Spring Core

About this course

Spring Boot was introduced in 2013 and came as an opinionated way to simplify Spring configuration. It is now widely used by Java and Kotlin developers. This course offers an extensive blend of fundamentals concepts, simple tips and best practices, empowering you to become an expert with developing Spring backend applications.

Core-Spring certification

This course comprehensively covers all the topics required for achieving Core-Spring certification (the Core-Spring certification is a separate exam which may be taken as a follow-up for this course)

Java or Kotlin?

This course includes 60% Labs and all of them are run with Kotlin or Java (depending on the learner's preference).

Goal for this course

With this course, you will be able to:

- Comprehend the foundational concepts of the Spring framework and Dependency Injection
- Harness the capabilities of Spring Boot to streamline application development and configuration
- Integrate Spring Boot with databases using Spring Data JPA and perform CRUD operations
- Understand Proxies and Aspect Oriented Programming
- Understand how Spring helps you manage Database Transactions
- Develop REST APIs using Spring Boot's powerful @RestController and OpenAPI
- Learn how to call a REST API and manage exceptions elegantly
- Implement testing strategies for Spring Boot applications, including unit tests and integration tests
- Understand how to organise your code elegantly so it can easily be migrated to microservices when needed
- Understand all the underlying techniques needed when moving to distributed systems and microservices

Training mode and course duration

Full-time, in-person or Live-online
40% Theory / 60% Labs
Duration: 3 days (21 hours)

Who should attend

The audience for this course includes Software Developers, Architects, and Engineering Managers with basic knowledge of Java or Kotlin

Course Outline

Introduction to Spring Boot

- What is the Spring framework
- Spring Boot as a way to reduce Spring configuration
- Overview of Spring Boot features, development workflow, and use cases
- Creating a simple Spring Boot project using start.spring.io
- Running the application and exploring the default behaviour

Dependency Injection with Spring

- Why should you use Dependency Injection? (examples with Singletons, swapping dependencies, Proxies)
- When should you use Dependency Injection? When not to use it?
- @Scope, @Qualifier and @Value annotations
- The Spring lifecycle

Aspect Oriented Programming with Spring

- Out-of-the-box versus custom Aspects
- How Spring uses Proxies to inject technical code
- Understanding common use-cases: Transactions, Security, Caching…
- Building your own Aspect class

Data Access with Spring Data JPA

- JDBC, JPA, Hibernate, Spring Data JPA: why so many libraries?
- Understanding JPA without Spring (mapping, JPA states, lazy loading)
- Writing queries with JPQL
- Using Spring Data repositories to simplify JPA interactions
- How to use loose relationships and aggregates in order to keep your queries simple, and be able to move to microservices when needed

Transactions with Spring

- Programmatic versus Declarative Transactions
- Managing transactions with Spring's declarative approach
- Using @Transactional inside a Unit Test
- Working with transaction propagation and rollback scenarios
- Getting ready for distributed systems: Transaction IDs and idempotency

Building RESTful APIs with Spring Boot

- How to expose REST services with Spring (get/post/put/delete)
- Design your APIs using OpenAPI and Swagger-UI
- Contract-first versus Contract-last
- Validating incoming requests (with or without @Valid)
- Customising JSON serialisation with Jackson annotations
- Understanding the Data Transfer Object pattern and when to use it
- Error management: how to handle Exceptions elegantly

Testing and Profiles

- Summary of Unit Testing tools used in this course
- How Spring caches the context to run tests faster
- Test Containers to run full-blown databases in your tests
- How to use Spring Profiles in order to use predefined configuration in development / UAT / production

Observability

- Difference between Logs, Metrics and Traces
- Introduction to Spring Boot Actuator
- How to monitor your application's health
- Changing the log level at runtime
- How to collect Metrics and Traces in a Distributed System

Microservices with Spring

- Monolith versus Microservices
- Why microservices are not always suitable
- How to organise your code and database from the start in order to be ready for microservices
- Distributed Systems Techniques
- Technical components that should be used in a microservices architecture