Kotlin 1.4 Online Event

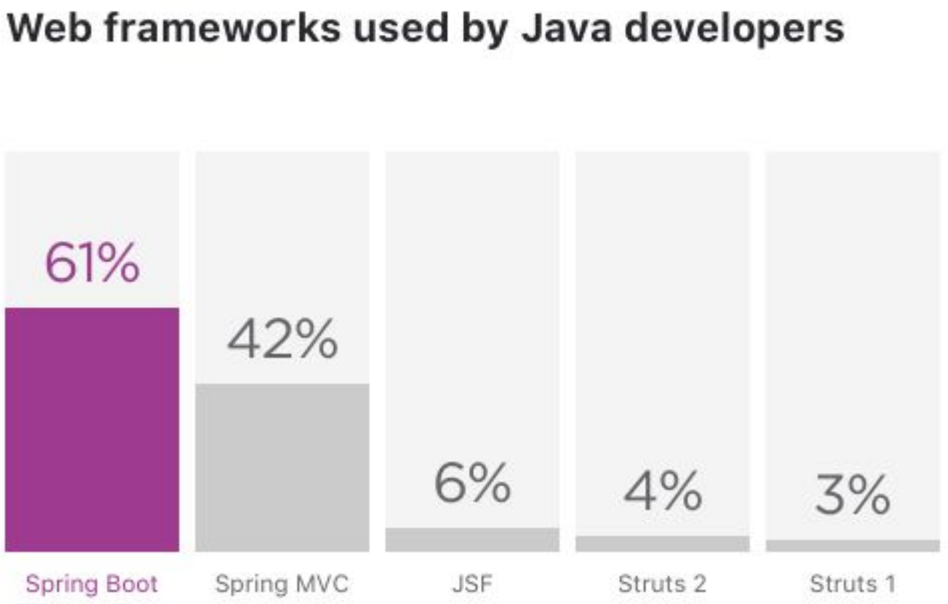
The State of Kotlin Support in Spring

Sébastien Deleuze

@sdeleuze

October 15, 2020

Spring is the server-side leader on the JVM



Source: a Picture of Java in 2020, JetBrains

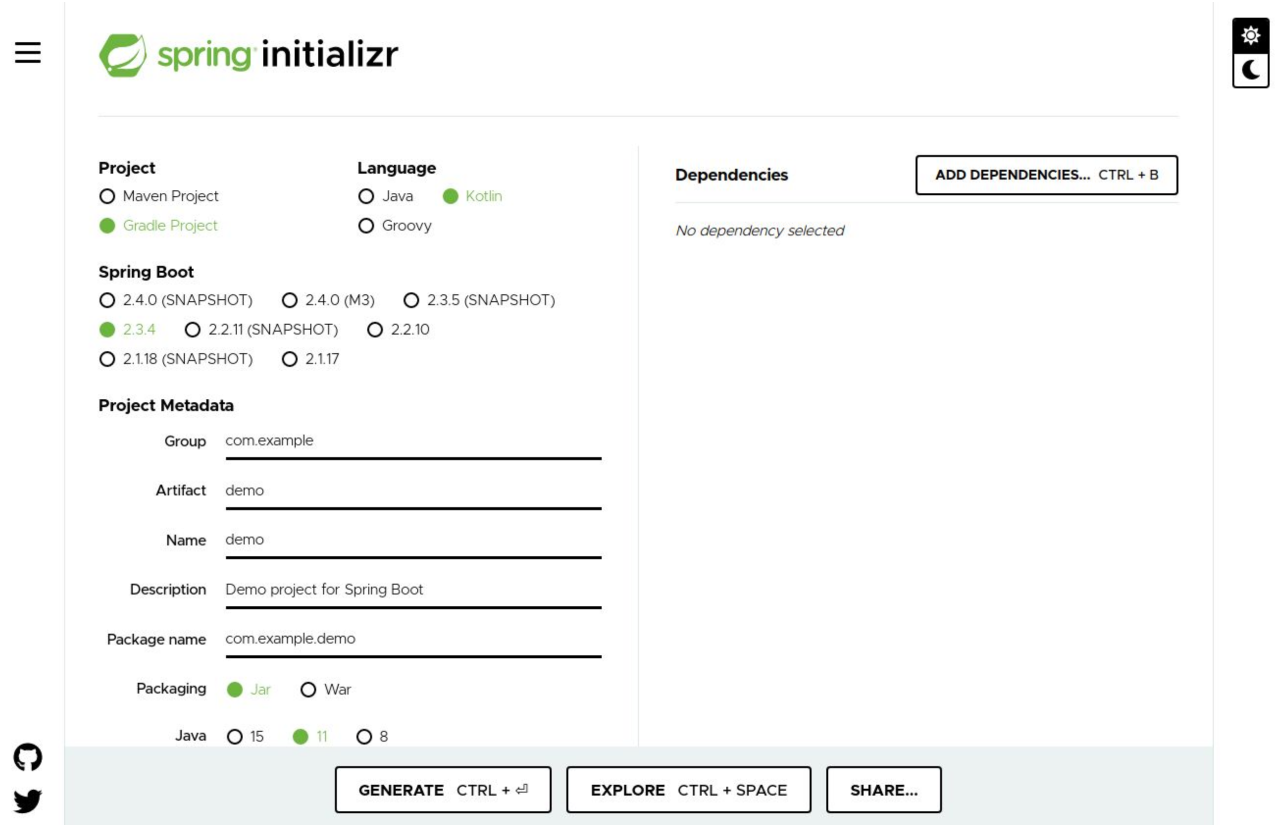
First class support for Java and Kotlin

More Spring Boot projects generated with Kotlin each year.

Let’s focus on Kotlin today

Getting started

Start your project on https://start.spring.io



spring initializr

Project
○ Maven Project
● Gradle Project

Language
○ Java ● Kotlin
○ Groovy

Spring Boot
○ 2.4.0 (SNAPSHOT) ○ 2.4.0 (M3) ○ 2.3.5 (SNAPSHOT)
● 2.3.4 ○ 2.2.11 (SNAPSHOT) ○ 2.2.10
○ 2.1.18 (SNAPSHOT) ○ 2.1.17

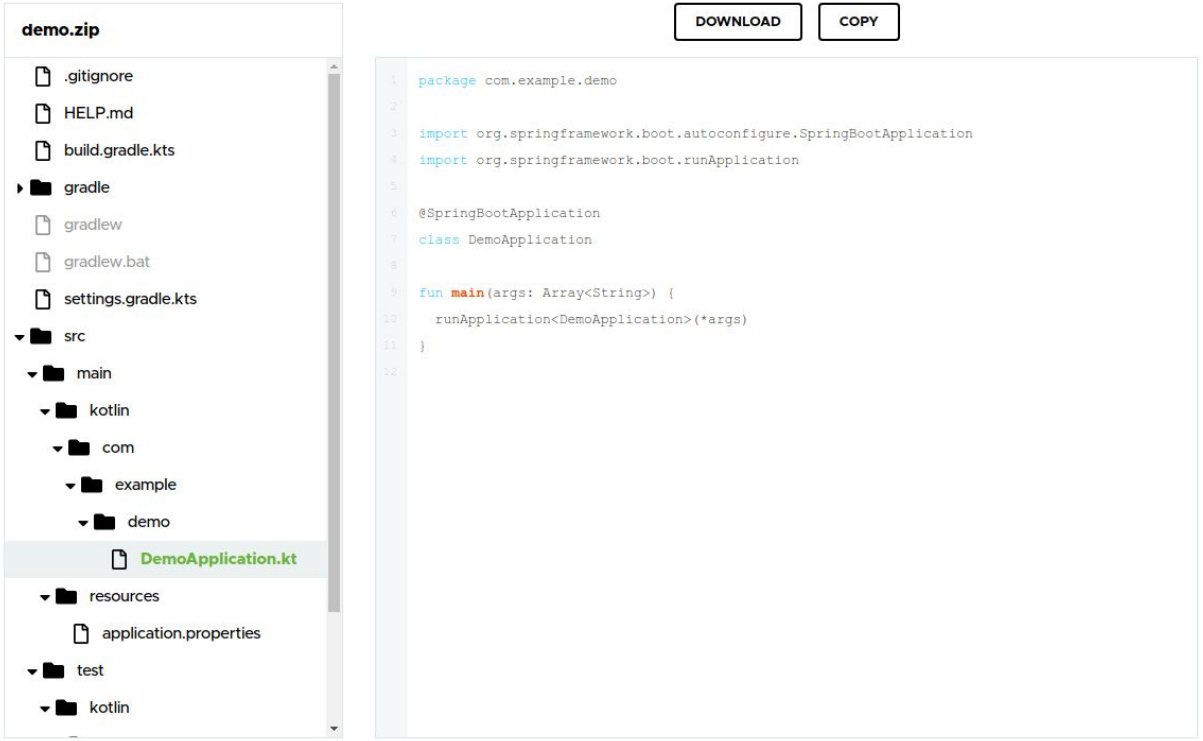
Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	● Jar ○ War
Java	○ 15 ● 11 ○ 8

