

Introducción a Kotlin para desarrolladores Java

Víctor Orozco

10 de septiembre de 2020

@tuxtor



ACADEMIK

Contexto

Características interesantes

Demo productiva

Kotlin vs. Java





ACADEMIK

Contexto



¿Java?

- Lenguaje (Java 14) ~~.....~~
- 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 con algunas cosas de 8)
 - ART/Dalvik
 - Bibliotecas/API (Java+Google Classpath)
-



@tuxtor

Java - Java como JVM

TheServerSide
Your Enterprise Java Community

Java management ▾ All Subtopics

Search TechTa

Transcript of James' TSSJS 2011 Discussion about Java and the JVM

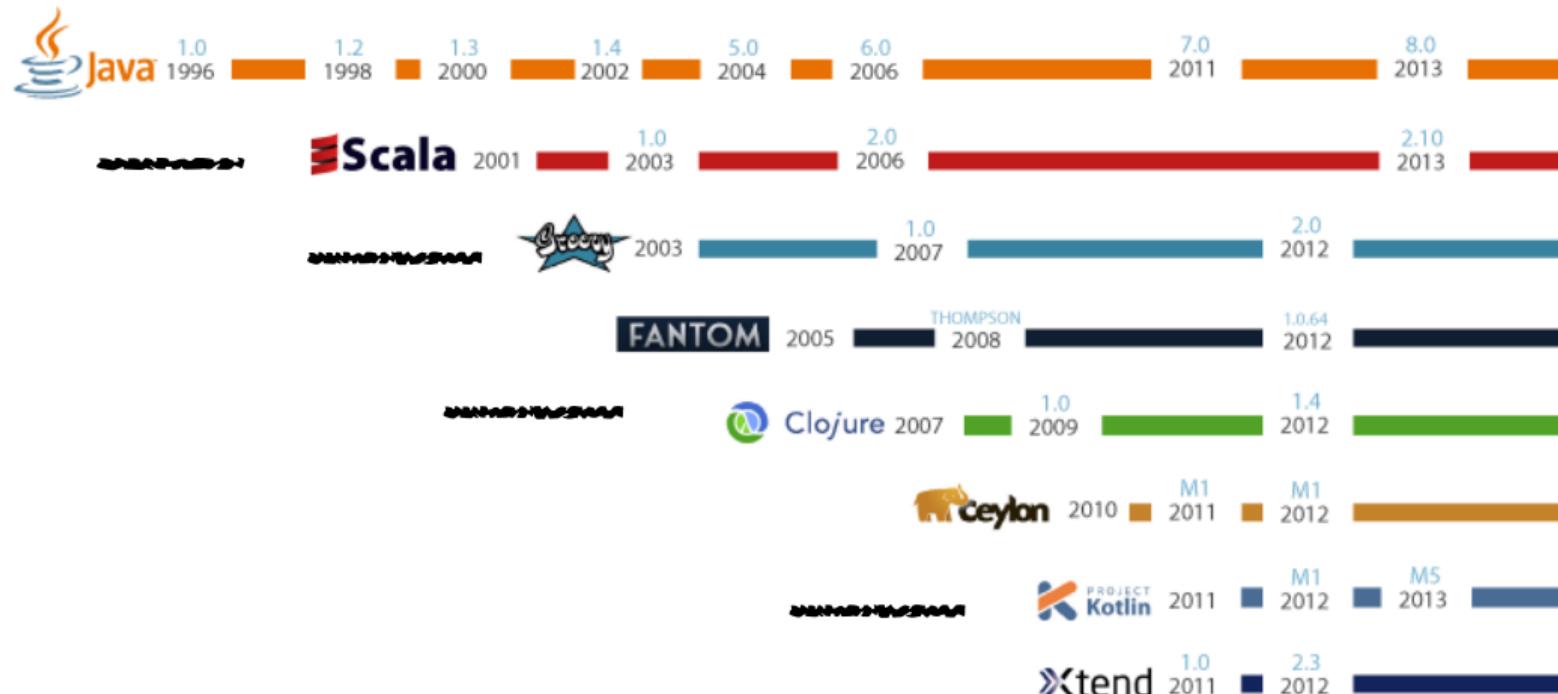
"At the core of the Java ecosystem is the JVM. Most people talk about Java the language, and this may sound odd coming from me, but I could hardly care less.

