Kotlin for Spring Developers

Antonio Leiva

devexperto.com
@devexperto1

1. Organization

2. What is Kotlin?

Features

- Created By Jetbrains
- Executed in JVM
- Statically typed
- Object oriented and functional
- Free and open source

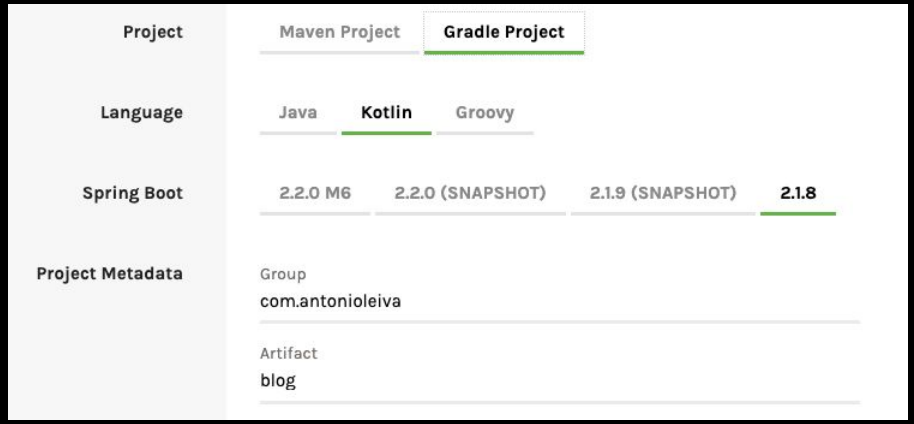
2. What is Kotlin?

Philosophy

- Pragmatic
- Concise
- Safe
- Interoperable

3. First Spring project using Kotlin

Project creation



3. First Spring project using Kotlin

Project creation

```
@SpringBootApplication
class BlogApplication

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

4. Basic elements

Functions

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
}
```

4. Basic elements

Functions

- Exercise: Create a function to print a message when the App starts

4. Basic elements

Variables

```
val x = 20
val y: Double
val z: List<Int> = listOf(1, 2, 3)
```

4. Basic elements

String templates

```
toast("Hello ${textView.text}")
```

4. Basic elements

String templates

- Exercise: Add an argument to the Run, save it to a variable and use a String template to write “Hello *name*”

4. Basic elements

Operador *Spread* y *vararg*

```
val numbers = arrayOf(1, 2, 3, 4, 5, 6)
listOf(7, *numbers, 8, 9)
```

4. Basic elements

Operador *Spread* y *vararg*

- Exercise: Write a function that receives a vararg and prints it

5. Classes and properties

Classes

```
class Developer : Person()
open class Person

val dev: Person = Developer()
```

5. Classes and properties

Properties

```
class Developer(
        val name: String,
        val age: Int
)
```

5. Classes and properties

Interfaces

```
interface CanWalk {

    fun doStep()

    fun walk(numSteps: Int) {
        repeat(numSteps) { doStep() }
    }
}
```

5. Classes and properties

- Create an HtmlController that runs the root of the site

5. Classes and properties

- Create an article class and show a list of articles

5. Classes and properties

Data Classes

```
data class MediaItem(val title: String, val thumbUrl: String)

val mediaItem = MediaItem( title: "Title",  thumbUrl: "Url")
val (title, url) = mediaItem
```

5. Classes and properties

Data Classes

- Exercise: Convert Article to a data class

7. Extension functions

and default values

```
fun String.println() {
    println(this)
}
```

7. Extension functions

and default values

- Exercise 1: Create an extension function called *toSlug()* that converts a String to a valid slug.

7. Extension functions

and default values

- Exercise 2: create a *set()* function for *Model*, to simplify *addAttribute()*

7. Extension functions

Reified functions

```
inline fun <reified T> T.printClass() {
    println(T::class.java.simpleName)
}
```

7. Extension functions

Reified functions

- Exercise: Create a *runApplication2()* function to encapsulate this code:

```
SpringApplication.run(BlogApplication::class.java, *args)
```

7. Operator overloading

Expression	Translated to
`a + b`	`a.plus(b)`
`a - b`	`a.minus(b)`
`a * b`	`a.times(b)`
`a / b`	`a.div(b)`
`a % b`	`a.rem(b)`, `a.mod(b)` (deprecated)
`a..b`	`a.rangeTo(b)`

7. Operator overloading

- Exercise 1: Create *set* operator overload for *Model*

8. Enums

```
enum class Type { PHOTO, VIDEO }
```

8. Enums

- Exercise: Create a new *Type* enum in *Article* class with two types: *Text* and *Video*

9. Flow control

when

```
videoIndicator.visibility = when (item.type) {
    MediaItem.Type.PHOTO → View.GONE
    MediaItem.Type.VIDEO → View.VISIBLE
}
```

