5/4/21 .NET Platform

Tuesday, May 4, 2021 7:31 AM

Components of .NET implementation:
One or more runtimes
A class library
Optionally, one or more application frameworks
Optionally, development tools

.NET compliant: C#, VB.NET, F#

Features of .NET:
- Language interoperability
  - All .NET compliant languages can work together in one solution

SDK VS Runtime

SDK: Includes everything you need to build and run .NET Core application, using command line tools and any editor (including Visual Studio)

Runtime: includes just the resources required to run existing .NET Core applications. The runtime is included in the SDK.

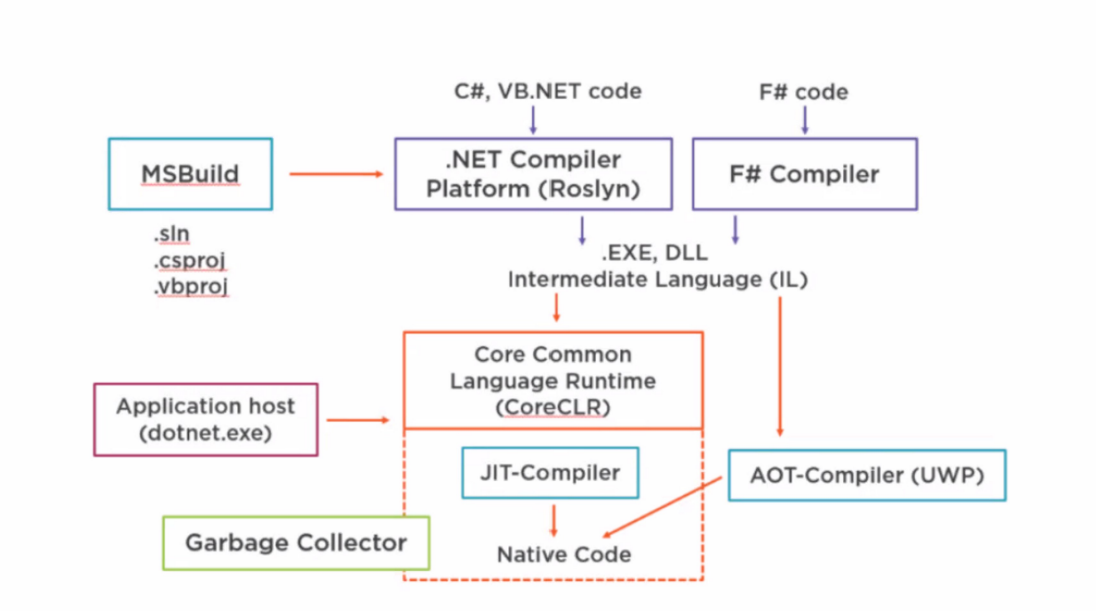
.NET Standard
A set of APIs that are implemented by the Base Class Library of a .NET implementation
The BCL provides classes and types that are helpful in performing day to day operations

CLR: Common language runtime
It's a runtime environment, that runs the code and provides services that make the development process easier. Some of these services include: Memory management, JIT complication, Exception handling support, automatic garbage collection
This runs just like the JVM

JIT compilation: Write once, run anywhere compiler, Just In Time compiler

Compilation process: C# file -- MSBUILD --> Intermediate Language -- CLR/JIT --> Machine Code (1's, 0's)
Jit compiler converts to the machine specific code.

JIT is not C# specific, Java has one too, which makes the language write once, use anywhere.



Managed code is whose execution is managed by a runtime.

EX. C#, any .NET compliant language

Unmanaged code is the code that is developed outside .NET. It does not leverage CLR features but can be executed by the help of wrapper classes.

EX. C

Garbage collection

CLR provides automatic memory management of your heap memory

Sometimes there are memory used that are outside the scope of CLR, such as files and DB connection. In such cases we have to clean them up ourselves

Types that utilize these unmanaged resources inherit from the IDisposable Interface

Garbage collection in CLR is done through generational garbage collection

When you create a new object, it's given gen 0. Gen 0 objects get checked more often for references than others, if it keeps getting referenced, then it is bumped up to the next generation.

Special cases: If it uses a lot of memory, then it gets put in to gen 2.. etc