



Sexy unit testy



czyli o kilku praktykach w testach jednostkowych

Agenda

KILKA

- sekund o samym sobie
- zdań o prezentacji
- kilka przemyśleń
- pomysłów na ułatwienie sobie życia
- pytań od publiczności

O mnie




- Absolwent WMil UMK
- Programista Java
- Wielbiciel Scruma i dobrej roboty
- W domu gitary, aparaty i kilka piłek

O Prezentacji

czyli co będzie, a czego nie usłyszycie

- 
- Kilka prawdopodobnie przydatnych praktyk

- 
- Argumentacja za pisanem testów
 - Recepta na idealne testy
 - Wady podanych praktyk
 - Parametryzacja testów
 - Piosenka na pożegnanie

Hasło

Bo jakoś trzeba zacząć...



“Not only working software,
but also well-crafted software”

(<http://manifesto.softwarecraftsmanship.org/>)



Kilka przemysłów



#1

Kod testowy jak żona, matka
i teść... kod produkcyjny –
zasługuje na dobrą opiekę



#2

Dobre testy są lepszą
dokumentacją niż słaby
java-doc





#3

Test może być znakomitym
wyznacznikiem jakości
Twojego kodu



#4

Dobrze napisany test jest
odwrotnością dziecka wracającego
po miesiącu z wakacji od dziadków

(dla bezdziejnych: nie dziwi Cię jego zachowanie)



#5

Sam fakt pokrycia niekoniecznie
doprowadza do happy-endu

Przykłady

Kilka rzeczy z życia programisty

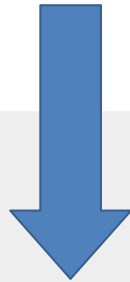


- Brzydki kod produkcyjny testuje się brzydko



☹ Item.java

```
public class Item {  
    private Status status;  
    private Item parent;  
  
    public void mergeItem(Item item) {  
        //...  
        if (isClosed(this) && isClosed(item)) {  
            //...  
        }  
        //...more if's here  
    }  
  
    private static boolean isClosed(Item item) {  
        return item.getStatus() == CLOSED || (item.getParent() != null  
            && item.getParent().getStatus() == CLOSED);  
    }  
    //...
```



Może tak ?

```
public boolean isClosed() {  
    return status == CLOSED || (parent != null && parent.getStatus() == CLOSED);  
}
```

• testOK – czytelne nazwy metod testowych



😊 PurchaseTest.java

```
public class PurchaseTest {  
  
    @Test  
    public void testOK() {}  
  
    @Test  
    public void testFail() {}  
  
    @Test  
    public void testException() {}  
}
```



Może tak ?

```
@Test  
public void shouldSellItemIfExists() {}  
  
@Test  
public void shouldNotSellItemIfNotAvailable() {}  
  
@Test  
public void shouldThrowExceptionForNotExistingClient() {}
```



```
@Test  
public void should_finalize_purchase_if_paid() {}
```

• Fluent API & Builders VS litania setterów



😊 PurchaseTest.java

```
@Test
public void shouldSellItemIfAvailable() {

    Item item = createItem(25000L, null, 1, null);

    //...

}

//...

private Item createItem(Long price, Long discount, Integer quantity, Integer popularity) {
    Item item = new Item();
    item.setPrice(price);
    item.setDiscount(discount);
    item.setQuantity(quantity);
    item.setPopularity(popularity);
    return item;
}
```



Może tak ?

```
@Test
public void shouldSellItemIfAvailable() {

    Item item = anItem().withPrice(25000L).withQuantity(1).build();

    //...

}
```

• Given/When/Then



😊 CartTest.java

```
@Test
public void shouldGetTotalPrice() {
    Item bike = anItem().withPrice(100000L).withQuantity(1).build();
    Item ball = anItem().withPrice(4000L).withQuantity(2).build();
    Cart cart = aCartBuilder().withItems(bike, ball).build();
    Long totalPrice = cart.getTotalPrice();
    assertEquals(108000L, totalPrice);
}
```



Može tak ?

```
@Test
public void shouldGetTotalPrice() {
    // given
    Item bike = anItem().withPrice(100000L).withQuantity(1).build();
    Item ball = anItem().withPrice(4000L).withQuantity(2).build();
    Cart cart = aCart().withItems(bike, ball).build();

    // when
    Long totalPrice = cart.getTotalPrice();

    // then
    assertEquals(108000L, totalPrice);
}
```


• Małe podsumowanie początków



☹️ CartTest.java

```
@Test
public void testOK() {
    Item bike = new Item();
    bike.setPrice(100000L);
    bike.setQuantity(1);
    Item ball = new Item();
    ball.setPrice(4000L);
    ball.setQuantity(2);
    Cart cart = new Cart();
    cart.setItems(Arrays.asList(bike, ball));
    Long totalPrice = cart.getTotalPrice();
    assertEquals(108000L, totalPrice);
}
```



```
@Test
public void shouldGetTotalPrice() {
    // given
    Item bike = anItem().withPrice(100000L).withQuantity(1).build();
    Item ball = anItem().withPrice(4000L).withQuantity(2).build();
    Cart cart = aCart().withItems(bike, ball).build();

    // when
    Long totalPrice = cart.getTotalPrice();

    // then
    assertEquals(108000L, totalPrice);
}
```

• Złośliwe buildery (default is evil?)



