C# MCA - C# Multithreading e Advanced I/O: Programmazione Concorrente e Asincrona

ITCore Sviluppo - C# - Sviluppo

Durata:	Lingue:	Certificazione:
3 Giorni	**Italiano**	**-**

Descrizione del corso

Il corso **"C# Multithreading e Advanced I/O"** è pensato per sviluppatori che desiderano approfondire i meccanismi avanzati della programmazione concorrente, asincrona e del file system utilizzando il linguaggio **C#**. La gestione di file, stream e la programmazione parallela sono competenze fondamentali per la creazione di applicazioni performanti, reattive e robuste.

A Chi è Rivolto Questo Corso

Questo corso avanzatissimo è destinato a sviluppatori con una conoscenza approfondita di C# e della programmazione orientata agli oggetti, che abbiano già esperienza con le basi della programmazione concorrente e asincrona. È ideale per chi lavora su applicazioni ad alte prestazioni e per chi vuole estendere le proprie competenze in progetti di sistema, infrastruttura e gestione di risorse condivise.

Obiettivo del corso:

Al termine del percorso, i partecipanti saranno in grado di:

• Sviluppare applicazioni in **C#** capaci di gestire file, stream e dati complessi;
• Scrivere codice concorrente e multi-thread sicuro ed efficiente;
• Utilizzare tecniche avanzate per la sincronizzazione e il coordinamento dei thread;
• Comprendere la teoria delle **Reti di Petri** come base per la programmazione parallela.

Prerequisiti: Conoscenza di base di C# e programmazione orientata agli oggetti.

Un corso ideale per chi vuole padroneggiare la **programmazione asincrona** e **concorrente** con il linguaggio **C#**, ottimizzando l'efficienza e la scalabilità delle applicazioni moderne.

Programma

1. **File e Directory Management**: gestione avanzata del file system.
2. **Serializzazione e Deserializzazione**: conversione dei dati tra oggetti e formati testuali/binarî.
3. **I/O Stream**: gestione dei flussi di input e output per dati sincroni e asincroni.
4. **Programmazione asincrona** con **async**, **await** e **Task**, per eseguire operazioni non bloccanti.
5. **Task Parallel Library (TPL)**: strumenti per semplificare la programmazione parallela.
6. **Threading**: concetti fondamentali e avanzati per la gestione dei **Thread**.
7. **Sincronizzazione**: utilizzo di strumenti come:
    - **Lock, Interlocked**
    - **Monitor, Mutex e SpinLock**
    - **Semaphore** e **CountdownEvent**
    - **ReadWriteLockSlim**
    - **Barrier** ed **EventWaitHandle**.
8. **Collezioni concorrenti**: gestione sicura dei dati in ambienti multi-thread con:
    - **BlockingCollection**, **ConcurrentBag**, **ConcurrentDictionary**, **ConcurrentQueue** e **ConcurrentTask**.
9. **Reti di Petri**: concetti teorici per la modellazione e gestione dei processi concorrenti.
10. **Dynamic e Reflection**: manipolazione avanzata dei tipi e metadati a runtime.