.NET Performance and Debugging

Course 4137 – 32 Hours

Overview

Performance oriented development allows you to stay on the edge of today's competition. It is crucial to write applications that can utilize stronger machines or scale out to distributed architectures. At the same time, it can be crucial to archive good performance on client application running on weak hardware. In addition, debugging an application in a production environment can be tricky, as it requires a different set of tools than those found on development machines, and a different state of mind.
During this course, we will focus on many common performance bottlenecks and best practices to solve them, providing a rich toolkit for both analyzing the performance issues and designing your application to avoid them in the first place. In addition, we will learn about tools allowing us to understand and pinpoint the problem without influencing the running application, to set up production and development environments in a way that will make this task easier, and when everything else fails, perform post mortem debugging and application dump analysis on those crashes that are so hard to reproduce.

On Completion, Delegates will be able to

- Understand performance issues and best practices for developing with the .NET Framework
- Obtain hands on experience with using advanced debugging tools and profilers

Who Should Attend

- .NET developers, who want to enhance their skills in the areas of performance and debugging

Prerequisites

- At least 2 years of experience with the .NET Framework (version 2.0 and above)
- C# language proficiency – Advantage
- Native development experience – Advantage

Course Contents

Module 1: .NET Internals

- Assemblies and the Loader
- Value and Reference type's differences and performance implications
- CLR memory management: Garbage Collection, Finalizer, Generations
- Deterministic finalization
- The Large Object Heap
- String interning

Module 2: Performance Monitoring and Profiling

- Performance monitoring – possible strategies
- Performance Counters
- Creating custom Performance Counters
- The CLR profiler
- Visual Studio profiler

Module 3: Debugging

- Debug environment setup
- The debugging process
- .NET debuggers
- Controlling the debugger
  - Advanced breakpoints
    - Conditional
    - Hit count
    - Function
- The debugging loop
  - Make it fail
  - Make a hypothesis
  - Collect evidence
  - Prove or refine the hypothesis
- Advanced techniques and tools
  - Exceptions
    - First and second chance exceptions
    - Unhandled exceptions
    - Exception handling model
  - Debugging Symbols
    - Source & Symbol Servers
  - Production Debugging & Dump Files
    - Generating Dumps
    - Controlling Dumps with GFlags
  - Debugging with Visual Studio and OzCode
    - Modifying debugger display
    - Advanced debugging techniques with Visual Studio
      - IntelliTrace
      - Using IntelliTrace files
      - Configuring custom collection plan
      - Production IntelliTrace
    - What's new in Visual Studio 2015 C# debugger
    - Advanced debugging techniques with OzCode
      - Predict into the future
      - Search
      - Compare
      - LINQ Debugging
- The future of debugging

Module 4: Additional Debugging Toolkit

- Windows based tools
- System Internals toolkit
- Dependency Walker
- DebugDiag

Module 5: Debugging Tools for Windows

- WinDbg
- DebugView
- ADPlus
- Loading symbols, attaching to processes, viewing threads and processes
- Setting breakpoints, viewing modules
- Advanced breakpoints
- Loading the SOS, SOSEX and PSSCOR4 extensions
  - Setting breakpoint on managed (Jitted) functions
  - Inspecting CLR data structure
    - Types, Instances, threads, stacks, AppDomains
  - Smart analysis of .NET problems:
    - Detecting deadlocks
    - Detecting memory leaks
    - Handling Interop problems
- Viewing the managed heap, objects, threads, call stack and more