☺ CurrencyServiceTest.java

```
@Test
public void shouldReturnNullIfNoPriceDefined() {

    // given
    Item item = anItem().build();

    // when
    Long amount = currencyService.getAmountIn(item.getPrice(), Currency.EUR);

    // then
    assertThat(amount).isNull();
}
```

FAILED
(not null)



Może tak ?

```
Item item = anItem().build();
```



```
Item item = anItem().withDefaultValues().build();
```

• Testowanie pojedynczych przypadków



😊 CartTest.java

```
@Test
public void testTotalPrice() {
    Item ball = anItem().withPrice(4000L).withQuantity(2).build();
    Cart cart = aCart().withItems(ball).build();
    Long totalPrice = cart.getTotalPrice();

    assertThat(totalPrice).isEqualTo(8000L);

    ball.setDiscount(1000L);
    totalPrice = cart.getTotalPrice();

    assertThat(totalPrice).isEqualTo(6000L);
}
```



Może tak ?

```
@Test
public void shouldGetWholeTotalPriceIfNoDiscount() {}
```

```
@Test
public void shouldGetDecreasedTotalPriceIfDiscountExists() {}
```

• Bezpośrednie używanie wartości #1



😊 LabelProviderTest.java

```
@Test
public void shouldGetItemLabel() {
    // given
    Item item = anItem().withName("Kosz na śmieci").withQuantity(2).build();

    // when
    String label = labelProvider.createLabel(item);

    // then
    assertThat(label)
        .isEqualTo(item.getName().toUpperCase() + " w ilości " + item.getQuantity() + " sztuk");
}
```



Może tak ?

```
assertThat(label)
    .isEqualTo("Kosz na śmieci w ilości 2 sztuk");
```

• Bezpośrednie używanie wartości #2



😊 OrderServiceTest.java

```
@Test
public void shouldExtractCustomerFromOrderRecord() {
    // given
    String record = StringUtils.join(
        new Object[] {ORDER_ID, CUSTOMER_ID, FIRST_NAME, LAST_NAME, DOB,
            ADDRESS, ITEM_ID, ITEM_PRICE, ITEM_QUANTITY}, DELIMITER);

    // when
    Customer customer = orderService.extractCustomerFromOrderRecord(record);

    // then
    assertThat(customer.getId()).isEqualTo(CUSTOMER_ID);
    assertThat(customer.getFirstName()).isEqualTo(FIRST_NAME);
    assertThat(customer.getLastName()).isEqualTo(LAST_NAME);
    assertThat(customer.getDOB()).isEqualTo(DOB);
    assertThat(customer.getAddress()).isEqualTo(ADDRESS);
}
```



Może tak ?

```
// given
String record = "1|233|ZDZISŁAW|SAMORÓB|740121|WESOŁA 21,10-111 BĄKOWO|3876|1000L|2";

// when
Customer customer = orderService.extractCustomerFromOrderRecord(record);

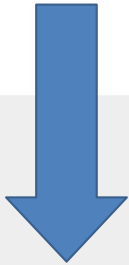
// then
assertThat(customer.getId()).isEqualTo(233);
assertThat(customer.getFirstName()).isEqualTo("ZDZISŁAW");
assertThat(customer.getLastName()).isEqualTo("SAMORÓB");
assertThat(customer.getDOB()).isEqualTo(parse("740121", forPattern("yyMMdd")).toDate());
assertThat(customer.getAddress()).isEqualTo("WESOŁA 21, 10-111 BĄKOWO");
```

• Asercje i fluent API



☺ CurrencyServiceTest.java

```
// then
assertEquals(IOException.class, exception.getClass());
assertEquals("Cannot read the stream.", exception.getMessage());
```



Może tak ?

```
// then
assertThat(exception)
    .isExactlyInstanceOf(IOException.class)
    .hasMessage("Cannot read the stream.");
```

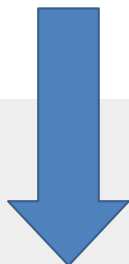
AssertJ (*fluent assertions for java*)

• Własne asercje



😊 CurrencyServiceTest.java

```
// then
assertThat(customer.getId()).isEqualTo(233L);
assertThat(customer.getFirstName()).isEqualTo("ZDZISŁAW");
assertThat(customer.getLastName()).isEqualTo("SAMORÓB");
assertThat(customer.getDOB()).isEqualTo(parse("740121", forPattern("yyMMdd")).toDate());
assertThat(customer.getAddress()).isEqualTo("WESOŁA 21, 10-111 BĄKOWO");
```