Dependencies

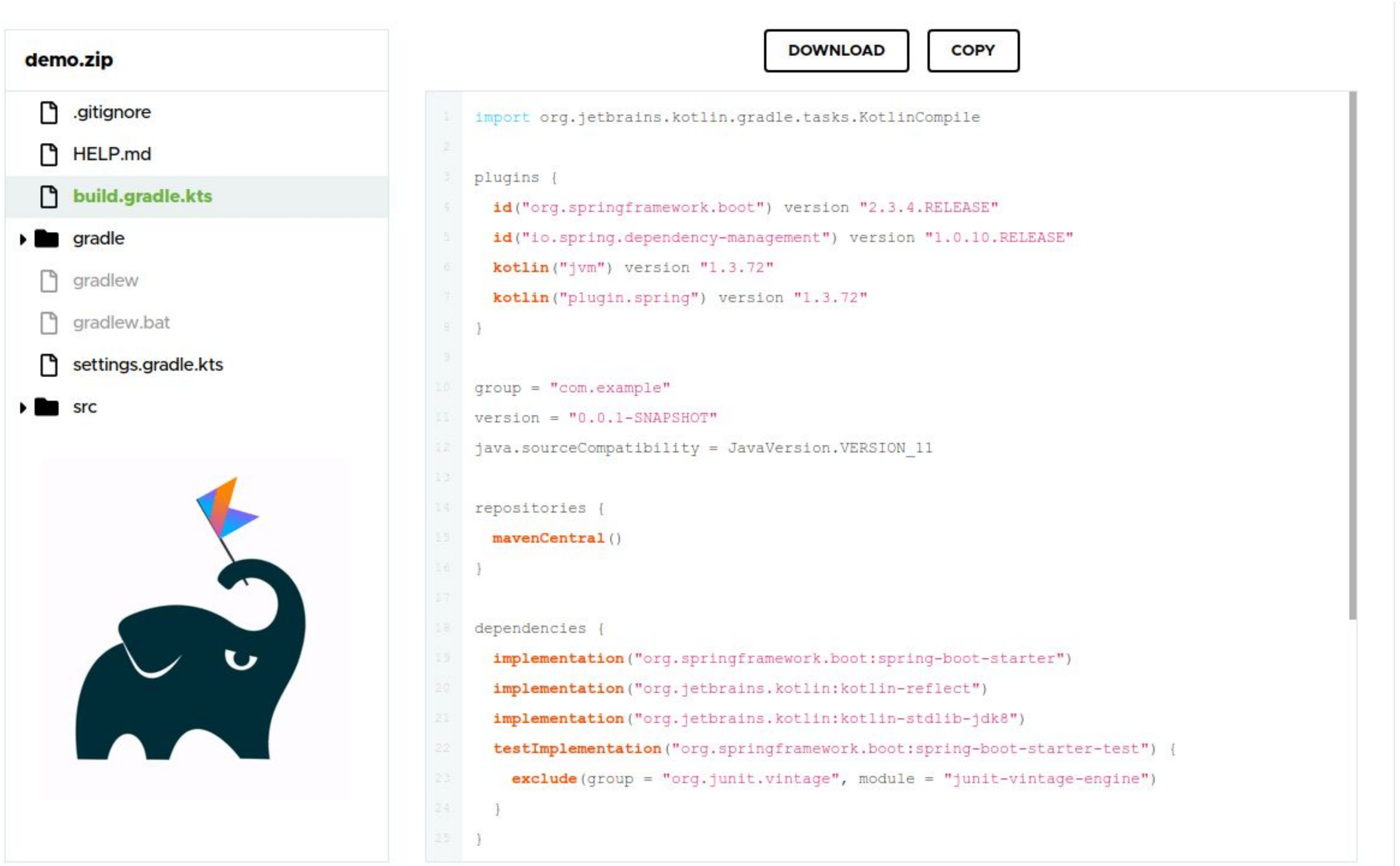
No dependency selected

Minimal Spring Boot Kotlin application



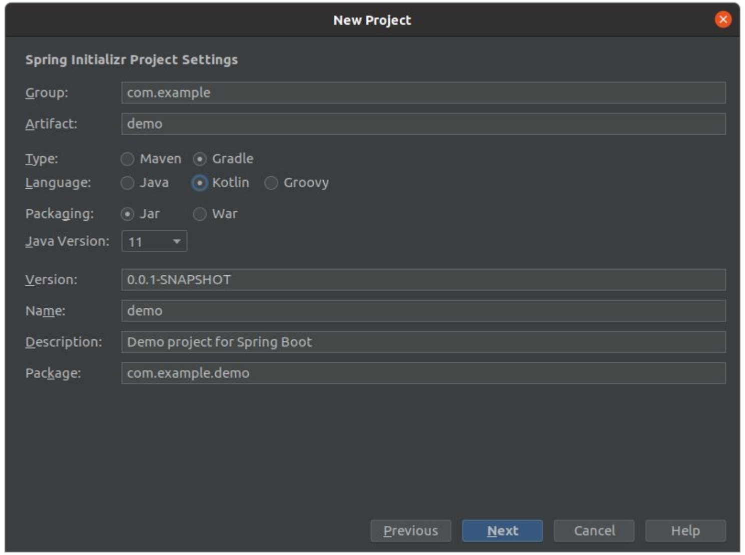
```kotlin
package com.example.demo

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication

@SpringBootApplication
class DemoApplication

fun main(args: Array<String>) {
  runApplication<DemoApplication>(*args)
}
```

Gradle Kotlin DSL



```
import org.jetbrains.kotlin.gradle.tasks.KotlinCompile

plugins {
  id("org.springframework.boot") version "2.3.4.RELEASE"
  id("io.spring.dependency-management") version "1.0.10.RELEASE"
  kotlin("jvm") version "1.3.72"
  kotlin("plugin.spring") version "1.3.72"
}

group = "com.example"
version = "0.0.1-SNAPSHOT"
java.sourceCompatibility = JavaVersion.VERSION_11

repositories {
  mavenCentral()
}

dependencies {
  implementation("org.springframework.boot:spring-boot-starter")
  implementation("org.jetbrains.kotlin:kotlin-reflect")
  implementation("org.jetbrains.kotlin:kotlin-stdlib-jdk8")
  testImplementation("org.springframework.boot:spring-boot-starter-test") {
    exclude(group = "org.junit.vintage", module = "junit-vintage-engine")
  }
}
```

