Rumos valorizamos profissionais!

Web Development with Spring MVC

- **Formato do curso:** Presencial
- **Preço:** 1750€
- **Duração:** 30 horas

The Spring Web MVC framework provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications.

In this 5-day course, students will learn how to build a Spring-powered Java application that demonstrates the Spring Framework and other Spring technologies.

Destinatários

This course targets professional Java programmers who want to learn how to develop Spring MVC applications

Pré-requisitos

- Participants of this course need to have a solid understanding of Java, HTML, JavaScript and CSS Development.
- The used IDE will be Eclipse.
- Previous knowledge of the Spring framework is not required.

Objectivos

- Develop web applications using Spring MVC
- Map requests to controllers using annotations
- Implement full SCRUD Web applications
- Display, validate and process forms
- Implement REST style interfaces to support Ajax requests
- Secure web applications using Spring security
- Create Micro Services with Spring Cloud

Programa

Introduction to Spring

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

Spring Java Configuration: A Deeper Look

- External properties & Property sources
- Environment abstraction
- Bean scope, bean profiles
- Spring Expression Language (SpEL)
- How it Works: Inheritance based proxies

Annotation-Based Dependency Injection

- Autowiring and component scanning
- Java configuration versus annotations, mixing.
- Lifecycle annotations: @PostConstruct and @PreDestroy
- Stereotypes and meta-annotations

XML Dependency Injection

- XML syntax, constructor & setter injection
- Resource prefixes
- Namespaces and best practices when using XML
- XML profile selection

The Bean Lifecycle: How Does Spring Work Internally?

- The init phase: available interceptors
- The init phase: what is the difference between XML, annotations and Java configuration?
- The use and destruction phases

Testing A Spring-Based Application

- Spring and Test Driven Development
- ContextConfiguration and @RunWith annotations
- Application context caching and the @DirtiesContext annotation
- Profile selection with @ActiveProfiles
- Easy test data setup with @Sql

Aspect-Oriented Programming

- What problems does AOP solve?
- Differences between Spring AOP and AspectJ

- Defining pointcut expressions
- Implementing an advice: Around, Before, After

Data Access and Jdbc with Spring

- How Spring integrates with existing data access technologies
- DataAccessException hierarchy
- Implementing caching using @Cacheable
- jdbc namespace and Spring‘s JdbcTemplate

Database Transactions with Spring

- Transactional annotation
- Transactions configuration: XML versus annotations
- Isolation levels, transaction propagation and rollback rules
- Transactions and integration testing
- Should you use read-only transactions?

JPA with Spring and Spring Data

- Quick introduction to ORM with JPA
- Benefits of using Spring with JPA
- JPA configuration in Spring
- Spring Data JPA dynamic repositories

Spring in A Web Application

- Configuring Spring in a Web application
- Introduction to Spring MVC, required configuration
- Controller method signatures
- Views and ViewResolvers
- Using @Controller and @RequestMapping annotations

Spring Boot

- Using Spring Boot to bypass most configuration
- Simplified dependency management with starter POMs
- Packaging options, JAR or WAR
- Easily overriding Spring Boot defaults

Spring Boot – Going Further

- Going beyond the default settings
- Customizing Spring Boot configuration
- Logging control
- Configuration properties using YAML
- Boot-driven testing

Spring Security

- What problems does Spring Security solve?
- Configuring authentication and intercepting URLs
- The Spring Security tag library for JSPs
- Security at the method level
- Customizing the Spring Security filter chain

REST with Spring MVC

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

Microservices with Spring Cloud

- Microservice Architectures
- Challenges with cloud-native applications
- Using Spring Cloud
- Developing microservice systems