Introducción a Kotlin para desarrolladores Java

Víctor Orozco

@tuxtor

22 de mayo de 2020

ACADEMIK

Contexto

Caracteristicas interesantes

Demo productiva

Kotlin vs. Java

ACADEMIK

Contexto

¿Java?

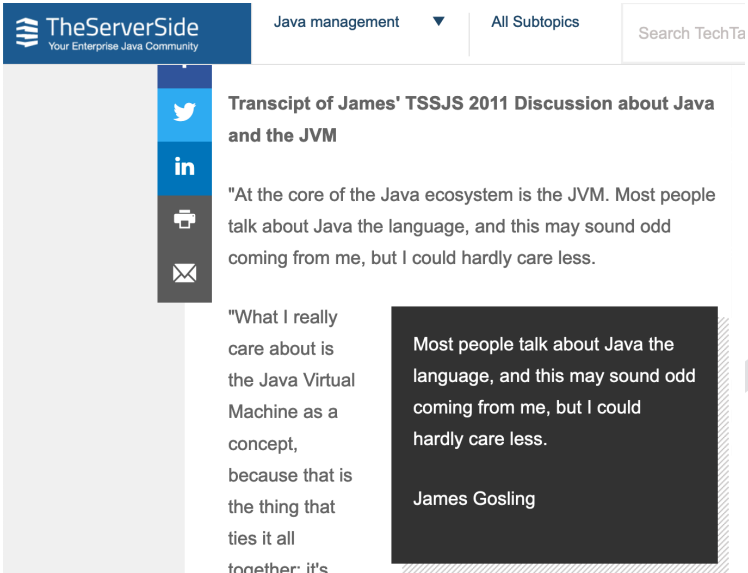
- Lenguaje (Java 11)
- OpenJDK (Java Virtual Machine)
- Bibliotecas/API (Java Classpath)

El conjunto de los 3 es la plataforma Java(TM) pero pueden usarse de forma independiente

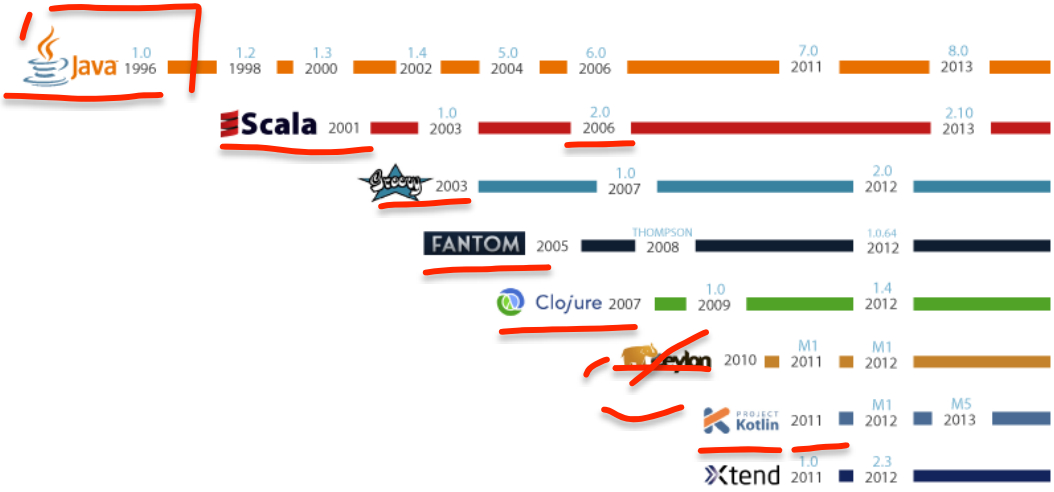
Java en Android

- Lenguaje (Java 7)
- ART/Dalvik
- Bibliotecas/API (Java + Google Classpath)

Java - Java como JVM



JVM



Kotlin

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

Kotlin en Android

- Lenguaje (Kotlin)
- ART/Dalvik
- Bibliotecas/API (Java+Google Classpath)
- kotlin-stdlib