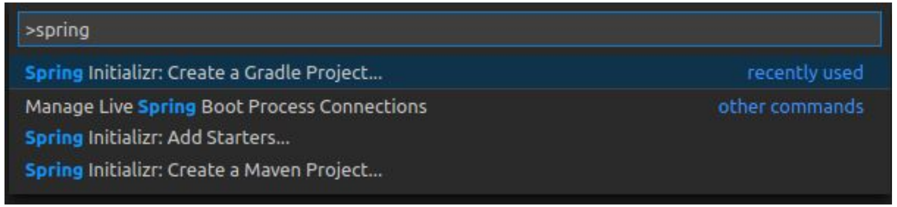
Also available in IDEs



Spring Initializr Project Settings

Group: com.example
Artifact: demo
Type: Maven [✓ Gradle]
Language: Java [✓ Kotlin] Groovy
Packaging: [✓ Jar] War
Java Version: 11
Version: 0.0.1-SNAPSHOT
Name: demo
Description: Demo project for Spring Boot
Package: com.example.demo

IntelliJ IDEA Ultimate

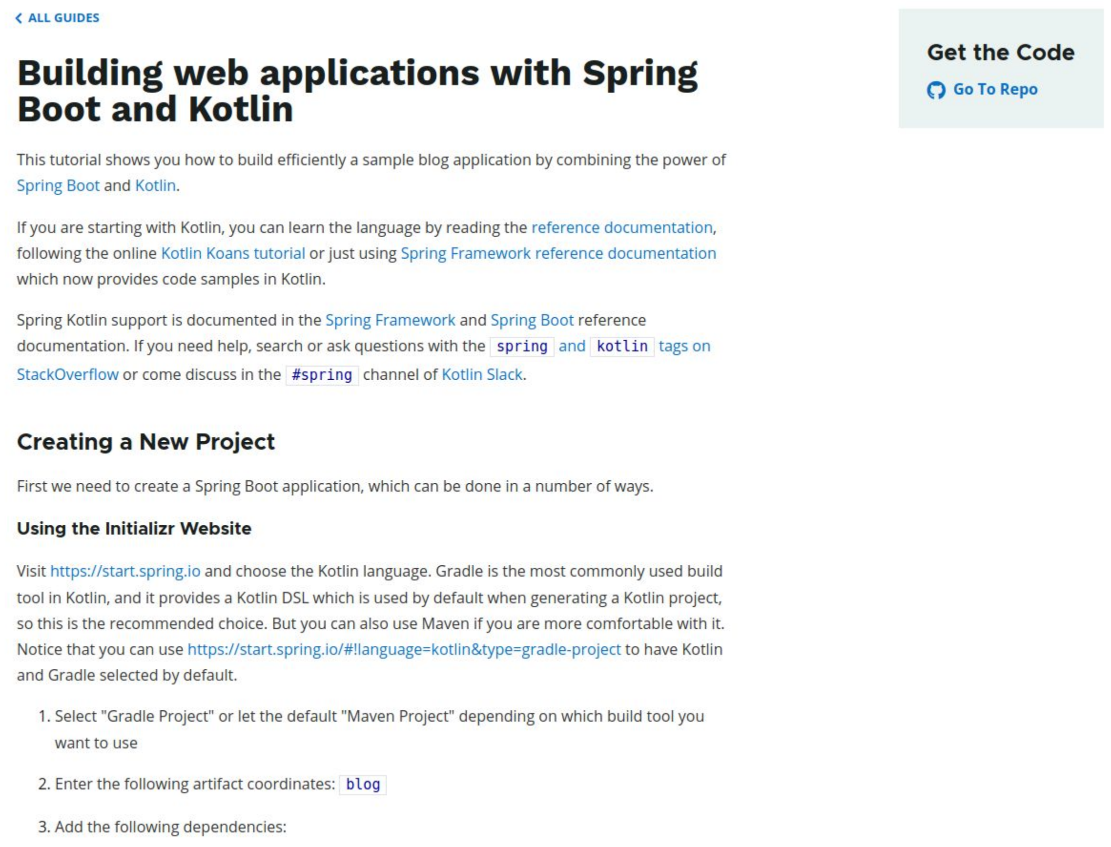


>spring

Spring Initializr: Create a Gradle Project... recently used
Manage Live Spring Boot Process Connections other commands
Spring Initializr: Add Starters...
Spring Initializr: Create a Maven Project...

VS code

Follow the tutorial on https://spring.io/guides



‹ ALL GUIDES

Building web applications with Spring Boot and Kotlin

This tutorial shows you how to build efficiently a sample blog application by combining the power of Spring Boot and Kotlin.

If you are starting with Kotlin, you can learn the language by reading the reference documentation, following the online Kotlin Koans tutorial or just using Spring Framework reference documentation which now provides code samples in Kotlin.

Spring Kotlin support is documented in the Spring Framework and Spring Boot reference documentation. If you need help, search or ask questions with the `spring` and `kotlin` tags on StackOverflow or come discuss in the `#spring` channel of Kotlin Slack.

Creating a New Project

First we need to create a Spring Boot application, which can be done in a number of ways.

Using the Initializr Website

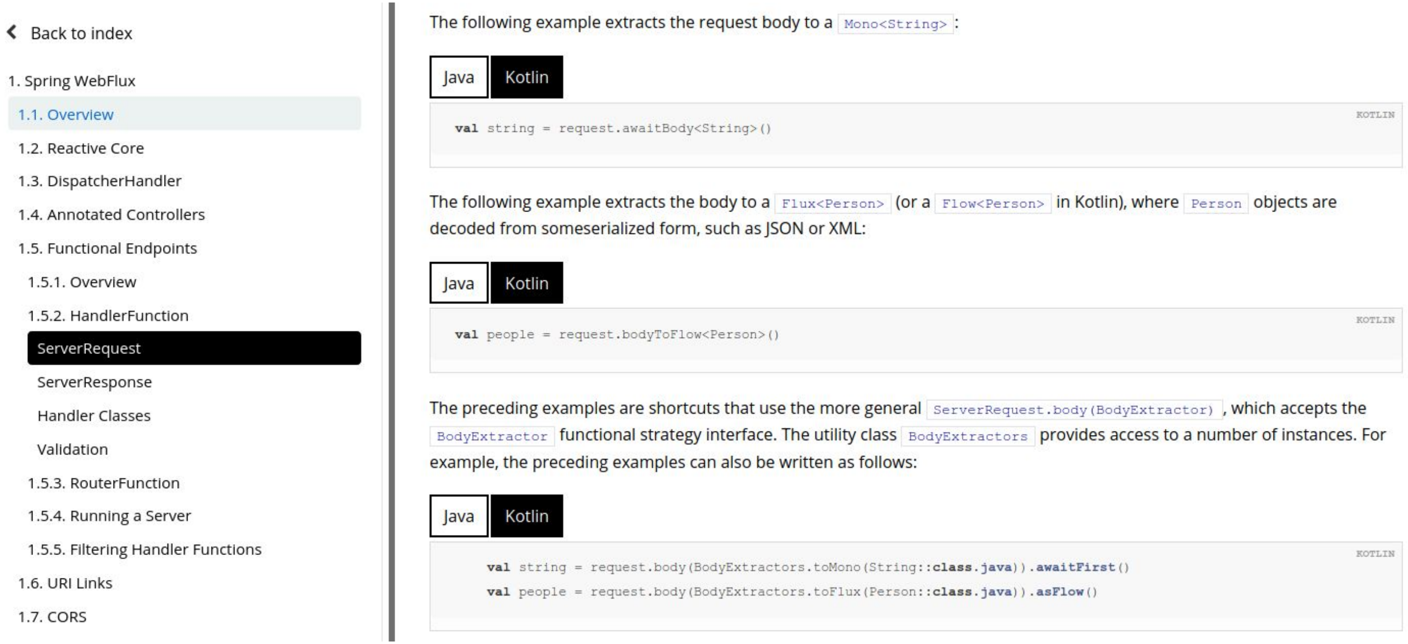
Visit https://start.spring.io and choose the Kotlin language. Gradle is the most commonly used build tool in Kotlin, and it provides a Kotlin DSL which is used by default when generating a Kotlin project, so this is the recommended choice. But you can also use Maven if you are more comfortable with it. Notice that you can use https://start.spring.io/#!language=kotlin&type=gradle-project to have Kotlin and Gradle selected by default.

