JUnit 5
Co nowego?

Kto ja?

Łukasz Rybka

Technical Director w Bond, Andiola & Compay, PC
Trener w infoShare Academy

Podstawowe ustalenia

- Nie jestem alfą ani omegą
- Pytania mile widziane, szczególnie w trakcie!
- Slajdy to tylko notatki, roadmapa

O czym porozmawiamy?

- Co to jest JUnit 5?
- Architektura biblioteki
- Struktura projektu
- Asercje
- Cykl życia
- Parametryzacja testu
- Tagowanie
- Q&A
- ...

Pytanie

Kto z Was pracuje zawodowo?

Pytanie

Kto z Was testuje jednostkowo swój kod?

Pytanie

Kto z Was używa JUnit3/4?

Pytanie

Kto z Was używa JUnit5?

Pytanie

Kto z Was używa TestNG?

Czym jest JUnit 5?

- Nowa generacja najpopularniejszego framework'a do testowania jednostkowego w Javie
- Bardziej przyjazna dla programistów
- Natywne wsparcie dla **Java 8** (m.in. Java 8 Lambdas)

Architektura biblioteki

JUnit Platform

+

JUnit Jupiter

+

JUnit Vintage

=

JUnit 5

JUnit Platform

- Platforma do uruchamiania framework'ów testowych na JVM
- Definiuje **TestEngine API** używane do tworzenia narzędzi testowych uruchamianych na platformie
- W skład platformy wchodzi m.in. **Console Launcher** oraz pluginy do budowania w Maven'ie oraz Gradle

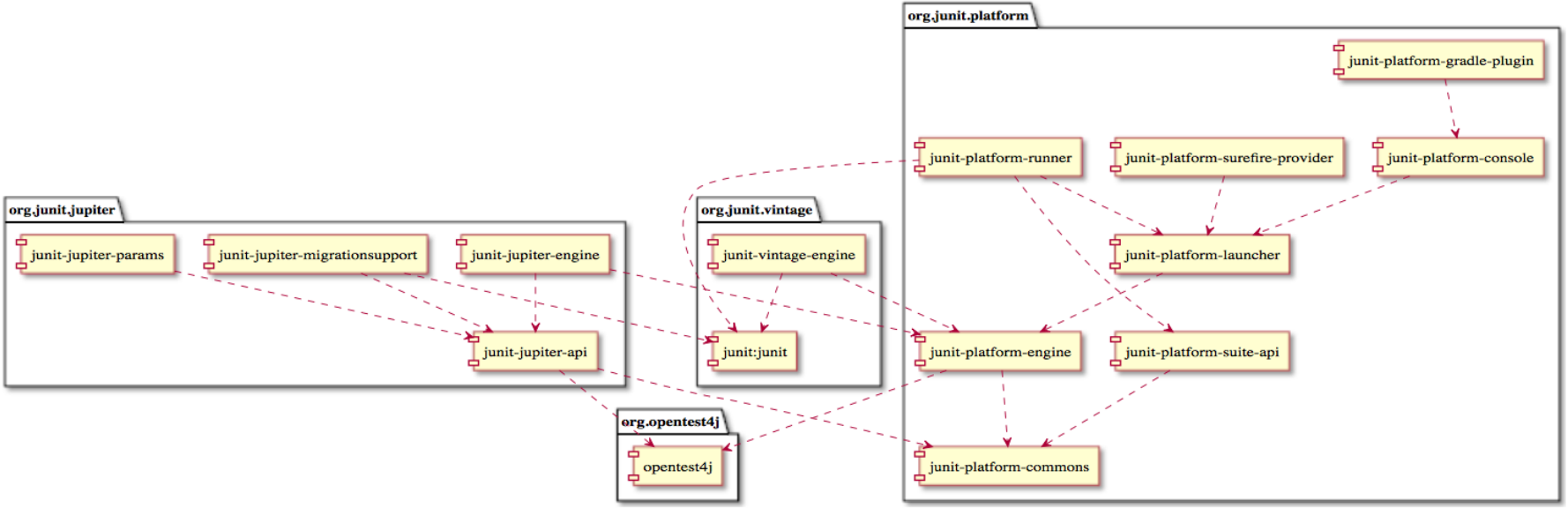
JUnit Jupiter

- Zawiera nowy model pisania testów i rozszerze w JUnit 5
- Dostarcza implementację TestEngine pozwalającą uruchamiać testy na JUnit Platform

JUnit Vintage

- Dostarcza implementację TestEngine pozwalającą uruchamiać testy napisane w JUnit 3 i JUnit 4 na JUnit Platform (kompatybilność wsteczna)

