Kotlin: MAX PAYNE

Artur Dryomov, Juno Lab

@arturdryomov

KONG WHISKEY
ROSCOE ST
THE BRO

Juno ❤️ Kotlin

Kotlin 💔 Juno

Juno Rider 🚙

- 3 years
- 100 % Kotlin
- Kotlin EAP in production 🔥
  - `1.1-EAP` — 6 months
  - `1.2-EAP` — 3 months
- 99.9 % crash-free
- 2 week sprints
- 4 developers, 1 QA

Juno Rider 🚙

- `cloc main/kotlin` — 56 K
- `cloc test/kotlin` — 49 K
- `cloc androidTest/kotlin` — 17 K

Style 💄

Java Tools

Nope ✨

Specifications 📐

- Java AST (Abstract Syntax Tree)
- PSI (Program Structure Interface)
- UAST (Universal AST)

Checkstyle

> Checkstyle works only with AST as we depend on ANTLR and Java grammar.

https://github.com/checkstyle/checkstyle/issues/4369

Kotlin Tools 🔦

- IntelliJ IDEA ❤️
- `detekt` + `ktlint`
- `android/lint`
- `kotlinc --force`

  ```kotlin
  tasks.withType<KotlinCompile> {
      kotlinOptions {
          allWarningsAsErrors = true
      }
  }
  ```

Infinity War 👿

```kotlin
val gameBuilder = StringBuilder()

gameBuilder.append("tic")
gameBuilder.append("tac")
gameBuilder.append("toe")

val game = gameBuilder.toString()
```

Infinity War 👿

```kotlin
val game = StringBuilder().also {
    it.append("tic")
    it.append("tac")
    it.append("toe")
}
.toString()
```

Infinity War 👿

```kotlin
val game = StringBuilder().let {
    it.append("tic")
    it.append("tac")
    it.append("toe")

    it.toString()
}
```

Infinity War 👿

```
val game = StringBuilder().apply {
    append("tic")
    append("tac")
    append("toe")
}
.toString()
```

Infinity War 👿

```kotlin
val game = StringBuilder().run {
    append("tic")
    append("tac")
    append("toe")

    toString()
}
```

Infinity War 👿

```kotlin
val game = with(StringBuilder()) {
    append("tic")
    append("tac")
    append("toe")

    toString()
}
```

Infinity War 🔦

- ☕?
- `CODE_STYLE.md`

Weird Parts 🎃

Destructuring

```kotlin
val (first, second) = Pair(1, 2)
val (first, second, third) = Triple(1, 2, 3)

listOf(1 to "one", 2 to "two").map { (number, text) ->

    println("number is $number, text is $text")
}
```

Destructuring

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

val (firstName, lastName) = Person("f", "l")
```

✨

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

val (firstName, lastName) = Person("f", "l") // It works!
```

Destructuring

```kotlin
data class Person(val firstName: String, val lastName: String)
{
    operator fun component1() = firstName
    operator fun component2() = lastName
}
```

Destructuring 🔦

❤️ `Pair`, `Triple`

💔 `data class`

`null` safety?

```
Observable.just(null)
```

✨

```
public static <T> Observable<T> just(T item) {
    ObjectHelper.requireNonNull(item, "The item is null");
    ...
```

`null` safety!!

`@Nullable` + `@NonNull`

`null` safety!! 🔦

- Docs.
- Annotations.
- Abstractions.

RxJava 2

```
val o: Observable<Int?> = Observable.just(null)

// Runtime: NullPointerException
```

✨

```
public static <T> Observable<T> just(T item) {
    ObjectHelper.requireNonNull(item, "The item is null");
    ...
```

`Optional<T>`

- Swift: `?`, `nil` ➡️ `enum Optional`
- Kotlin: `?`, `null` ➡️ `null`

Koptional 🔦

https://github.com/gojuno/koptional

```
val some = Some(value)
val none = None

val optional = nullableValue.toOptional()
val nullable = o.toNullable()
```

Mockito

```kotlin
interface Api {
    fun call(id: String)
}

class Service(private val api: Api) {
    fun call() = api.call(generateId())
}
```

```kotlin
@Test fun test() {
    val api = Mockito.mock<Api>()
    Service(api).call()

    verify(api).call(Mockito.any())
}

// Runtime: NullPointerException
```

Mockito

```
class Mockito {

    public static <T> T any() {
        return null;
    }
}
```

Mockito

```
fun <T> anything(): T = Mockito.any<T>()
```

```
@Test fun test() {
    val api = Mockito.mock<Api>()
    Service(api).call()

    verify(api).call(anything())
}
```

Mockito 🔦

```
fun <T> eq(value: T): T = Mockito.eq<T>(obj)
fun <T> anything(): T = Mockito.any<T>()

fun <T> whenever(call: T) = Mockito.`when`(call)
```

Gson

```
data class Model {

    @SerializedName("model_field")
    val field: String

}

val model = Gson().fromJson("{}", Model::javaClass)

// Runtime: OK

model.field.toString()

// Runtime: NullPointerException
```

Gson

Java Reflection

Gson

```kotlin
data class Model {

    @SerializedName("model_field")
    val field: String

} : Validatable {

    override fun validate() {
        if (field == null) throw ParseException()
    }
}
```

Gson 🔦

Moshi vs. Jackson

Kotlin Reflection 👼 (2.5 MiB)

`null` safety!! 📖

Kotlin: The Problem with `null`

https://arturdryomov.online/posts/kotlin-the-problem-with-null/

Optimizations 🚀

`inline`

```
private fun createId(): Int = 42
```

vs.

```
@Suppress("nothing_to_inline")
private inline fun createId(): Int = 42
```

`inline`

⭐ https://youtrack.jetbrains.com/issue/KT-16769

Revisit compiler and IDE warning about inline functions

`const`

```
companion object {
    private val NUMBER = 42
}
```

⬇

```
public static final class Companion {
    private final int NUMBER = 42;

    private final int getNUMBER() {
        return NUMBER;
    }
}
```

`const`

```
companion object {
    private const val NUMBER = 42
}
```

⬇

```
public static final class Companion {
    private static final int NUMBER = 42;
}
```

`const`

- Primitives + `String`
- `kotlinc`?

