Faysal Aziz

Java Performance Management

Optimize, Debug, and Deliver High-Performance Java Applications in a Single Day!

FA-0271

Software Development

Introduction

Is your Java application struggling to keep up? Join us for an intensive workshop where you'll master the tools and techniques to diagnose and fix performance bottlenecks and memory leaks.

Our industry veteran instructor will guide you through hands-on exercises using real-world scenarios and modern profiling tools. By day's end, you'll be equipped to optimize your applications for speed, efficiency, and reliability - skills you can apply immediately to deliver better user experiences and reduce operational costs.

Learning outcomes

By the end of this training, participants will:

- Understand the key factors that affect Java application performance.
- Optimize Java code for runtime efficiency and reduced memory overhead.
- Gain a deep understanding of the Java Virtual Machine (JVM) and its internal workings.
- Identify and fix performance bottlenecks using profiling tools.
- Master memory management concepts, including garbage collection tuning.
- Conduct performance testing and benchmarking to validate optimizations.
- Explore advanced options like Just-In-Time (JIT) compilation and alternative JVMs like GraalVM.

Prerequisites

- Basic knowledge of Java programming.
- Familiarity with object-oriented programming concepts.
- Experience with Java development tools like Eclipse or IntelliJ IDEA.

Training outline

1. 1. Fundamentals of Java Performance Tuning

- Introduction to Java performance issues and their business impact.
- Understanding the trade-offs between performance, memory usage, and maintainability.
- Overview of the Java performance tuning workflow.

2. 2. Deep Dive into the Java Virtual Machine (JVM)

Anatomy of the JVM:

- JVM architecture and execution model.
- Bytecode and class loading.

Just-In-Time (JIT) Compilation:

- How JIT optimizes code during runtime.
- Configuring JIT settings for performance gains.

Overview of JVM alternatives (e.g., GraalVM, OpenJ9):

- Use cases and benefits for specific workloads.

3. 3. Memory Management in Java

How Java allocates and manages memory:

- Heap vs. stack memory.
- Object lifecycle and references.

Garbage Collection (GC):

- Types of garbage collectors (e.g., G1, ZGC, Shenandoah).
- Tuning GC for low-latency and high-throughput scenarios.

Troubleshooting memory leaks:

- Identifying memory leaks using tools like VisualVM and Eclipse MAT.
- Best practices to prevent memory leaks in Java applications.

4. 4. Analyzing and Debugging Performance Bottlenecks

Introduction to performance profiling tools:

- JProfiler, VisualVM, and YourKit.

Profiling methodology:

- Identifying hotspots in the code.
- CPU vs. memory profiling.

Debugging common issues:

- High CPU usage.
- Thread contention and deadlocks.
- I/O bottlenecks.

5. 5. Optimizing Code for Performance

Best practices for efficient Java code:

- Choosing the right data structures (e.g., ArrayList vs. LinkedList).
- Avoiding common performance anti-patterns.

Threading and concurrency optimization:

- Managing thread pools and executor services.
- Minimizing contention in multi-threaded applications.

Reducing I/O overhead:

- Optimizing database queries and network communication.

6. 6. Performance Testing and Benchmarking

- The importance of benchmarking in performance tuning.

  Designing effective performance tests:

  - Tools: JMH (Java Microbenchmark Harness) and Apache JMeter.
  - Simulating real-world workloads.
- Analyzing and interpreting benchmarking results.

7. 7. Advanced JVM Configurations and Tools

Tuning JVM runtime parameters:

- Key JVM flags for performance (e.g., -Xms, -Xmx, -XX:+UseG1GC).

Exploring advanced profiling techniques:

- Decompiling bytecode for debugging.

- Understanding method inlining and escape analysis.

This one-day course will be packed with actionable insights, hands-on exercises, and industry-best practices to help you develop Java applications that are fast, reliable, and resource-efficient.

© 2026 Faysal Aziz · Kuala Lumpur