Agenda

- Spring Boot & Kotlin Demo
- Spring Framework & Spring Boot
- Spring Boot & Kotlin @ REWE digital
- Resources

Spring Boot & Kotlin Demo

Spring Framework & Spring Boot

Spring Framework

"The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications."

— *spring.io*

Spring Framework

- "An inversion of control container and an application framework for the Java Platform"[1]
- 17+ years old
- Many modules and side projects

[1] Wikipedia

Support for Kotlin in Spring Framework

- Null Safety
- Extention Functions
- DSLs
  - Web Router DSL
  - Mock Web MVC DSL
  - ...
- Coroutines
  - see also Going Reactive with Spring, Coroutines and Kotlin Flow

Spring Boot

"Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can 'just run'."

— *spring.io*

Spring Boot Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly
- Provide opinionated 'starter' dependencies to simplify build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration

Support for Kotlin in Spring Boot

- Official Support since Spring Boot 2.x
  - dependency managment
  - extention functions
  - immutable configuration properties
- Initial Project Setup via Spring Initializr
- See also "The State of Kotlin Support in Spring" Talks by Sébastien Deleuze on YouTube

Reactive Spring & Kotlin Coroutines

- Going Reactive with Spring, Coroutines and Kotlin Flow
- Deepdive into Reactive Spring with Coroutines and Kotlin Flow by Sébastien Deleuze

Spring Fu

"Incubator for Java and Kotlin Configuration DSL designed to configure Spring Boot explicitly with code in a declarative way."

— *Spring Fu*

```
val app = webApplication {
    beans {
        bean<SampleService>()
    }
    webMvc {
        port = if (profiles.contains("test")) 8181 else 8080
        router {
            val service = ref<SampleService>()
            GET("/") {
                ok().body(service.generateMessage())
            }
            GET("/api") {
                ok().body(Sample(service.generateMessage()))
            }
        }
        converters {
            string()
            jackson
}   }   }

fun main() { app.run() }
```

How we started using Kotlin with Spring Boot

1. REWE Angebote & Lieferservice app, 2016 with Kotlin
2. A few teams built a few new µ-services with SB and Kotlin
3. Teams in FF built Android apps and SB Backends in Kotlin
4. Internal coding dojos did the Kotlin Koans
5. Kotlin was often used in regular coding dojos @ REWE digital
6. Some devs took Kotlin for Java Developers @ Coursera
7. FF built a shared Kafka consumer library in Kotlin

Resources

Resources - Kotlin

- kotlinlang.org
- Learn Kotlin by Example
- Kotlin Koans
- Kotlin for Java Developers (Coursera)

Resources - Spring Boot with Kotlin

- Spring Framework Reference
- Spring Boot Reference
- spring.io Tutorial
- The State of Kotlin Support in Spring (YouTube)

Appendix

Kotlin - Extention Functions

Instead of this

```
fun <T> swap(list: MutableList<T>, index1: Int, index2: Int) {
    // implementation omitted ...
}

val list = mutableListOf(1, 2, 3)

swap(list, 1, 2)
```

Kotlin - Extention functions

... we can write this

```
fun <T> MutableList<T>.swap(index1: Int, index2: Int) {
    // now the list is bound to 'this'
}

val list = mutableListOf(1, 2, 3)

list.swap(1,2)
```

Kotlin - Reified Type Parameters

- Functions marked with `inline` will be inlined by the compiler
- In inlined generic functions, type parameters can be marked with `reified` and passed in at call side

Kotlin - Reified Type Parameters

```
inline fun <reified T> TreeNode.findFirstAncestorOfTypeOrNull(): T? {
    var p = this.parent
    while (p != null && p !is T) { // no reflection!
        p = p.parent
    }
    return p as T?
}

// usage:
treeNode.findParentOfType<MyTreeNode>()
```