.NET Core 3 Multithreaded & Parallel Programming
Concepts, Kernel Objects, Threads, Synchronization, Tasks, TPL, PLINQ, Parallel Collections, TPL Dataflow

[Sample: presentation] This course examines how to use .NET Core 3 to build sophisticated architectures using multithreaded and parallel programming. When designed correctly, these can substantially increase application performance and responsiveness to distributed clients and end-users. The kernel object is the opaque foundation for Windows multithreading – based on this are .NET constructs for the process, thread & various synchronization objects - mutex, event, semaphore, waitable timer and more – each of which targets different needs. Thread activity, lifetimes and inter-thread communication must be co-ordinated.

A server must efficiently multiplex many I/O requests over a few threads – which is precisely the goal of threadpools in .NET Core 3. We see major benefits from the addition of parallel programming to .NET, especially with TPL, PLINQ, concurrent collections and TPL Dataflow. We explore the critical role of `Task` in this new approach.We see scope for custom enhancements to TPL in a variety of scenarios.

This course supplies attendees with a clear understanding of .NET multithreaded and parallel programming, together with experience of their use.

Target Audience

System architects and experienced developers who need to gain an in-depth understanding of .NET Core 3 multithreaded and parallel programming.

Prerequisites

Attendees must have experience of systems-level programming.

Attendance at our *.NET Core 3 CLR Programming Using C# 8* course or equivalent experience needed.

Contents of One-Day Training Course

Multithreading Concepts

Thread definition
Scheduling vs. synchronization
Parallelism and concurrency
Compute-bound and I/O bound apps
Race conditions, deadlock, starvation, priority inversion

OS Foundation (e.g. kernel objects)

Win32 kernel objects
Usage counting / Kernel object handles
Sharing handles among processes

Managed Threads

OS threads and managed threads
Creating a new managed thread
The ThreadStart delegate
Thread priority and processor affinity
Thread management & lifetime

Synchronization

Monitor (C# lock), Mutex, Event, Timer
Waiting (once, many), SpinWait
The “Protect data, not code” principle
The Interlocked class / Semaphore

Memory, Threads and SIMD

Thread data slots
VolatileRead/Write
SIMD - Single instruction, multiple data Intrinsics

Managed Thread Pool

The thread pool is a runtime-managed pools of threads for processing I/O, work-items and timer handlers
Architecture of large multithreaded app

Parallel Programming in .NET

Takes a higher level view of concurrency
Parallel querying
Parallel algorithms and supporting types

Task

What is in System.Threading.Task
Detailed look at Task class and how to use
Parallel / TaskFactory / TaskExtensions

PLINQ - Parallel Linq

Applying the ideas of Linq to Objects across multiple threads
Query operartors and threading
Additional capabilities of PLINQ

Concurrent Collections

Original .NET collections are not thread-safe, and why this can be good/bad
New concurrent collections offer similar API surface to original, but now threadsafe
Deep dive: System.Collection.Concurrent

Advanced Tasks & PLINQ

Schedulers
Lambda expressions
Custom operators
Partitioners

TPL Dataflow

Library of dataflow constructs
Message passing
Based around composition of blocks
Very useful for certain kinds of workloads

Multithreaded Project

A complete multithreaded embedded HTTP web server that uses a thread pool to efficiently manage very many requests