.NET Core 2.2 CLR Programming

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

The .NET Core 2.2 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 and Linux. The .NET Core CLR 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*-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.

Target Audience

System architects and senior software engineers who need to rapidly get up to speed with .NET Core CLR programming.

Prerequisites

General understanding of high level .NET Core concepts.

Attendance at our *C# Language* course or equivalent experience is needed.

Contents of One-Day Training Course

Overview

Multiple implementations of .NET
.NET Standard is a specification that mandates how all should work
.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

CLR Architecture

Feature-rich runtime for many languages
What capabilities does it provide?

Assemblies

An assembly is a (DLL-like) delivery & management unit for CLR types
Strong names / versioning
Modules and assemblies
Locating assemblies

Types

Importance of type / CLS & CTS
Types and different languages
System.Object & identity

Advanced Types

Methods – virtual, abstract, overloaded
Type hierarchy / relationships / generics
Properties, enumerations, constants, fields

Instances

Types & instances
Reference type and value type
Finalisation / IDisposable
Garbage collection / stack and heap
Calling methods/anonymous methods
Callback techniques

Compiler Options

Native compiler vs. JIT - contrast
Using .NET Native
.NET on macOS and Linux

Nuget

Using .NET's package manager
Creating your own packages

.NET Core 2.2 Framework

Extensive quality class library
Solves problems with .NET Framework
Portable and small

Config

Externalised Configuration
Handling configuration data
Format of config files & custom settings
Application and machine configuration

Custom Attributes

Advantages of using attributes in IL
Pseudo-custom attributes
Creating & detecting attributes

Metadata & Reflection

What data is stored along with code
Manifest & metadata tables
Peaking inside an assembly
Discovering types

Security

Identity issues & overview of .NET Core security architecture
Certificates & permissions
Command-line and UI security tools

The Unmanaged World

P/Invoke
How to call C libraries