O estado atual do Kotlin no backend

Víctor Orozco - Nabenik

1 de dezembro de 2021

@tuxtor

Kotlin

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

Paradigmas de desenvolvimento

Desenvolvimento na JVM

Paradigma

- Thread
- Async

Abragencia 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

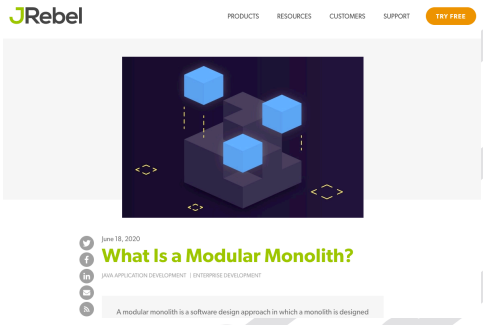
Reactive based -e.g. Async JAX-RS, Spring WebFlux, Actors, Verticles-

```
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

Arquitetura

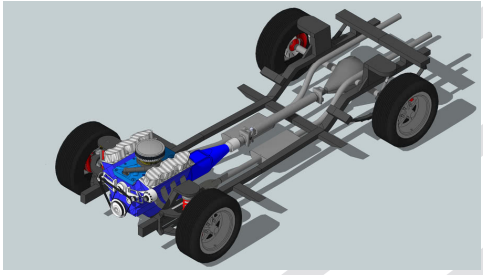
- Monólito
- Microserviço
- Micromonolito



Runtime

"Simples/DIY"

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



Runtime/Framework

"Micro"

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

Complexo

"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 no ano 2021

Arquiteto

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

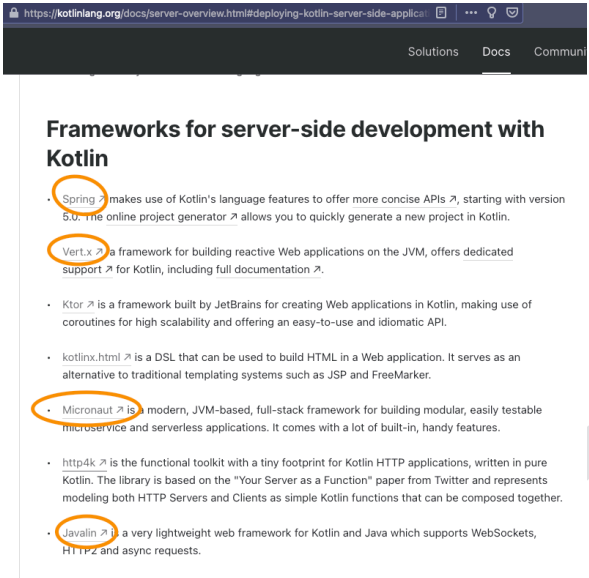
Kotlin no backend

Kotlin no backend - Fatos

Fato #1

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

Kotlin no backend - Fatos



Fato #2

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

Fato #3

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

Kotlin no backend - Fatos

Reactive based - DIY

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

Http4k

Kotlin no backend - Fatos

Reactive based - Micro

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

Ktor

Java - Morrendo desde 1995

- 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>
```

Kotlin com Maven - kotlin-maven-plugin

```
<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

Víctor Orozco

- vorozco@nabenik.com
- @tuxtor
- https://vorozco.com