Faysal Aziz

OJT with Secure Coding in Spring Boot and Java

FA-0342 · 2-day

Software Development · Cybersecurity

Introduction

Advanced 2-day course

This is an advanced course tailored for experienced Java and Spring Boot developers. In this training, we aim to bridge the gap between standard application development and secure coding practices. With a focus on security, we will delve into principles that fortify your code against vulnerabilities, tackle migration strategies in Spring Boot, and address issues raised by code analysis tools like SonarQube. Additionally, we'll explore how to integrate and leverage tools such as SonarQube and Swagger for enhancing the security and documentation of your Spring Boot applications.

Learning outcomes

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

- Understand the fundamentals and importance of secure coding practices in Java and Spring Boot applications.
- Recognize common security pitfalls in Java code and how to avoid them.
- Apply secure coding principles to new and existing Spring Boot projects.
- Conduct migrations in Spring Boot applications effectively, ensuring minimal disruption and enhanced security.
- Interpret and act on issues and vulnerabilities reported by SonarQube.
- Optimize the use of SonarQube for continuous inspection of code quality.
- Implement Swagger for API documentation and testing with a focus on secure coding practices.

- Formulate strategies for resolving library and dependency issues with consideration of security implications.

Prerequisites

Participants should have:

- Proficiency in Java programming.
- Practical experience with Spring and Spring Boot frameworks.
- Familiarity with Maven or Gradle build tools.
- Basic knowledge of application security principles.
- Previous exposure to SonarQube and Swagger is beneficial but not mandatory.

Training outline

1. 1: Secure Coding Fundamentals in Java

- 1.1: Overview of Security in Java
- 1.2: Common Security Vulnerabilities (OWASP Top 10)
- 1.3: Secure Coding Best Practices
- 1.4: Security Features in JDK

2. 2: Enhancing Spring Boot Security

- 2.1: Spring Security Framework
- 2.2: Authentication vs. Authorization Mechanisms
- 2.3: OAuth2 and JWT Integration
- 2.4: Best Practices for Secure Session Management

3. 3: Spring Boot Migration Strategies

- 3.1: Strategies for Effective Migration
- 3.2: Managing Spring Boot Version Upgrades
- 3.3: Migrating Security Configurations
- 3.4: Testing and Verification Post-Migration

4. 4: Dependency Management and Library Upgrades

- 4.1: Assessing and Updating Dependencies
- 4.2: Handling Deprecated Libraries
- 4.3: Automated Tools for Dependency Checks

5. 5: Utilizing SonarQube for Secure Code

- 5.1: Setting Up SonarQube with Spring Boot
- 5.2: Analyzing and Interpreting SonarQube Reports
- 5.3: Addressing Security Hotspots and Vulnerabilities
- 5.4: Integrating SonarQube in CI/CD Pipeline

6. 6: Swagger for API Documentation and Security

- 6.1: Swagger Integration with Spring Boot
- 6.2: Secure API Documentation Practices
- 6.3: Testing APIs with Swagger UI
- 6.4: Customizing Swagger for Enhanced Security

7. 7: Advanced Secure Coding Techniques

- 7.1: Implementing Advanced Java Security Features
- 7.2: Security Logging and Monitoring
- 7.3: Incident Response and Vulnerability Patching

8. 8: Hands-on Exercises and Best Practices

- 8.1: Refactoring Legacy Code with Security in Mind
- 8.2: Case Studies: Resolving Real-World Security Issues
- 8.3: Peer Review and Pair Programming for Secure Code
- 8.4: Developing a Security-Focused Mindset in Coding

9. Conclusion and Next Steps

- Review of Key Course Takeaways
- Continued Learning Resources
- Establishing a Routine for Secure Coding Practices

This comprehensive outline will guide the instruction over the course of the two-day session, ensuring that participants not only understand secure coding principles but also learn to effectively apply these practices in their Spring Boot applications.

© 2026 Faysal Aziz · Kuala Lumpur