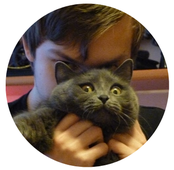
KotlinConf

lyft

Testing Kotlin at Scale: Spek



Artem Zinnatullin
@artem_zin

talk passing

KotlinConf

- Productivity

KotlinConf

- Productivity
- Reviewability

KotlinConf

- Productivity
- Reviewability
- **Maintainability**

KotlinConf

- Patterns
- Principles
- OOP/FP
- Common Sense

But

KotlinConf

We focus on production code

KotlinConf

- Productivity
- Reviewability
- Maintainability
- Developer Happiness

When it comes to test code

KotlinConf

- Patterns
- Principles
- OOP/FP
- Common Sense

When it comes to test code

Aren’t we being hypocrite to ourselves?

Do we *really* understand how much **Effort** do we put in Tests?

KotlinConf

There is a simple metric though

KotlinConf

Tests can actually take more LOCs than production code

RxJava (library):

- $ cloc src/main — 82 K LOC

https://github.com/ReactiveX/RxJava

KotlinConf

RxJava (library):

- $ cloc src/main — 82 K LOC
- $ cloc src/test — 159 K LOC

https://github.com/ReactiveX/RxJava

November 2017

KotlinConf

RxJava

159 / 82 = 1,94 x

KotlinConf

RxJava: 159 / 82 = 1,94 x

OkHttp: 27 / 15 = 1,8 x

KotlinConf

RxJava: 159 / 82 = 1,94 x

OkHttp: 27 / 15 = 1,8 x

Retrofit: 4,9 / 2,8= 1,75 x

!

😐

KotlinConf

Is there something common between these projects?

KotlinConf

- Assertion libraries

KotlinConf

- Assertion libraries ✅

KotlinConf

- Assertion libraries ✅

- Mocking libraries

KotlinConf

- Assertion libraries ✅

- Mocking libraries ✅

KotlinConf

Test framework though?

KotlinConf

They all use JUnit (4)

And it works

KotlinConf

JUnit 4 ❤️

- It’s robust

KotlinConf

JUnit 4 ❤️

- It’s robust
- It’s straightforward

KotlinConf

JUnit 4 ❤️

- It’s robust
- It’s straightforward
- You don’t have to debug it

KotlinConf

JUnit 4 ❤️

- It’s robust
- It’s straightforward
- You don’t have to debug it
- All build systems and IDEs support it

KotlinConf

JUnit 4 ❤️

- It’s robust
- It’s straightforward
- You don’t have to debug it
- All build systems and IDEs support it
- Everybody is familiar with it

KotlinConf

JUnit 4 ❤️

- It’s robust
- It’s straightforward
- You don’t have to debug it
- All build systems and IDEs support it
- Everybody is familiar with it
- It’s a standard.