Tools ➡️ Kotlin ➡️ Show Kotlin Bytecode

Tooling 🔨

Bugs 🐛

```
fun multiply(double: Double) = double * double
```

⬇️ 1.1.3

```
org.jetbrains.kotlin.kapt3.diagnostic.KaptError
Error while annotation processing
```

Bugs 🐛

⭐ https://youtrack.jetbrains.com/issue/KT-18377

Syntax error while generating kapt stubs

Bugs 🐛

Are a thing!

Assemble 🕘

- `javac`
- ➕ `kotlinc`
- ➕ `kapt`
- ✖️ 💯

`kapt`

Kotlin Annotation Processing Tool

- Breaks incremental compilation.
- Throws errors.

  ```
  *_MemberInjector.java: error: package does not exist
  ```

`kapt` 📖

A Dagger to Remember

https://arturdryomov.online/posts/a-dagger-to-remember/

Assemble 🕘

- 💻 — `i5-5257U @ 2.70GHz`
- 📺 — `i7-6700K @ 4.00GHz`

Mainframer

https://github.com/gojuno/mainframer

```
$ rsync laptop mainframe
$ make
$ rsync mainframe laptop
```

Mainframer

```
$ ./gradlew assembleDebug
```

👎 4 minutes 10 seconds 🔥

```
$ mainframer ./gradlew assembleDebug
```

👍 1 minute 10 seconds

Assemble 🕘 🔦

- Remove everything possible.
- Gradle build cache.
- Mainframer.

Gradle Kotlin DSL 🎲

Kotlin versions

- Gradle 4.2 — 1.1.4-3
- Project — 1.2.0-beta-31

```
Runtime files in the classpath should have the same version.
These files were found in the classpath:

.../dists/.../kotlin-stdlib-1.1.4-3.jar (version 1.1)
.../caches/.../kotlin-stdlib-1.2.0-beta-31.jar (version 1.2)
```

Kotlin versions 🔦

https://github.com/gradle/kotlin-dsl/issues/519

Groovy

```
unitTests.all {
    maxParallelForks = 2
}
```

⬇

```
unitTests.all(KotlinClosure1<Test, Test>({
    apply {
        maxParallelForks = 2
    }
}, this))
```

Groovy

```
import groovy.lang.Closure;

public void unitTests(Closure closure)
```

⭐ https://issuetracker.google.com/issues/78547461

Groovy

- 2012: `groovy.lang.Closure`
- 2017: `org.gradle.api.Action`

Groovy 🔦

Promote DSL-agnostic API

Gradle Kotlin DSL

- IDE 🔥
- Dynamic Groovy 🔥
- Workflow 🔥
- Alpha! 👻

Spek

JUnit

```kotlin
class SystemTest {

    lateinit var system: System

    @BeforeEach fun create() {
        system = System()
    }

    @BeforeEach fun start() {
        system.boot()
    }

    @Test fun `has boot status`() {
        assertThat(system.status()).isEqualTo(BOOTED)
    }

    @Test fun `has positive uptime`() {
        assertThat(system.uptime()).isNotZero()
    }
}
```

Spek

```
class SystemSpec : Spek({

    val system by memoized { System() }

    context("boot") {

        beforeEachTest {
            system.boot()
        }

        it("has boot status") {
            assertThat(system.status()).isEqualTo(BOOTED)
        }

        it("has positive uptime") {
            assertThat(system.uptime()).isNotZero()
        }
    }
})
```

Versions

- `1.0.89`: `DescribeBody` ➡️ `Dsl` 💣
- `1.1.0-beta3`: `Dsl` ➡️ `SpecBody` 💣
- `1.0`, `1.1.19`: do not exist.
- JUnit 4 ➡️ 5 💣

`memoized` before and after

```kotlin
class JustSpec : Spek({

    val system by memoized { System() }

    beforeEachTest {
        println("before $system") // system_before
    }

    afterEachTest {
        println("after $system") // system_after
    }
})

// system_before != system_after
```

`memoized` before and after 🔦

⭐ https://github.com/JetBrains/spek/issues/213

Memoized value is not the same before and after test

JUnit

- JUnit 5!
  - Android Studio 💔
- JUnit 4?
  - `@RunWith(JUnitPlatform::class)`
- `fon`, `fit` ➡️ Spek IJ plugin.
  - JUnit 4 💔
- ✨ Fun ✨

Spek

- `jetbrains/spek` ➡️ `spekframework/spek`
- Compatibility 💣
- Docs 💣
- Bugs 💣
- 100 K tests 😡
- `2.0` 👼

Spek 🔦

Spectrum, KotlinTest

Conclusion 🎱

https://youtrack.jetbrains.com/issues/KT

> ...
>
> Kotlin is a thin layer on top of Java, and so it perpetuates through a lot of the Javaisms in its model.
>
> ...

Chris Lattner (Swift, LLVM, Clang) 🐦

🎩

- Java — Executive Committee
  - Oracle, Red Hat, Intel, IBM, Eclipse, Twitter, JetBrains...
  - 24 members
- Kotlin — Language Committee
  - JetBrains, Google
  - 3 members

THE FALL OF MAX PAYNE

Conclusion 🎱

- JVM 👻
- Tooling 👻
- Language ❤️

KONG WHISKEY
ROSCOE ST
THE BRO