9. Flow control

if

```
videoIndicator.visibility = if (item.type == Type.PHOTO) {
    View.GONE
} else {
    View.VISIBLE
}
```

9. Flow control

for

```
for (type in Type.values()) {
    // ...
}

for (i in 0 until 10) {
    // ...
}
```

9. Flow control

- Exercise: Use *for* and *when* to convert Articles to a new *RenderedArticle* class that sets a Video suffix in the title if the type is *Video*

9. Flow control

- Exercise2: Create a render() extension function for *Article* that returns a *RenderedArticle*

17. Nullability

```
val item: MediaItem? = null
item.print()
```

```
if (item ≠ null) {
    item.print()
}
```

```
val item: MediaItem? = null
item?.print()
```

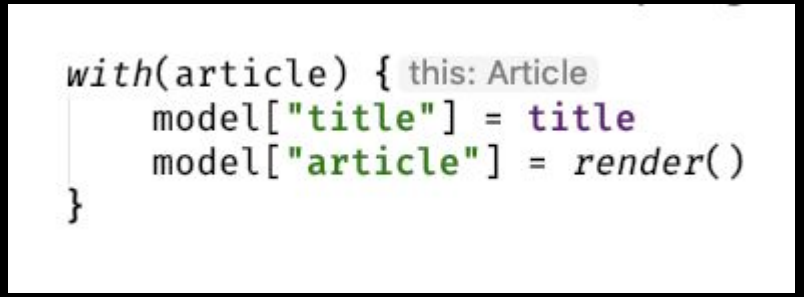
```
val item: MediaItem? = null
item!!.print()
```

17. Nullability

- Exercise: Create the navigation to the article

11. Scope functions

with



11. Scope functions

- Exercise: Use *with* to simplify the *Model* binding.

11. Scope functions

apply

```
val person = Person().apply { this: Person
    name = "Tom"
    age = 20
}
```

11. Scope functions

let

```
findArticleBySlug(slug)?.let { it: Article
    //
}
```

11. Scope functions

- Exercise: Use *let* to solve the previous example in a different way.

12. Lambdas

```
fun asyncOp(value: Int, callback: (String) → Unit) {
    // ...
}

asyncOp( value: 20) { result → toast(result) }
```

12. Lambdas

lambda: (X, Y) → Z

12. Lambdas

$$\text{sum: (Int, Int)} \rightarrow \text{Int}$$

12. Lambdas

sum: (Int, Int) → Int

```
{ x, y ->
    val z = x + y
    z
}
```

12. Lambdas

sum: (Int, Int) → Int

```
{ x, y ->
    x + y
}
```

12. Lambdas

- Exercise: Transform the *TitleRenderer* we created using an interface into a lambda.

13. Property delegation

lazy

```
private val articleRepository by lazy { ArticleRepository() }
```

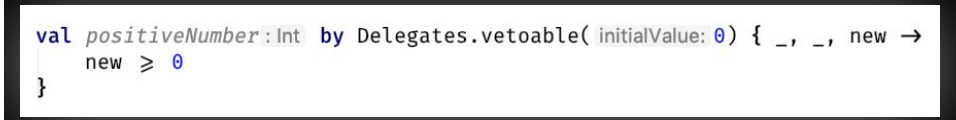
13. Property delegation

observable

```
val observedNumber by Delegates.observable( initialValue: 0){ _, old, new →
    Log.d( tag: "observedNumber",  msg: "old value: $old, new value $new" )
}
```

13. Property delegation

vetoable



13. Property delegation

lateinit

```
@MockkBean
private lateinit var articleRepository: ArticleRepository
```

13. Property delegation

- Exercise: Use *lazy* to create an ArticleRepository and instantiate it only when used

13. Property delegation

- Exercise: Use *observable* to update the *popular* tag of an article when the number of likes is 3 or more

14. Collections and Ranges

Collections

```
val urls = list
        .filter { it.type == MediaItem.Type.PHOTO }
        .sortedBy { it.title }
        .map { it.thumbUrl }
```

14. Collections and Ranges

Collections

- Exercise: use collection functions to replace *for* loops

14. Collections and Ranges

Ranges

```
for (i in 1..4) print(i)

(1..4).forEach(::print)

(1 until 4).forEach(::print)

(4 downTo 1).forEach(::print)

val x: String = "c"

val y = when (x) {
    in ("a".."e") → 1
    in ("f".."z") → 2
    else → 3
}
```

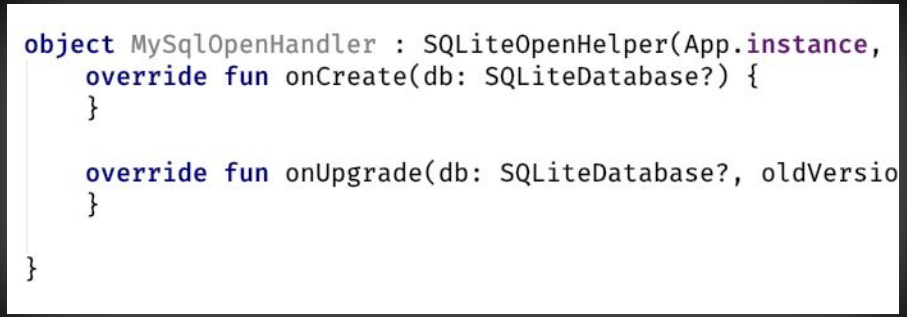
14. Collections and ranges

Ranges

- Exercise: Create a list of 10 articles by using a range.