KotlinConf

But it has problems

KotlinConf

“Code Repetition”

```kotlin
@Test
fun updateCoarseLocationSync_Foregrounded() {
    `when`(appForegroundDetector.isForegrounded).thenReturn(true)

    val first = AndroidLocationBuilder()
            .withProvider(AndroidLocation.Provider.FUSED)
            .withLat(0.0)
            .withLng(0.0)
            .withTime(1L)
            .build()
    locationIngestService.updateCoarseLocationSync(first)

    val argumentCaptor = ArgumentCaptor.forClass(IngestLocationsRequestDTO::class.java)
    verify(locationIngestApi, times(1)).postLocations(argumentCaptor.capture())

    val value = argumentCaptor.value

    assertThat(value).isNotNull()
    assertThat(value.locations).isNotNull().hasSize(1)
    assertThat(value.locations[0]).isNotNull()
    assertThat(value.locations[0].source).isEqualTo(Location.SIGNIFICANT_LOCATION_CHANGE_FG)
}
```

```kotlin
@Test
fun updateCoarseLocationSync_NotForegrounded() {
    `when`(appForegroundDetector.isForegrounded).thenReturn(false)

    val first = AndroidLocationBuilder()
            .withProvider(AndroidLocation.Provider.FUSED)
            .withLat(0.0)
            .withLng(0.0)
            .withTime(1L)
            .build()
    locationIngestService.updateCoarseLocationSync(first)

    val argumentCaptor = ArgumentCaptor.forClass(IngestLocationsRequestDTO::class.java)
    verify(locationIngestApi, times(1)).postLocations(argumentCaptor.capture())

    val value = argumentCaptor.value

    assertThat(value).isNotNull()
    assertThat(value.locations).isNotNull().hasSize(1)
    assertThat(value.locations[0]).isNotNull()
    assertThat(value.locations[0].source).isEqualTo(Location.SIGNIFICANT_LOCATION_CHANGE_BG)
}
```

```
@Test
fun updateCoarseLocationSync_NotForegrounded() {

    `when`(appForegroundDetector.isForegrounded).thenReturn(false)

    val first = AndroidLocationBuilder()
            .withProvider(AndroidLocation.Provider.FUSED)
            .withLat(0.0)
            .withLng(0.0)
            .withTime(1L)
            .build()
    locationIngestService.updateCoarseLocationSync(first)

    val argumentCaptor = ArgumentCaptor.forClass(IngestLocationsRequestDTO::class.java)
    verify(locationIngestApi, times(1)).postLocations(argumentCaptor.capture())

    val value = argumentCaptor.value

    assertThat(value).isNotNull()
    assertThat(value.locations).isNotNull().hasSize(1)
    assertThat(value.locations[0]).isNotNull()
    assertThat(value.locations[0].source).isEqualTo(Location.SIGNIFICANT_LOCATION_CHANGE_BG)
}
```

```kotlin
@Test
fun updateCoarseLocationSync_NotForegrounded() {
    `when`(appForegroundDetector.isForegrounded).thenReturn(false)

    val first = AndroidLocationBuilder()
            .withProvider(AndroidLocation.Provider.FUSED)
            .withLat(0.0)
            .withLng(0.0)
            .withTime(1L)
            .build()
    locationIngestService.updateCoarseLocationSync(first)

    val argumentCaptor = ArgumentCaptor.forClass(IngestLocationsRequestDTO::class.java)
    verify(locationIngestApi, times(1)).postLocations(argumentCaptor.capture())

    val value = argumentCaptor.value

    assertThat(value).isNotNull()
    assertThat(value.locations).isNotNull().hasSize(1)
    assertThat(value.locations[0]).isNotNull()
    assertThat(value.locations[0].source).isEqualTo(Location.SIGNIFICANT_LOCATION_CHANGE_BG)
}
```

JUnit 4: “Code Repetition” Problem

KotlinConf

“Test case/context naming”

```kotlin
@Test
fun removeAccessTokenShouldStopLoggedinScopeAndStartLoggedOutScope() {
    accessTokenRepository.removeAccessToken()

    val scopeChange1 = scopeManager.scopeChanges[0]

    assertThat(scopeChange1.scope).isEqualTo(PassengerScopes.LOGGED_IN)
    assertThat(scopeChange1.started).isFalse()

    val scopeChange2 = scopeManager.scopeChanges[1]

    assertThat(scopeChange2.scope).isEqualTo(PassengerScopes.LOGGED_OUT)
    assertThat(scopeChange2.started).isTrue()
}
```

```kotlin
@Test
fun `remove access token should stop logged in scope and start logged out scope`() {

    accessTokenRepository.removeAccessToken()

    val scopeChange1 = scopeManager.scopeChanges[0]

    assertThat(scopeChange1.scope).isEqualTo(PassengerScopes.LOGGED_IN)
    assertThat(scopeChange1.started).isFalse()

    val scopeChange2 = scopeManager.scopeChanges[1]

    assertThat(scopeChange2.scope).isEqualTo(PassengerScopes.LOGGED_OUT)
    assertThat(scopeChange2.started).isTrue()
}
```

JUnit4: “Test case/context naming” Problem

Still bad 😿

```
@Test
fun `remove access token should stop logged in
scope and start logged out scope`()
```

