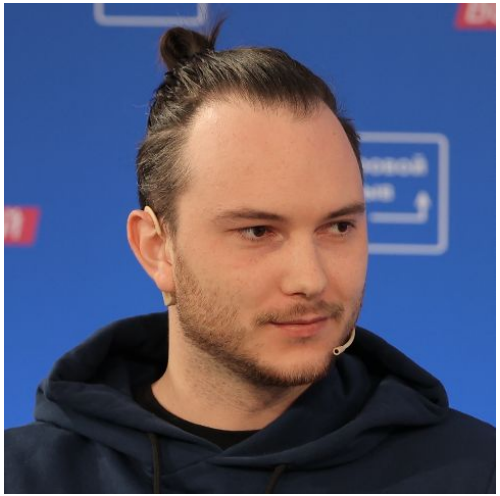
Бэкенд и Kotlin. Есть ли жизнь после Java?

Московский клуб программистов

07.07.2022

Кто я?



Вячеслав Аксёнов

Создаю бэкенды в финтехе больше 3х лет на Java и Kotlin

Написал больше 20 статей

Подготовился к Oracle Java SE Certification за 2 дня

linkedin.com/in/viacheslav-aksenov/

Twitter: @StratStuff

Что такое Java?

- строго типизированный
- компилируется в байткод
- байткод запускается на JVM вне зависимости от системы
- Широко распространен

Что такое Kotlin?

Язык, разработанный командой JetBrains под руководством Андрея Бреслава.

Выглядит красиво

Компилируется в Java байткод

Который может запускаться на JVM

Лоб в лоб. Привет, мир!

```kotlin
class TestClassKotlin

fun main() {
    println("Hello, Kotlin")
}
```

```java
public class TestClassJava {

    public static void main(String[] args) {
        System.out.println("Hello, Java");
    }
}
```

Обертки и примитивы. Привет, Java!

```java
static int pIntUndefined;

private static void printPrimitiveInt() {
    int pInt = 10;
    System.out.println("Print primitive int:");
    System.out.println(pInt);
    System.out.println(pIntUndefined);
}
```

```java
static Integer wIntUndefined;

private static void printWrappedInt() {
    Integer wInt = 20;
    System.out.println("Print wrapped Integer:");
    System.out.println(wInt);
    System.out.println(wIntUndefined);
}
```

Обертки и примитивы. Привет, Java!

```java
static int pIntUndefined;

private static void printPrimitiveInt() {
    int pInt = 10;
    System.out.println("Print primitive int:");
    System.out.println(pInt);
    System.out.println(pIntUndefined);
}
```

```java
static Integer wIntUndefined;

private static void printWrappedInt() {
    Integer wInt = 20;
    System.out.println("Print wrapped Integer:");
    System.out.println(wInt);
    System.out.println(wIntUndefined);
}
```

```
Print primitive int:
10
0
```

```
Print wrapped Integer:
20
null
```

Обертки и примитивы. Kotlin way

```
const val intUndefined: Int

fun printInt() {
    val intDefined: Int = 1
    println("print Kotlin Int:")
    println(intDefined)
    println(intUndefined)
}
```

Обертки и примитивы. Kotlin way

```
const val intUndefined: Int = 0

fun printInt() {
    val intDefined = 1
    println("print Kotlin Int:")
    println(intDefined)
    println(intUndefined)
}
```

Обертки и примитивы. Kotlin way

```kotlin
const val intUndefined: Int = 0

fun printInt() {
    val intDefined = 1
    println("print Kotlin Int:")
    println(intDefined)
    println(intUndefined)
}
```

```
print Kotlin Int:
1
0
```

Мутабельность. Kotlin way

```
val text = "super text"
var mutableText = "mutable text"
println(text + mutableText)
mutableText = "new text"
println(text + mutableText)
```

Мутабельность. Kotlin way

```
val text = "super text"
var mutableText = "mutable text"
println(text + mutableText)
mutableText = "new text"
println(text + mutableText)
```

```
super textmutable text
super textnew text
```

Мутабельность. Kotlin way

```
val text = "super text"
var mutableText = "mutable text"
println(text + mutableText)
mutableText = "new text"
println(text + mutableText)
```

```
text = "new text"
Val cannot be reassigned
Change to 'var'
```

Data Transfer Object

(или data class)
(или record?)