Kotlin en JS

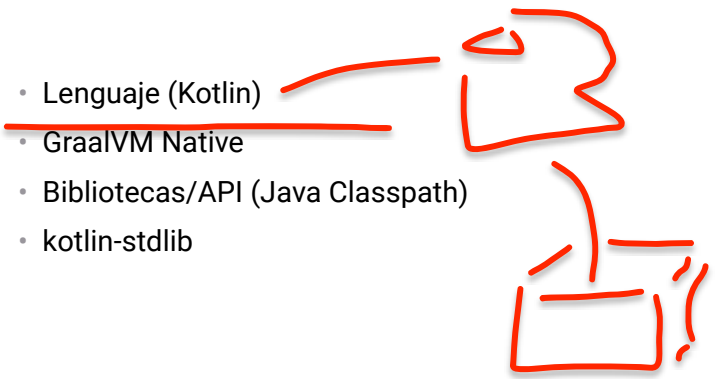
- Lenguaje (Kotlin)
- V8/SpiderMonkey
- Bibliotecas/API (ECMA 6 + Web)
- kotlin-stdlib

Kotlin Nativo

- Lenguaje (Kotlin)
- LLVM
- GLibc (Linux)
- kotlin-stdlib

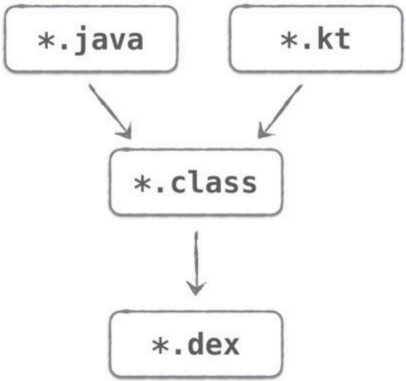
Kotlin Nativo/GraalVM

- Lenguaje (Kotlin)
- GraalVM Native
- Bibliotecas/API (Java Classpath)
- kotlin-stdlib



Kotlin como lenguaje

- Típado estático con inferencia de tipos
- OOP y funcional
- Funciones son ciudadanos de primer nivel
- Interoperable con Java
- Compilador genera Bytecode nivel Java 6 (Android)



ACADEMIK

Caracteristicas interesantes

Kotlin - Inferencia de tipos (constantes y variables)

Similar a lo visto en TypeScript o Swift, promueve la inmutabilidad

```
1  // Mutable
2  var answer = 42
3
4  // Inmutable
5  val phrase = "JVM Rocks!"
6
7  // Declaración explicita
8  val pi : Double = "3.14159"
9
10 // Inferencia por retorno
11 val auto = crearAuto()
```

Kotlin - Funciones

Pueden ser *top level, nested y a su vez pueden ser bloques o expresiones*

```
1  //Bloque
2  fun sumar(x: Int, y: Int): Int{
3      return x + y
4  }
5
6  //One-line - Expresion y default parameter
7  fun sumar2(x: Int, y: Int = 99) = x + y
8
9  //Infix - AKA operador
10 infix fun Int.sumar3(y: Int) = this + y
11 ...
12 //Uso
13 2 sumar3 4
```

Kotlin - Clases

Todas las clases heredan de *Any*

```
1  //Clase
2  class Automovil: Vehiculo{
3      ...
4      constructor(conductor:Persona): super(conductor){
5          ...
6      }
7  }
8
9  //Clase concisa
10 class Automovil(conductor:Persona): Vehiculo(conductor){
11     ...
12 }
```

Kotlin - Propiedades y clases

Combinación del campo y métodos de acceso

```
1  //Clase
2  class Automovil: Vehiculo{
3      var marca: String
4      var modelo: Int = 2020
5      var motor: String = ""
6        set(value) {
7              field = value + "CC"
8          }
9       get() = field + " extra info"
10 }
```

Kotlin - Clases

Kotlin permite escribir clases y data carriers de forma concisa

```
1  //Forma corta
2  class Automovil(val marca: String, val color: String="Rojo")
3  ...
4  //Data class (metodos universales como equals, hash code,
       toString)
5  data class Automovil(val marca: String,
6      val color: String="Rojo")
```

Kotlin - Object AKA Singleton

Creación de instancias únicas

```
1  object Automovil: Vehiculo {
2      override fun correr(){
3          ...
4      }
5  }
6
7  //Invocamos comportamiento
8  Automovil.correr()
9
10
11 //Y lo usamos como objeto
12 fun iniciarVehiculo(Automovil)
```

Kotlin - No static keyword

```
1  class Automovil {
2      companion object {
3          fun correr() {
4              ...
5          }
6      }
7  }
8
9  object Automovil {
10     override fun correr(){
11     ...
12     }
13 }
```