KotlinConf

“What does this test test?”

```kotlin
@Test
fun `remove access token should stop logged in scope and start logged out scope`() {
    accessTokenRepository.removeAccessToken()

    val scopeChange1 = scopeManager.scopeChanges[0]

    assertThat(scopeChange1.scope).isEqualTo(PassengerScopes.LOGGED_IN)
    assertThat(scopeChange1.started).isFalse()

    val scopeChange2 = scopeManager.scopeChanges[1]

    assertThat(scopeChange2.scope).isEqualTo(PassengerScopes.LOGGED_OUT)
    assertThat(scopeChange2.started).isTrue()
}
```

```kotlin
@Test
fun `remove access token should stop logged in scope and start logged out scope`() {
    accessTokenRepository.removeAccessToken()

    val scopeChange1 = scopeManager.scopeChanges[0]

    assertThat(scopeChange1.scope).isEqualTo(PassengerScopes.LOGGED_IN)
    assertThat(scopeChange1.started).isFalse()

    val scopeChange2 = scopeManager.scopeChanges[1]

    assertThat(scopeChange2.scope).isEqualTo(PassengerScopes.LOGGED_OUT)
    assertThat(scopeChange2.started).isTrue()
}
```

```kotlin
@Test
fun `remove access token should stop logged in scope and start logged out scope`() {
    accessTokenRepository.removeAccessToken()

    val scopeChange1 = scopeManager.scopeChanges[0]

    assertThat(scopeChange1.scope).isEqualTo(PassengerScopes.LOGGED_IN)
    assertThat(scopeChange1.started).isFalse()

    val scopeChange2 = scopeManager.scopeChanges[1]

    assertThat(scopeChange2.scope).isEqualTo(PassengerScopes.LOGGED_OUT)
    assertThat(scopeChange2.started).isTrue()
}
```

4 checks 😿

Some motivation speech here

Spek

Spek v0, 2012

Commits on Dec 15, 2012

Redone project due to failure

hhariri committed on Dec 15, 2012

ab43a96

Newer Older

Spek v0, 2012

Tree: ab43a96eb5 ▾ **spek** / **lib** /

Create new file | Upload files | Find file | History

hhariri Redone project due to failure — Latest commit ab43a96 on Dec 15, 2012

..

hamcrest-core-1.3-javadoc.jar	Redone project due to failure	5 years ago
hamcrest-core-1.3-sources.jar	Redone project due to failure	5 years ago
hamcrest-core-1.3.jar	Redone project due to failure	5 years ago
junit-4.11-javadoc.jar	Redone project due to failure	5 years ago
junit-4.11-sources.jar	Redone project due to failure	5 years ago
junit-4.11.jar	Redone project due to failure	5 years ago
kotlin-runtime.jar	Redone project due to failure	5 years ago
mockito-all-1.9.5-javadoc.jar	Redone project due to failure	5 years ago
mockito-all-1.9.5-sources.jar	Redone project due to failure	5 years ago
mockito-all-1.9.5.jar	Redone project due to failure	5 years ago

Spek v0, 2012

Kotlin 0.4.297

Spek v0, 2012

FAQ

Q: What is Kotlin?

Kotlin is an Apache 2 OSS Language targetted at the JVM and JavaScript and is developed by JetBrains It is aimed at being a concise modern language for general use. It also rocks!

Q: Is Kotlin free to use?

While this is not a Kotlin FAQ, it is important to note that Kotlin is free to use and you can use the command line or the Community Edition of IntelliJ to develop with it (which is free and OSS). Obviously IntelliJ Ultimate also works! There's also an Eclipse plugin in the works. Check the project site for updates.