"What I really care about is the Java Virtual Machine as a concept, because that is the thing that ties it all together: it's

Most people talk about Java the language, and this may sound odd coming from me, but I could hardly care less.

James Gosling

JVM



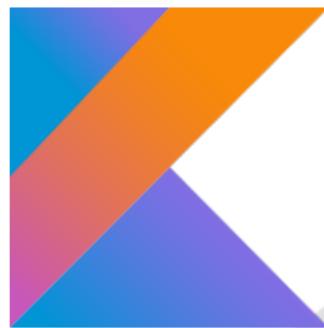
Kotlin

- 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



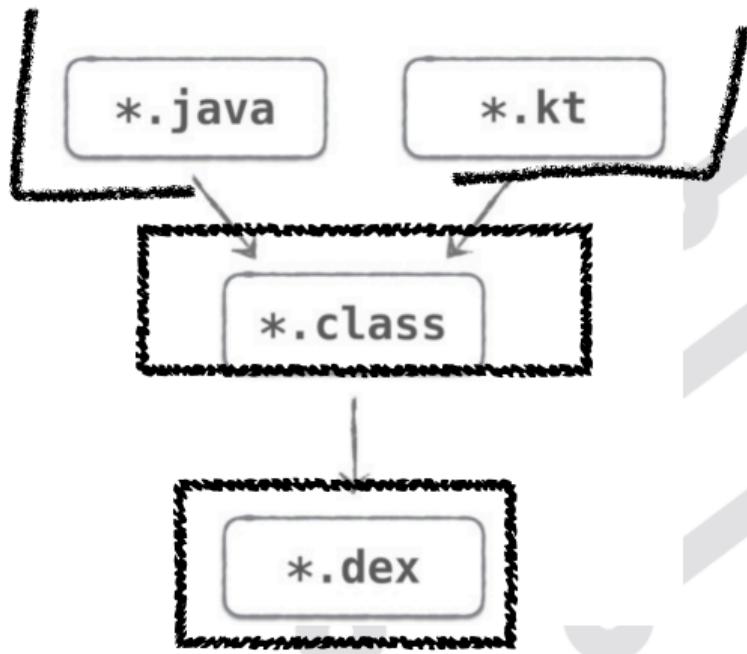
Kotlin Nativo/GraalVM

- 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

Características 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 explícita
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*

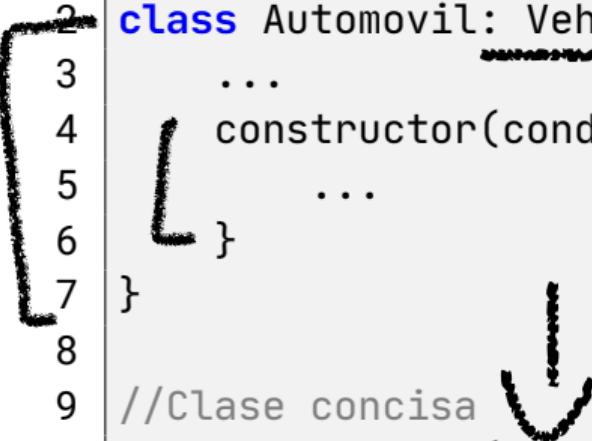
```
1 //Bloque
2 fun sumar(x: Int, y: Int): Int{
3     return x + y
4 }
5
6 //One-line - Expresion y default parameter
7 fun sumar2(x: Int, y: Int = 99) = x + y
8
9 //Infix - AKA operador
10 infix fun Int.sumar3(y: Int) = this + y
11 ...
12 //Uso
13 A 2 sumar3 4
```



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,  
5      toString)  
6 data class Automovil(val marca: String,  
7                      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 }
```

@JvmStatic

Kotlin - Extension functions

Posibilidad de extender funcionalidad en clases existentes