Data transfer object. Oldschool Java

```java
public class PokemonJava {
    private Long id;
    private String name;
    private Integer weight;

    public PokemonJava(Long id, String name, Integer weight) {
        this.id = id;
        this.name = name;
        this.weight = weight;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
```

```java
    public void setName(String name) {
        this.name = name;
    }

    public Integer getWeight() {
        return weight;
    }

    public void setWeight(Integer weight) {
        this.weight = weight;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        PokemonJava that = (PokemonJava) o;
        return Objects.equals(id, that.id) && Objects.equals(name, that.name) && Objects.equals(weight, that.weight);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, name, weight);
    }
}
```

Data transfer object. Oldschool Java

```
static PokemonJava pikachuJava = new PokemonJava(
        id: null,
        name: "pikachu",
        weight: 2
);
```

Data transfer object. Kotlin

```kotlin
data class Pokemon(
    val id: Long? = null,
    val name: String,
    val weight: Int
)
```

```kotlin
val pikachu = Pokemon(
    name = "Pikachu",
    weight = 2
)
```

Data transfer object. Newschool Java 16+

```java
public record PokemonRecord(
        Long id,
        String name,
        Integer weight
) {
}
```

```java
static PokemonRecord pikachuRecord = new PokemonRecord(
        id: null,
        name: "pikachu",
        weight: 2
);
```

Stream Api. Java

```
List<PokemonJava> notBulbasaur = pokemons
        .stream()
        .filter(pokemon -> !pokemon.getName().equals("bulbasaur"))
        .toList();

Map<String, List<PokemonJava>> groupedByName = pokemons
        .stream()
        .collect(Collectors.groupingBy(PokemonJava::getName));
```

Stream Api. Kotlin

```kotlin
val notBulbasaur: List<Pokemon> =
    pokemons.filter { it.name != "bulbasaur" }

val groupedByName =
    pokemons.groupingBy { it.name }
```

Optional

Null Safety?

Classic Java

```
public void test() {
    PokemonJava pokemonJava = new PokemonJava( id: 12L,  name: "pikachu",  weight: 3);
    PokemonJava pokemonJavaNull = null;
    doSomething(pokemonJava);
    doSomething(pokemonJavaNull);
}

private void doSomething(PokemonJava pokemon) {
    System.out.println(pokemon.getName());
}
```

Classic Java

```java
public void test() {
    PokemonJava pokemonJava = new PokemonJava( id: 12L,  name: "pikachu",  weight: 3);
    PokemonJava pokemonJavaNull = null;
    doSomething(pokemonJava);
    doSomething(pokemonJavaNull);
}

private void doSomething(PokemonJava pokemon) {
    System.out.println(pokemon.getName());
}
```

```
pikachu
Exception in thread "main" java.lang.NullPointerException Create breakpoint : Cannot invoke
 "com.example.pokemonapp.info.model.PokemonJava.getName()" because "pokemon" is null
    at Tests.doSomething(Tests.java:14)
    at Tests.test(Tests.java:10)
    at TestClassJava.main(TestClassJava.java:17)

Process finished with exit code 1
```

Kotlin with Java way

```java
    PokemonJava pokemonJava = null;
    doSomething(pokemonJava);
}

private void doSomething(PokemonJava pokemon) {
    System.out.println(pokemon.getWeight());
}
```

```kotlin
    val pokemon = Pokemon( id: 12,  name: "pikachu",  weight: 3)
    doSomething(pokemon)
    doSomething( pokemon: null)
}

fun doSomething(pokemon: Pokemon?) {
    println(pokemon.name)
}
```

Only safe (?.) or non-null asserted (!!.) calls are allowed

Only safe (?.) or non-null asserted (!!.) calls are allowed on a nullable receiver of type Pokemon?

Kotlin way

```
    val pokemon = Pokemon( id: 12,  name: "pikachu",  weight: 3)
    doSomething(pokemon)
    doSomething( pokemon: null)
}

fun doSomething(pokemon: Pokemon?) {
    println(pokemon?.name)
}
```

```
pikachu
null

Process finished with exit code 0
```

Kotlin way

```
    val pokemon = Pokemon( id: 12,  name: "pikachu",  weight: 3)
    doSomething(pokemon)
    doSomething( pokemon: null)
}

fun doSomething(pokemon: Pokemon?) {
    println(pokemon?.name)
}
```

```
    val pokemon = Pokemon( id: 12,  name: "pikachu",  weight: 3)
    doSomething(pokemon)
    doSomething( pokemon: null)
}

fun doSomething(pokemon: Pokemon) {
    println(pokemon.name)
}
```

Но ведь есть Lombok?

Что есть Lombok?

Project Lombok

Project Lombok is a java library that automatically plugs into your editor and build tools, spicing up your java. Never write another getter or equals method again, with one annotation your class has a fully featured builder, Automate your logging variables, and much more.

Reinier Zwitserloot

Roel Spilker

```
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
```

Lombok Java vs Kotlin. DTO

```java
public class PokemonJava {
    private Long id;
    private String name;
    private Integer weight;

    public PokemonJava(Long id, String name, Integer weight) {
        this.id = id;
        this.name = name;
        this.weight = weight;
    }

    public Long getId() { return id; }

    public void setId(Long id) { this.id = id; }

    public String getName() {
        return name;
    }

    public void setName(String name) { this.name = name; }

    public Integer getWeight() { return weight; }

    public void setWeight(Integer weight) { this.weight = weight; }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
```

```java
@Value
public class PokemonJava {
    Long id;
    String name;
    Integer weight;
}
```

```kotlin
data class Pokemon(
    val id: Long? = null,
    val name: String,
    val weight: Int
)
```