Spek v0, 2012

```
8   spec public fun calculatorSpecs() {
9
10      given("a calculator")
11      {
12          val calculator = Calculator()
13
14          on("calling sum with two numbers")
15          {
16
17              val sum = calculator.sum(2, 4)
18
19
20              it("should return the result of adding the first number to the second number")
21              {
22                  shouldEqual(6, sum)
23              }
24          }
```

Hadi Hariri @JetBrains

2016

Spek v1.0.25, 2016

We started to feel JUnit 4 problems really badly

Spek v1.0.25, 2016

JetBrains / **spek**

raniejade / **kspec**

kotlintest / **kotlintest**

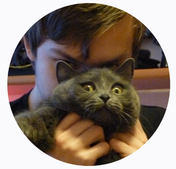
Spek v1.0.25, 2016

JetBrains / spek

Spek v1.0.25, 2016

JetBrains / spek

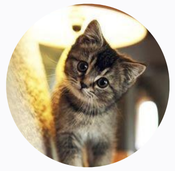
We started using it



Spek v1.0.25, 2016

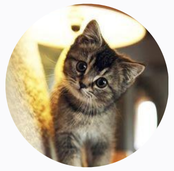
JetBrains / **spek**

raniejade / **kspec**



Spek v1.0.25, 2016

JetBrains / spek



Please fix Spek versioning, compatibility and publishing. #170

Edit New issue

Open artem-zinnatullin opened this issue on Jan 24 · 7 comments

artem-zinnatullin commented on Jan 24 Collaborator

Currently Spek has veery crazy versioning principles.

- We're currently on `1.0.25` which is known as pre-v1...
- `1.0.89+` known as post-v1, somewhere near to this version `DescribeBody` was renamed to `Dsl`, **compatibility broken**.
- `1.1.0-beta3` is version shown by badge in GitHub README, in this version `Dsl` was renamed to `SpecBody`, **compatibility broken again**.
- Spek website's root page says that current version is `1.0`, but there are **no jars** with such version.
- Spek website's documentation page points to *latest* version which is `1.1.0-beta3` while Spek website's downloads page points to `1.0` which is actually `1.0.89`.
- Also, there is `1.1.19` in both repos, but `1.1.0-beta3` pointed everywhere else as latest...
- And there is a GitHub releases page which only contains `1.1.0-beta3` and `v1`, no mentions of stable `1.0.89` or actually latest `v1.1.19`.

To get these versions you need to find bintray repo which is used in `build.gradle` in docs example, some artifacts are published to jcenter and 1.0.25 and some other versions live in internal jetbrains repo.

I mean thank you for Spek, we have lots of specs in the project, but we're trying to migrate to *latest* Spek and this is just insane...

24 4 6

Assignees

artem-zinnatullin

Labels

enhancement

Projects

None yet

Milestone

2.0.0

Notifications

Unsubscribe

You're receiving notifications because you authored the thread.

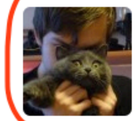
2 participants

Lock conversation

Please fix Spek versioning, compatibility and publishing. #170

Edit New issue

Open artem-zinnatullin opened this issue on Jan 24 · 7 comments



artem-zinnatullin commented on Jan 24 Collaborator

Currently Spek has veery crazy versioning principles.

- We're currently on `1.0.25` which is known as pre-v1...
- `1.0.89+` known as post-v1, somewhere near to this version `DescribeBody` was renamed to

Assignees

artem-zinnatullin

Labels

enhancement

I see what you did there, Hadi

Spek 2.x

KotlinConf

Spek 2.x
: BetterTestingFuture

```kotlin
class SuperTypicalJUnitTest {

    val calculator = Calculator()

    @Test
    fun `2 + 4 = 6`() {
        val result = calculator.add(2, 4)
        assertEquals(6, result)
    }
}
```

Typical Spec

```kotlin
class CalculatorSpec : Spek({

    val calculator by memoized { Calculator() }

    context("2 + 4") {

        val result by memoized { calculator.add(2, 4) }

        it("equals 6") {
            assertEquals(6, result)
        }
    }
})
```

Typical Spec

Spek: Basic API

```
context(“2 + 4") {
}
```

Spek: Basic API

```
context("2 + 4") {
}

describe(“2 + 4") {
}
```

Spek: Basic API

```
context("2 + 4") {
}

describe("2 + 4") {
}

given(“2 + 4") {
}
```

