Parallel Programming with .NET 4.5

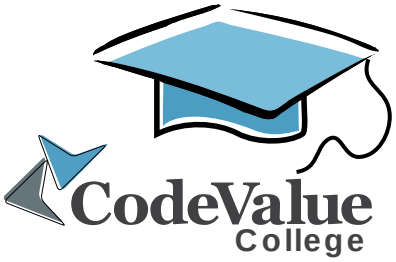
Course Summary Table

Duration:	32 hours instructor-led course
Target Audience:	.NET developers
Objectives:	Understand the concepts of parallel programming Gain hands-on experience in parallel programming with .NET Use .NET's synchronization constructs, TPL, PLINQ and concurrent collections
Pre Requisites:	.NET developers with at least 1 year of experience C# language proficiency – an advantage (all code samples are given in C#) Familiarity with multi-threading and synchronization concepts is an advantage

Abstract

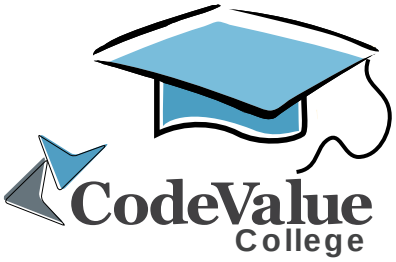
At the beginning of 2005 Herb Sutter had an article stating that the developer's free lunch is over. We had an assumption that more transistors in the CPU imply better application execution speed - the CPU executes the code in a sequential manner hence the performance of the CPU-bound code is directly related to CPU frequency. This used to be our "Free Lunch": an old program runs faster on a new CPU. The only problem is that using this assumption with modern low power consumption multi-core CPUs is wrong, and we might even find that an old program runs slower on a newer CPU! Since performance is no longer tied to CPU frequency we need to leverage parallelism – our new free lunch.

In this 4-day course we will see the abstractions, libraries and tools that Microsoft provides for .NET developers. We will start with understanding the concepts involved in parallel programming such as threads and locks. We will then see how these concepts are applied in the .NET Framework and continue on to advanced abstractions and techniques that the framework provides. These abstractions include the Task Parallel Library (TPL), Parallel LINQ, Concurrent Collections and more.



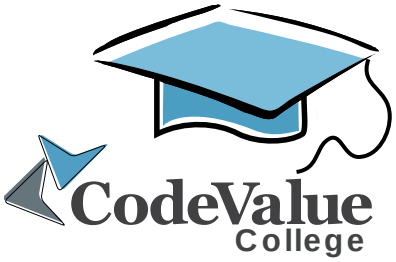
Syllabus

- Module 1: Introduction to Parallel Programming
  - What is parallel programming?
  - The need for parallel programing
  - Moore's Law
  - Threads & Processes
  - Scale-Up vs. Scale-Out
  - Amdahl's Law
  - Software & hardware obstacles
- Module 2: Basic .NET Threading and Synchronization
  - System.Thread
  - The *lock* keyword
  - Monitor
  - Synchronization constructs (Mutex, Semaphore, Wait Handles etc.)
  - Interlocked methods
  - The .NET ThreadPool
  - The Asynchronous Programming Model (APM)
- Module 3: Concurrent Collections
  - The need for concurrent collections
  - Concurrent collection usage
  - *System.Collections.Concurrent* namespace
- Module 4: TPL Basics
  - Task Parallel Library overview
  - What is a task?
  - Creating and using tasks
  - Waiting for task completion
  - Task coordination
  - Exception Handling
- Module 5: Parallel Control Structures
  - Parallel.For
  - Parallel.ForEach
  - Parallel.Invoke
  - Cancelling a loop
  - Partitioning
- Module 6: Parallel LINQ
  - Introduction to LINQ
  - Parallelizing LINQ
- Module 7: Advanced TPL
  - Task cancellation
  - Parent/Child tasks
  - Task schedulers
  - The *async* & *await* keywords



- Asynchronous methods and WCF
- Asynchronous methods and the Windows 8 API

- Module 8: TPL Data Flow Networks
  - The Actor Model
  - Action Blocks
  - Building a data flow network
- Module 9: Diagnostics & Profiling
  - Parallel Tasks window
  - Parallel Threads window
  - Concurrency Profiler



Course Compatibility Questionnaire

Please answer the following questions as accurately as possible:

Name: _____ Email: _____

Company: _____ Phone: _____

Language / Technology / Platform	Years of Experience						Level of Familiarity				
.NET	0-1	1-2	2-3	3-4	4-5	5+	1	2	3	4	5
C#	0-1	1-2	2-3	3-4	4-5	5+	1	2	3	4	5
Multi-Threading	0-1	1-2	2-3	3-4	4-5	5+	1	2	3	4	5
Windows Internals	0-1	1-2	2-3	3-4	4-5	5+	1	2	3	4	5
LINQ	0-1	1-2	2-3	3-4	4-5	5+	1	2	3	4	5
Other _____	0-1	1-2	2-3	3-4	4-5	5+	1	2	3	4	5

Can you describe what a thread context-switch is?

_____

_____

_____

What is a deadlock?

_____

_____

_____

What are your expectations from the course?

_____

_____

_____

Thanks!

http://college.codevalue.net/