



Spek - Kotlin Test Framework

Metas (Oak)

Growth Session #20 - XX - December 20-21 2018

What is Spek?

- Spek - A test framework written in kotlin
- It provides ways of writing tests in BDD (Behaviour Driven Development) style, by using DSL extensively
- It provides 2 styles for writing tests
 - Specification - RSpec style
 - Gherkin - Cucumber style
- It uses JUnit5

Sample of Specification

```
object TextUtilsSpekTest : Spek({  
    describe("To TitleCase") {  
        context("given an empty string") {  
            it("returns empty") {  
                assertEquals("", "".toTitleCase())  
            }  
        }  
        context("given an lowercase string") {  
            it("returns string by capitalize first letter") {  
                assertEquals("Helloworld", "helloworld".toTitleCase())  
            }  
        }  
        context("given an lowercase string with spaces") {  
            it("returns string as Title Case") {  
                assertEquals("Hello World", "hello world".toTitleCase())  
            }  
        }  
    }  
})
```

Test Results	9 ms
TextUtilsSpekSpecificationTest	9 ms
To TitleCase	9 ms
given an empty string	7 ms
returns empty	7 ms
given an lowercase string	1 ms
returns string by capitalize first letter	1 ms
given an lowercase string with spaces	1 ms
returns string as Title Case	1 ms

Sample of Gherkin

```
object TextUtilsGherkinTest : Spek({  
    Feature("To TitleCase") {  
        Scenario("empty") {  
            Then("returns empty string") {  
                assertEquals("", "".toTitleCase())  
            }  
        }  
        Scenario("lowercase") {  
            Then("returns string with first letter capitalized") {  
                assertEquals("Helloworld", "helloworld".toTitleCase())  
            }  
        }  
        Scenario("multiple lower case words") {  
            Then("returns each words with first letter capitalized") {  
                assertEquals("Hello World", "hello world".toTitleCase())  
            }  
        }  
    }  
})
```

▼ ✓ Test Results	8 ms
▼ ✓ TextUtilsGherkinTest	8 ms
▼ ✓ Feature: To TitleCase	8 ms
▼ ✓ Scenario: empty	8 ms
✓ Then: returns empty string	8 ms
▼ ✓ Scenario: lowercase	0 ms
✓ Then: returns string with first letter	0 ms
▼ ✓ Scenario: multiple lower case words	0 ms
✓ Then: returns each words with first	0 ms

Tests on the Red Planet project

Standard JUnit

▼ ✓ SelfCheckInRoomPreferencesModelImplTest	2 s 209 ms
✓ When click continue, it saves room preference	2 s 2 ms
✓ When click continue but error occurred, it saves	107 ms
✓ When arrival time item is snapped, it emits signal	25 ms
✓ When the same floor type is clicked, it clears	19 ms
✓ When elevator type is clicked, it emits new elevator	13 ms
✓ When the same elevator type is clicked, it clears	12 ms
✓ When floor type is clicked, it emits new floor	12 ms
✓ When arrival time item is clicked, it emits item	10 ms
✓ When click continue but not all room preferences	9 ms

Spek

▼ ✓ Test Results	1 s 627 ms
▼ ✓ SelfCheckInRoomPreferencesModelImplTest	1 s 627 ms
▼ ✓ initialize	1 s 613 ms
▼ ✓ elevator	1 s 608 ms
✓ returns NONE	1 s 608 ms
▼ ✓ floor	2 ms
✓ returns NONE	2 ms
▼ ✓ estimated arrival time	3 ms
▶ ✓ default position	3 ms
▼ ✓ estimated arrival time	3 ms
▼ ✓ snapped the item	1 ms
✓ returns snapped item position	1 ms
▼ ✓ clicked at the item	2 ms
✓ returns clicked item position	2 ms
▼ ✓ elevator preferences	7 ms
▼ ✓ click elevator type FAR	2 ms
✓ returns elevator type as FAR	2 ms
▼ ✓ click elevator type FAR 2 times	5 ms
✓ returns elevator type as NONE	5 ms
▼ ✓ floor preferences	4 ms
▼ ✓ click floor type HIGH	2 ms
✓ returns floor type as HIGH	2 ms
▼ ✓ click floor type HIGH 2 times	2 ms
✓ returns floor type as NONE	2 ms

Gotcha: The expected order of test execution



```
given("a NewsManager") {  
  // 1. setup code here  
  
  beforeEach {  
    // 2. clean and setup for each "on" section  
  }  
  
  on("service returns something") {  
    // 3. execute code with specific parameters for this section  
    it("should receive something") {  
      // 4. asserts  
    }  
  }  
  
  on("another mocked service call") {  
    // so on...  
  }  
}
```

Gotcha: The actual order of test execution




```
given("a NewsManager") {  
  // 1)  
  
  beforeEach {  
    // 4) before every 'it'  
  }  
  
  on("service returns something") {  
    // 2)  
    beforeEach {  
      // 5) before every 'it' in this 'on' section  
    }  
    it("should receive something") {  
      // 6)  
    }  
  }  
  on("another mocked service call") {  
    // 3)  
    ...  
  }  
}
```

Achievement and Progress

- Write simple unit tests - converting string to title case
- Write unit tests on model class in Red Planet project on

Conclusion

- Spek let us write tests as small as possible using the **describe { .. },** **given { .. },** and so on
- Test order execution is difference, which need times to get used to it
- Test becomes easier to understand (If we don't go too crazy nesting it too much)



Having tests is good and
having tests that are also
easier to read is even better!

Thanks!

Contact Nimble

nimblehq.co

hello@nimblehq.co

Bangkok

399 Interchange 21 Sukhumvit Road, Unit
#2402-03, Klong Toei, Wattana, Bangkok
10110, Thailand

Singapore

28C Stanley St, Singapore 068737

Hong Kong

20th Floor, Central Tower
28 Queen's Road, Central, Hong Kong

