



# Kotlin

**introduction for Java developers**

**Karol Dominiak**



Clojure



The JetBrains logo is a large, stylized letter 'J' composed of several overlapping, rounded rectangular segments in shades of pink, magenta, orange, and yellow. It is centered on the page.

**JET  
BRAINS**

**CL**  
—

**WS**  
—

**TC**  
—

**IJ**  
—

**R#**  
—

**PS**  
—

**PC**  
—

**RD**  
—

# Use cases:

## 1. Android development

...

## 2. Back-end:

- Spring
- Vert.x
- Ktor
- [kotlinx.html](https://kotlinlang.org/docs/reference/server-overview.html)



<https://kotlinlang.org/docs/reference/server-overview.html>

# Differences

Java / Kotlin

# Java

```
@Service
@RequiredArgsConstructor
public class JavaAnimalsComClient {

    private final RestTemplate restTemplate;
    private final AnimalsComProperties animalsComProperties;

    public String getCat(final String id) {
        return getAnimal(animalsComProperties.getCatsUrl(), id);
    }

    public String getDog(final String id) {
        return getAnimal(animalsComProperties.getDogsUrl(), id);
    }

    private String getAnimal(final URI uri, final String animalId) {
        final URI uriWithId = expandWithPathSegment(uri, animalId);
        final HttpHeaders httpHeaders = new HttpHeaders();
        httpHeaders.setContentType(MediaType.APPLICATION_JSON);
        final HttpEntity entity = new HttpEntity(httpHeaders);
        try {
            final String animal = restTemplate.exchange(uriWithId, HttpMethod.GET, entity, String.class).getBody();
            if (animal == null) {
                throw new AnimalNotFoundException(animalId);
            }
            return animal;
        } catch (final RestClientException ex) {
            throw new AnimalComCommunicationException("Communication with animals.com failed!", ex);
        }
    }

    private URI expandWithPathSegment(final URI uri, final String additionalSegment) {
        return UriComponentsBuilder.fromUri(uri)
            .pathSegment(additionalSegment)
            .encode()
            .build()
            .toUri();
    }
}
```

# Kotlin

```
@Service
class KotlinAnimalsComClient(private val restTemplate: RestTemplate,
    private val animalsComProperties: AnimalsComProperties) {

    fun getCat(id: String) = getAnimal(animalsComProperties.catsUrl, id)

    fun getDog(id: String) = getAnimal(animalsComProperties.dogsUrl, id)

    private fun getAnimal(uri: URI, animalId: String): String {
        val uriWithId = uri.expandWithPathSegment(animalId)
        val httpHeaders = HttpHeaders().apply { contentType = MediaType.APPLICATION_JSON }
        val entity = HttpEntity<String>(httpHeaders)
        try {
            val animal = restTemplate.exchange(uriWithId, HttpMethod.GET, entity, String::class.java).body
            return animal ?: throw AnimalNotFoundException(animalId)
        } catch (ex: RestClientException) {
            throw AnimalComCommunicationException("Communication with animals.com failed!", ex)
        }
    }

    private fun URI.expandWithPathSegment(additionalSegment: String) = UriComponentsBuilder.fromUri(this)
        .pathSegment(additionalSegment)
        .encode()
        .build()
        .toUri()
}
```

# Methods



## Java

```
public String getDog(final String id) {  
    return getAnimal(animalsComProperties.getDogsUrl(), id);  
}
```

## Kotlin



```
fun getDog(id: String): String {  
    return getAnimal(animalsComProperties.dogsUrl, id)  
}
```

or...

```
fun getDog(id: String) = getAnimal(animalsComProperties.dogsUrl, id)
```



# Variables

## Java

```
String zmienna = "sample text";  
final String stala = "sample text";
```

## Kotlin

```
var zmienna = "sample text"  
val stala = "sample text"
```



# Data class & Nullable



## Java

```
@Data
public class Person {
    @NotNull
    private String firstName;
    private String lastName;
}
```



## Kotlin

```
data class Person(
    val firstName: String,
    val lastName: String?
)
```



# Inheritance

## & override & optional class body

```
@Component
class SampleFilter : OncePerRequestFilter() {

    override fun doFilterInternal(request: HttpServletRequest,
                                   response: HttpServletResponse,
                                   filterChain: FilterChain) {
        filterChain.doFilter(request, response)
    }
}
```

```
@ResponseStatus(HttpStatus.NOT_FOUND)
class AnimalNotFoundException(animalId: String)
    : RuntimeException("Animal with id: $animalId not found.")
```

# All args constructor (dependency injection)

## Java

```
@Service
@RequiredArgsConstructor
public class JavaAnimalsComClient {

    private final RestTemplate restTemplate;
    private final AnimalsComProperties animalsComProperties;
```