```
1 fun String.ultimoCar(): Char = this.get(this.length - 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

```
1 val auto1: Vehiculo = ...
2
3 if (auto1 is Automovil){
4     auto1.cosasDeAutos()
5 }
6
7 when (auto1) {
8     is Automovil -> auto1.cosasDeAutos()
9     is Motocicleta -> auto1.cosasDemotos()
10    else -> throw Exception("El vehiculo de los ojos tristes")
11 }
```



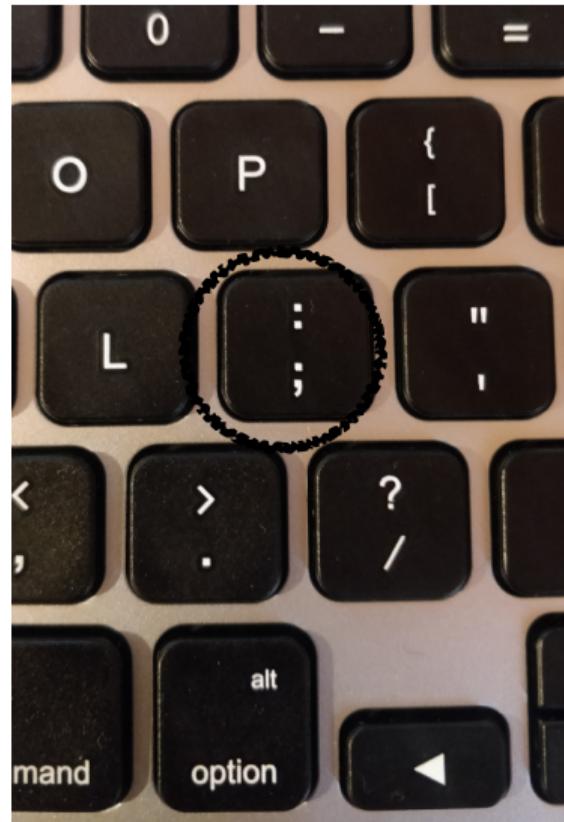
Kotlin - Collections

Metodos convenientes y expresiones lambda, por defecto Inmutables

```
1 //Con expresiones lambda  
2 list0f(1, 2, 3).filter{ i -> i % 2 == 0}  
3  
4 //Con expresiones cortas (predicado)  
5 list0f(1, 2, 3).filter{i % 2 == 0}
```

Kotlin - Convenciones

- Convenciones de nombrado Java
- Tipos 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





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)

Eclipse MicroProfile with Payara 5

```
1 <dependency>
2   <groupId>org.eclipse.microprofile</groupId>
3   <artifactId>microprofile</artifactId>
4   <type>pom</type>
5   <version>3.2</version>
6   <scope>provided</scope>
7 </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

```
1 <execution>
2   <id>default-compile</id>
3   <phase>none</phase>
4 </execution>
5 <execution>
6   <id>default-testCompile</id>
7   <phase>none</phase>
8 </execution>
9 <execution>
10  <id>java-compile</id>
11  <phase>compile</phase>
12  <goals> <goal>compile</goal> </goals>
13 </execution>
14 <execution>
15  <id>java-test-compile</id>
16  <phase>test-compile</phase>
17  <goals> <goal>testCompile</goal> </goals>
18 </execution>
```

Kotlin with Maven - kotlin-maven-plugin

```
1 <compilerPlugins>
2   <plugin>all-open</plugin>
3 </compilerPlugins>
4 ...
5 <option>all-open:annotation=javax.ws.rs.Path</option>
6 <option>all-open:annotation=javax.enterprise.context.RequestScoped</option>
7 <option>all-open:annotation=javax.enterprise.context.SessionScoped</option>
8 <option>all-open:annotation=javax.enterprise.context.ApplicationScoped</option>
9 <option>all-open:annotation=javax.enterprise.context.Dependent</option>
10 <option>all-open:annotation=javax.ejb.Singleton</option>
11 <option>all-open:annotation=javax.ejb.Stateful</option>
12 <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