Może tak ?

```
CustomerAssertion.assertThat(customer)
    .hasId(233L)
    .hasFirstName("ZDZISŁAW")
    .hasLastName("SAMORÓB")
    .wasBorn(new DateTime().withYear(1974).withMonthOfYear(01).withDayOfMonth(21))
    .hasAddress("WESOŁA 21, 10-111 BĄKOWO");
```



```
.wasBorn("1974-01-21")
```

• Różne realne wartości testowe



☺ CartTest.java

```
@Test
public void shouldExtractCustomerFromOrderRecord() {
    // given
    String record = "1|1|SOME_STRING|SOME_STRING|740101|SOME_STRING,SOME_STRING|1|1|1";

    // when
    Customer customer = orderService.extractCustomerFromOrderRecord(record);

    // then
    CustomerAssertion.assertThat(customer)
        .hasId(1L)
        .hasFirstName("SOME_STRING")
        .hasLastName("SOME_STRING")
        .wasBorn("19740101")
        .hasAddress("SOME_STRING");
```



Po porawieniu wartości...

```
CustomerAssertion.assertThat(customer)
    .hasId(233L)
    .hasFirstName("ZDZISŁAW")
    .hasLastName("SAMORÓB")
    .wasBorn("19740101")
    .hasAddress("WESOŁA 21, 10-111 BĄKOWO");
```

FAILED

• Czytelne nazwy zmiennych



☺ CartTest.java

```
@Test
public void shouldCopyAllFromDirectory() {
    // given
    Directory directory1 =
        aDirectory().withPath("/home/docs").withFiles("report_1.txt", "report_2.txt").build();
    Directory directory2 =
        aDirectory().withPath("/home/download").withFiles("readme.txt").build();

    // when
    directoryService.copy(directory1, directory2);

    // then
    // ...
}
```



A może tak?

```
@Test
public void shouldCopyAllFromDirectory() {
    // given
    Directory source =
        aDirectory().withPath("/home/docs").withFiles("report_1.txt", "report_2.txt").build();
    Directory target =
        aDirectory().withPath("/home/download").withFiles("readme.txt").build();

    // when
    directoryService.copy(source, target);

    // then
    // ...
}
```

• Przydatne biblioteki do tworzenia mocków



EASYMOCK

JMockit



☺ ReportServiceTest.java

```
@Mock
private CurrencyService currencyService;

@Mock
private OrderService orderService;

@InjectMocks
private OrderService orderService

@Test
public void shouldGenerateReportInEur() {
    // given
    given(currencyService.getRate("EUR")).willReturn(415L);
    given(orderService.getOrders(eq(DateTime.parse("2014-05-01")), eq(DateTime.parse("2014-05-31"))))
        .willReturn(asList(
            anOrder().withId(20L).withDate("2014-05-01").withTotalAmount(2500L).build(),
            anOrder().withId(30L).withDate("2014-05-31").withTotalAmount(3000L).build()));

    // when
    String csvReport = reportService.createMonthlyOrdersReport(ReportType.CSV,
        new DateTime().withYear(2014).withMonthOfYear(5));

    // then
    // ...
}
```

• Nadmiarowe dane i mockowanie



😊 PurchaseServiceTest.java

```
@Mock
EmailService emailService;

@Mock
GeoService geoService;

@Mock
ClientService clientService;

@InjectMocks
PurchaseService purchaseService = new PurchaseService();

@Before
public void setup() {
    when(clientService.getClientByHash(anyString())).thenReturn(
        ClientBuilder.aClient().withId(10L).withName("Jan Kowalski").withSex(Sex.FEMALE)
            .withDOB("1981-04-03").withAddress("Batorego 10 / 3").build());

    when(geoService.getCountry(anyString())).thenReturn(Country.PL);
    //...
}

@Test
public void shouldDecreaseQuantityOnceSold() {
    //...
    // when
    boolean confirmed = purchaseService.confirm("234ij32jf2a83a");

    // then
    assertThat(confirmed).isTrue();
}
```



catch-exception*

ReportServiceTest.java

```
@Test
public void shouldThrowExceptionWhenGeneratingReportInEurWithRateNotFound() {
    // given
    given(currencyService.getRate("EUR")).willReturn(null);
    given(orderService.getOrders(any(DateTime.class), any(DateTime.class)))
        .willReturn(asList(anOrder().withId(10L).withDate("2014-05-20")
            .withTotalAmount(1000L).build())));

    // when
    catchException(reportService).createMonthlyOrdersReport(ReportType.CSV,
        new DateTime().withYear(2014).withMonthOfYear(5));

    // then
    assertThat(caughtException()).isExactlyInstanceOf(IOException.class)
        .hasMessage("Cannot find the rate for EUR.");
}
```

*) „Java 8's lambda expressions will make catch-exception redundant. Therefore, this project won't be maintained any longer”





Podsumowanie

Pytania i odpowiedzi



Kontakt:
Łukasz Sawicki
sawickil@gmail.com



Sexy unit testy ■ ■ ■
czyli o kilku praktykach w testach jednostkowych