## Kotlin

```
@Service
class KotlinAnimalsComClient(private val restTemplate: RestTemplate,
                             private val animalsComProperties: AnimalsComProperties) {
```

# Elvis operator ?:

~(if null then ...)

```
val car = carService.findCar() ?: "default car"  
authService.getHeader(name: "login") ?: throw RuntimeException()
```


## return if-else

equivalent to Java ternary operator (? ... : ...)

```
private fun getNumber(value: Boolean): Int {  
    return if (value) 1 else 0  
}
```

# Triple-quoted String

```
class ApiModelExamples {  
    companion object {  
        const val WEIGHT = """  
            [  
                {  
                    "weight":72.80000305175781,  
                    "id":"a7c1b983-cf11-4cb4-ab04-33f56fbf3797",  
                    "comment":"comment"  
                }  
            ]  
        """  
    }  
}
```



# Default method parameters

```
fun invoke() {  
    calculate()  
}  
  
fun calculate(indexStart: Int = 0): String {  
    if (indexStart == 10) {  
        return "DONE"  
    }  
    return calculate(indexStart: indexStart + 1)  
}
```

# when

```
when (car) {  
  is Mercedes -> drive(car) // smart cast  
  is Opel -> dontDrive(opel = car) // argument opel is car  
  else -> playGames()  
}
```

```
when (age) {  
  in 1..19 -> goToSchool()  
  in 65..Int.MAX_VALUE -> stayAtHome()  
  30,31 -> celebrate30th()  
}
```

# return when / return try etc.

```
return when {  
    value == "Janusz Korwin Mikke" -> 4.76  
    value === "Konrad Berkowicz" -> 4.76  
    value.endsWith(suffix: "Mikke") -> 4.76  
    value.length < 3 -> -1.0  
    else -> throw Exception()  
}
```

```
return try {  
    objectMapper.readValue(value, Patient::class.java)  
} catch (ex: IOException) {  
    throw RuntimeException()  
}
```



# Extension functions

```
fun doSomething() {  
    fun String.expandWith(additionalString: String): String {  
        return this + additionalString  
    }  
    "Karol".expandWith( additionalString: " Dominiak")  
}
```

# Functions

**apply** / run / **let** / also

# 1 - apply

Java

```
final Person person = new Person();  
person.setFirstName("Karol");  
person.setLastName("Dominiak");  
  
final HttpHeaders httpHeaders = new HttpHeaders();  
httpHeaders.setContentType(MediaType.APPLICATION_JSON);  
return httpHeaders;
```

Kotlin

```
val person = Person().apply { this: Person  
    firstName = "Karol"  
    lastName = "Dominiak"  
}  
  
return HttpHeaders().apply { contentType = MediaType.APPLICATION_JSON }
```

## 2 - let

### Java

```
if (person.getLastName() != null) {  
    doSomething(person.getLastName());  
}
```

### Kotlin

```
person.lastName?.let { lastName -> doSomething(lastName) }
```

or...

```
person.lastName?.let { doSomething(it) }
```

or...

```
person.lastName?.let { doSomething(it) }  
person.lastName?.run { doSomething( lastName: this) }  
person.lastName?.also { doSomething(it) }  
person.lastName?.apply { doSomething( lastName: this) }
```

# **Danger**

**Spring Boot compatibility & workarounds...**

# Jackson support

```
<dependency>
  <groupId>com.fasterxml.jackson.module</groupId>
  <artifactId>jackson-module-kotlin</artifactId>
  <version>2.9.8</version>
</dependency>
```

```
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.module.kotlin.KotlinModule
import com.fasterxml.jackson.module.kotlin.registerKotlinModule


val doThis = ObjectMapper().registerModule(KotlinModule())
val or = ObjectMapper().registerKotlinModule()
```

# lateinit var

```
@Component
@ConfigurationProperties(prefix = "animals.com")
class AnimalsComProperties {

    lateinit var dogsUrl: URI
    lateinit var catsUrl: URI
}
```

**or...**



```
@Value(value = "\\${animals.com.dogsUrl}")
lateinit var dogsUrl: URI
```



# Dependency injection

## ...between Java and Kotlin

### package-private

```
@RestController  
@RequestMapping( ...value: "/animals")  
class AnimalsController(private val javaNonPublicService: JavaNonPublicService) {
```

### public

```
@RestController  
@RequestMapping( ...value: "/animals")  
class AnimalsController(private val javaPublicService: JavaPublicService) {
```

# Lombok and Kotlin 1

don't mix it!!!

```
@Getter  
public class Person {  
    private final String firstName = "Karol";  
}
```