Kotlin - Extension functions

Posibilidad de extender funcionalidad en clases existentes

```
1  fun String.ultimoCar(): Char = this.get(this.lenght - 1)
2
3  val frase: Char = "Yo amo la JVM".ultimoCar()
```

Kotlin - Verificación de nulos

En Kotlin la verificación de nulos y declaración de variables .ªbiertas.ᵉˢ explicita

```
1  val talvez: String? = ...
2
3  talvez.length //Error de compilacion
4  talvez?.length
5
6  fun forzarNull(s: String?) {
7      println(s!!.length)
8  }
```

Kotlin - Smart Cast y Pattern Matching

El casting se da automático en ciertos bloques y expresiones

```kotlin
val auto1: Vehiculo = ...

if (auto1 is Automovil){
    auto1.cosasDeAutos()
}

when (auto1) {
    is Automovil -> auto1.cosasDeAutos()
    is Motocicleta -> auto1.cosasDemotos()
    else -> throw Exception("El vehiculo de los ojos tristes")
}
```

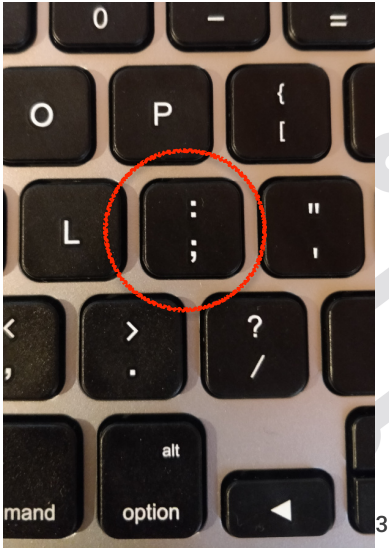
Kotlin - Collections

Metodos convenientes y expresiones lambda, por defecto Inmutables

```
1  //Con expresiones lambda
2  listOf(1, 2, 3).filter{ i -> i % 2 == 0}
3
4  //Con expresiones cortas (predicado)
5  listOf(1, 2, 3).filter{i % 2 == 0}
```

Kotlin - Convenciones

- Convenciones de nombrado Java
  - Típos en Uppercase
  - Metodos y propiedades en lower camelCase
  - Punto y coma son opcionales
  - Convención reversa en nombrado de paquetes
  - Multiples clases por archivo
  - Los paquetes en código no deben coincidir con nombres de directorios



Kotlin para desarrolladores Java

- Lenguaje (Kotlin)
- V8/SpiderMonkey
- Bibliotecas/API (ECMA 6 + Web)
- kotlin-stdlib

ACADEMIK

Demo productiva

Eclipse MicroProfile - 1, 2, 3 with Kotlin

1. Maven or Gradle config
2. MicroProfile dependency and your extras (Jakarta EE, Arquillian, JUnit, . . .)
3. Maven plugin (maven-compiler-plugin)
4. Kotlin plugin (kotlin-maven-plugin)

```
<dependency>
    <groupId>org.eclipse.microprofile</groupId>
    <artifactId>microprofile</artifactId>
    <type>pom</type>
    <version>2.1</version>
    <scope>provided</scope>
</dependency>
```

Kotlin with Maven - Dependency

```
1 <dependency>
2     <groupId>org.jetbrains.kotlin</groupId>
3     <artifactId>kotlin-stdlib-jdk8</artifactId>
4     <version>${kotlin.version}</version>
5 </dependency>
```