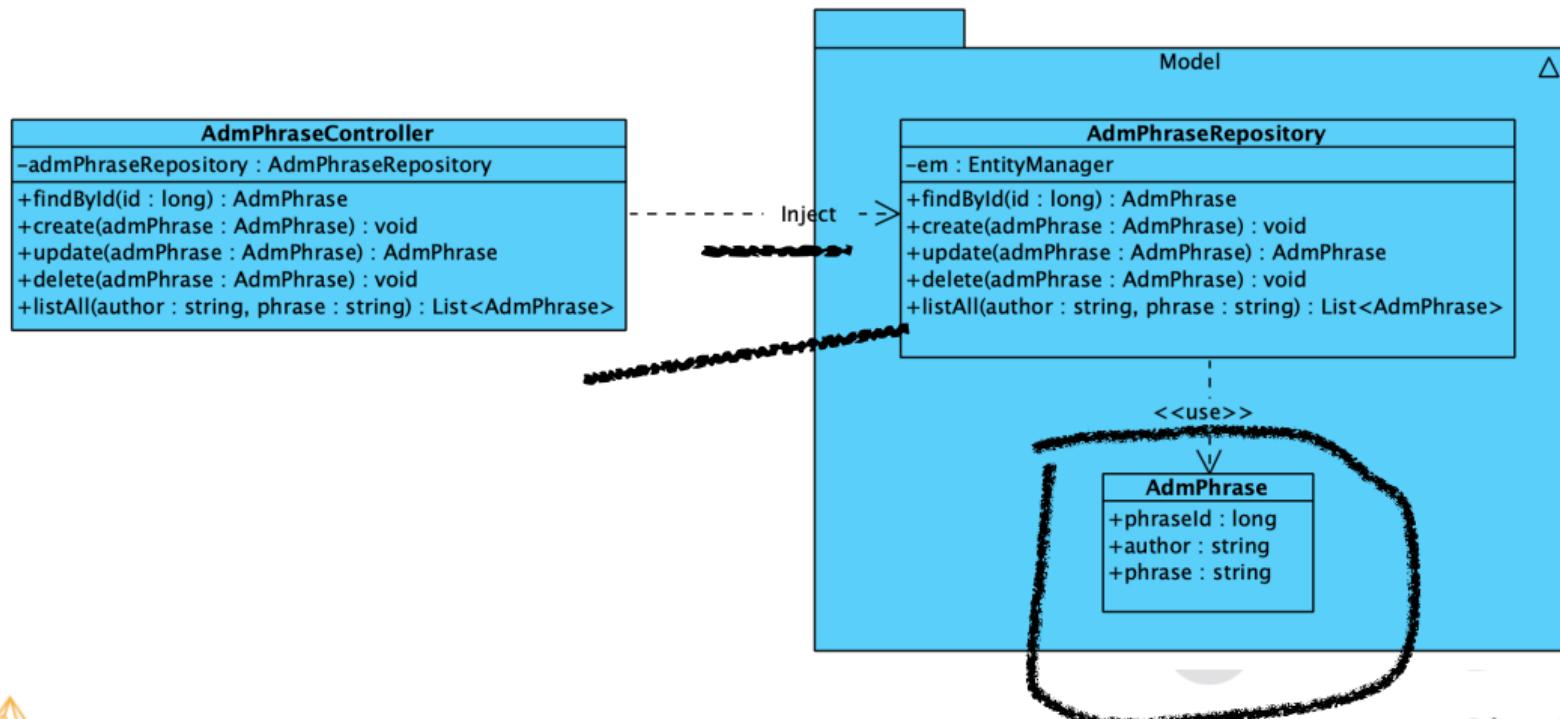
- Kotlin 1.4.x
- Libraries - SLF4J, Flyway, PostgreSQL
- Jakarta EE 8 - EJB, JPA
- MicroProfile - CDI, JAX-RS, MicroProfile Config
- Testing - Arquillian, JUnit, Payara Embedded

