Faysal Aziz

Spring Boot for Enterprise Applications

High-Performance, Distributed Systems Using Spring Boot - 2 days

FA-0419 · 2 days

Software Development · DevOps, Cloud & Infrastructure

Introduction

This **2-day advanced Spring Boot** course is designed for seasoned **Java developers** and those already familiar with Spring Boot foundations. As the industry moves toward microservices architectures, caching strategies, and robust message-driven systems, mastering these technologies has become essential for delivering scalable and high-performance enterprise applications.

In this hands-on course, we delve deep into **microservices**, **Spring Cloud**, **caching with Spring Cache**, and **integration with messaging systems like JMS**. Leveraging remote Linux environments via **VS Code's SSH plugin**, we simulate real-world cloud environments, ensuring that participants walk away with practical knowledge ready for immediate implementation. The focus is on giving participants **industry-relevant tools and techniques** that go beyond the basics and dive into the challenges and solutions of modern distributed systems.

Instructor Profile: Your instructor has over **30 years of industry experience**, having built and architected complex, distributed systems for global enterprises. The course is purely focused on practical, real-world scenarios—no academic theories, just **battle-tested industry best practices**.

Learning outcomes

By the end of this course, participants will be able to:

- Understand and implement a **microservices architecture** using **Spring Boot** and **Spring Cloud**.
- Efficiently manage and optimize **distributed caching** using **Spring Cache** and various caching providers.
- Integrate Spring Boot applications with **message-driven systems** such as **JMS** and **RabbitMQ**.

- Secure microservices with **OAuth 2.0** and **JWT tokens** using **Spring Security**.
- Manage **inter-service communication** using **Spring Cloud** components like **Feign Clients** and **Circuit Breakers**.
- Implement **asynchronous messaging** for **event-driven microservices** architecture.

Prerequisites (mandatory):

- Advanced knowledge of **Java** and **object-oriented programming** principles.
- Solid understanding of **Spring Boot**, including REST APIs, Dependency Injection, and Spring Data JPA.
- Experience with the basics of **distributed systems** and web architectures.
- Experience with **remote Linux environments** for development and deployment.

Training outline

1. Introduction to Microservices Architecture

What are microservices?

1. Differences from monolithic architectures
2. Benefits and challenges of adopting microservices

2. Key components of microservices architecture: **API Gateway**, **Service Discovery**, **Load Balancing**

Designing microservices with **Spring Boot**

1. Decomposing a monolithic application into microservices
2. Best practices for structuring microservices for maintainability and scalability

2. Spring Cloud for Microservices Development

Overview of **Spring Cloud** and its components

1. **Spring Cloud Netflix** (Eureka, Ribbon, Feign, Hystrix)
2. **Spring Cloud Config** for centralized configuration management

Setting up **service discovery** with **Eureka Server**

1. Registering services with Eureka
2. Configuring clients to communicate via Eureka

Load balancing with **Spring Cloud Ribbon**

1. Configuring Ribbon for client-side load balancing

Implementing **RESTful inter-service communication** using **Feign Clients**

1. Simplifying HTTP API calls between microservices

3. Circuit Breakers and Resilience with Spring Cloud

Understanding the need for **circuit breakers** in microservices

1. Handling service failures gracefully

Implementing **Hystrix** for fault tolerance

1. Setting up circuit breakers to monitor and control service interactions
2. Configuring fallbacks for failed services

3. Integrating **Spring Cloud Resilience4j** as a modern alternative to Hystrix

4. Securing Microservices with Spring Security and OAuth 2.0

Introduction to **OAuth 2.0** and its role in securing microservices

1. Understanding the OAuth flow (Authorization Code, Implicit, etc.)

Implementing **OAuth 2.0** authentication and authorization in Spring Boot

1. Securing microservices with **OAuth** and **JWT tokens**
2. Managing user roles and scopes across distributed services

Implementing **API Gateway security** for microservices

1. Protecting endpoints using **Spring Cloud Gateway**

5. Introduction to Spring Cache for Performance Optimization

Why caching is critical in microservices and enterprise systems

1. Understanding **distributed caching** and cache invalidation strategies

Setting up **Spring Cache** in a Spring Boot application

1. Basic caching annotations: **@Cacheable**, **@CacheEvict**, and **@CachePut**

Integrating Spring Boot with **popular caching providers**

1. Using **Ehcache**, **Redis**, and **Hazelcast** for distributed caching
2. Configuring a **Redis-based cache** for high performance

Optimizing cache performance with **TTL (Time-to-Live)** and **cache eviction policies**

1. Handling cache synchronization in distributed systems

6. Messaging and Asynchronous Communication in

Microservices

Why asynchronous communication is essential in modern distributed systems

1. Introduction to **event-driven architectures**
2. Comparison of **synchronous** vs **asynchronous** communication in microservices

Java Messaging Service (JMS) overview

1. Setting up a **JMS broker** (ActiveMQ) in a Spring Boot project
2. Implementing **message producers** and **message listeners** using **Spring JMS**

Integrating **RabbitMQ** as a message broker

1. Using **Spring AMQP** for working with RabbitMQ
2. Building a message-driven system for microservices communication

Implementing **message queues** and **publish-subscribe** patterns

1. Managing reliability and message durability in distributed systems
2. Ensuring **message acknowledgments** and **retry mechanisms** are in place

7. Distributed Transactions in Microservices

Understanding **distributed transactions** and their challenges

1. Why **two-phase commit** is not ideal for microservices
2. Implementing **eventual consistency** and **saga patterns**

Using **Spring Boot** and **JMS** for managing distributed transactions

1. Orchestrating long-running transactions using **Compensating Transactions**

8. Final Project: Building and Deploying a Microservices-based Application

Developing a complete microservices-based application using **Spring Boot**

1. Multiple services communicating asynchronously via **JMS** and **RabbitMQ**
2. Caching strategies for optimizing performance in a distributed system
3. Using **Spring Cloud Gateway** for routing and load balancing

Deploying microservices to a **remote Linux server**

1. Configuring **Nginx** as a reverse proxy for microservices
2. Monitoring service health and logs using **Spring Boot Actuator**

This course will give you the expertise needed to architect and develop advanced, enterprise-grade systems using **Spring Boot**, **Spring Cloud**, and **JMS**. You'll be equipped with practical skills in **microservices**, **distributed caching**, and **asynchronous messaging** that are highly sought after in today's development landscape.

© 2026 Faysal Aziz · Kuala Lumpur