1. Select "Gradle Project" or let the default "Maven Project" depending on which build tool you want to use
2. Enter the following artifact coordinates: `blog`
3. Add the following dependencies:

Get the Code

Go To Repo

Spring Framework documentation in Kotlin



The following example extracts the request body to a `Mono<String>`:

Java | Kotlin

```kotlin
val string = request.awaitBody<String>()
```

The following example extracts the body to a `Flux<Person>` (or a `Flow<Person>` in Kotlin), where `Person` objects are decoded from someserialized form, such as JSON or XML:

Java | Kotlin

```kotlin
val people = request.bodyToFlow<Person>()
```

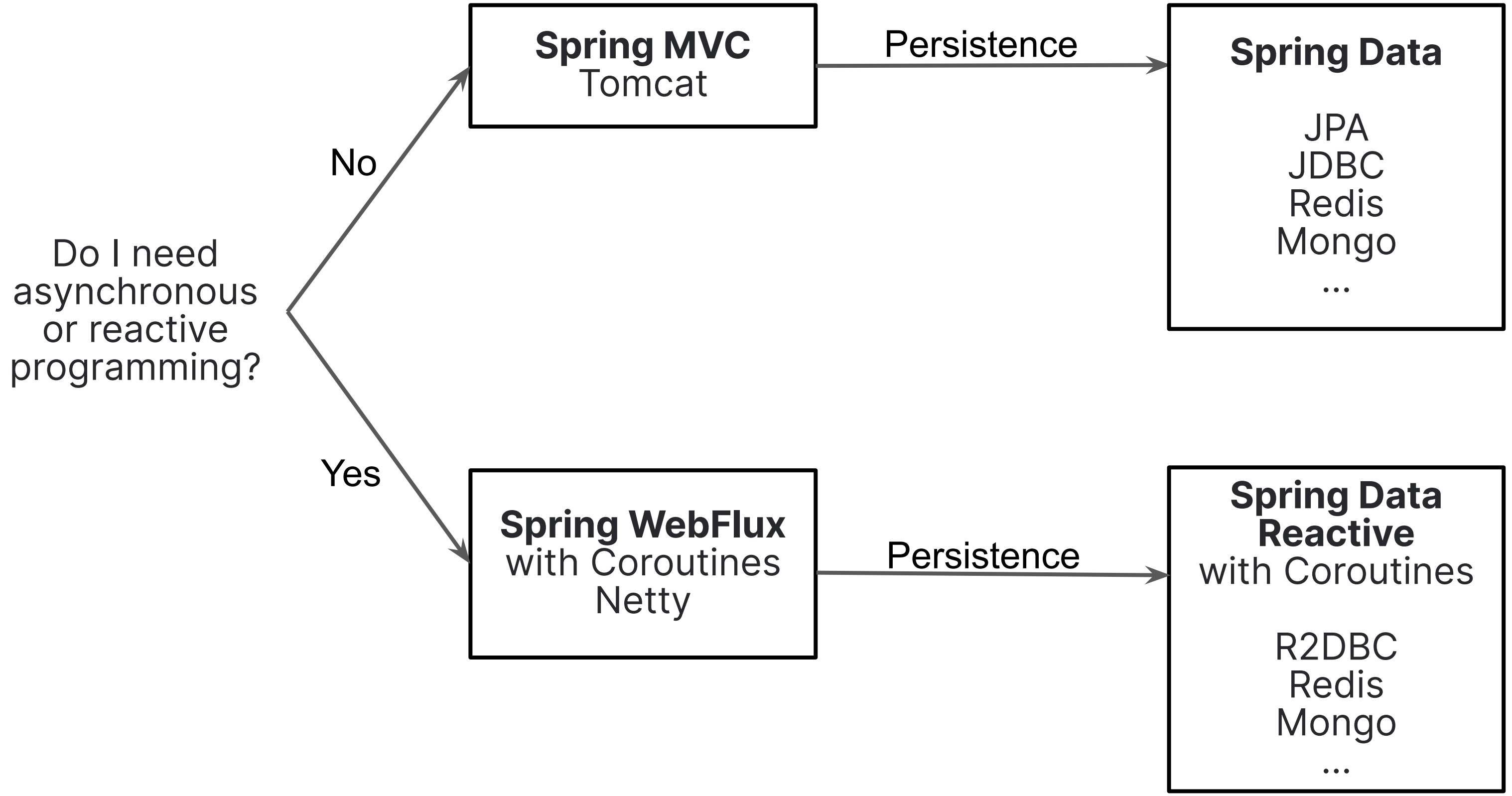
The preceding examples are shortcuts that use the more general `ServerRequest.body(BodyExtractor)`, which accepts the `BodyExtractor` functional strategy interface. The utility class `BodyExtractors` provides access to a number of instances. For example, the preceding examples can also be written as follows:

Java | Kotlin

```kotlin
    val string = request.body(BodyExtractors.toMono(String::class.java)).awaitFirst()
    val people = request.body(BodyExtractors.toFlux(Person::class.java)).asFlow()
```

Choose your style

Choose your web server stack



Choose your programming model

Do you prefer annotations?

```kotlin
@RestController
@RequestMapping("/api/article")
class ArticleController(private val repository: ArticleRepository) {

  @GetMapping("/")
  fun findAll() = repository.findAllByOrderByAddedAtDesc()

  @GetMapping("/{slug}")
  fun findOne(@PathVariable slug: String) =
        repository.findBySlug(slug) ?:
            throw ResponseStatusException(NOT_FOUND)

}
```

Or functional APIs?

```
@Bean
fun route(repository: ArticleRepository) = router {
   "/api/article".nest {
       GET("/") {
           ok().body(repository.findAllByOrderByAddedAtDesc())
       }
       GET("/{slug}") {
           val slug = it.pathVariable("slug")
           val article = repository.findBySlug(slug) ?:
               throw ResponseStatusException(NOT_FOUND)
           ok().body(article)
       }
   }
}
```

Spring supports both,
so up to you.

Coroutines

Allow to go reactive with a great trade-off between imperative and functional programming.

Coroutines are the default way to go reactive in Spring with Kotlin.

First class Coroutines support

- Spring WebFlux
- Spring MVC (new in Spring Boot 2.4)
- Spring Data Reactive
- Spring Messaging (RSocket)
- Spring Vault

Suspending functions
Spring MVC and WebFlux

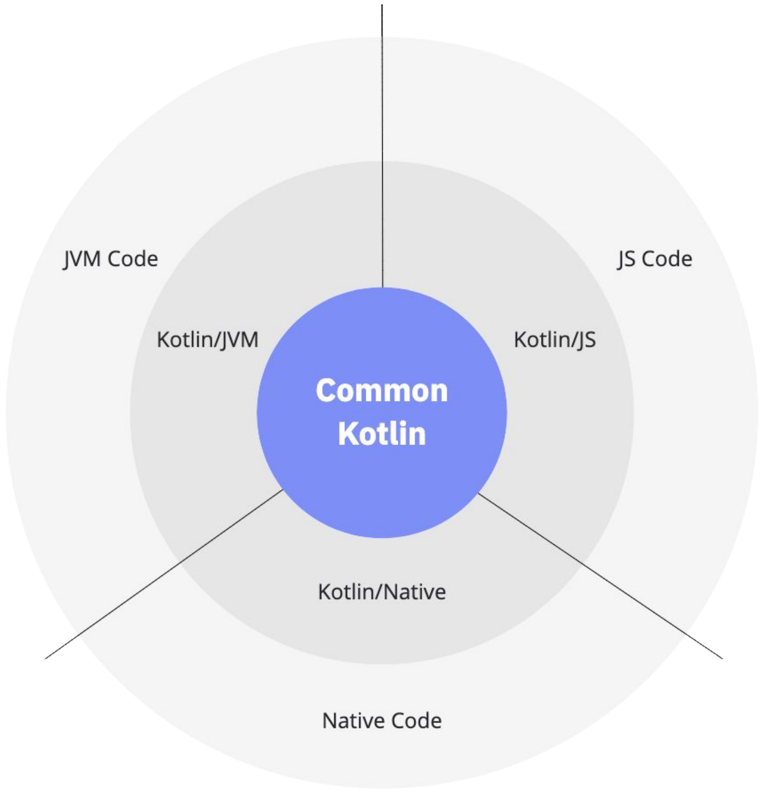
```kotlin
@GetMapping("/api/banner")
suspend fun suspendingEndpoint(): Banner {
  delay(10)
  return Banner("title", "Lorem ipsum")
}
```

