

O estado atual do Kotlin no backend

Víctor Orozco - Nabenik

1 de dezembro de 2021

@tuxtor



NABENIK

Kotlin

- Linguagem (Kotlin)
- OpenJDK (Java Virtual Machine)
- Bibliotecas/API (Java Classpath)
- kotlin-stdlib





Paradigmas de desenvolvimento



Paradigma

- Thread
- Async

Abrangencia do toolkit

- DIY
- Micro
- Pilhas inclusas



Concorrência com threads

Thread based -e.g. EJB, CDI, JAX-RS-. Uma conexão http = 1 thread

```
@Inject
SayService say;

@GET
@Produces(MediaType.TEXT_PLAIN)
public String hello() {
    return say.hello();
}
```

JakartaEE/MicroProfile, Spring

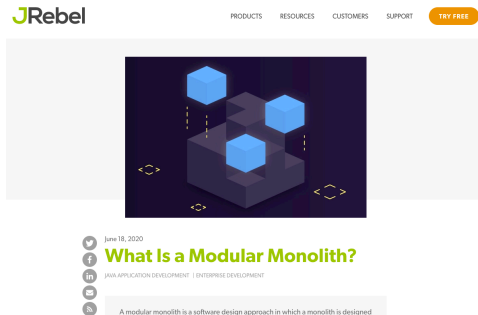
Baseado em Async

Reactive based -e.g. Async JAX-RS, Spring WebFlux, Actors, Verticles-

```
public class Server extends AbstractVerticle {  
    public void start() {  
        vertx.createHttpServer().requestHandler(req -> {  
            req.response()  
                .putHeader("content-type", "text/plain")  
                .end("Hello from Vert.x!");  
        }).listen(8080);  
    }  
}
```

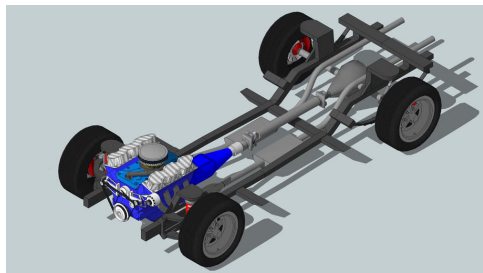
Vert.x, Spring WebFlux, Zio, Akka

- Monólito
- Microserviço
- Micromonolito



"Simples/DIY"

- Servlet based -e.g. Tomcat, Jetty-
- Custom I/O -e.g. Undertow, Netty, Vert.x-



"Micro"

- Custom micro -e.g. Jooby, Spark, Javalin, Helidon SE-
- MicroProfile based -e.g. Quarkus, Helidon-
- Micronaut
- Spring Boot



"Complexo"

- Java EE/Jakarta EE -e.g. JBoss, WebSphere Liberty-
- Spring
- Akka



Arquiteto

A pessoa que tem que decidir entre pegar um framework pronto *complexo* ou pegar um runtime *ligero* e criar a estrutura toda -i.e Bibliotecas, estilo arquitetural, SCM (Maven)-.

A pessoa que tem que decidir entre chatear os dev Java tradicionais no modelo de desenvolvimento Async ou chatear os desenvolvedores Kotlin por não aproveitar os Coroutines.

A pessoa que tem que avaliar o pulo para o Kotlin conservando (ou não) os stacks tradicionais.

Arquiteto

A pessoa que vai ser odiada por não dar o pulo para JavaScript só pela vontade de re-escrever a base de código que vem fazendo sucesso há 15 anos

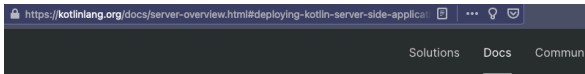


Kotlin no backend



Fato #1

Todo framework Java pode virar framework Kotlin. Mas nem todo framework Java é testado no Kotlin.



