Spring 4.x and Microservices (TT3340)

Modality: Virtual Classroom

Duration: 3 Days

About this course:

The spring system is an application structure that gives a lightweight container that supports the production of easy to-complex parts in a non-obtrusive style. Spring's transparency and flexibility are supportive and congruent with incremental testing and development. The structure of the framework helps the layering of functionality, for example, view-oriented frameworks, transactions, persistence, and enterprise capabilities and systems.

The target of this course is spring 4.x, which incorporates full help for Java 8 and JEE 7 (Java's prior versions and JEE keep on being supported). Spring supports the utilization of method references and lambda expressions in huge numbers of its APIs.

Spring makes easier the development of JEE. Spring makes simple the basic assignments and empowers great structure based on programming to interfaces. The configuration is now easier with the help of spring and decreases the requirement for some patterns of JEE designs. Spring puts the design of OO back into your application of JEE, and it coordinates pleasantly with many view innovations and the new HTML5 features.

Note that our training of Spring is highly modularized and include the entire spectrum. In that capacity, we can customize courses to your particular needs. Coming up next is a significant level listing of the topics of Spring to consider in building your modified training of Spring:

- Core Framework of Spring (including Dependency Injection, Inversion of Control, and Aspect-Oriented Programming).
- Advanced Framework Projects and Features (comprising Handling Application Events, Spring Boot, and Spring Security).
- Persistence and Spring (counting Transactions, Spring DAOs, and Spring Data).
- The Web and Spring (comprising HTML5, Spring MVC and Web Flow supporting Web Sockets, and asynchronous processing).
- Incorporating Spring into the Enterprise (comprising Integration working of Spring with JMS and other remoting alternatives).

The normal pay of a Java Spring Developer is $117,087 annually.

Course Objective:

Understudies are taken on a top to bottom voyage through the essential Spring system, before jumping into what microservices are and how they are upheld.

Working in a dynamic, environment of lab-escalated hands-on coding, understudies will figure out how to:

- Analyze the ecosystem that supports and surrounds microservices, including monitoring, typical stack, deploying, containerizing, logging, and orchestrating microservices.
- Implement and Design REST services
- Comprehend the steps associated in the progress of Microservices
- Understand how and why of fundamental microservices
- Understand the patterns and anti-patterns of microservice.
- Examine the basics of Spring
- Perform with microservices best practices.
- Look at the difficulties of utilizing microservices.
- Work with OpenId and OAuth.
- Work with JavaConfig and Spring Boot to more efficiently and effectively developing the applications of Spring.

Audience:

This training course is for intermediate-level Spring 4.x, designed for developers who want to comprehend when and how to use Spring in the applications of Java and JEE.

Prerequisite:

Participants should have practical experience in basic Java development.

Course Outline:

Module 1: Introduction to Spring 4

Lesson: The Spring Framework

- Spring Architecture
- Dependency Injection
- Spring DI Container
- Bean Creation Using Factories
- Configuration Options: XML, Annotations, or JavaConfig
- Use of Lambda Expressions and Method References in Spring
- Tutorial: Setup Eclipse Neon for Using Maven
- Exercise: Hello World Spring Application
- Exercise: Configuring Dependencies

Lesson: Spring Beans and Advanced Configuration

- Spring's Pre-Built Factory Beans
- PropertyPlaceholderConfigurer
- Custom Property Editors
- Lazy Bean Resolution
- Ordered Autowiring

- Using Configuration Classes
- Organizing Configuration Classes
- Exercise: Advanced Configuration

Module 2: Microservices

Lesson: From Monolithic to Microservices

- Understand and discuss the continuing trend to reduce the monolithic nature of applications
- Explain the principles and characteristics of microservices
- Recognize both good and poor candidates for microservices

Lesson: Designing Microservices

- Work on defining boundaries for microservices
- Make other design decisions relating to communication styles, orchestration and many other issues
- Understand some of the challenges to designing and implementing microservices