Flow
Spring MVC and WebFlux

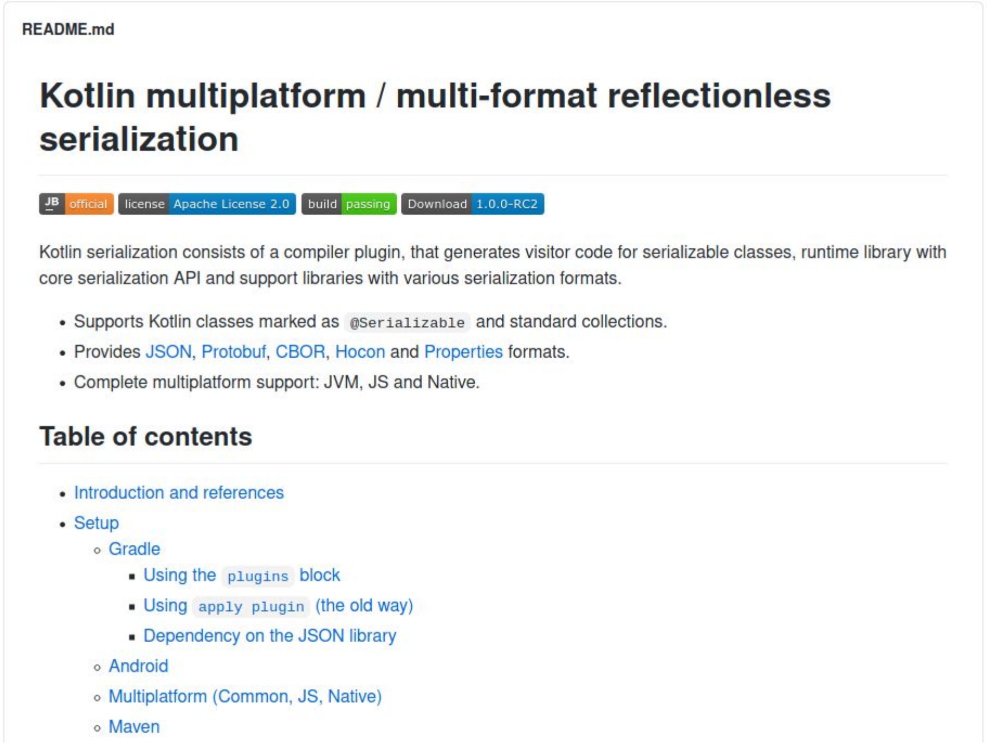
```
@GetMapping("/banners")
suspend fun flow(): Flow<Banner> = client.get()
        .uri("/messages")
        .accept(MediaType.TEXT_EVENT_STREAM)
        .retrieve()
        .bodyToFlow<String>()
        .map { Banner("title", it) }
```

Multiplatform

Multiplatform



kotlinx.serialization



README.md

Kotlin multiplatform / multi-format reflectionless serialization

JB official | license Apache License 2.0 | build passing | Download 1.0.0-RC2

Kotlin serialization consists of a compiler plugin, that generates visitor code for serializable classes, runtime library with core serialization API and support libraries with various serialization formats.

- Supports Kotlin classes marked as `@Serializable` and standard collections.
- Provides JSON, Protobuf, CBOR, Hocon and Properties formats.
- Complete multiplatform support: JVM, JS and Native.

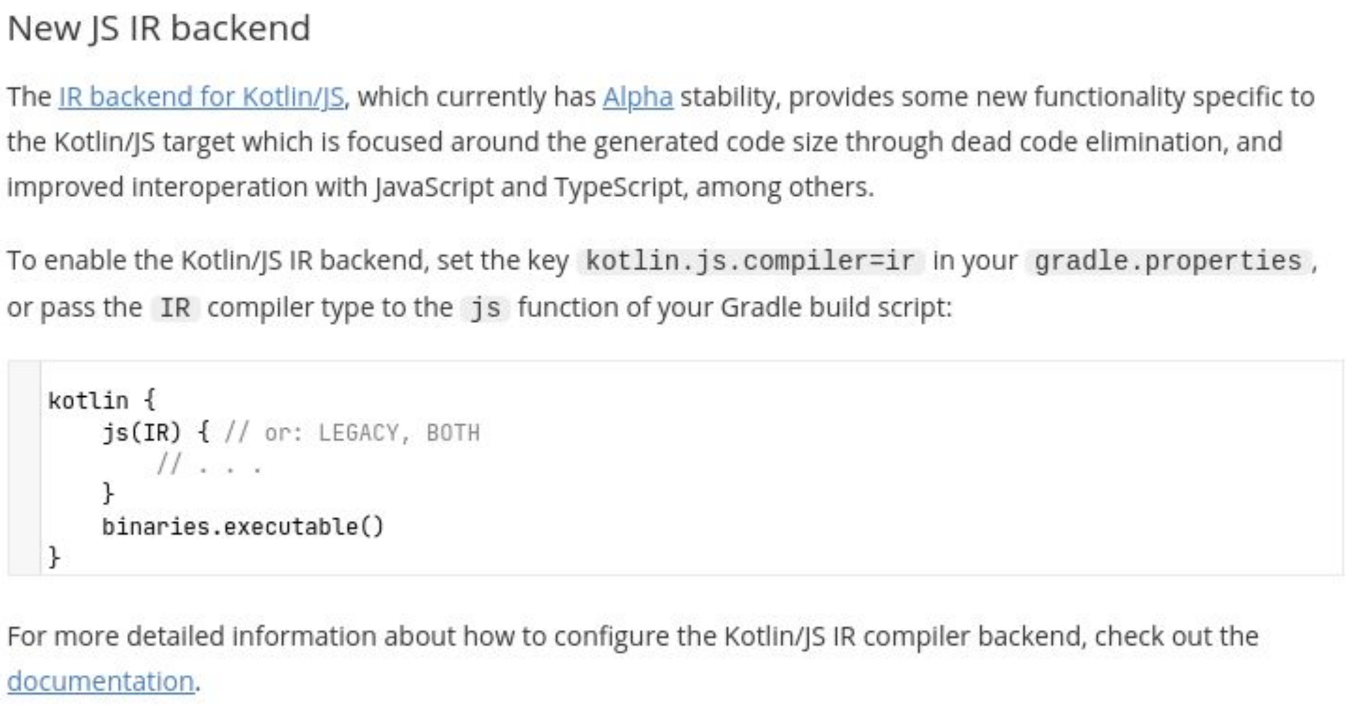
Table of contents

kotlinx.serialization support
New in Spring Boot 2.4

- More lightweight than Jackson
- Designed for Kotlin
- Multiplatform serialization
- Allows same code for model and validation across server, frontend and mobile!

```
implementation("org.springframework.boot:spring-boot-starter-web") {
   exclude(module = "spring-boot-starter-json")
}
implementation("org.jetbrains.kotlinx:kotlinx-serialization-json:1.0.0")
```

Kotlin/JS



New JS IR backend

The IR backend for Kotlin/JS, which currently has Alpha stability, provides some new functionality specific to the Kotlin/JS target which is focused around the generated code size through dead code elimination, and improved interoperation with JavaScript and TypeScript, among others.