Spek: Basic API

```
describe("2 + 4") {
}

context("2 + 4") {
}

given("2 + 4") {
}
```

```
group(“2 + 4")
```

Spek: Basic API

```
group("") ~= Test class in JUnit
```

```
group("") ~= Test class in JUnit
```

You can nest groups **naturally**

```
it("equals 6") {
    assertThat(result).isEqualTo(6)
}
```

Spek: Basic API

```kotlin
it("equals 6") {
    assertThat(result).isEqualTo(6)
}

@Test
fun `2 + 4 = 6`() {
    val result = calculator.add(2, 4)
    assertEquals(6, result)
}
```

Spek: Basic API

```
it("equals 6") {
    assertThat(result).isEqualTo(6)
}

@Test
fun `2 + 4 = 6`() {
    val result = calculator.add(2, 4)
    assertEquals(6, result)
}
```

Spek: Basic API

```kotlin
it("equals 6") {
    assertThat(result).isEqualTo(6)
}

@Test
fun `2 + 4 = 6`() {
    val result = calculator.add(2, 4)
    assertEquals(6, result)
}
```

Spek: Basic API

```kotlin
it("equals 6") {
    assertThat(result).isEqualTo(6)
}

@Test
fun `2 + 4 = 6`() {
    val result = calculator.add(2, 4)
    assertEquals(6, result)
}
```

Spek: Basic API

```
it("") = @Test in JUnit
```

```
it("") = @Test in JUnit
```

You can have as many `it` in a `group` **as needed**

Groups and Tests create natural structure
that scales very well

```kotlin
val calculator by memoized { Calculator() }
```

You avoid state sharing betwen tests with `memoized`

Let’s rewrite real JUnit test with Spek

```kotlin
@Test
fun `remove access token should stop logged in scope and start logged out scope`() {
    accessTokenRepository.removeAccessToken()

    val scopeChange1 = scopeManager.scopeChanges[0]

    assertThat(scopeChange1.scope).isEqualTo(PassengerScopes.LOGGED_IN)
    assertThat(scopeChange1.started).isFalse()

    val scopeChange2 = scopeManager.scopeChanges[1]

    assertThat(scopeChange2.scope).isEqualTo(PassengerScopes.LOGGED_OUT)
    assertThat(scopeChange2.started).isTrue()
}
```

4 checks 😿

```kotlin
@Test
fun `remove access token should stop logged in scope and start logged out scope`() {
    accessTokenRepository.removeAccessToken()

    val scopeChange1 = scopeManager.scopeChanges[0]

    assertThat(scopeChange1.scope).isEqualTo(PassengerScopes.LOGGED_IN)
    assertThat(scopeChange1.started).isFalse()

    val scopeChange2 = scopeManager.scopeChanges[1]

    assertThat(scopeChange2.scope).isEqualTo(PassengerScopes.LOGGED_OUT)
    assertThat(scopeChange2.started).isTrue()
}
```

```
context("remove access token") {
}
```

Let’s rewrite real JUnit test with Spek

```
context("remove access token") {
    beforeEachTest {
        accessTokenRepository.removeAccessToken()
    }
}
```

Let’s rewrite real JUnit test with Spek

```
context("remove access token") {

    beforeEachTest {
        accessTokenRepository.removeAccessToken()
    }
}
```

Let’s rewrite real JUnit test with Spek

```kotlin
@Test
fun `remove access token should stop logged in scope and start logged out scope`() {
    accessTokenRepository.removeAccessToken()

    val scopeChange1 = scopeManager.scopeChanges[0]

    assertThat(scopeChange1.scope).isEqualTo(PassengerScopes.LOGGED_IN)
    assertThat(scopeChange1.started).isFalse()

    val scopeChange2 = scopeManager.scopeChanges[1]

    assertThat(scopeChange2.scope).isEqualTo(PassengerScopes.LOGGED_OUT)
    assertThat(scopeChange2.started).isTrue()
}
```

```kotlin
@Test
fun `remove access token should stop logged in scope and start logged out scope`() {
    accessTokenRepository.removeAccessToken()

    val scopeChange1 = scopeManager.scopeChanges[0]

    assertThat(scopeChange1.scope).isEqualTo(PassengerScopes.LOGGED_IN)
    assertThat(scopeChange1.started).isFalse()

    val scopeChange2 = scopeManager.scopeChanges[1]

    assertThat(scopeChange2.scope).isEqualTo(PassengerScopes.LOGGED_OUT)
    assertThat(scopeChange2.started).isTrue()
}
```

```
describe("first scope change") {
}
```

