

# Automatizované testovanie

jUnit, assertJ, PowerMock, EasyMock

**Matej Kováč**

# Obsah

- **Úvod**

- Prečo testovať
- Čomu sa budeme venovať

- **Teória v kocke + prerekvizity**

- Typická štruktúra

- **Mocking**

- Čo je a prečo
- Čo všetko vieme „namockovať“

- **Asserting**

- Zaujímavé funkcie a usecases knižnice assertJ

# Obsah

- **Novinky v spomenutých frameworkoch**
  - Všeobecne
  - junit 4 vs. junit 5
- **Integračné testovanie**
  - Ukážka, analogické princípy k unit testom
  - Best practices
- **Zdroje**
- **Záver**

Úvod >>

# Prečo testujeme ?



# Prečo testujeme automatizovane ?

- **Včasné zachytenie chýb**
- **Regresná analýza**
- **Konzistencia**
- **CI/CD, nasadzovanie**
- **Kvantita**
- **Test driven development**
- **Juniori / nové nezainteresované osoby**

# Typy testov / spôsoby testovania

- **Unit testy**
- **Integračné testy**
- **API Testy**
- **Manuálne testovanie**
- **Funkčné, Závažové, Bezpečnostné, Databázové, Concurrency, Regresné, Recovery, Compliance, Usability, Load balancer testy ...**

# Typická štruktúra

- **Arrange**
- **Act**
- **Assert**

# Typická štruktúra

```
public class CalculatorTest {  
  
    private Calculator calculator;  
  
    @BeforeEach  
    public void setUp() {  
        // Arrange  
        calculator = new Calculator();  
    }  
  
    @Test  
    public void testAddition() {  
        // Act  
        int result = calculator.add(2, 3);  
  
        // Assert  
        Assertions.assertEquals(5, result);  
    }  
}
```



# Mocking

- **Izolácia**
- **Kontrola**
- **Rýchlosť / úspora systémových prostriedkov**
- **Reprodukovateľnosť**
- **Okrajové prípady**
- **Zjednodušenie konfigurácie**

# jUnit, Mockito

- **public metóda**
- **public metóda, akýkoľvek vstup**
- **public metóda, partial mock**
- **private atribút**
- **private final atribút**
- **statická public metóda**

## jUnit, PowerMock,

- **privátne metódy**
- **public static atribút**
- **private static atribút**

```
testImplementation 'org.powermock:powermock-module-junit4:2.0.9'  
testImplementation 'org.powermock:powermock-api-mockito2:2.0.9'  
testImplementation 'org.powermock:powermock-api-support:2.0.9'  
testImplementation 'org.powermock:powermock-core:2.0.9'  
testImplementation 'org.objenesis:objenesis:3.1'
```

# Spring profily, viacero implementácií

```
public interface MyService {  
    void doSomething();  
}
```

```
@Service  
@Profile("junittest")  
public class FirstServiceImpl implements MyService {
```

```
    @Override  
    public void doSomething() {  
        ...  
    }  
}
```

```
@Service  
public class SecondServiceImpl implements MyService {
```

```
    @Override  
    public void doSomething() {  
        ...  
    }  
}
```

# Asserting

- **Závěrečná část v teoriií testovacího postupu**
- **Kontrola výstupov**
- **Ukážka častých a zaujímavých usecases**

# Základné

```
String text = "Hello, AssertJ";  
assertThat(text).isNotNull();  
assertThat(text).isNotEmpty();  
assertThat(text).isEqualTo("Hello, AssertJ");  
assertThat(text).isNotEqualTo("Hello, JUnit");
```

```
int number = 42;  
assertThat(number).isGreaterThan(30);  
assertThat(number).isLessThan(50);
```

```
Object object = new Integer(42);  
assertThat(object).assertInstanceOf(Integer.class);
```

```
Runnable operation = () -> {  
    throw new IllegalArgumentException("Invalid argument");  
};
```

```
assertThatThrownBy(operation)  
    .assertInstanceOf(IllegalArgumentException.class)  
    .hasMessage("Invalid argument");
```

# Základné

## Číselné hodnoty:

```
assertThat(age).isBetween(18, 35);  
assertThat(price).isCloseTo(19.99, within(0.01));
```

## Text:

```
assertThat(text).contains("important");  
assertThat(email).matches("[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.\\.[a-zA-Z]{2,4}");
```

## Dátum/čas:

```
assertThat(birthDate).isBefore(becameAdultDate);  
assertThat(expirationDate).isWithin(1, TimeUnit.DAYS).of(now);
```

## Listy:

```
assertThat(names).containsExactly("Alice", "Bob", "Charlie");
```

## Mapy:

```
assertThat(scores).containsEntry("Alice", 95);
```

## Súbor:

```
assertThat(file).hasContent("Hello, World!");
```

# Kolekcia filtrovaná cez Stream

```
List<Person> persons = new ArrayList<>();  
    persons.add(new Person("Alice", 25));  
    persons.add(new Person("Bob", 30));  
    persons.add(new Person("Charlie", 15));  
    persons.add(new Person("David", 17));  
  
assertThat(persons)  
    .filteredOn(person -> person.getAge() >= 18)  
    .extracting(Person::getName)  
    .contains("Alice", "Bob");
```