To enable the Kotlin/JS IR backend, set the key `kotlin.js.compiler=ir` in your `gradle.properties`, or pass the `IR` compiler type to the `js` function of your Gradle build script:

```
kotlin {
    js(IR) { // or: LEGACY, BOTH
        // . . .
    }
    binaries.executable()
}
```

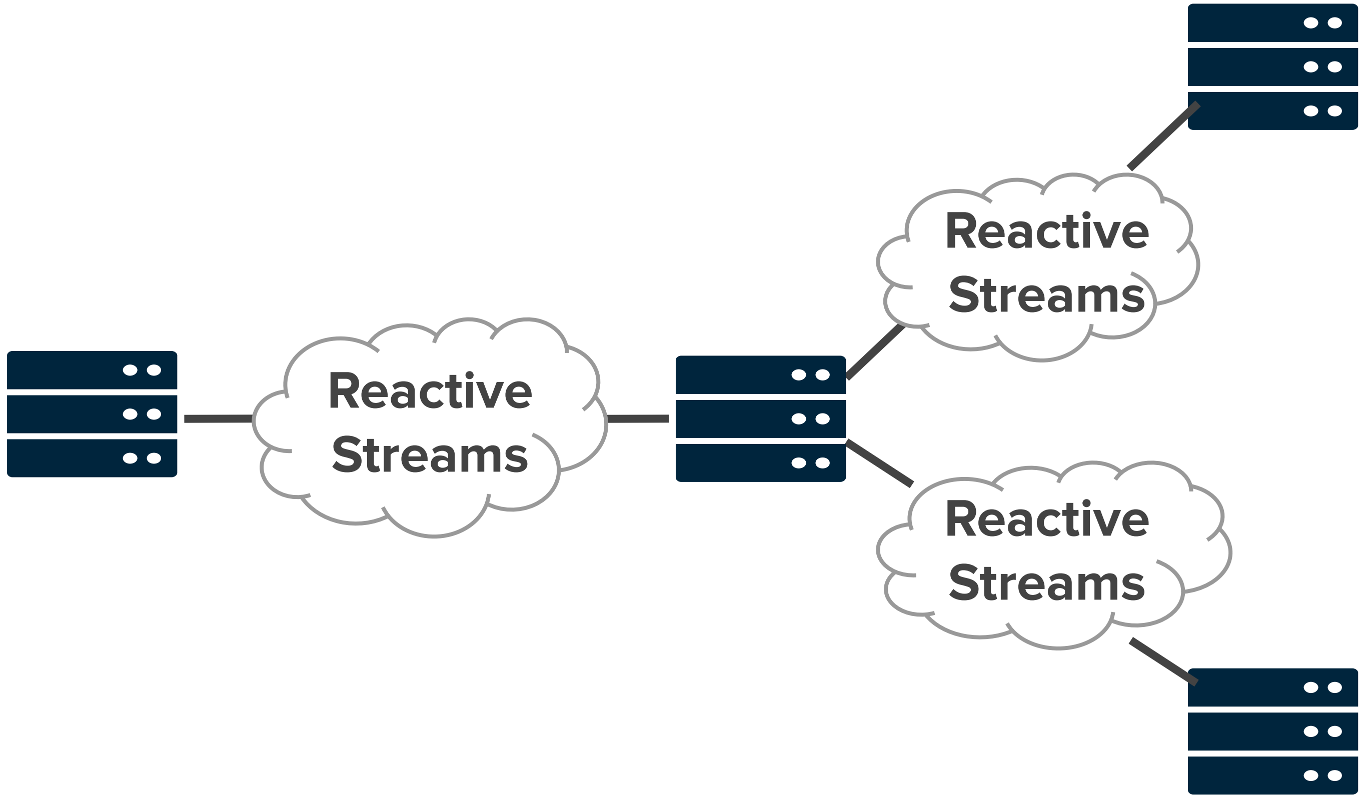
For more detailed information about how to configure the Kotlin/JS IR compiler backend, check out the documentation.

