/)

The Mono runtime can be used as a stand-alone process, or it can be [embedded into applications](http://www.mono-project.com/docs/advanced/embedding/)

Embedding the Mono runtime allows applications to be extended in C# while reusing all of the existing C and C++ code. For more details, see the [Embedding Mono](http://www.mono-project.com/docs/advanced/embedding/) page and the [Scripting With Mono](http://www.mono-project.com/docs/advanced/embedding/scripting/) page.

**Supported Platforms**

Mono has support for both 32 and 64 bit systems on a number of architectures as well as a number of operating systems.

**Operating Systems**

* [Android](https://developer.xamarin.com/guides/android/)
* [Apple iOS](https://developer.xamarin.com/guides/ios), [iOS](http://www.mono-project.com/docs/about-mono/supported-platforms/iphone/)
* [Apple macOS](http://www.mono-project.com/docs/about-mono/supported-platforms/osx/),
* [Apple tvOS](https://developer.xamarin.com/guides/ios/tvos/)
* [Apple watchOS](https://developer.xamarin.com/guides/ios/watch/)
* [BSD](http://www.mono-project.com/docs/about-mono/supported-platforms/bsd/) - OpenBSD, FreeBSD, NetBSD
* [Linux](http://www.mono-project.com/docs/about-mono/supported-platforms/linux/)
* [Microsoft Windows](http://www.mono-project.com/docs/getting-started/install/windows/)
* [Nintendo Wii](http://www.mono-project.com/docs/about-mono/supported-platforms/wii/)
* [Sony PlayStation 3](http://www.mono-project.com/docs/about-mono/supported-platforms/playstation3/)
* [Sony PlayStation 4](http://www.mono-project.com/docs/about-mono/supported-platforms/playstation4/)
* [Sun Solaris](http://www.mono-project.com/docs/about-mono/supported-platforms/solaris/)

In [computing](https://en.wikipedia.org/wiki/Computing), just-in-time (JIT) compilation, also known as dynamic translation, is [compilation](https://en.wikipedia.org/wiki/Compiler) done during execution of a program – at [run time](https://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase)) – rather than prior to execution. Most often this consists of translation to [machine code](https://en.wikipedia.org/wiki/Machine_code), which is then executed directly, but can also refer to translation to another format. A system implementing a JIT compiler typically continuously analyses the code being executed and identifies parts of the code where the speedup gained from compilation would outweigh the overhead of compiling that code.

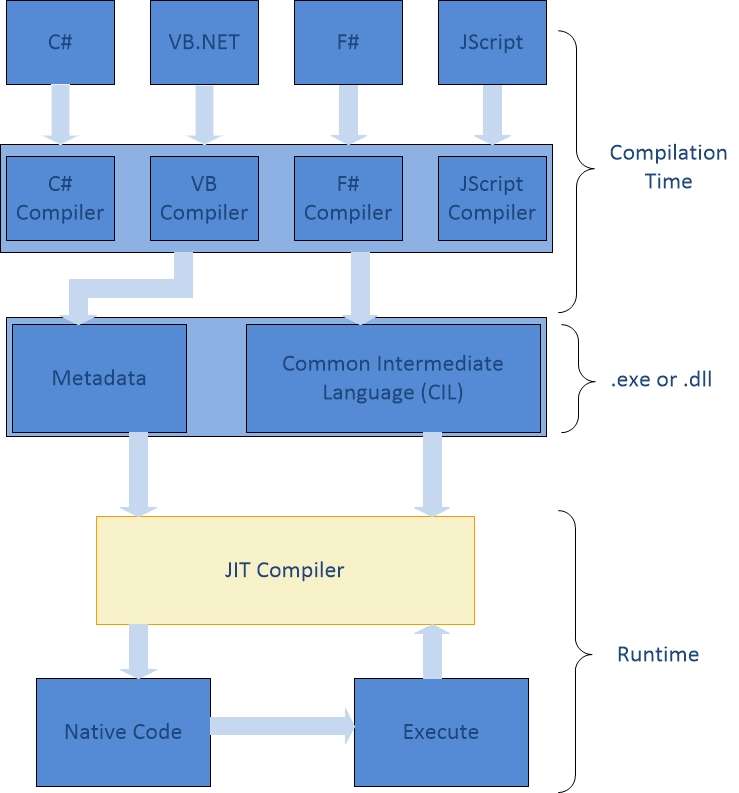
JIT compilation can be applied to some programs, or can be used for certain capacities, particularly dynamic capacities such as [regular expressions](https://en.wikipedia.org/wiki/Regular_expression). For example, a text editor may compile a regular expression provided at runtime to machine code to allow faster matching – this cannot be done ahead of time, as the pattern is only provided at runtime

**Explicit Compilation**

Explicit compilation converts the upper level language into object code prior to program execution. Ahead of time (AOT) compilers are designed to ensure that, the CPU can understand every line in the code before any interaction takes place.