Kotlin with 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 with 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>
```

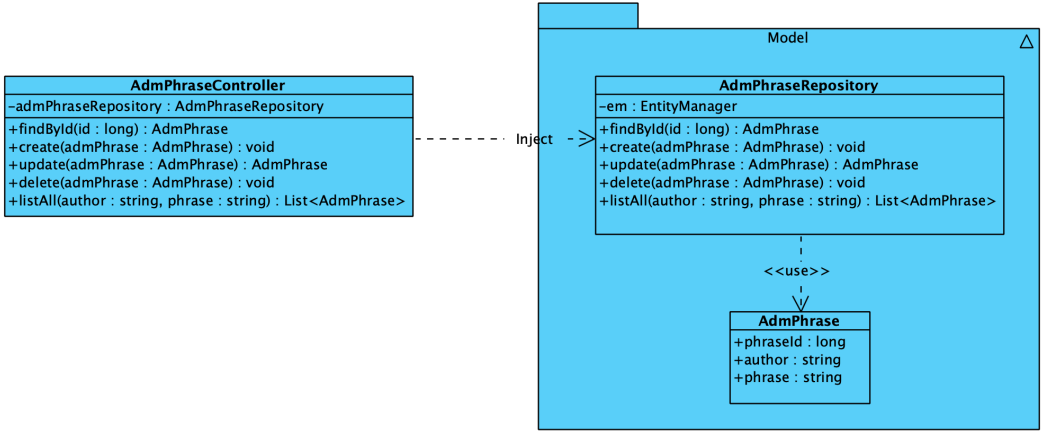
Idea general: Agregar todas las anotaciones arquitecturales de JakartaEE (CDI and EJB)

Kotlin + Jakarta EE + MicroProfile - Demo

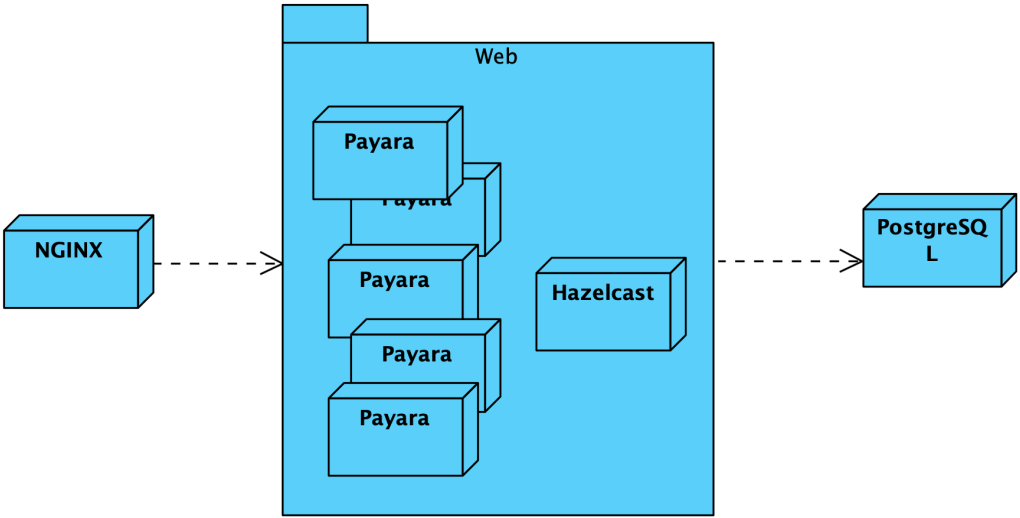
- Kotlin 1.3
- Libraries - SLF4J, Flyway, PostgreSQL
- Jakarta EE 8 - EJB, JPA
- MicroProfile - CDI, JAX-RS, MicroProfile Config
- Testing - Arquillian, JUnit, Payara Embedded

https://dzone.com/articles/
the-state-of-kotlin-for-jakarta-eemicroprofile-tra
https://github.com/tuxtor/integrum-ee

Kotlin + Jakarta EE + MicroProfile - Demo



Kotlin + Jakarta EE + MicroProfile - Demo



Kotlin - JPA entity

```
1  @Entity
2  @Table(name = "adm_phrase")
3  @TableGenerator(...)
4  data class AdmPhrase(
5      @Id
6      @GeneratedValue(strategy = GenerationType.TABLE,
7          generator = "admPhraseIdGenerator")
8      @Column(name = "phrase_id")
9      var phraseId:Long? = null,
10     var author:String = "",
11     var phrase:String = ""
12 )
```

Data Clases, Nullable Types

Kotlin - CDI Repository

```
1  @RequestScoped
2  class AdmPhraseRepository {
3
4      @Inject
5      private lateinit var em:EntityManager
6
7      ...
8
9  }
```

Lateinit (nullable type)

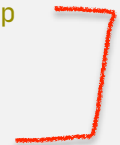
Kotlin - CDI Repository

```
1  fun create(admPhrase:AdmPhrase) = em.persist(admPhrase)
2
3  fun update(admPhrase:AdmPhrase) = em.merge(admPhrase)
4
5  fun findById(phraseId: Long) =
6  em.find(AdmPhrase::class.java, phraseId)
7
8  fun delete(admPhrase: AdmPhrase) = em.remove(admPhrase)
9  . . .
```

Single expression functions (One line methods)

Kotlin - CDI Repository

```
1  fun listAll(author: String, phrase: String):
2      List<AdmPhrase> {
3
4      val query = """SELECT p FROM AdmPhrase p
5      where p.author LIKE :author
6      and p.phrase LIKE :phrase
7      """
8
9      return em.createQuery(query, AdmPhrase::class.java)
10         .setParameter("author", "%$author%")
11         .setParameter("phrase", "%$phrase%")
12         .resultList
13 }
```



Multiline string

Kotlin - JAX-RS Controllers

```kotlin
@Path("/phrases")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
class AdmPhraseController{

    @Inject
    private lateinit var admPhraseRepository: AdmPhraseRepository

    @Inject
    private lateinit var logger: Logger
    ...

