

# .NET Core 3 CLR Programming Using C# 8

## Architecture, Assemblies, Type/instance, Attributes, Reflection, Security, Nuget, Unmanaged

The .NET Core 3 Common Language Runtime (CLR) is the foundation for all aspects of the modern .NET Core initiative. It offers leading edge services for applications on Windows, macOS, Linux and more. The .NET Core provides a “managed” runtime environment in the sense that an in-process runtime engine, separate from your code, to assist with its execution. Its modern feature-set includes a system-wide object model, full control of memory management, granular security, innovative metadata, sophisticated type loading, and virtualized contracts between types (so that their physical layouts may be defined / optimized at runtime).

Regardless of which .NET Core application type you are building – rich client UI, web UI, web service, class library, Windows Service, console - you must understand how it works with the CLR and how it can programmatically interact with the capabilities of the CLR. The .NET Core Framework works with the CLR to provide a rich [.NET Standard 2.1](#)-based class library.

.NET Core is becoming the platform of choice for new projects. It is highly regarded and is gaining a reputation as a solid foundation for innovative solutions.

<b>Contents of One-Day Training Course</b>	
<p><b>Target Audience</b> System architects and senior software engineers who need to rapidly get up to speed with .NET Core CLR programming.</p> <p><b>Prerequisites</b> General understanding of high level .NET Core concepts.</p> <p>Attendance at our <i>C# 8 Language</i> course or equivalent experience is needed.</p> <p>This course covers .NET Core 3 using C# 8 and Visual Studio 2019.</p>	<p><b>Overview</b> Multiple implementations of .NET .NET Standard is a specification that mandates how all should work: best=2.1 .NET Framework: stable, mature (20yrs), slowly-evolving/Windows only, in OS .NET Core: new, Windows/macOS/Linux, ships with your app (so faster updates) For modern codebases, choose .NET Core</p> <p><b>CLR Architecture</b> Feature-rich runtime for many languages What capabilities does it provide?</p> <p><b>Assemblies</b> An assembly is a (DLL-like) delivery &amp; management unit for CLR types Strong names / versioning Modules and assemblies Locating assemblies</p> <p><b>Types</b> Importance of type / CLS &amp; CTS Types and different languages System.Object &amp; identity</p> <p><b>Advanced Types</b> Methods – virtual, abstract, overloaded Type hierarchy / relationships / generics Properties, enumerations, constants, fields</p> <p><b>Instances</b> Types &amp; instances Reference type and value type Finalisation / IDisposable Garbage collection / stack and heap Calling methods/anonymous methods Callback techniques</p> <p><b>Compiler Options</b> Native compiler vs. JIT - contrast Using .NET Native .NET on macOS and Linux</p> <p><b>Nuget</b> Using .NET’s package manager Creating your own packages</p> <p><b>.NET Core 3 API</b> Extensive quality class library Solves problems with .NET Framework Portable; small; new features (e.g. <a href="#">SIMD</a> &amp; <a href="#">AssemblyDependencyResolver</a> )</p> <p><b>Config</b> Externalized Configuration Handling configuration data Format of config files &amp; custom settings Application and machine configuration</p> <p><b>Custom Attributes</b> Advantages of using attributes in IL Pseudo-custom attributes Creating &amp; detecting attributes</p> <p><b>Metadata &amp; Reflection</b> What data is stored along with code Manifest &amp; metadata tables Peeking inside an assembly Discovering types</p> <p><b>Security</b> Identity issues &amp; overview of .NET Core security architecture Certificates &amp; permissions Command-line and UI security tools</p> <p><b>The Unmanaged World</b> P/Invoke &amp; how to call C libraries</p>