Kotlin/WASM has a huge potential

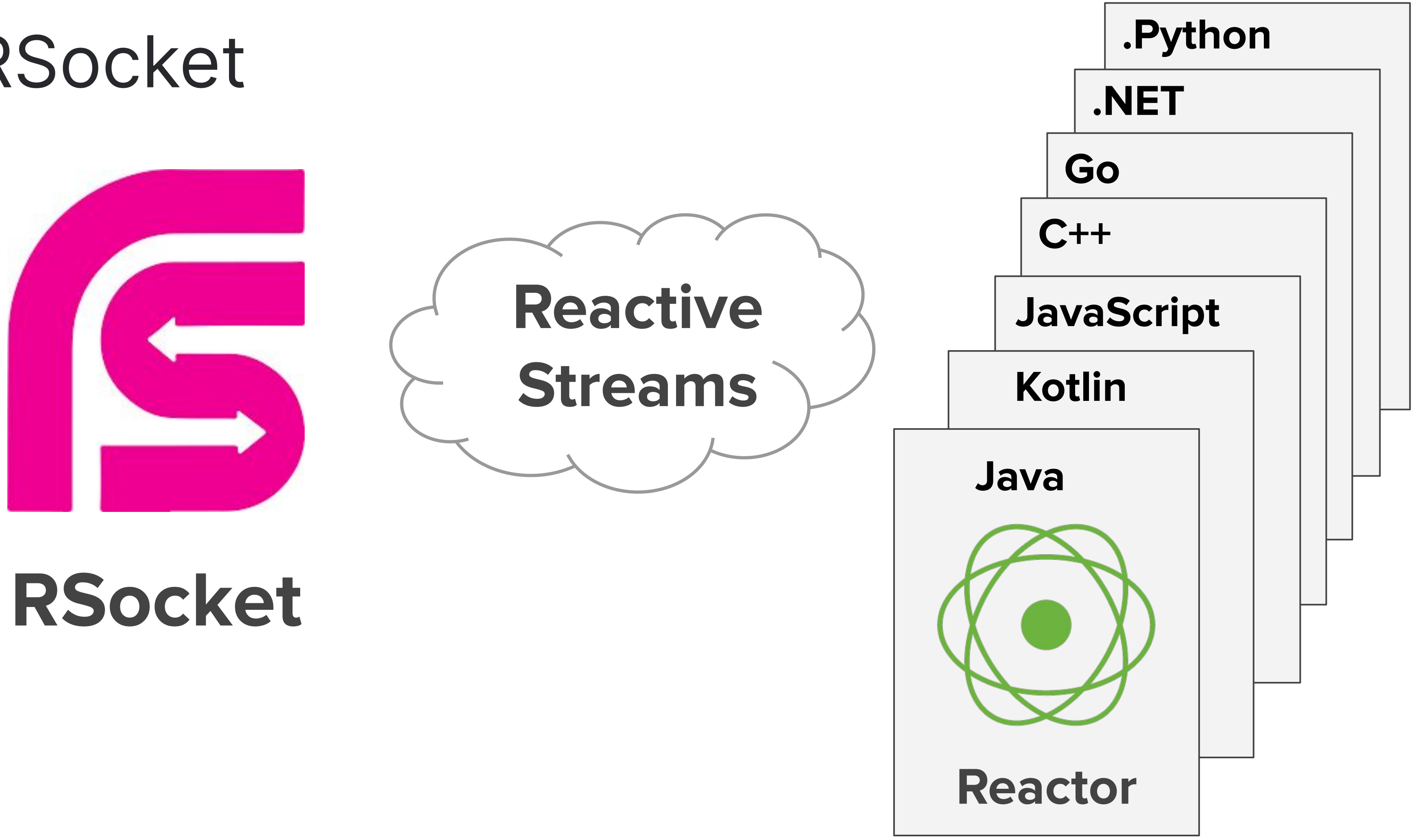
RSocket

RSocket

RSocket



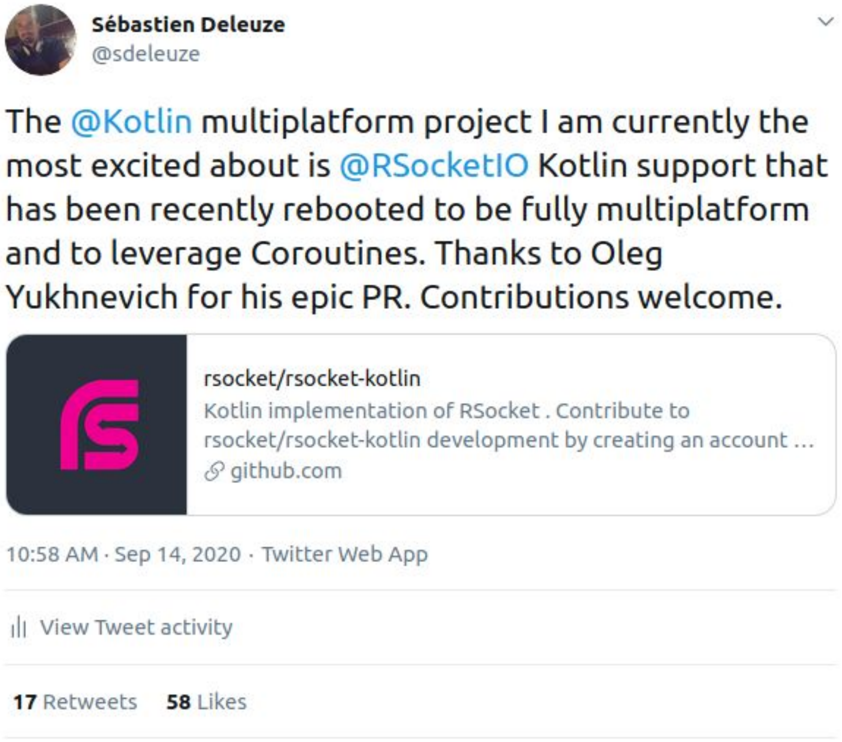
RSocket


RSocket
Reactive Streams
.Python
.NET
Go
C++
JavaScript
Kotlin
Java
Reactor

RSocket support in Spring messaging

```kotlin
class MessageHandler(private val builder: RSocketRequester.Builder) {

  // ...

  suspend fun stream(request: ServerRequest): ServerResponse {
     val requester = builder
          .dataMimeType(APPLICATION_CBOR)
          .connectTcpAndAwait("localhost", 9898)
     val replies = requester
          .route("bot.messages")
          .dataWithType(processor)
          .retrieveFlow<Message>()
     val broadcast = requester.route("bot.broadcast").retrieveFlow<Message>()
     val messages = flowOf(replies, processor.asFlow(), broadcast).flattenMerge()
     return ok().sse().bodyAndAwait(messages)
  }
}
```

rsocket-kotlin



Sébastien Deleuze
@sdeleuze

The @Kotlin multiplatform project I am currently the most excited about is @RSocketIO Kotlin support that has been recently rebooted to be fully multiplatform and to leverage Coroutines. Thanks to Oleg Yukhnevich for his epic PR. Contributions welcome.

rsocket/rsocket-kotlin
Kotlin implementation of RSocket . Contribute to rsocket/rsocket-kotlin development by creating an account …
github.com

10:58 AM · Sep 14, 2020 · Twitter Web App

View Tweet activity

17 Retweets **58** Likes

Other key points

100% of Spring Framework API
with null-safety annotations
→ no NPE for Spring applications
written in Kotlin

@ConfigurationProperties data classes

```kotlin
@ConstructorBinding
@ConfigurationProperties("blog")
data class BlogProperties(val title: String, val banner: Banner) {
  data class Banner(val title: String? = null, val content: String)
}
```

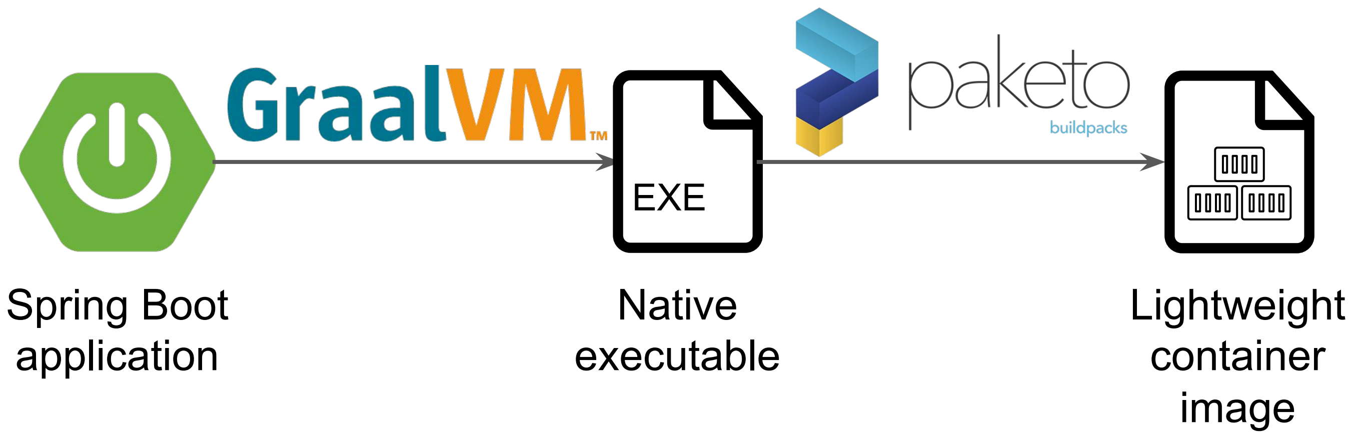
Spring Security Kotlin DSL

New in Spring Security 5.4

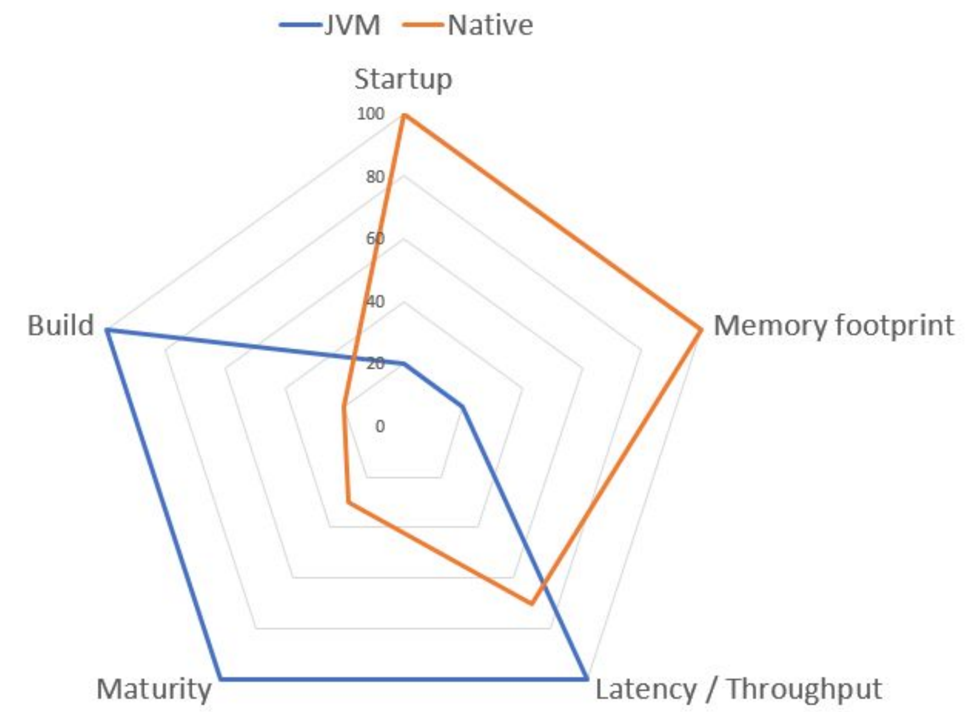
```
override fun configure(http: HttpSecurity) {
  http {
     authorizeRequests {
        authorize("/css/**", permitAll)
        authorize("/user/**", hasAuthority("ROLE_USER"))
     }
     formLogin {
        loginPage = "/log-in"
     }
  }
}
```

