

**Spring Boot with Kotlin**

**Course Number:** MBL-224  
**Duration:** 3 days

**Overview**

This Spring Boot with Kotlin training course teaches developers how to use all the features of Kotlin, refactor Kotlin codebases in IntelliJ, perform TDD using KotlinTest, and generate mock objects using Mockito. Students learn how to develop microservices via both Spring MVC and WebFlux and create basic Android applications.

**Prerequisites**

Students must be confident Java developers, however, we would be delighted to add content from the [Introduction to Kotlin](file:////training/kotlin-introduction) course for students who are new to the language. Attendees should also have a basic understanding of the architecture of RESTful services and Single Page Applications.

**Materials**

All Kotlin training attendees receive comprehensive courseware.

Course outline and materials are copyrighted and owned by [Instil Software](https://instil.co/).

**Software Needed on Each Student PC**

* Windows or Mac minimum 8 GB RAM
* Android Studio installed
* Provided lab files from Accelebrate

**Objectives**

All attendees will learn how to:

* Write RESTful services in Kotlin via Spring Boot, MVC, and WebFlux
* Write unit tests for Spring including using automatic mock injection
* Use Spring Boot starters for data access and security
* Leverage Kotlin to simplify building Spring Boot applications

**Outline**

* Introduction to Spring
  + Weaknesses in the original JEE architecture
  + Why Spring arose as a cure for the maladies of JEE
  + Configuring Spring via XML, YAML, Groovy and Kotlin
  + How to choose between the configuration options
  + Springs growth from library to framework to ecosystem
  + Choosing between the Spring platform and JEE8
* Annotation Based Dependency Injection in Spring Core
  + The role of the
  + Different ways of providing ‘bean wiring’ to the context object
  + Understanding scopes and selecting the correct scope for a bean
  + Using
  + Using
  + Using
  + Declaring beans using
  + Creating bean provider methods via
  + A detailed introduction to Spring Expression Language (Spring EL)
  + Populating fields via Spring EL using
* Other Forms of Dependency Injection in Spring Core
  + The XML based Bean Description Language and Schema Extensions
  + The Groovy and Kotlin based Domain Specific Languages
  + Support for standard properties files and YAML
* Unit Testing Spring Beans
  + How a DI container aids unit and integration testing
  + Configuring the Spring specific test runner for JUnit
  + Injecting dependencies into JUnit tests via Spring
  + Creating configurations for different testing scenarios
  + Combining mocking frameworks like Mockito with Spring
* Introducing Aspect Oriented Development
  + The notion of cross-cutting concerns (aka Aspects)
  + Key terms (Aspects, Advice, Pointcuts, Weaving etc…)
  + A detailed guide to AspectJ Pointcut Expressions
  + Support for AOP in Spring Core via auto-proxying
  + Why only method calls can be intercepted in Spring
  + Declaring Advice and Pointcuts using annotations
  + Understanding the five different kinds of advice
  + How AOP is used within Spring Security and Transactions
* Introduction to Spring Boot
  + The need for a meta-framework to manage Spring itself
  + Creating Spring Boot projects via the ‘Spring Initializr’
  + How Spring Boot configures other parts of Spring as modules
  + Customizing the Maven / Gradle build file to manage dependencies
  + Options for overriding the default configurations in Spring Boot
  + Building and testing command line applications in Spring Boot
* Using Spring MVC within Spring Boot
  + How MVC evolved from a Web Framework to a Microservices Platform
  + MVC Design (Dispatcher Servlet, Handler Mappings and View Resolvers)
  + Registering controllers via annotations and component scanning
  + The difference between
  + Deploying MVC Apps as Microservices via Spring Boot and Cloud Services
  + Overriding the default configurations and registering JEE components
* Basic Configuration of Spring Controllers
  + Associating controller beans with URL patterns
  + Mapping methods to HTTP verbs (GET, POST, PUT etc…)
  + Triggering methods based on parameters and headers
  + Passing objects from the Servlet API into methods
  + Injecting individual parameters and populating JavaBeans
  + Injecting information from HTTP headers and cookies
  + Using path variables to inject information from the URL
  + Marshalling the body of the request into JSON and/or XML
  + Customizing XML marshalling via the JAXB annotations
  + Cusomizing JSON marshalling via Jackson annotations
* Advanced Configuration of Spring Controllers
  + Wrapping the response type in
  + Creating
  + Customizing the response code and manipulating HTTP headers
  + Validating input via the JSR-303 Bean Validation annotations
  + Registering your own validators for cross-field validations
  + Using the
  + Defining model attributes and exception handler methods
  + Redirecting output to server pages via view resolvers
  + Configuring Thymeleaf as a sample server page library
* Writing Tests and Clients for Spring Controllers
  + The
  + Creating a Web Application Context within a JUnit test
  + Sending requests to controllers via the Dispatcher Servlet
  + Using the fluent API to specify requests and check responses
  + Writing clients for RESTful Services via the JAX-RS Client API
  + Writing clients for RESTful Services via the Spring
* Enhancements in Spring 5 and Spring Boot 2
  + Support for Functional and Reactive Programming in Spring 5
  + The new ‘WebFlux’ model for services in Spring Boot 2
  + Creating WebFlux based services via MVC annotations
  + Creating WebFlux services fia the functional model
  + Using the Kotlin based DSL for functional services
* Securing and Monitoring Spring Microservices
  + Combining Spring Security with Spring Boot Applications
  + Different options for adding authentication to endpoints
  + Using Spring Actuator to collect metrics from running services
  + Customizing and extending the built in metrics and health checks
* Database Access with Spring Data
  + The famously intractable ‘Object Relational Mismatch’
  + Review of ORM frameworks such as Hibernate and the JPA
  + How Spring Data simplifies the creation of repositories
  + Customizing and extending your repository components
* Conclusion