Let’s rewrite real JUnit test with Spek

```kotlin
describe("first scope change") {
    val firstScopeChange by memoized { scopeManager.scopeChanges[0] }
}
```

Let’s rewrite real JUnit test with Spek

```kotlin
@Test
fun `remove access token should stop logged in scope and start logged out scope`() {
    accessTokenRepository.removeAccessToken()

    val scopeChange1 = scopeManager.scopeChanges[0]

    assertThat(scopeChange1.scope).isEqualTo(PassengerScopes.LOGGED_IN)
    assertThat(scopeChange1.started).isFalse()

    val scopeChange2 = scopeManager.scopeChanges[1]

    assertThat(scopeChange2.scope).isEqualTo(PassengerScopes.LOGGED_OUT)
    assertThat(scopeChange2.started).isTrue()
}
```

```kotlin
describe("first scope change") {

    val firstScopeChange by memoized { scopeManager.scopeChanges[0] }

    it("is 'logged in' scope") {
        assertThat(firstScopeChange.scope).isEqualTo(LOGGED_IN)
    }

    it("is not started") {
        assertThat(firstScopeChange.started).isFalse()
    }
}
```

Let’s rewrite real JUnit test with Spek

```kotlin
class JUnitTest {

    private var scopeManager = MockScopeManager()

    private val accessTokenRepository = AccessTokenRepository(
            RuntimeEnvironment.application,
            scopeManager
    )

    lateinit var scopeChange1: MockScopeManager.ScopeChange
    lateinit var scopeChange2: MockScopeManager.ScopeChange

    @Before
    fun `remove access token`() {
        accessTokenRepository.removeAccessToken()
        scopeChange1 = scopeManager.scopeChanges[0]
        scopeChange2 = scopeManager.scopeChanges[1]
    }

    @Test
    fun `first scope change is 'logged in' scope`() {
        assertThat(scopeChange1.scope).isEqualTo(LOGGED_IN)
    }

    @Test
    fun `first scope change is not started`() {
        assertThat(scopeChange1.started).isFalse()
    }

    @Test
    fun `second scope change is 'logged out' scope`() {
        assertThat(scopeChange2.scope).isEqualTo(LOGGED_OUT)
    }

    @Test
    fun `first scope change is started`() {
        assertThat(scopeChange2.started).isTrue()
    }
}
```