}
```

Kotlin - JAX-RS Controller

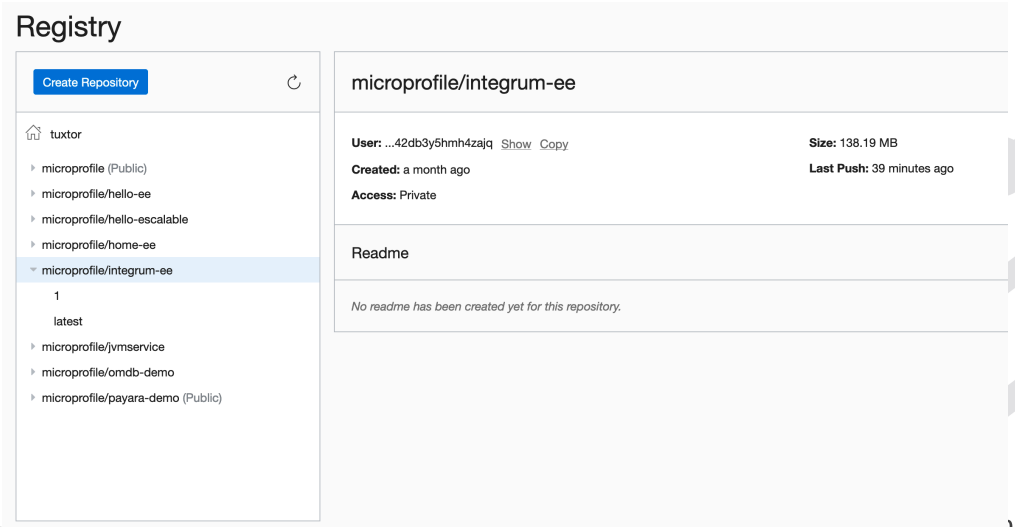
```
1
2  @GET
3  fun findAll(
4  @QueryParam("author") @DefaultValue("%") author: String,
5  @QueryParam("phrase") @DefaultValue("%") phrase: String) =
6      admPhraseRepository.listAll(author, phrase)
7
8  @GET
9  @Path("/{id:[0-9][0-9]*}")
10 fun findById(@PathParam("id") id:Long) =
11     admPhraseRepository.findById(id)
12
13 @PUT
14 fun create(phrase: AdmPhrase): Response {
15     admPhraseRepository.create(phrase)
16     return Response.ok().build()
17 }
```

Kotlin - JAX-RS Controller

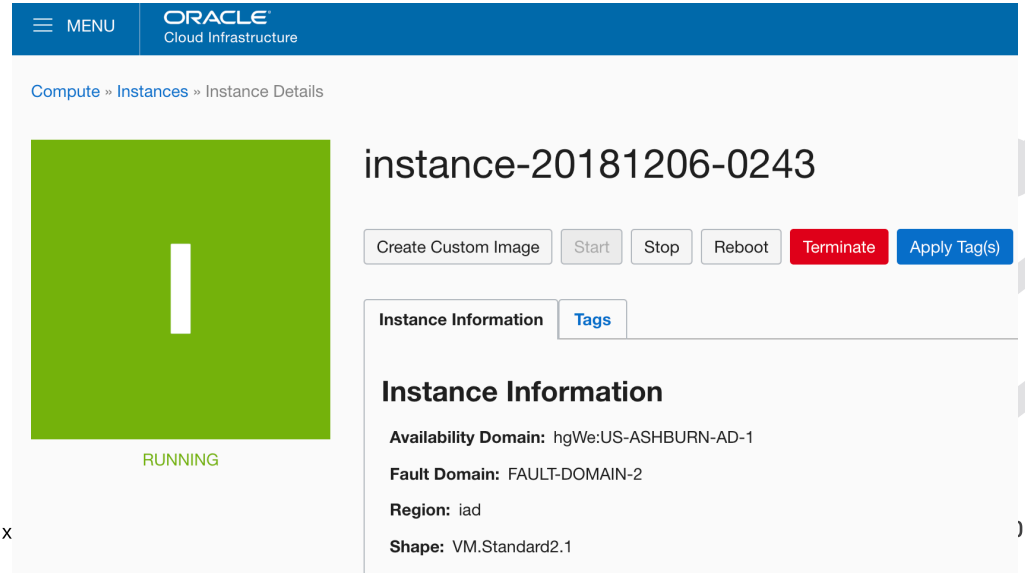
Elvis operator as expression

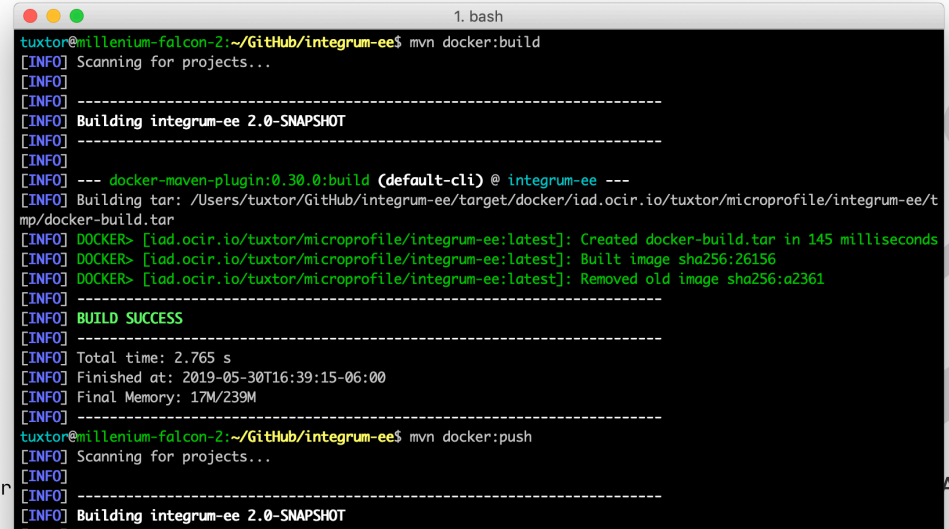
```
@POST
@Path("/{id:[0-9][0-9]*}")
fun update(@PathParam("id") id: Long?, phrase: AdmPhrase)
    :Response {
    if(id != phrase.phraseId)
        return Response.status(Response.Status.NOT_FOUND).build()