15. Objects

Ranges



15. Objects

- Exercise: Use an Object for *ArticlesRepository*.

18. Sealed classes

```
sealed class Operation{
    class Add(val value: Int) : Operation()
    class Subtract(val value: Int) : Operation()
    class Multiply(val value: Int) : Operation()
    class Divide(val value: Int) : Operation()
}
```

19. Type Alias

```
typealias Listener = (MediaItem) → Unit
```

19. Type Alias

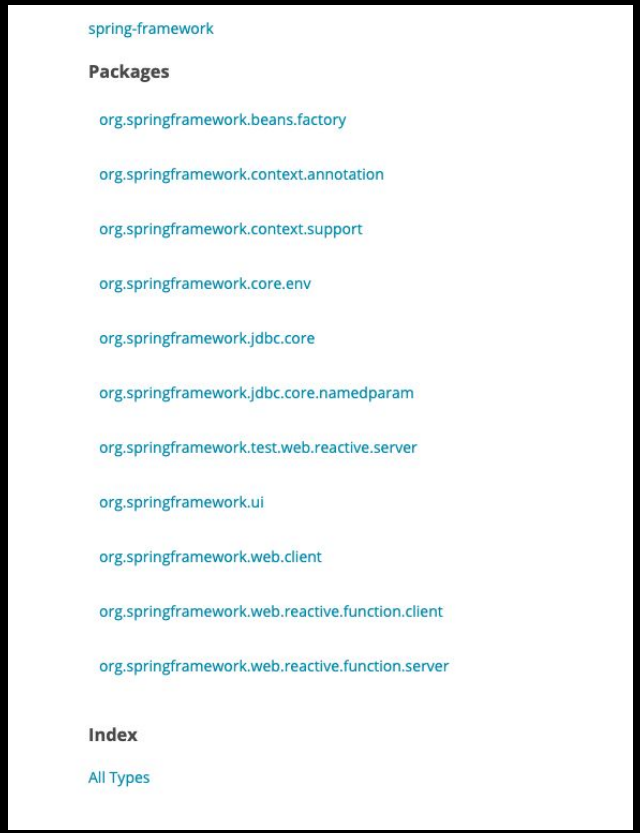
- Exercise: Use Type Alias to provide a name for *titleRenderer* lambda.

20. Coroutines

```
GlobalScope.launch(Dispatchers.Main) { this: CoroutineScope
    val cats = async(Dispatchers.Default) { MediaProvider.dataSync( dataType: "cats") }
    val nature = async(Dispatchers.Default) { MediaProvider.dataSync( dataType: "nature") }
    updateData( media: cats.await() + nature.await(), filter)
}
```

20. Spring + Kotlin

Extensions



20. Spring + Kotlin

Beans DSL

```kotlin
val beans = beans {
        bean<UserHandler>()
        bean<Routes>()
        bean("webHandler") {
                RouterFunctions.toWebHandler(ref<Routes>().router(), HandlerStrategies.buil
        }
        bean("messageSource") {
                ReloadableResourceBundleMessageSource().apply {
                        setBasename("messages")
                        setDefaultEncoding("UTF-8")
                }
        }
        bean {
                val prefix = "classpath:/templates/"
                val suffix = ".mustache"
                val loader = MustacheResourceTemplateLoader(prefix, suffix)
                MustacheViewResolver(Mustache.compiler().withLoader(loader)).apply {
                        setPrefix(prefix)
                        setSuffix(suffix)
                }
        }
        profile("cors") {
                bean("corsFilter") {
                        CorsWebFilter { CorsConfiguration().applyPermitDefaultValues() }
                }
        }
}
```

20. Spring + Kotlin

WebFlux Functional DSL

```
@Configuration
class RouterConfiguration {

        @Bean
        fun mainRouter(userHandler: UserHandler) = coRouter {
                accept(TEXT_HTML).nest {
                        (GET("/user/") or GET("/users/")).invoke(userHandler::findAllView)
                        GET("/users/{login}", userHandler::findViewById)
                }
                accept(APPLICATION_JSON).nest {
                        (GET("/api/user/") or GET("/api/users/")).invoke(userHandler::findAll)
                        POST("/api/users/", userHandler::create)
                }
        }

}
```