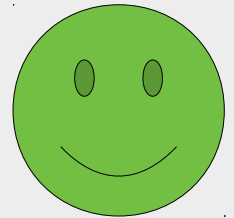


```
val person = Person()  
val firstName = person.firstName // COMPILATION ERROR !!!
```

# Lombok and Kotlin 2

## solution...

```
public class Person {  
    private final String firstName = "Karol";  
  
    public String getFirstName() {  
        return firstName;  
    }  
}
```



```
val person = Person()  
val firstName = person.firstName // it works. :-)
```

# Annotations

<https://kotlinlang.org/docs/reference/annotations.html>

```
@XmlElement(name = "userBaseIndividualV0")
data class PersonalInformation(

    @get: JacksonXmlProperty
    val familyName: String?,

    @get: JacksonXmlProperty
    val givenName: String?,

    @JsonFormat(pattern = "yyyyMMdd")
    @get: JacksonXmlProperty
    val birthDate: Date?
)
```



# How to start?

# CTRL + SHIFT + ALT + K on Java class in IntelliJ IDEA

```
package pl.karoldominiak.example;  
  
import ...  
  
@EnableAsync  
@SpringBootApplication  
public class ExampleApplication {  
  
    public static void main(final String[] args) { SpringApplication.run(ExampleApplication.class, args); }  
}
```

Convert Java to Kotlin



Some code in the rest of your project may require corrections after performing this conversion. Do you want to find such code and correct it too?

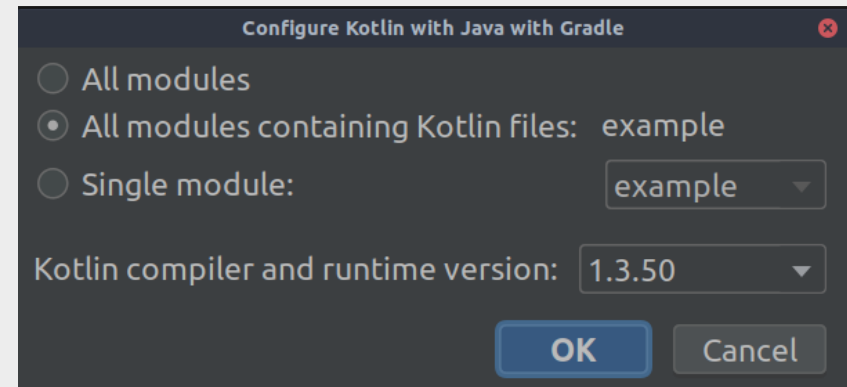
OK

Cancel

# CTRL + SHIFT + ALT + K on Java class in IntelliJ IDEA

Kotlin not configured

Configure Ignore



```
buildscript {  
    ext.kotlin_version = '1.3.50'  
    dependencies {  
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"  
    }  
}  
  
apply plugin: 'kotlin'  
  
compileKotlin {  
    kotlinOptions {  
        jvmTarget = "1.8"  
    }  
}  
  
compileTestKotlin {  
    kotlinOptions {  
        jvmTarget = "1.8"  
    }  
}
```


# Result should be like this..

it doesn't work properly in some cases and usually needs some tweaking. :-P

```
package pl.karoldominiak.example

import ...

@EnableAsync
@SpringBootApplication
open class ExampleApplication {
    companion object {
        @JvmStatic
        fun main(args: Array<String>) {
            SpringApplication.run(ExampleApplication::class.java, *args)
        }
    }
}
```






