

# Kurs-Dokumentation



**Zentrum für Informatik ZFI AG**

## **Efficient Concurrent Programming in .NET (NCPG-0511) -IT Ausbildung nach Mass**

<http://www.zfi.ch/NCPG-0511>

Weitere Infos finden Sie unter [www.zfi.ch](http://www.zfi.ch) oder via Adresse:

**Zentrum für Informatik ZFI AG  
Zentralsekretariat  
Technoparkstrasse 1  
CH-8005 Zürich  
Telefon: 044 732 40 00  
Telefax: 044 732 40 09**

**Zürich, Basel, Bern, Zürich, Schweiz**

<b>Titel</b>	<b>Efficient Concurrent Programming in .NET</b>
<b>Untertitel</b>	<b>Multicore- und Multiprozessor-Computer richtig ausnutzen</b>
<b>Einleitung</b>	<p>Mit der Verbreitung der Multicore- und Multiprozessor-Computer hat die Bedeutung von Parallelität und Nebenläufigkeit in heutigen Softwaresystemen stark zugenommen - nicht zuletzt, weil die sequentielle Rechenleistung in den letzten Jahren nicht mehr wesentlich erhöht werden konnte. Aus diesem Grund bieten moderne Systeme wie das .NET Framework ein umfangreiches Spektrum von Features für die nebenläufige Programmierung.</p> <p>Jedoch stellt die Nebenläufigkeit den Softwareentwickler vor neue und anspruchsvolle technische Probleme und Herausforderungen: Wann und wie lässt sich überhaupt mit dem Einsatz von Threads und Prozessen die Software beschleunigen? Wie lassen sich die Gefahren von Race Conditions und Deadlocks systematisch verhindern? Auf welche Starvation-Probleme sind bei Prioritäten und Synchronisationen zu achten? Was für Modelle und Design Patterns eignen sich für die nebenläufige Programmierung? Welche spezifischen Probleme gibt es in .NET? Wie lässt sich die .NET Task Parallel Library optimal einsetzen?</p> <p>In diesem zweitägigen Kurs wird der effiziente und sichere Einsatz von Nebenläufigkeit in der Praxis für das aktuelle .NET Framework 4.0 behandelt. Die Sprache C# wird dabei für Programmbeispiele eingesetzt. Die Teilnehmer haben ferner die Gelegenheit, ihr gewonnenes Wissen in praktischen Übungen anzuwenden und zu vertiefen.</p>
<b>Ihr Nutzen</b>	Der Kurs vermittelt einen detaillierten Überblick über die Konzepte der Nebenläufigkeit in .NET sowie deren sicheren und effizienten Anwendung in der Praxis. Der Kurs ist so gestaltet, dass er nicht nur .NET-Standardwissen vermittelt, sondern auch über den Horizont der aktuellen Literatur hinausgeht. So werden auch weitergehende Konzepte und besondere Risiken in .NET diskutiert.
<b>Voraussetzungen</b>	Erfahrung als Software-Entwickler von professioneller Software oder als System/Software-Architekt
<b>Teilnehmerkreis</b>	Der Kurs richtet sich vor allem an Software-Ingenieure und Architekten, die sich mit der Nebenläufigkeit in .NET vertieft auseinandersetzen möchten. Er kann auch für Software-Projektleiter und IT-Verantwortliche dienen, sich einen genaueren Überblick über die Möglichkeiten und Herausforderungen der Nebenläufigkeit in .NET zu verschaffen.
<b>Unterlagen</b>	Praxisorientiertes Studienmaterial
<b>Folgekurse</b>	
<b>Inhalt</b>	<ul style="list-style-type: none"> <li>- Introduction, system support for parallelism</li> <li>- Motivation: Why concurrent programming?</li> <li>- The free lunch is over</li> <li>- Levels of parallelism</li> <li>- Parallel computer architectures</li> </ul>

- **Parallelism and concurrency**
- **Processes versus threads**
- **Resources per process and thread**
- **Processes and thread scheduling**
- **Processor multiplexing**
- **Context switches**
  
- **Threads and Critical Sections in .NET**
- **Thread implementation, creation and start**
- **Thread parameter passing**
- **Thread join**
- **Thread termination**
- **Thread passivation**
- **Thread lifecycle**
  
- **Critical Sections and Mutual Exclusion in .NET**
- **Access to shared resources**
- **Critical sections**
- **Naive approaches: typical mistakes**
- **Synchronization with Mutex**
- **Thread-safe data structures**

- **Monitors in .NET**
- **Monitor concept**
- **Monitor support in .NET**
- **Monitor lock**
- **Monitor wait and pulse**
- **.NET monitor: typical pitfalls**
- **Correct .NET monitor usage**
- **Sleeping barber optimization**
- **.NET monitor limitations**
  
- **Specific synchronization primitives in .NET**
- **Barrier**
- **Rendez-vous**
- **Semaphores**
- **Reader-Writer Lock**
  
- **Dangers of Concurrency: Races, Deadlocks, Starvation**
- **Race condition errors**
- **Race condition detection**
- **Race condition prevention**
- **Deadlock errors**

- **Deadlock detection**
- **Deadlock prevention**
- **Architectural deadlock prevention**
- **Deadlocks with reader-writer locks**
- **Starvation error**
- **Starvation prevention**
- **Priority inversion problem**
- **Concurrency correctness criteria**
  
- **Thread pools and task parallelism with TPL in .NET**
- **Thread pool concept**
- **Thread pool benefits**
- **Thread pool limitations and deadlock risks**
- **.NET Task Parallel Library (TPL)**
- **Task implementation and start**
- **Task parameter passing**
- **Task start and wait**
- **Task with result values**
- **Multi task start and wait**
- **Nested tasks**
- **Tasks: miscellaneous**
- **Children tasks**
- **Thread pool implementation in TPL**
- **Task parallelism: typical pitfalls**

- Data parallelism with TPL in .NET
- Data parallelism with TPL: Overview
- Parallel statement execution
- Parallel loop execution
- Synchronization for data parallelism
- Aggregate parallel loop results
- Stop and break parallel loops
- Performance tuning for data parallelism
- Parallel loop partitioning
- Performance considerations with data parallelism
- Parallel language-integrated querying (PLINQ)
  
- Asynchronous programming in .NET
- Asynchronous method calls
- Event-based asynchronous programming
- Asynchronous execution with TPL
- Continuation-style programming with TPL
- Task status monitoring
  
- Implementing efficient monitors in .NET

- Standard .NET monitor: shortcomings
- Enhanced monitor concept
- Eggshell model
- Monitor signal models
- Enhanced monitor implementation
- Monitor versions: comparison
  
- Specific .NET concurrency aspects
- Race conditions with finalizers
- Threading with GUI
- Multi-processor memory model
- Atomic instructions (.NET Interlocked classes)
- Memory fences
- Spin locks
- Visual Studio Parallel Diagnostic Tools
  
- Summary and outlook
- «Lessons learned» summary
- Outlook to future concurrency
- Conclusion

**Beitrag**

Der Teilnehmerbeitrag versteht sich rein netto. Das ZFI ist (gemäss MwSt-Gesetz) nicht Mehrwertsteuerpflichtig und erhebt somit keine MwSt. Bei länger als einen Monat dauernden Lehrgängen ist die Zahlung des Teilnehmerbeitrages in mehreren Raten möglich (pro rata temporis).

# Bildungsweg Visual Studio .NET