Struktura projektu (Maven)



Struktura projektu (Maven)

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java.version>1.8</java.version>
    <junit.jupiter.version>5.0.2</junit.jupiter.version>
    <junit.platform.version>1.0.2</junit.platform.version>
</properties>
```

```
<dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>${junit.jupiter.version}</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Struktura projektu (Maven)

```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.7.0</version>
            <configuration>
                <source>${java.version}</source>
                <target>${java.version}</target>
            </configuration>
        </plugin>
        <plugin>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.19.1</version>
            <configuration>
                <includes>
                    <include>**/Test*.java</include>
                    <include>**/*Test.java</include>
                    <include>**/*Tests.java</include>
                    <include>**/*TestCase.java</include>
                </includes>
            </configuration>
            <dependencies>
                <dependency>
                    <groupId>org.junit.platform</groupId>
                    <artifactId>junit-platform-surefire-provider</artifactId>
                    <version>${junit.platform.version}</version>
                </dependency>
            </dependencies>
        </plugin>
    </plugins>
</build>
```

Struktura projektu (Maven)

```java
package com.infoshareacademy.karierait.junit5;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class FirstJUnit5Test {
    @Test
    void myFirstTest() { assertEquals(2, 1 + 1); }
}
```

Asercje
Nowy pakiet

- Dawniej (JUnit 4):

```
import org.junit.Test;

import static org.junit.Assert.assertEquals;
```

- JUnit 5

```
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;
```

Asercje
Usunięta assercja - assertThat

```
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.is;
import static org.hamcrest.Matchers.equalTo;

boolean a;
boolean b;

// all statements test the same
assertThat(a, equalTo(b));
assertThat(a, is(equalTo(b)));
assertThat(a, is(b));
```

Asercje
Wiadomość

- Dawniej (JUnit 4):

```
assertEquals( "The optional assertion message.",1, 1);
```

- JUnit 5

```
assertEquals(1, 1, "The optional assertion message.");
```

Asercje
Java 8 Lambdas

```
@Test
public void shouldReturnConcatenatedArguments() {
    String arg1 = "argument 1";
    String arg2 = "argument 2";
    String expected = "argument 1argument 2";

    String concatenated = StringUtils.concat(arg1, arg2);

    assertEquals(concatenated, expected, () ->
        "Expected concatenation of " + arg1 + " with " + arg2 + " to result in " + expected
    );
}
```

Asercje
assertAll

```
@Test
public void shouldReturnConfigWithDefaultSettings() {
    Config config = ConfigFactory.getDefault();

    assertAll("Default database settings", () -> {
        assertEquals("http://localhost", config.getDatabaseHost());
        assertEquals(9999, config.getDatabasePort());
    });

    assertEquals("karierait", config.getDomain());
}
```

Testowanie wyjątków
JUnit4 - expected

```
@Test(expected = IndexOutOfBoundsException.class)
public void empty() {
     new ArrayList<Object>().get(0);
}
```

Testowanie wyjątków
JUnit4 - rules

```
@Rule
public ExpectedException thrown = ExpectedException.none();

@Test
public void shouldTestExceptionMessage() throws IndexOutOfBoundsException {
    List<Object> list = new ArrayList<Object>();

    thrown.expect(IndexOutOfBoundsException.class);
    thrown.expectMessage("Index: 0, Size: 0");
    list.get(0); // execution will never get past this line
}
```

Testowanie wyjątków
JUnit5 - assertThrows

```
@Test
public void shouldThrowExceptionWhenHostIsNull() {
    assertThrows(IllegalArgumentException.class, () -> {
        ConfigFactory.getForDatabase(null, 3000);
    });
}

@Test
public void shouldThrowExceptionWhenHostIsEmpty() {
    Throwable exception = assertThrows(IllegalArgumentException.class, () -> {
        ConfigFactory.getForDatabase("", 3000);
    });

    assertEquals("Host cannot be empty!", exception.getMessage());
}
```

Cykl życia
Anotacje

JUnit 4	JUnit 5
@BeforeClass	@BeforeAll
@BeforeEach	@Before
@AfterEach	@After
@AfterClass	@AfterAll