    val updatedEntity = admPhraseRepository.update(phrase)
    return Response.ok(updatedEntity).build()
}

@DELETE
@Path("/{id:[0-9][0-9]*}")
fun delete(@PathParam("id") id: Long): Response {
    val updatedEntity = admPhraseRepository.findById(id) ?:
    return Response.status(Response.Status.NOT_FOUND).build()
    admPhraseRepository.delete(updatedEntity)
    return Response.ok().build()
}
```

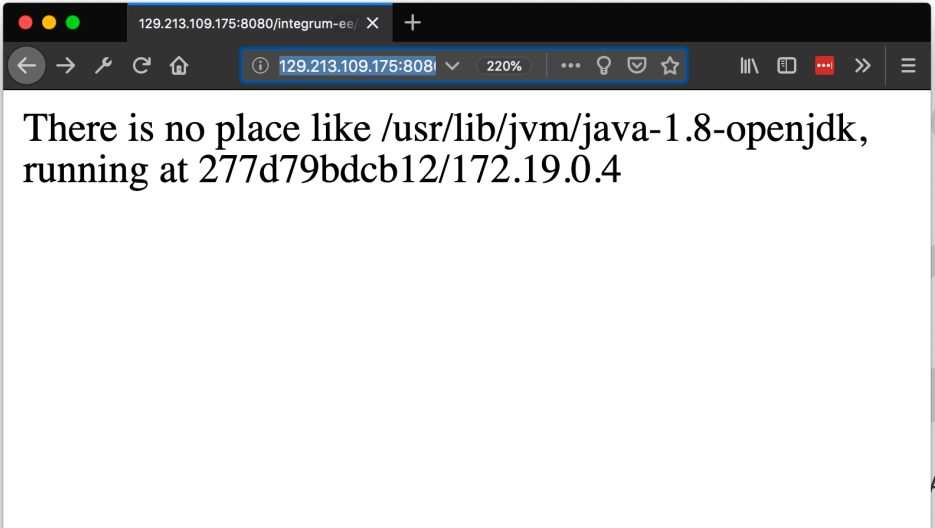
```
<groupId>io.fabric8</groupId>
<artifactId>docker-maven-plugin</artifactId>
<version>0.30.0</version>
...
<image>
    <name>iad.ocir.io/tuxtor/microprofile/integrum-ee</name>
    <build>
        <dockerFile>${project.basedir}/Dockerfile</dockerFile >
    </build>
</image>
```


Registry
Create Repository
tuxtor
microprofile (Public)
microprofile/hello-ee
microprofile/hello-escalable
microprofile/home-ee
microprofile/integrum-ee
1
latest
microprofile/jvmservice
microprofile/omdb-demo
microprofile/payara-demo (Public)
microprofile/integrum-ee
User: ...42db3y5hmh4zajq Show Copy
Created: a month ago
Access: Private
Size: 138.19 MB
Last Push: 39 minutes ago
Readme
No readme has been created yet for this repository.

Oracle Cloud




1. bash
tuxtor@millenium-falcon-2:~/GitHub/integrum-ee$ mvn docker:build
[INFO] Scanning for projects...
[INFO]
[INFO] --
[INFO] Building integrum-ee 2.0-SNAPSHOT
[INFO] --
[INFO]
[INFO] --- docker-maven-plugin:0.30.0:build (default-cli) @ integrum-ee ---
[INFO] Building tar: /Users/tuxtor/GitHub/integrum-ee/target/docker/iad.ocir.io/tuxtor/microprofile/integrum-ee/tmp/docker-build.tar
[INFO] DOCKER> [iad.ocir.io/tuxtor/microprofile/integrum-ee:latest]: Created docker-build.tar in 145 milliseconds