<https://dzone.com/articles/>

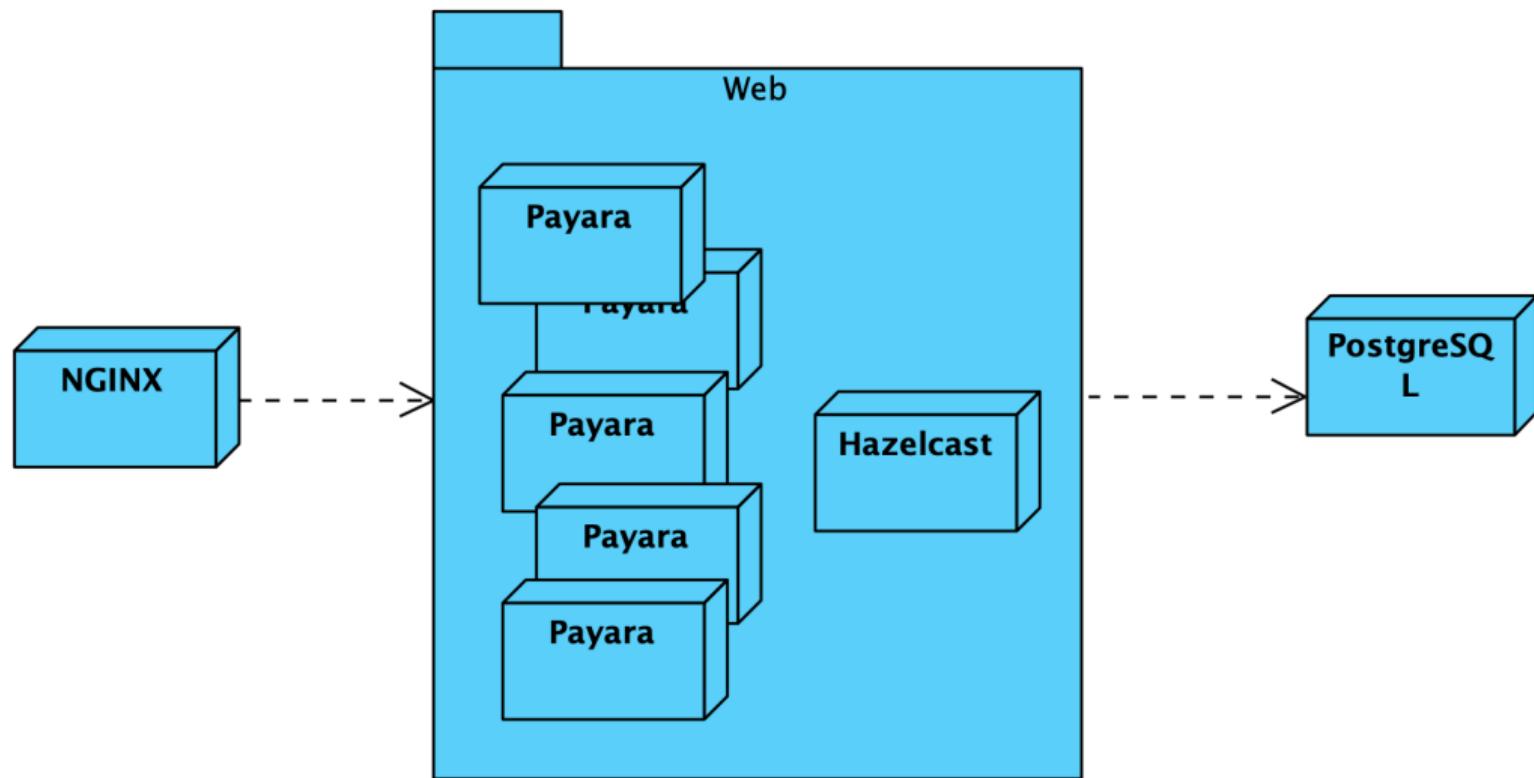
the-state-of-kotlin-for-jakarta-eemicroprofile-tra

<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 )
```

Kotlin - CDI Repository

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

Lateinit (nullable type)



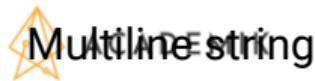
Kotlin - CDI Repository

```
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 }
```



@tuxtor

Kotlin - JAX-RS Controllers

```
1 @Path("/phrases")
2 @Produces(MediaType.APPLICATION_JSON)
3 @Consumes(MediaType.APPLICATION_JSON)
4 class AdmPhraseController{
5
6     @Inject
7     private lateinit var admPhraseRepository: AdmPhraseRepository
8
9     @Inject
10    private lateinit var logger: Logger
11    ...
12
13 }
```

Kotlin - JAX-RS Controller

