**Course: Spring Core**

**Duration: 4 Days**

**Day -1**

Lesson -1: Spring Overview

* What is the Spring Framework?
* The DI Container
* The Spring Framework History and EcoSystem

Lesson – 2: Java Configuration

* Java configuration and the Spring application context
* @Configuration and @Bean annotations
* @Import: working with multiple configuration files
* Defining bean scopes
* Launching a Spring Application and obtaining Beans

Lesson – 3: More Java Configuration

* External properties & Property sources
* Environment abstraction
* Using bean profiles
* Spring Expression Language (SpEL)

Lesson – 4: Annotation and Component Scanning

* Component scanning
* Autowiring using @Autowired
* Java configuration versus annotations, mixing.
* Lifecycle annotations: @PostConstruct and @PreDestroy
* Stereotypes and meta-annotations

Lesson – 5: Inside the Spring Container

* The Spring Bean Lifecycle
* The BeanFactoryPostProcessor interception point
* The BeanPostProcessor interception point
* Spring Bean Proxies
* @Bean method return types

Lesson – 6: Introducing Aspect-oriented programming

* What problems does AOP solve?
* Defining pointcut expressions
* Implementing various types of advice

**Day -2**

Lesson -7: Testing a Spring-based Application

* Spring and Test-Driven Development
* Spring 5 integration testing with JUnit 5
* Application context caching and the @DirtiesContext annotation
* Profile selection with @ActiveProfiles
* Easy test data setup with @Sql

Lesson-8: JDBC Simplification with JdbcTemplate

* How Spring integrates with existing data access technologies
* Spring‘s JdbcTemplate
* DataAccessException hierarchy

Lesson-9: Transaction Management with Spring

* Transaction overview
* Transaction management with Spring
* Transaction propagation and rollback rules
* Transactions and integration testing

Lesson- 10: Spring Boot Feature Introduction

* Introduction to Spring Boot Features
* Value Proposition of Spring Boot
* Creating a simple Boot application using the Spring Initializer website

Lesson-11: Spring Boot – A closer look

* Dependency management using Spring Boot starters
* How auto-configuration works
* Configuration properties
* Overriding auto-configuration
* Using CommandLineRunner

**Day -3**

Lesson-12: Spring Boot – Spring Data JPA

* Quick introduction to ORM with JPA
* Benefits of using Spring with JPA
* JPA configuration in Spring
* Configuring Spring JPA using Spring Boot
* Spring Data JPA dynamic repositories

Lesson – 13: Web Applications with Spring Boot

* Introduction to Spring MVC and request processing
* Controller method signatures
* Configuring Spring MVC with Spring Boot
* Spring Boot packaging options, JAR or WAR

Lesson – 14: RESful Application with Spring Boot

* An introduction to the REST architectural style
* Controlling HTTP response codes with @ResponseStatus
* Implementing REST with Spring MVC, @RequestMapping, @RequestBody and @ResponseBody
* Spring MVC’s HttpMessageConverters and automatic content negotiation

**Sample Project:**

**Notes and Comment Management Application implementation using microservices architecture.**

(Detailed description is provided in the appendix)

**Day -4**

**Sample Project contd.,**

Lesson – 15: Spring Boot Testing

* Spring Boot testing overview
* Integration testing using @SpringBootTest
* Web slice testing with MockMvc framework
* Slices to test different layers of the application

Lesson – 16: Securing REST Application with Spring Security

* What problems does Spring Security solve?
* Configuring authentication
* Implementing authorization by intercepting URLs
* Authorization at the Java method level
* Understanding the Spring Security filter chain
* Spring security testing

Lesson – 17: Actuators, Metrics and Health Indicators

* Exposing Spring Boot Actuator endpoints
* Custom Metrics
* Health Indicators
* Creating custom Health Indicators
* External monitoring systems

Appendix – Sample Project

Title: Notes and Comments Management Application

Description: Notes and Comments Management Application is a sample project to implement the microservices architecture. This application will allow the registered users to post their notes and other users can be able to view all notes and can comment on the post, the author of the notes can be able to view all the comments received for his/her notes, and he/she can be able to delete his/her own notes and can view all others notes as well. The commenter who commented on a particular note can also be able to delete or update the comment.

Scope: In this project, two microservices viz., note-service and comment-service are created, where note-service will provide notes-related activities and comment service will provide comment-related services, where note service will communicate to comment service to get the comments received for a particular note. Note-service and comment-service will be in localhost in two different tomcat servers in different ports.

Architecture: Microservices

Design Patterns: Aggregator, API Gateway, Circuit Breaker, Database or Shared Data Pattern

Backend Database: MongoDb / MySQL

Technologies: Spring, Springboot and Springdata JPA

Servers: Tomcat, Eureka (For service discovery and load balancing)

Gateway: Spring Cloud Gateway