Lombok Java vs Kotlin. Val / var

```
val pokemonJava = new PokemonJava( id: 12L,  name: "pikachu",  weight: 3);
var mutablePokemon = new PokemonJava( id: 12L,  name: "pikachu",  weight: 3);
mutablePokemon = new PokemonJava( id: 13L,  name: "Bulbasaur",  weight: 33);
pokemonJava = new PokemonJava( id: 13L,  name: "Bulbasaur",  weight: 33);
```

Cannot assign a value to final variable 'pokemonJava'

```
val pokemonJava = Pokemon( id: 12L,  name: "pikachu",  weight: 3)
var mutablePokemon = Pokemon( id: 12L,  name: "pikachu",  weight: 3)
mutablePokemon = Pokemon( id: 13L,  name: "Bulbasaur",  weight: 33)
pokemonJava = Pokemon( id: 13L,  name: "Bulbasaur",  weight: 33)
```

Val cannot be reassigned

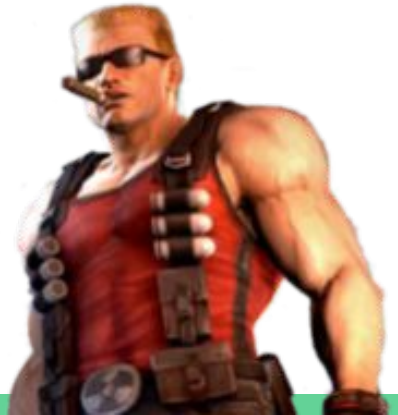
Lombok Java vs Kotlin. NonNull

```java
    PokemonJava pokemonJava = new PokemonJava( id: 12L,  name: "pikachu",  weight: 3);
    PokemonJava pokemonJavaNull = null;
    doSomething(pokemonJava);
    doSomething(pokemonJavaNull);
}

private void doSomething(@NonNull PokemonJava pokemon) {
    System.out.println(pokemon.getName());
}
```

```kotlin
    val pokemon = Pokemon( id: 12,  name: "pikachu",  weight: 3)
    doSomething(pokemon)
    doSomething( pokemon: null)
}

fun doSomething(pokemon: Pokemon) {
    println(pokemon.name)
}
```

Lombok Java bonuses

val
var
@Builder
@Data
@Getter
@Setter
@With
@Slf4j
....
Profit?



One annotation to rule them all?

```
@Entity
@Table(name = "pokemons", schema = DbSchemaName.CMS)
@Builder
@EqualsAndHashCode(callSuper = true)
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@SuppressWarnings({"PMD.AvoidDuplicateLiterals", "PMD.PreserveStackTrace", "PMD.TooManyFields", })
public class PokemonEntity extends AbstractEntity {
```

Kotlin достопримечательности

Kotlin context

```
fun doSomething(pokemon: Pokemon?) {
    val name = pokemon?.let { it: Pokemon
        println(it.name)
        it.name ^let
    } ?: "undefined name"
}
```

```
val name = pokemon
    ?.also { println(it.name) }
    ?.name
    ?: "undefined name"
```

```
fun withExample(pokemon: Pokemon) {
    with(pokemon) { this: Pokemon
        val nameWeight = name + weight
        // еще что-нибудь
    }
}
```

Kotlin упрощение

```kotlin
fun getSomething(pokemon: Pokemon) = "pretty" + pokemon.name
```

```java
public String getSomething(PokemonJava pokemon) {
    return "pretty" + pokemon.getName();
}
```

Kotlin extensions

```
    val pikachu = Pokemon( id: 12,  name: "pikachu",  weight: 3)
    val bulbazaur = Pokemon( id: 12,  name: "Bulbazaur",  weight: 55)
    println("pikachu: " + pikachu.myExtensionFun( default: 20))
    println("bulbazaur: " + bulbazaur.myExtensionFun( default: 20))
}

fun Pokemon.myExtensionFun(default: Int) =
    if (name == "pikachu") 33 else default
```

Kotlin extensions

```
    val pikachu = Pokemon( id: 12,  name: "pikachu",  weight: 3)
    val bulbazaur = Pokemon( id: 12,  name: "Bulbazaur",  weight: 55)
    println("pikachu: " + pikachu.myExtensionFun( default: 20))
    println("bulbazaur: " + bulbazaur.myExtensionFun( default: 20))
}

fun Pokemon.myExtensionFun(default: Int) =
    if (name == "pikachu") 33 else default
```

```
pikachu: 33
bulbazaur: 20

Process finished with exit code 0
```

Что рассмотрели?

- Java в бытовой жизни
- DTO
- Stream API
- Optional
- Lombok
- приятные фичи Kotlin

Что можно исследовать самостоятельно?

- Многопоточность в Kotlin (coroutine)
- Компиляция Java + Kotlin в рамках одного проекта
- Библиотеки для тестирования Kotlin
- Легковесные фреймворки Kotlin
- .... и другое

Спасибо за внимание!