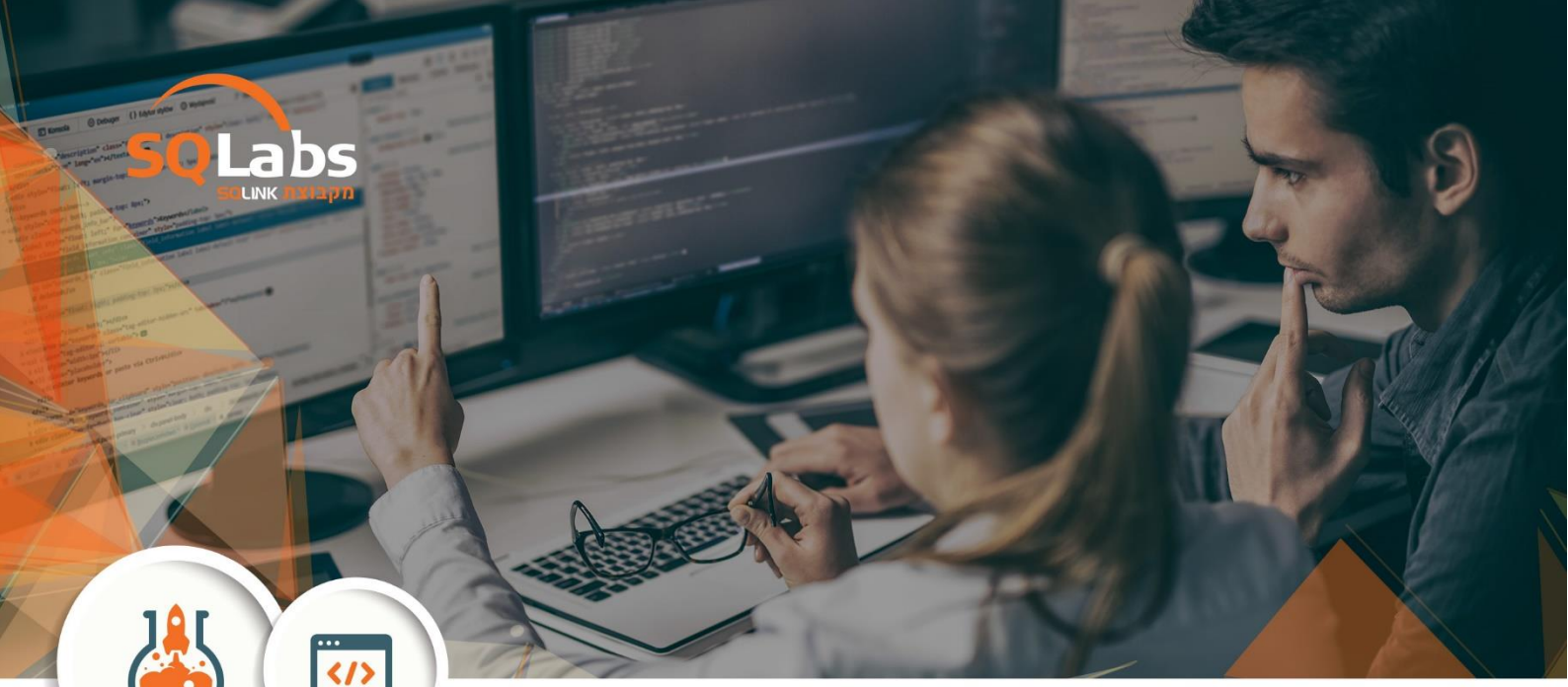

# Microservices with Spring Boot and Spring Cloud

**Course Duration: 16 Hours**

## Overview

The Spring Framework is the world's most popular open source framework for Java developers. Spring facilitates a dynamic framework for high productivity, modular and maintainable applications and system development.
This allows Spring developers to understand the mechanisms behind the way Spring works, how to avoid common pitfalls and how to design better architectures using Spring.
Students will learn how to improve code quality using Spring testing with JUnit & Mockito.

Web services, REST APIs and the cloud have entered every corner of the software development world. Course participants will learn how to leverage Spring to develop Microservices and REST architectures, and will strengthen this knowledge using hands-on exercises with Spring MVC, Spring Boot, and Spring Cloud.

## Who Should Attend

- Java developers who want to develop Microservices and web APIs with Spring
- Developers who want to know how Spring leverages cloud technologies
- Developers migrating from Spring to Spring Boot

## Prerequisites

- Experience developing with the Spring Framework and Java
- Basic familiarity with networking concepts such as HTTP

## Course Contents

### Refresher: Introduction to the Spring Framework

- Course introduction
- The importance of Dependency Injection
- Manual Dependency Injection Demo
- Dependency Injection in Spring
- Inversion of Control (IOC)
- Frameworks vs Libraries
- Spring Hello world
- Beans and Creating Objects
- XML vs Annotations
- Singletons vs Prototypes
- Working with Maven
- Common Mistakes
- @Component vs @Bean
- Lazy vs Eager
- Beans with Parameters
- DI using Autowired and Inject
- Custom Creation Functions
- Choosing between injection Candidates

### Spring testing

- Junit – brief
- Mocking – introduction
- SpringBoot test platform
- Understanding testing application context
- Tier oriented testing

### Spring Beans Exercises

- Creating Beans
- Creating Beans with Constructor Args

- Managing Singletons
- Injecting Dependencies
- Choosing between dependencies

**Introduction to Spring MVC**

- Core concepts of Spring MVC
- Spring Web Pages vs Web Services
- Introducing Spring Boot
- Spring Boot and Spring MVC
- Creating Spring MVC controllers
- Defining Spring MVC web services
- POST vs Get, working with JSON
- Layering Spring at the UI,  Logic and DB tiers
- Sessions and additional bean scopes
- RESTful Web Service development
- RestTemplate for REST clients

**Spring Web Services Exercises**

- Creating A Spring web service
- Creating A Spring Rest API service

**Spring Boot**

- Introduction
- start.spring.io & extensions
- Installation
- Configuring POM
- Setting class dependencies
- Coding & @EnableAutoConfiguration
- Creating executable jars

**Microservices, REST and the Cloud**

- Breaking the Monolith
- Netflix's Horror Story
- HTTP, REST, and SOAP
- SOA vs Microservices
- Microservices advantages and challenges

- Microservices and  the cloud
- Best Practices
- Cloud Patterns
- Load Balancing

**Spring Cloud**

- Configuration server - Eureka
- Service discovery - Eureka
- API gateway - Zuul
- Circuit breaker (load-balancing) - Hystrix
- Monitoring – Hystrix
- Load-balancing – Spring Ribbon
- Feign clients

**Spring Cloud  Exercises**

- Configuration server exercise
- Service discovery exercise