[INFO] DOCKER> [iad.ocir.io/tuxtor/microprofile/integrum-ee:latest]: Built image sha256:26156
[INFO] DOCKER> [iad.ocir.io/tuxtor/microprofile/integrum-ee:latest]: Removed old image sha256:a2361
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 2.765 s
[INFO] Finished at: 2019-05-30T16:39:15-06:00
[INFO] Final Memory: 17M/239M
[INFO] --
tuxtor@millenium-falcon-2:~/GitHub/integrum-ee$ mvn docker:push
[INFO] Scanning for projects...
[INFO]
[INFO] --
[INFO] Building integrum-ee 2.0-SNAPSHOT


129.213.109.175:8080/integrum-ee/
129.213.109.175:808
220%
There is no place like /usr/lib/jvm/java-1.8-openjdk, running at 277d79bdcb12/172.19.0.4

ACADEMIK

Kotlin vs. Java

Kotlin - Cosas más interesantes para mi

- Static typing
- Java inter-op
- OO + FP
- Null safety
- Extension functions
- Operator overloading
- Data classes
- Functions as expressions

Kotlin - Datos interesantes

- Effective Java - Immutability, builder, singleton, override, final by default, variance by generics
- Elvis - Groovy
- Type inference - Scala
- Immutability - Scala
- Identifiers - Scala
- Null values management - Groovy
- Functions - Groovy

Java - Muriendo desde 1995

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

Kotlin

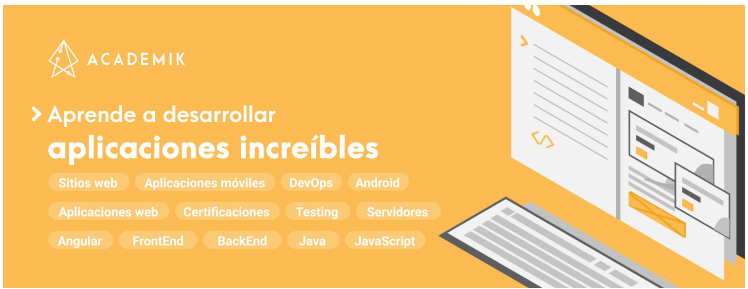
Ventajas

- Código conciso si se aprenden los nuevos bloques y expresiones
- Java inter-op
- Una oportunidad de Backend para desarrolladores Android
- Un nuevo abordaje "Full-stack"

Desventajas

- IntelliJ IDEA Ultimate
- Curva de aprendizaje más pronunciada
- Compiler (time)
- Thread-managed vs Co-routines
- Amber, Loom, Valhalla, Panama (Java 18?)

Academik



Víctor Orozco

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