Frameworks for server-side development with Kotlin

- **Spring** makes use of Kotlin's language features to offer more concise APIs, starting with version 5.0. The online project generator allows you to quickly generate a new project in Kotlin.
- **Vert.x** is a framework for building reactive Web applications on the JVM, offers dedicated support for Kotlin, including full documentation.
- Ktor is a framework built by JetBrains for creating Web applications in Kotlin, making use of coroutines for high scalability and offering an easy-to-use and idiomatic API.
- kotlinx.html is a DSL that can be used to build HTML in a Web application. It serves as an alternative to traditional templating systems such as JSP and FreeMarker.
- **Micronaut** is a modern, JVM-based, full-stack framework for building modular, easily testable microservice and serverless applications. It comes with a lot of built-in, handy features.
- http4k is the functional toolkit with a tiny footprint for Kotlin HTTP applications, written in pure Kotlin. The library is based on the "Your Server as a Function" paper from Twitter and represents modeling both HTTP Servers and Clients as simple Kotlin functions that can be composed together.
- **Javalin** is a very lightweight web framework for Kotlin and Java which supports WebSockets, HTTP2 and async requests.

Fato #2

Geralmente o problema de não testar são os annotation processors -e.g. DeltaSpike-, muitos são feitos para o Java

Fato #3

O ecossistema Kotlin esta criando Frameworks Kotlin-first/Kotlin-exclusive

Reactive based - DIY

```
fun HelloWorld(): HttpHandler {  
    return routes("/") bind GET to { Response(OK).body("hello world!") }  
}
```

Http4k

Reactive based - Micro

```
fun main(args: Array<String>) {  
    embeddedServer(Netty, 8080) {  
        routing {  
            get("/") {  
                call.respondText("Hello, world!", ContentType.Text.Html)  
            }  
        }  
    }.start(wait = true)  
}
```

Ktor

- Spring Boot, Micronaut, MicroProfile, GraalVM . . .
- Raw performance (Beam, Spark, Hadoop)
- Tooling - IDE, Maven, Drivers RDBMS
- JVM - (Twitter, Alibaba, Spotify, etc.)
- OpenJDK





Projeto tradicional com Kotlin



Projeto Java so que não

1. Maven
2. Dependencias (MicroProfile, Jakarta EE, Arquillian, JUnit, . . .)
3. Maven plugin (maven-compiler-plugin)
4. Kotlin plugin (kotlin-maven-plugin)

Kotlin com Maven - Dependency

```
<dependency>  
  <groupId>org.jetbrains.kotlin</groupId>  
  <artifactId>kotlin-stdlib-jdk8</artifactId>  
  <version>${kotlin.version}</version>  
</dependency>
```

Kotlin com Maven - maven-compiler-plugin

```
<execution>
  <id>default-compile</id>
  <phase>none</phase>
</execution>
<execution>
  <id>default-testCompile</id>
  <phase>none</phase>
</execution>
<execution>
  <id>java-compile</id>
  <phase>compile</phase>
  <goals> <goal>compile</goal> </goals>
</execution>
<execution>
  <id>java-test-compile</id>
  <phase>test-compile</phase>
  <goals> <goal>testCompile</goal> </goals>
</execution>
```



```
<compilerPlugins>
<plugin>all-open</plugin>
</compilerPlugins>
...
<option>all-open:annotation=javax.ws.rs.Path</option>
<option>all-open:annotation=javax.enterprise.context.RequestScoped</option>
<option>all-open:annotation=javax.enterprise.context.SessionScoped</option>
<option>all-open:annotation=javax.enterprise.context.ApplicationScoped</option>
<option>all-open:annotation=javax.enterprise.context.Dependent</option>
<option>all-open:annotation=javax.ejb.Singleton</option>
<option>all-open:annotation=javax.ejb.Stateful</option>
<option>all-open:annotation=javax.ejb.Stateless</option>
```

Ideia geral: As anotações Java viram open classes por causa do proxy-classes metapadrão



- vorozco@nabenik.com
- @tuxtor
- <https://voroeco.com>



This work is licensed under a
Creative Commons
Attribution-ShareAlike 3.0.