

## **J2EE Effective Coding for Performance**

### **Course Summary**

#### **Description**

Creating High Performance Java Based Web/Server Based Software is a skill and an art. This course teaches practical Java Enterprise Edition (JEE) design and coding patterns for creating high performance server based software.

#### **Objectives**

At the end of this course, students will be able to:

- Understand the challenges and goals of achieving high performance
- Specify Service Level Agreements (SLAs)
- Know how to obtain Performance Measurements
- Know how to create simulated workloads
- Know general distributed programming performance techniques
- Know Web Server Tier Performance Issues and Patterns
- Be able to use Business Component Tier Performance Patterns
- Be able to tune the Data Tier
- Understand Performance issues in interfacing to other (Java and Non Java) Systems

#### **Topics**

- Specifying Service Level Agreements
- Obtaining Performance Measurements
- Create Simulated Workloads
- General Distributed Programming Performance Techniques
- Web Server Tier Performance Issues and Patterns
- Business Component Tier Performance Patterns
- The Data Tier
- Performance Issues in interfacing to other (Java and Non Java) Systems

#### **Audience**

This course will benefit developers (programmers, team leads and architects) who design and program using Java and J2EE.

#### **Prerequisites**

Students should have at least 1 year of experience with programming in Java and J2EE; including JSPs and Databases. Students should also know the basics of SQL and JDBC.

#### **Duration**

Five days

## J2EE Effective Coding for Performance

### Course Outline

#### I. Overview

- A. The Performance Problem
- B. How Fast is Fast Enough?
- C. Hiding Bad Performance
- D. The 6 main Performance costs: Memory, CPU Time, Network Traffic, I/O,
- E. System Calls, Resource Blocking
- F. Trading performance for maintainability
- G. Trading performance for security
- H. Trading performance for data integrity - Synchronization and Transactions

#### II. Specifying Service Level Agreements

- A. The Four Types of Performance Measures
- B. End-User Response Time
- C. Request Throughout
- D. Resource Utilization
- E. Application Availability
- F. Specifying a Service Level Agreement
- G. Average loads
- H. Peak loads
- I. Degradation of performance for super high loads

#### III. Obtaining Performance Measurements

- A. Using the Server Console
- B. Using Connection Pools that provide Statistics
- C. JMX Consoles
- D. OS Tools
- E. Creating Performance Prototypes
- F. Simulated Workload Generators
- G. Operating System tools
- H. Simultaneous Users vs. Individual User Response Time
- I. Rendering Time
- J. Estimating Unknown Times
- K. Back-end Throughput

#### IV. Create Simulated Workloads

- A. Workload Generation Tools
- B. Using JMeter
- C. Creating Graduated Loads
- D. Test Load Strategies
- E. Other workload generators: Grinder and OpenSTA

#### V. General Distributed Programming Performance Techniques

- A. Balancing Work Between the Tiers
- B. Migrating Code Between Tiers
- C. Stateless Objects
- D. Immutable Objects
- E. Content to Overhead Ratios of Distributed Calls
- F. Sending References instead of the Data Itself
- G. Service Activators and Locators
- H. Balancing Data Validation with Performance
- I. Balancing Security with Performance
- J. Balancing Transactions with Performance
- K. Balancing Resource Locking with Performance
- L. Across the wire data formats. Serialization and XML
- M. Perceived vs. Real Response Time

#### VI. Web Server Tier Performance Issues and Patterns

- A. Conserving Memory - Zillions of Classes - for JSPs
- B. Images and GIFS and long downloads
- C. Caching Strategies - Using Headers
- D. Moving Work to the Client, Java Script
- E. Creating the Illusion of Performance
- F. Session Objects and Memory Consumption

#### VII. Business Component Tier Performance Patterns

- A. Session Facade
- B. EJBs vs. Regular Java Beans
- C. Comparing the cost of Remote Interfaces, Local Interfaces and Regular Function
- D. Calls
- E. Direct JDBC or Entity Beans?
- F. Can you perform better than Container Managed Persistence
- G. Should code activate and create grab resources?
- H. Who keeps track of the current session?
- I. Stateless Beans vs. Stateful Beans
- J. Transforming Stateful Session Beans to Stateless Session Beans
- K. Infinite Loops in Message Queue Handling Errors

## **J2EE Effective Coding for Performance**

### **Course Outline (cont'd)**

#### **VIII. The Data Tier**

- A. JDBC Drivers - Prepared Statements are stored on a "per connection" basis
- B. Using JDBC Connection Pools
- C. Moving Work to the Database
- D. Controlling the Size of Result Sets
- E. Granularity of Data
- F. Prepared Statements
- G. Using Connection Pools
- H. Freshness / Staleness of Data
- I. Data Access Objects
- J. Value Object
- K. Using Read Only or Read Mostly Data
- L. Composite Entity
- M. Value List Assembler
- N. Dealing with Large Data Sets
- O. Closing Transactions Quickly

#### **IX. Performance issues in interfacing to other (Java and Non Java) Systems**

- A. HTTP
- B. JNI
- C. CORBA
- D. Messaging
- E. Java Serialization
- F. XML APIs - SAX vs. DOM vs. STaX and Text vs. Binary
- G. Messaging vs. Synchronous Calls
- H. Using Interpreted Languages