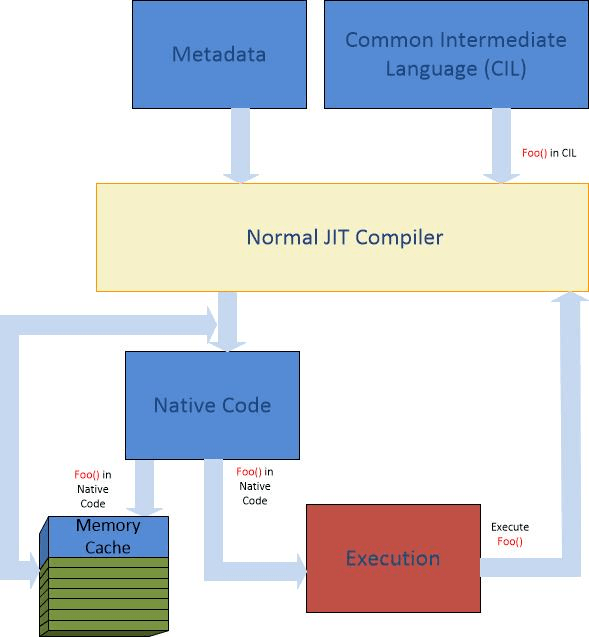
**Implicit Compilation**

Implicit compilation is a two-step process.  The first step is converting the source code to intermediate language (IL) by a language-specific compiler. The second step is converting the IL to machine instructions. The main difference with the explicit compilers is that only executed fragments of IL code are compiled into machine instructions, at runtime. The .NET framework calls this compiler the JIT (Just-In-Time) compiler.



**Normal JIT Compilation**

With the *Normal JIT Compiler (*figure 2*)*methods are compiled when called at runtime. After execution this method is stored in the memory and it is commonly referred as “jitted”. No further compilation is required for the same method. Subsequent method calls are accessible directly from the memory cache.

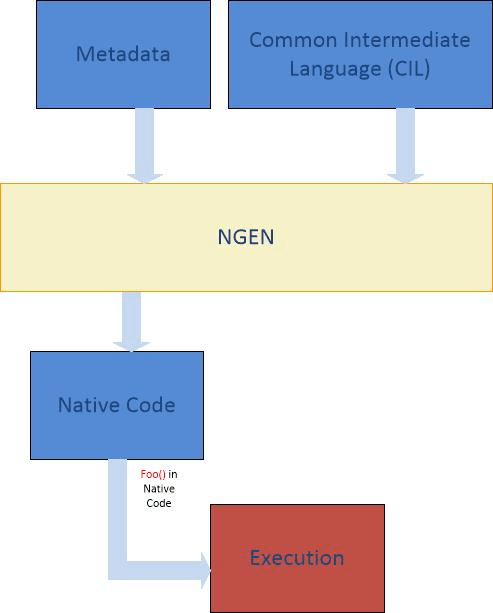


**Econo JIT Compilation**

The *Econo JIT Compiler* is displayed in figure 3*.*It compiles methods when called at runtime and removes them from memory after execution.

**Pre-JIT Compilation**

Another form of compilation in .NET is called *Pre-JIT compilation*. It compiles the entire assembly instead of used methods. In .NET languages, this is implemented in *Ngen.exe (Native Image Generator)*. All CIL instructions are compiled to native code before startup, as shown in figure 4. This way the runtime can use native images from the cache instead of invoking the JIT Compiler.



**What is CLR**

The Common Language Runtime (CLR), the [virtual machine](https://en.wikipedia.org/wiki/Virtual_machine) component of [Microsoft's](https://en.wikipedia.org/wiki/Microsoft) [.NET framework](https://en.wikipedia.org/wiki/.NET_framework), manages the execution of .NET programs. A process known as [just-in-time compilation](https://en.wikipedia.org/wiki/Just-in-time_compilation) converts compiled code into machine instructions which the computer's [CPU](https://en.wikipedia.org/wiki/CPU) then executes.[[1]](https://en.wikipedia.org/wiki/Common_Language_Runtime#cite_note-msdn-clr-1) 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.