```
1  @GET
2  fun findAll(
3      @QueryParam("author") @DefaultValue("%") author: String,
4      @QueryParam("phrase") @DefaultValue("%") phrase: String) =
5          admPhraseRepository.listAll(author, phrase)
6
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

```
1 @POST
2 @Path("/{id:[0-9][0-9]*}")
3 fun update(@PathParam("id") id: Long?, phrase: AdmPhrase)
4     :Response {
5     if(id != phrase.phraseId)
6         return Response.status(Response.Status.NOT_FOUND).build()
7
8     val updatedEntity = admPhraseRepository.update(phrase)
9     return Response.ok(updatedEntity).build()
10 }
11
12 @DELETE
13 @Path("/{id:[0-9][0-9]*}")
14 fun delete(@PathParam("id") id: Long): Response {
15     val updatedEntity = admPhraseRepository.findById(id) ?:
16     return Response.status(Response.Status.NOT_FOUND).build()
17     admPhraseRepository.delete(updatedEntity)
18     return Response.ok().build()
```



```
1 <groupId>io.fabric8</groupId>
2 <artifactId>docker-maven-plugin</artifactId>
3 <version>0.30.0</version>
4 ...
5 <image>
6   <name>iad.ocir.io/tuxtor/microprofile/integrum-ee</name>
7   <build>
8     <dockerFile>${project.basedir}/Dockerfile</dockerFile >
9   </build>
10 </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](#)

Size: 138.19 MB

Created: a month ago

Last Push: 39 minutes ago

Access: Private

Readme

No readme has been created yet for this repository.



tor

☰ MENU ORACLE®
Cloud Infrastructure

Compute » Instances » Instance Details



RUNNING

instance-20181206-0243

Create Custom Image Start Stop Reboot **Terminate** Apply Tag(s)

Instance Information **Tags**

Instance Information

Availability Domain: hgWe:US-ASHBURN-AD-1
Fault Domain: FAULT-DOMAIN-2
Region: iad
Shape: VM.Standard2.1

tor



tor

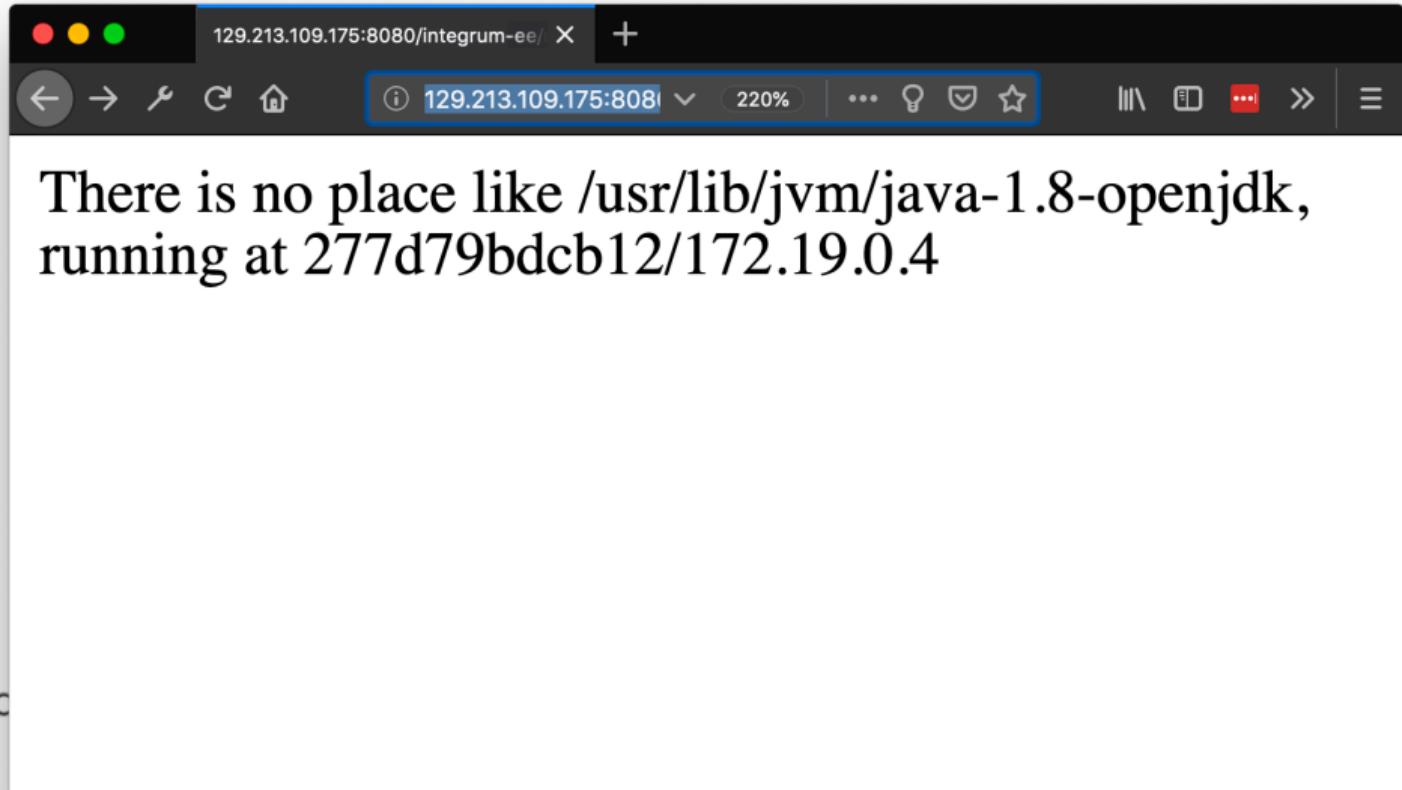
Oracle Cloud