# Spring Initializr

<https://start.spring.io/>

← → ↻ start.spring.io



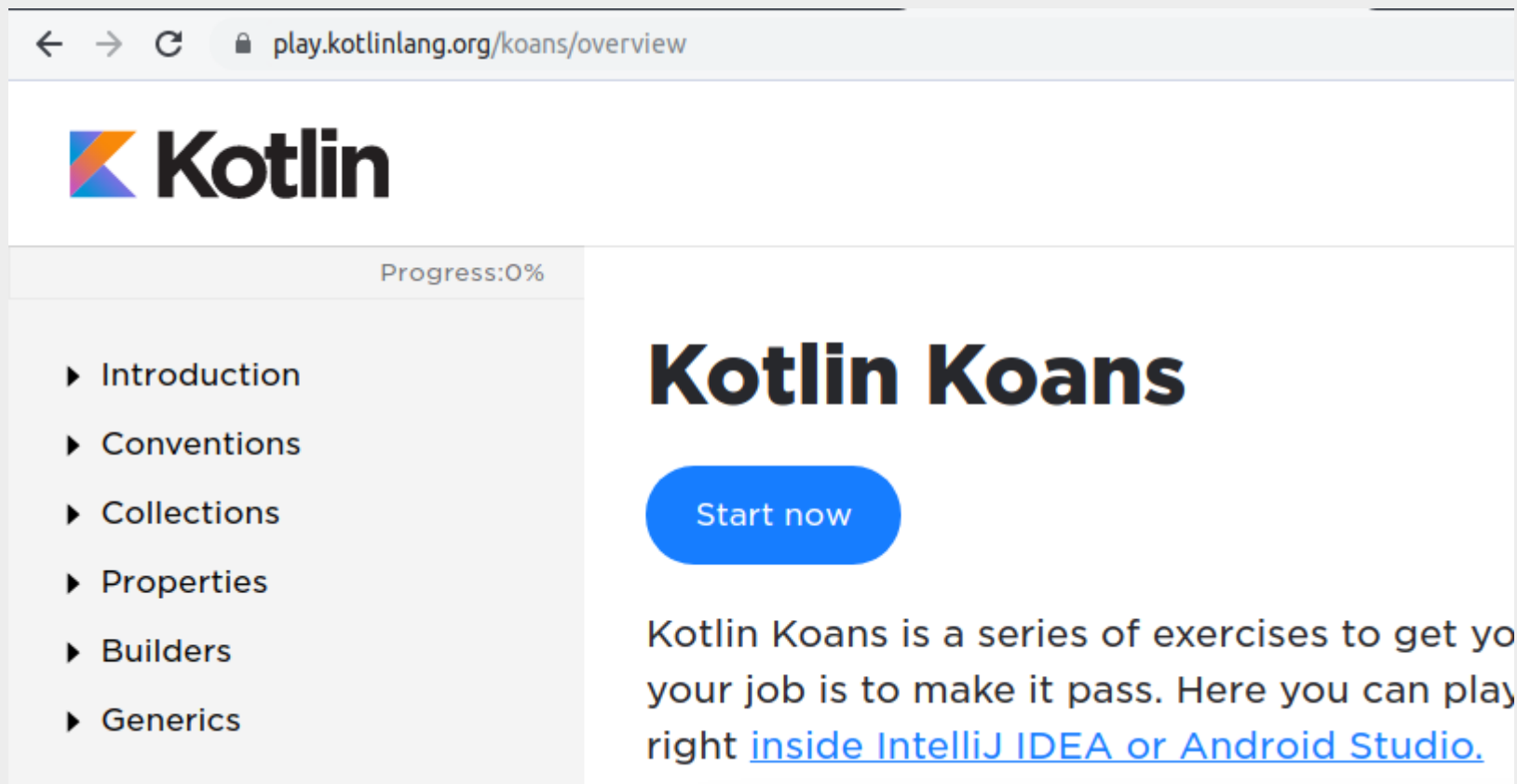
Spring **Initializr**  
Bootstrap your application

Project	Maven Project	Gradle Project		
Language	Java	Kotlin	Groovy	
Spring Boot	2.2.0 RC1	2.2.0 (SNAPSHOT)	2.1.10 (SNAPSHOT)	2.1.9

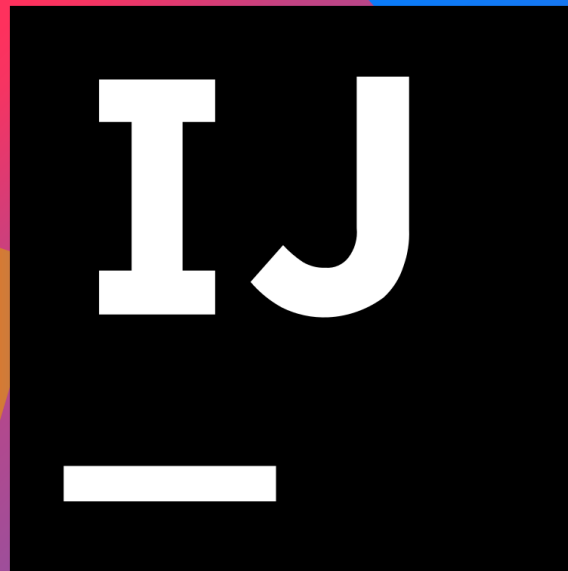
A red arrow points from the 'Gradle Project' option to the 'Kotlin' language option.

# Kotlin Koans

interactive tutorial in a browser  
<https://play.kotlinlang.org/koans>



# Use IntelliJ IDEA



The background is composed of several large, overlapping triangles in various colors: red, orange, yellow, teal, blue, and purple. The triangles are separated by thin white lines, creating a dynamic, geometric pattern.

**Thank you!**

**@copyright: Karol Dominiak**