Spring Boot native applications

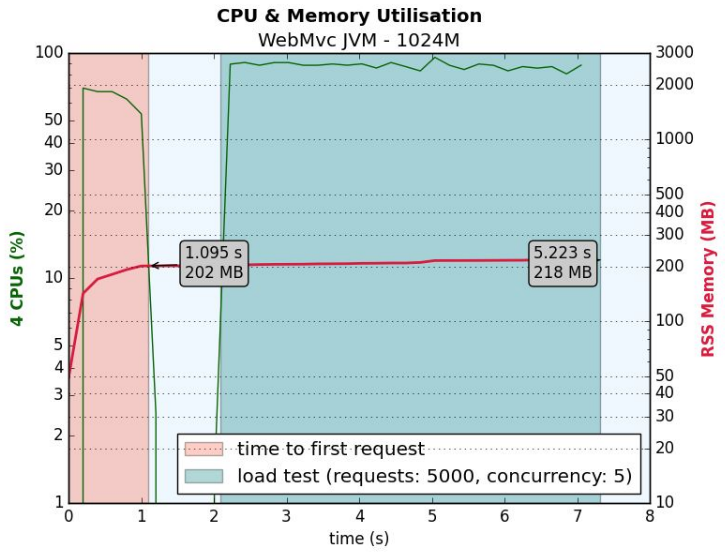
With GraalVM native


GraalVM
paketo buildpacks
EXE
Spring Boot application
Native executable
Lightweight container image

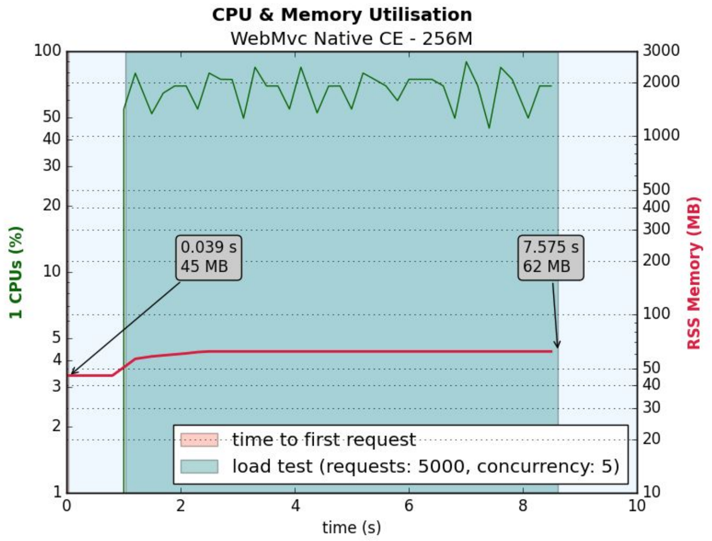
JVM and native executables offer different trade-offs



Instant startup and cheaper hosting

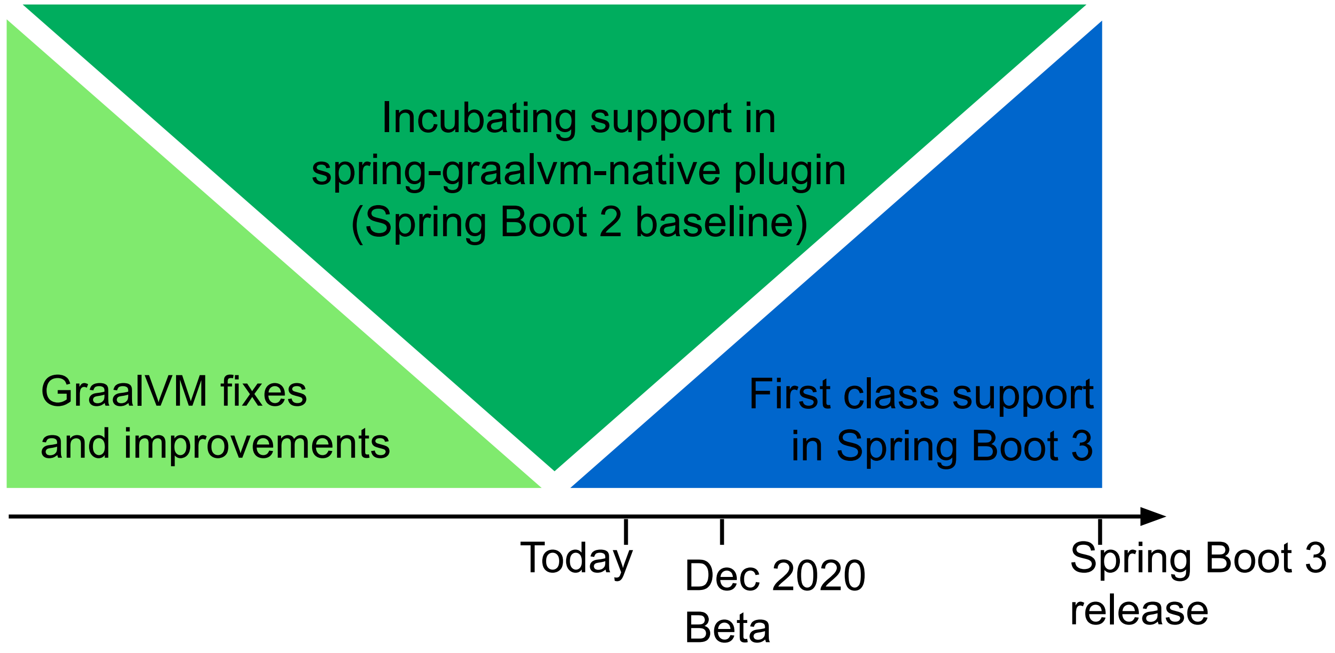


Spring Boot on JVM,
4 vCPU, 1G RAM



Spring Boot on Native,
1 vCPU, 256M RAM

Spring support for native executables



Demo

What’s next?

Programmatic configuration for Spring Boot using a Kotlin DSL

Spring Fu is an incubator for a functional flavor of Spring Boot

KoFu

JaFu

What is the same than Spring Boot?

- https://start.spring.io
- Based on Spring Boot infrastructure
- Spring configuration for the JVM ecosystem
- Dependency management
- Starters
- Actuators
- Standalone executable JAR or container deployment

What changes?

Spring Boot regular flavor	Spring Fu flavor
Conventions and automatic configuration	Explicit declaration
Annotations-based configuration	Functional configuration
Reflection-based infrastructure	Lambda-based infrastructure
Production ready	Incubating

Spring Boot configured with KoFu

```
val app = webApplication {
  beans {
     bean<SampleService>()
     bean<SampleHandler>()
  }
  webMvc {
     port = if (profiles.contains("test")) 8181 else 8080
     router {
        val handler = ref<SampleHandler>()
        GET("/", handler::hello)
        GET("/api", handler::json)
     }
     converters {
        string()
        jackson {
           indentOutput = true
        }
     }
  }
}

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

Links

https://start.spring.io
https://spring.io/guides/tutorials/spring-boot-kotlin/
https://github.com/spring-projects-experimental/spring-graalvm-native
https://github.com/spring-projects-experimental/spring-fu

Thanks!
Have a nice Kotlin!

@sdeleuze