Cykl życia
@TestInstance

```java
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
public class LifecycleTest {
    private Date firstDate = new Date();
    private Date secondDate;

    @BeforeAll
    public void beforeAll() {
        this.secondDate = this.firstDate;
    }

    @BeforeEach
    public void beforeEach() {
        assertEquals(firstDate, secondDate);
    }

    @AfterEach
    public void afterEach() {
        assertEquals(firstDate, secondDate);
    }

    @AfterAll
    public void afterAll() {
        assertEquals(firstDate, secondDate);
    }

    @Test
    public void firstTest() {
        assertEquals(firstDate, secondDate);
    }

    @Test
    public void secondTest() {
        assertEquals(firstDate, secondDate);
    }
}
```

Parametryzacja @RepeatedTest

```java
@RepeatedTest(5)
public void repeatedTest(RepetitionInfo repetitionInfo) {
    int current = repetitionInfo.getCurrentRepetition();
    int total = repetitionInfo.getTotalRepetitions();

    System.out.println("Checking for the #" + current + " time out of " + total + " trials...");

    assertTrue(true);
}
```

Parametryzacja
@TestInfo / @TestReporter

```java
public class ParameterizedTest {
    @BeforeAll
    public void beforeAll(TestInfo info) {
        System.out.println(info.getTestClass() + " :: beforeAll");
    }

    @BeforeAll
    public void beforeAll(TestReporter reporter) {
        reporter.publishEntry("parameterizedEntry", "someValue");
    }

    @AfterAll
    public void afterAll(TestInfo info) {
        System.out.println(info.getTestClass() + " :: afterAll");
    }

    @Test
    public void dummyTest(TestInfo info) {
        System.out.println(info.getTestClass() + " :: " + info.getDisplayName());
    }
}
```

Parametryzacja @ParameterizedTest

- @ValueSource
- @CSVFileSource
- @MethodSource
- @ArgumentsSource
- @EnumSource

Parametryzacja
@ParameterizedTest

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-params</artifactId>
    <version>${junit.jupiter.version}</version>
    <scope>test</scope>
</dependency>
```

Parametryzacja @ValueSource

```
@ParameterizedTest
@ValueSource(ints = {1, 2, 3, 4, 5})
public void valueSourceTest(int argument) {
    assertTrue(argument > 0);
}
```

Parametryzacja @MethodSource

```java
@ParameterizedTest
@MethodSource(value = {"generatePositiveNumbers"})
public void methodSourceTest(int argument) {
    assertTrue(argument > 0);
}

private static int[] generatePositiveNumbers() {
    return new int[]{1, 2, 3, 4, 5};
}

@ParameterizedTest
@MethodSource(value = {"generateStreamOfArguments"})
public void streamOfArgumentsTest(int number, int expectedResult) {
    assertEquals(expectedResult, Math.pow(number, 2));
}

private static Stream<Arguments> generateStreamOfArguments() {
    return Stream.of(
        Arguments.of(1, 1),
        Arguments.of(2, 4),
        Arguments.of(3, 9),
        Arguments.of(4, 16),
        Arguments.of(5, 25)
    );
}
```

Parametryzacja @CsvFileSource

```
@ParameterizedTest
@CsvFileSource(resources = {"/math_pow_test_data.csv"}, delimiter = ';')
public void csvFileSourceTest(int number, int expectedResult) {
    assertEquals(expectedResult, Math.pow(number, 2));
}
```

Tagowanie pom.xml

```
<plugin>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.19.1</version>
    <configuration>
        <includes>
            <include>**/Test*.java</include>
            <include>**/*Test.java</include>
            <include>**/*Tests.java</include>
            <include>**/*TestCase.java</include>
        </includes>
        <properties>
            <excludeTags>console</excludeTags>
        </properties>
    </configuration>
    <dependencies>
        <dependency>
            <groupId>org.junit.platform</groupId>
            <artifactId>junit-platform-surefire-provider</artifactId>
            <version>${junit.platform.version}</version>
        </dependency>
    </dependencies>
</plugin>
```

Tagowanie anotacja

```
@Target({ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Tag("console")
public @interface Console {
}
```

Tagowanie zastosowanie

```java
@RepeatedTest(5)
@Console
public void repeatedTest(RepetitionInfo repetitionInfo) {
    int current = repetitionInfo.getCurrentRepetition();
    int total = repetitionInfo.getTotalRepetitions();

    System.out.println("Checking for the #" + current + " time out of " + total + " trials...");

    assertTrue(true);
}
```

```java
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
@Console
public class ParameterizedTest {
```

Co dalej?

- TestSuits z @SelectClasses oraz @IncludeTags/@ExcludeTags
- ConsoleLauncher
- DynamicTests
- Timeouts
- Disable/Ignore
- Testy zagnieżdżone
- ...

Pytania?

Thanks!

https://github.com/Smoczysko

https://github.com/infoshareacademy