JUnit 5

Nowości i migracja

Paweł Nowak

Historia wersji JUnit

“

The goal is to create an up-to-date foundation for developer-side testing on the JVM. This includes focusing on Java 8 and above, as well as enabling many different styles of testing.

Architektura

Vintage
JUnit 4.12

Jupiter
JUnit 5

Platforma
Launcher korzystający z engine

Runners
Konsola, SureFire, Gradle itp.

Zmiany, zmiany...

Lambdy
Lambdy mogą być używane w wielu miejscach, choćby asercjach i do generacji komunikatów w asercjach.

Widoczność
Metody i klasy nie muszą już być publiczne (ale nie mogą być prywatne).

Zmiana adnotacji
@Test zostaje, ale:

- @BeforeAll zastępuje @BeforeClass
- @BeforeEach zastępuje @SetUp
- @AfterEach zastępuje @TearDown
- @AfterAll zastępuje @AfterClass
- @Disabled zastępuje @Ignore

Asercje

Asercje

Bez zasadniczych zmian, ale możemy wykonywać wszystkie bez zatrzymywania się na pierwszej oblanej:

```
void shouldReturnProperValuesFromGetters() {
        Address address = new Address("Zielona Góra", "Piłsudskiego", "10”);

        assertAll("address",
                        () -> assertEquals("Zielona Góra", address.getCity()),
                        () -> assertEquals("Piłsusdksiego", address.getStreet()),
                        () -> assertEquals("10", address.getNumber())
        );
}
```

@DisplayName

@DisplayName
Adnotacja dla metod i klas. Nadaje nazwę wyświetlaną zamiast nazwy metody.

```
@Test
@DisplayName(„powinno wyrzucić błąd dla bilansu < 0”)
void shouldReturnErrorForBalanceLessThanZero() {
...
```

@Nested

@Nested
Adnotacja dla klas wewnętrznych – pozwala na tworzenie hierarchii testów.

```
class OuterTest {

    @BeforeEach
    void setUp() {
        ...
    }

    @Test
    void test1() {
        ...
    }

    @Nested
    class InnerTest {
        ...
```

Wyjątki

assertThrows
Nowa metoda sprawdzająca czy wyjątek zostanie wyrzucony

expectThrows
Pozwala na badanie wyjątku.

```
void shouldThrowExcpetionWithAMessage() {
     Exception exception = assertThrows(Exception.class,
this::throwing);
     assertEquals("Hello!", exception.getMessage());
}
```

Zaawansowane (z lambdami)

Testy dynamiczne
Można tworzyć fabryki zwracające testy w formie listy `Executable`.

Testy parametryzowane
Mechanizm testów dynamicznych może być wykorzystany do tworzenia w wygodny sposób testów parametryzowanych w wygodniejszy sposób.

Mechanizm rozszerzeń
JUnit 5 oferuje punkty rozszerzeń w ramach cyklu życia testów. Nigdy więcej rozszerzeń za pomocą runnerów!

Wsparcie w IDE

Na razie wszelkie wsparcie jest dość eksperymentalne. Zespół JUnit ma jednak obejścia dla IDE z obsługą wersji czwartej.

IntelliJ
Wsparcie od wersji 2016.2

Eclipse
Oxygene ma wstępną wersję obsługi, ale trzeba ją doinstalować ręcznie z repozytorium.

Migracja (wersja łatwa)

Migrujemy z Junit 4 do 5
Wystarczy zmienić zależności na:

```
<dependency>
     <groupId>org.junit.jupiter</groupId>
     <artifactId>junit-jupiter-engine</artifactId>
     <version>5.0.0-M4</version>
     <scope>test</scope>
</dependency>

<dependency>
     <groupId>org.junit.vintage</groupId>
     <artifactId>junit-vintage-engine</artifactId>
     <version>4.12.0-M4</version>
     <scope>test</scope>
</dependency>
```

I możemy pisać nowe testy w 5...

Migracja (wersja trudna)

Migrujemy z Junit 4 do 5

1) Zmieniamy pakiety na nowe
2) Zmieniamy anotacje (na przykład `Ignore` na `Disabled`)
3) I uruchamiamy testy