JGiven

więcej miłości dla dev'a

Adam Kluźniak

Java developer @ jntive

Coto poco
- Typowy unit test

```
@Test
public void shouldRetrieveCorrectOrderData() {
    Customer customer = new Customer();
    customer.setFirstName("Jan");
    customer.setLastName("Kowalski");
    customer.setPesel("92101308297");
    customerService.saveCustomer(customer);
    Order order1 = new Order("12345");
    orderService.saveOrder(order1);
    Order order2 = new Order("12346");
    OrderItem orderItem = new OrderItem("tubajfor");
    order2.addItem(orderItem);
    orderService.saveOrder(order2);
    List<Order> result = orderService.getAllOrders();
    assertThat(result).isNotNull();
    assertThat(result).hasSize(2);
    assertThat(result.get(2).getItems()).contains(orderItem);
}
```

Coto poco
- Typowy unit test

WADY:

- Dużo szumu informacyjnego
- Cel testu może być trudny do ogarnięcia
- Może być ciężki w utrzymaniu
- Potencjalne powielanie kodu
- Nie może być użyty jako dokumentacja

Coto poco
- Popularny BDD: Cucumber

```
Feature: Orders


Scenario: Retrieving orders return correct data
    Given  A Customer
    And    An Order
    And    Another Order
    And    An order item for last order
    When   Order list is retrieved
    Then   Result contains 2 orders
    And    Order nr 2 contains 1 item
```

Coto poco
- Popularny BDD: Cucumber

ZALETY:

- Dużo bardziej czytelny
- Poszczególne zdania można użyć ponownie w innych testach
- Sam w sobie jest dokumentacją
- Może być pisany przez analityków

WADY:

- Piszemy go w innym języku
- Aspekt dokumentacyjny nie jest idealny
- Słabe wsparcie ze strony IDE
- Dodatkowy narzut

Coto poco
- Test z JGiven

```
@Description(”Orders”)
public class OrdersTests extends ScenarioTest<GivenOrders, WhenOrders, ThenOrders> {

    @Test
    public void retrieving_orders_returns_correct_data() {
        given().a_customer()
            .and().an_order()
            .and().another_order()
            .with().an_order_item();
        when().order_list_is_retrieved();
        then().result_contains_$_orders(2)
            .and().order_nr_$_contains_$_item(2, 1);
    }
}
```

Coto poco
- Test z JGiven

Test Class: com.importantco.OrdersTests

Retrieving orders returns correct data

Given a customer
And an order
And another order
With an order item
When order list is retrieved
Then result contains 2 orders
And order nr 2 contains 1 item

Coto poco
- Test z JGiven

- Testowanie w duchu BDD
  - w Javie

Coto poco
- Czemu w Javie

- Raporty generowane w HTML (jest ładniej )
- Definicje kroków są metodami Javy
- Wyszukiwanie wywołań w IDE
- Nie potrzeba innego języka
- Możliwość użycia wszelkich narzędzi oferowanych prze IDE (refactoring)
- Możliwość uzupełnienia testów o dodatkową logikę w Javie
  - With great power comes great responsibility
- Łatwiejsze do ogarnięcia i uzupełniania przez developerów
- Bliższy związek QA i Dev (wielka miłość )
- W związku z tym poprawne utrzymanie testów jest bardziej prawdopodobne

Co i jak
- jak to działa

- Testy są uruchamiane w Junit lub TestNG
- Płynne (fluent) API
- Pobiera większość informacji z kodu Java
- Raporty:
  - Tekstowy
  - HTML

Co i jak
- jak to działa

- Scenario
- Stage
- Współdzielony stan
- Parametry metod – zdań
- Formatowanie parametrów
- Rozgałęzienia (if)
- Tagi
- Wywołania przed/po stage/scenario
- notimplementedyet

Porady

- Wykorzystaj buildery do przygotowywania danych w stage’ach
- Warto aby buildery miały dużo danych domyślnych
- Do wielokrotnych wywołań z parametrami trzeba użyć osobnej biblioteki (DataProvider runner)

Podsumowując

- Testowanie w duchu BDD
- **Przyjazny developerom**
- Wystarczy Java, nie potrzeba używać innych języków
- Raporty z testów **dokumentują** nam funkcję tworzonej aplikacji

Koniec

- Pytania o JGiven
- Pytania o Intive
- Pytania o ?????
- Pytania prywatne
- Inne pytania

Koniec

http://jgiven.org/