# Vlastná implementácia

```
public class UserAssert extends AbstractObjectAssert<UserAssert, User> {  
  
    private UserAssert(User actual) {  
        super(actual, UserAssert.class);  
    }  
  
    public static UserAssert assertThat(User actual) {  
        return new UserAssert(actual);  
    }  
  
    public UserAssert isPremium() {  
        isNotNull();  
  
        if (!actual.isPremium()) {  
            failWithMessage("Expected the user to be premium, but it is not.");  
        }  
  
        return this;  
    }  
}  
  
User user = new User("John", true);  
assertThat(user).isPremium();
```

Asserting >>

## „Soft“ assertions

```
SoftAssertions softAssertions = new SoftAssertions();  
softAssertions.assertThat(age).isEqualTo(25);  
softAssertions.assertThat(name).isEqualTo("John");  
softAssertions.assertAll();
```

- vyhodí chyby až na konci, nezastavuje test

# Novinky

## **PowerMock**

- Podpora junit 5 v blízkej budúcnosti
- Pomerne nedávno prechod z 1.x.x na 2.x.x

## **Mockito - aktuálne v5, zrýchlenie číslovania verzií, zásadné zmeny vo v3**

- Mockito-inline - mockovanie final tried a metód
- Handling default metód v interfaces
- Podpora pre Kotlin
- Zlepšené stack traces a error messages

## **EasyMock - útlm v aktivite, preferované predošlé 2**

# jUnit 4 verzus jUnit 5

v4: `@Test(expected = Exception.class)`

v5: `assertThrows(Exception.class, () -> {...})`

v4: `@Test(timeout="1000")`

v5: `assertTimeout(Duration.ofMillis(1000), () -> {...})`

v4: `@RunWith(MockitoJUnitRunner.class)`

v5: `@ExtendWith(MockitoExtension.class)`

V4: 1 test runner

V5: viacero

V4: jeden JAR

V5: modulárnejší, možno importovať jednotlivé knižnice

V4: vyvinutý pre Javu 7

V5: Java 8+

V4: `@Before`, `@After`, `@BeforeClass`, `@AfterClass`, `@Ignore`

V5: `@BeforeEach`, `@AfterEach`, `@BeforeAll`, `@AfterAll`, `@Disabled`

# Ďalej vo verzii junit 5

## Chained assertions:

```
@Test
public void shouldAssertAllTheGroup() {
    List<Integer> list = Arrays.asList("alpha", "beta", "gamma");
    Assertions.assertAll("Some assertion message",
        () -> Assertions.assertEquals(list.get(0), "alpha"),
        () -> Assertions.assertEquals(list.get(1), "beta"),
        () -> Assertions.assertEquals(list.get(2), "gamma"));
}
```

# Ďalej vo verzii junit 5

## Parametrizované testy:

```
@Test  
public void string1IsNotNull() {  
    assertNotNull("Hello");  
}
```

```
@Test  
public void string2IsNotNull() {  
    assertNotNull("world");  
}
```

```
@ParameterizedTest  
@ValueSource(strings = {"Hello", "World"})  
void shouldPassNonNullMessageAsMethodParameter(String message) {  
    assertNotNull(message);  
}
```

# Ďalej vo verzii junit 5

## Vnáranie:

```
public class NestedTest {  
  
    @Nested  
    class FirstNestedClass {  
  
        @Test  
        void test() {  
            System.out.println("FirstNestedClass.test()");  
        }  
    }  
}
```

# Ďalej vo verzii jUnit 5

## Tagy a filtrovanie podľa nich:

```
@Test
@Tag("IntegrationTest")
public void testAddEmployeeUsingSimpleJdbcInsert() {
    ...
}
```

```
@SelectPackages("com.baeldung.tags")
@IncludeTags("UnitTest")
public class EmployeeDAOUnitTestSuite {
    ...
}
```

```
<plugin>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>2.22.2</version>
  <configuration>
    <groups>UnitTest</groups>
  </configuration>
</plugin>
```



# Integračné testovanie v Spring apps

- **Ukážka**

- <https://github.com/K0V0/paster-backend/blob/main/src/test/java/com/kovospace/paster/user/integration/UserControllerLoginTest.java>

# Best practices

- **Vhodne použiť mocking**
- **Rýchlosť testov**
  - Vyhnúť sa ak možno @DirtiesContext
  - Test scenár v transakcii @Transactional
- **Eliminovať závislosť na externých dátových zdrojoch a službách**
- **In-memory dátové zdroje**
- **Dostatočné asserty** (JUnit asserts, Hamcrest matchers, assertJ)