JUnit is not structured, it’s flat

Let’s rewrite real JUnit test with Spek

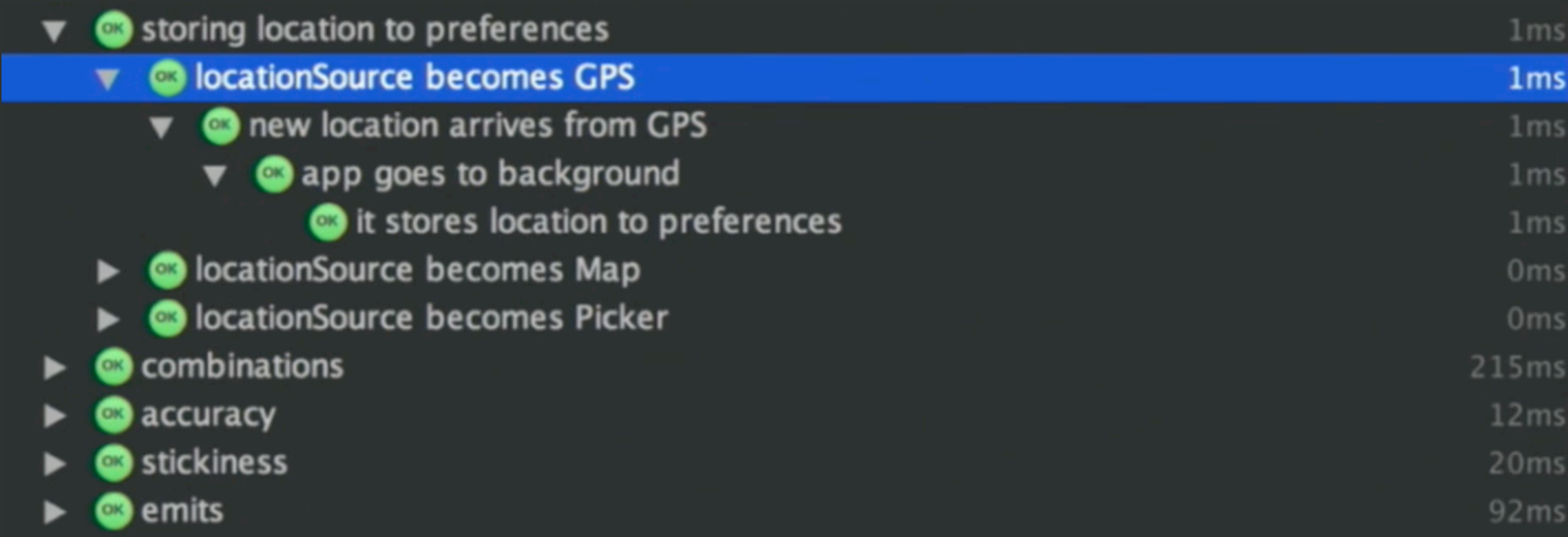
```kotlin
class Spec : Spek({

    val scopeManager by memoized { MockScopeManager() }

    val accessTokenRepository by memoized {
        AccessTokenRepository(RuntimeEnvironment.application, scopeManager)
    }

    context("remove access token") {

        beforeEachTest {
            accessTokenRepository.removeAccessToken()
        }

        describe("first scope change") {

            val firstScopeChange by memoized { scopeManager.scopeChanges[0] }

            it("is 'logged in' scope") {
                assertThat(firstScopeChange.scope).isEqualTo(LOGGED_IN)
            }

            it("is not started") {
                assertThat(firstScopeChange.started).isFalse()
            }
        }

        describe("second scope change") {

            val secondScopeChange by memoized { scopeManager.scopeChanges[1] }

            it("is 'logged out' scope") {
                assertThat(secondScopeChange.scope).isEqualTo(LOGGED_OUT)
            }

            it("is started") {
                assertThat(secondScopeChange.started).isTrue()
            }
        }
    }
})
```

Spek has structure

Let’s rewrite real JUnit test with Spek

Test code can be structured

with Spek



Hierarchical Report in IntelliJ

Spek Tips

You can iterate stuff

Iterate things, Spek is just a code

```
class StringExtensionsSpec : Spek({

    listOf(null, "", " ", "\t").forEach { string ->

        context("null or blank string '$string'") {

            val isNullOrBlank by memoized { string.isNullOrBlank() }

            it("is indeed null or blank") {
                assertThat(isNullOrBlank, equalTo(true))
            }
        }
    }

})
```

```
when (spek) ?
```

Spek 2.x release plans

Developer Preview by the end of November

Hopefully

Spek 2.x release plans

~month is exactly enough to get sick of how we typically write tests

Most likely

Spek 2.x release plans

github.com/spekframework/spek

Spek 2.x release plans

github.com/spekframework/spek

twitter.com/artem_zin

Spek 2.x release plans

github.com/spekframework/spek

twitter.com/artem_zin

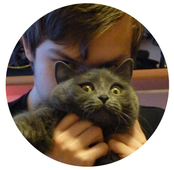
Spek 2.x release plans

github.com/spekframework/spek

twitter.com/artem_zin

End of November

Thank you!



Artem Zinnatullin
@artem_zin

Ranie Jade Ramiso
@raniejade

Hadi Hariri
@hhariri

Artur Dryomov
@arturdryomov

#kotlinconf17