Module Title	**:**	**Spring: Core Training**
Duration	**:**	**4 days**

OVERVIEW

This 4-day course offers hands-on experience with the major features of Spring and Spring Boot, which includes configuration, data access, REST, AOP, auto-configuration, actuator, security, and Spring testing framework to build enterprise and microservices applications. On completion, participants will have a foundation for creating enterprise and cloud-ready applications.

This course prepares students for the Spring Professional certification exam.

COURSE OBJECTIVES

By the end of the course, you should be able to meet the following objectives:

- Spring configuration using Java Configuration and Annotations
- Aspect oriented programming with Spring
- Testing Spring applications using JUnit 5
- Spring Data Access - JDBC, JPA and Spring Data
- Spring Transaction Management
- Simplifying application development with Spring Boot
- Spring Boot auto-configuration, starters and properties
- Build a simple REST application using Spring Boot, embedded Web Server and fat JARs or classic WARs
- Implementing REST client applications using RestTemplate
- Utilize Spring Boot enhancements to testing
- Spring Security
- Enable and extend metrics and monitoring capabilities using Spring Boot actuator

PREREQUISITES

Some developer experience using Java, an IDE (Eclipse, STS or IntelliJ) and build tools such as Maven or Gradle.

TARGET AUDIENCE

Application developers who want to increase their understanding of Spring and Spring Boot with hands-on experience and a focus on fundamentals

COURSE MODULES

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

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

More Java Configuration
- External properties & Property sources
- Environment abstraction
- Using bean profiles
- Spring Expression Language (SpEL)

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

Inside the Spring Container
- The Spring Bean Lifecycle
- The BeanFactoryPostProcessor interception point
- The BeanPostProcessor interception point
- Spring Bean Proxies
- @Bean method return types

Introducing Aspect-oriented programming

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

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

JDBC Simplification with JdbcTemplate

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

Transaction Management with Spring

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

Spring Boot Feature Introduction

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

Spring Boot – A closer look

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

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

Web Applications with Spring Boot

- Introduction to Spring MVC and request processing
- Controller method signatures
- Using @Controller, @RestController and @GetMapping annotations
- Configuring Spring MVC with Spring Boot
- Spring Boot packaging options, JAR or WAR

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

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

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

Actuators, Metrics and Health Indicators

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