Lesson: Supporting and Managing Microservices

- Understand how scaling is achieved using microservices
- Discuss the many options for deploying and managing microservices

Module 3: The Microservices Ecosystem

Lesson: Working with Microservices

- Typical Microservices Stack
- Monitoring Microservices
- Logging
- Containerizing with Docker
- Deploying into Docker
- Orchestration of Microservices

Lesson: Microservice Best Practices

- Motivation and Mindset
- Minimum Viable Product
- Challenges of Data and Data Islands
- Spring Data and Microservices
- PrePersist, PreUpdate, and Repository Interface
- A DevOps-Style Microservice Life Cycle
- Continuous Delivery Pipeline
- Governance
- Tracking APIs and API Consumers

Lesson: Microservice Patterns

- Aggregator Pattern
- Branch Pattern
- Proxy Pattern
- Chained Pattern
- Circuit Breaker/Bulkhead Isolation Pattern
- Continuous Integration/Delivery Pattern
- Shared Resources as an Anti-Pattern
- Shared Resources as a Pattern
- Async Messaging Pattern

Lesson: Microservice Anti-Patterns and Challenges

- Microservice Costs
- When to Apply and NOT Apply
- Data Islands
- Dependency management
- Cohesion Creep
- Avoiding Versioning

Lesson: Overview of OAuth and OpenID

- OAuth 2.0 Terminology and Concepts
- Usage Models
- OAuth 2.0 Tokens
- OAuth 2.0 Details
- Example: Google Usage Models
- OpenID Connect Overview
- OpenID Providers

Module 4: Working with REST

Lesson: Overview of REST

- REpresentational State Transfer
- REST Characteristics
- REST Elements
- REST Architectural Principles
- REST and HTTP
- REST/HTTP: Representation-Oriented
- REST Design Principles
- Exercise: Working With REST

Lesson: Designing RESTful Services

- Effectively Designing RESTful Services

- Best Practices for Endpoint Definition
- Using Query Parameters
- Working with HTTP GET and DELETE
- Working with HTTP PUT
- Working with HTTP POST
- Best Practices for HTTP Methods
- Handling Additional Operations

Module 5: Implementing REST with Spring

Lesson: RESTful Services in Spring

- Understand how Spring supports the implementation of RESTful services
- Use Spring to map URIs and extract values from the URI
- Work with @RequestMapping to support routing decisions based on what type should be processed by associated method
- Handle response codes
- Work with view resolvers, HTTP message converters, and content negotiation
- Exercise: Working with Spring REST

Lesson: RESTful Clients in Spring

- Understand how Spring supports browser-based RESTful clients
- Understand how Spring supports Spring-based RESTful clients
- Exercise: Injection in Spring REST
- Exercise: Exception Mapping in Spring REST
- Exercise: Content Negotiation in Spring REST

Module 6: Spring Boot

Lesson: Spring IO Platform

- Understand the Spring IO Platform
- Understand the IO Bill of Materials
- Understand the IO Foundation
- Learn how the IO Execution will be leveraged
- Learn how Spring Cloud is used for Platform Coordination

Lesson: Spring Boot Overview

- What is Spring Boot
- Explore Spring Boot starters
- Examine Spring Boot’s AutoConfiguration as well as its command-line interface (CLI)
- Understand the Spring Boot Actuator

Lesson: Spring Boot Introduction

- Spring Boot JPA Starter
- Examine Spring Boot’s AutoConfiguration
- Understand the Spring Conditionals
- Understand Spring Boot DevTools
- Exercise: Create a "REST JPA Repository"

Lesson: Advanced Spring Boot

- Explore additional Spring Boot starters
- Bootstrapping Spring Boot
- Understand Spring Boot Actuators
- Create and run a Spring Thymeleaf MVC application
- Exercise: Create a “Thymeleaf MVC With JPA Repository”