```
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
```



AC

tuxtor





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



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



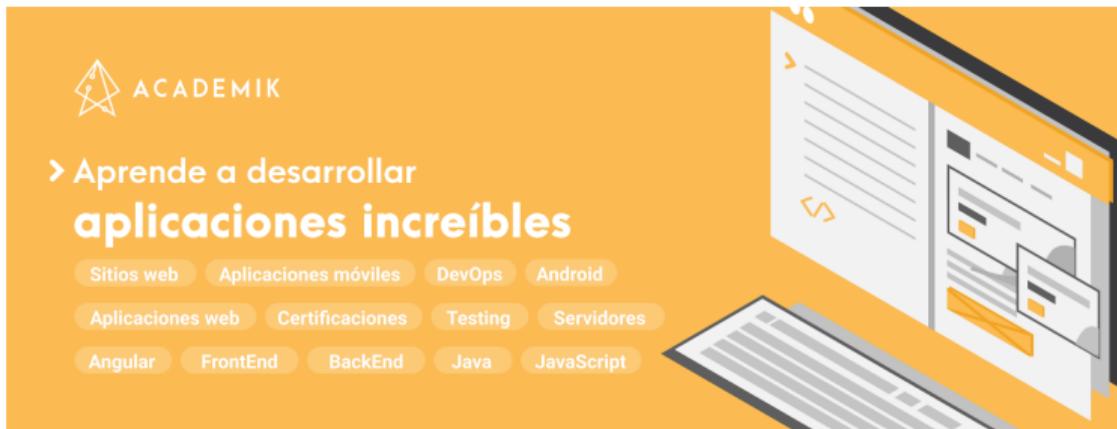
@tuxtor

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?)



The image shows a landing page for 'ACADEMIK'. At the top left is the logo, which consists of a stylized orange star-like icon followed by the word 'ACADEMIK' in a sans-serif font. Below the logo is a large, bold, orange header with the text 'Aprende a desarrollar aplicaciones increíbles'. Underneath this header is a horizontal row of ten colored buttons, each containing a technology term: 'Sitios web', 'Aplicaciones móviles', 'DevOps', 'Android', 'Aplicaciones web', 'Certificaciones', 'Testing', 'Servidores', 'Angular', 'FrontEnd', 'BackEnd', 'Java', and 'JavaScript'. To the right of the text and buttons is a graphic of a computer monitor displaying a web browser with multiple tabs open, and a keyboard in front of it.



@tuxtor



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



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