Testowanie automatyczne

Behaviour Driven Development
Cucumber

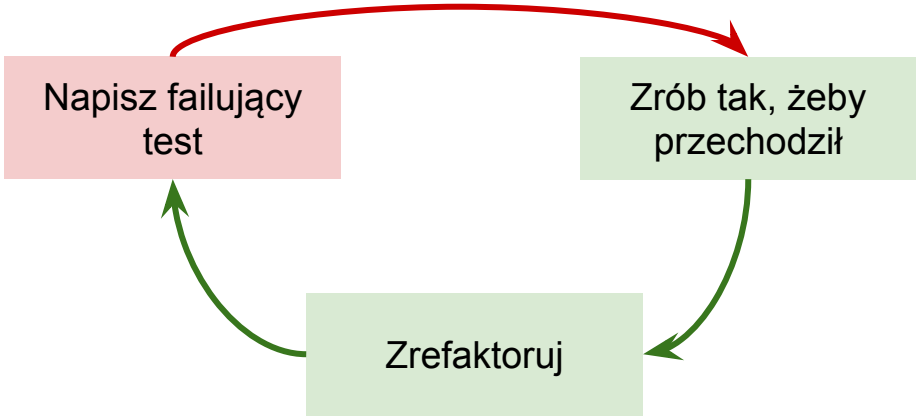
TDD - Test Driven Development
BDD - Behaviour Driven Development

TDD w pigułce

1. Napisz test
2. Sprawdź, czy wykonuje się nieprawidłowo
3. Napraw
4. Zrefaktoruj

BDD - to samo, ale w inny sposób

1. Scenario – nazwa scenariusza
2. Given – zakładając, że mamy…
3. When – dzieje się…
4. Then – wydarzy się…



Behaviour Driven Development

Zasadniczą różnicą między TDD i BDD jest sposób definiowania testów

TDD opiera się na testach jednostkowych, czyli de facto kodzie w języku programowania

BDD zakłada pisanie testów w języku (prawie) naturalnym, zrozumiałym dla osób nietechnicznych

… a może jeszcze dalej?

ATDD to inaczej Acceptance Test Driven Development - najpierw automatyczne testy akceptacyjne

Co dzięki temu osiągamy?

Stosując podejście “najpierw test” zbliżamy się do “ideału” zapewniającego

Pewność

- zapewniamy, program działa poprawnie, został przetestowany na tyle, ile to możliwe

Refaktoring

- nigdy nie jest tak, że kod programu od razu jest doskonały, zwykle jest potrzeba przepisania różnych fragmentów w nowy, “lepszy” sposób. Pokrycie testami umożliwia wykonanie tej czynności bez stresu, że coś zepsujemy

Co dzięki temu osiągamy? (2)

Zmiany w projekcie

- to nie to samo co refaktoring (!)
- zmiany specyfikacji projektu są w wielu wypadkach nieuniknione

Dokumentacja

- testy jednostkowe pisane w technice TDD są świetną dokumentacją kodu
- testy pisane w technice BDD mogą stać się dokumentacją systemu na ogólniejszym poziomie - dokumentację wymagań

(…)DD w pigułce

TDD - czy kod jest poprawny

BDD - czy testujemy to, co powinniśmy testować

ATDD - czy system robi to, co powinien robić

Gherkin - składnia

Główne słowa kluczowe

- Feature
- Rule (as of Gherkin 6)
- Example (or Scenario)
- Given, When, Then, And, But (steps)
- Background
- Scenario Outline (or Scenario Template)
- Examples

Dodatkowe znaczniki

- """ (Doc Strings)
- | (Data Tables)
- @ (Tags)
- # (Comments)

Feature

Wysokopoziomowy opis funkcjonalności do przetestowania

Celem jest zdefiniowanie istotnych aspektów funkcjonalności w sposób zrozumiały dla wszystkich zainteresowanych

Feature: Pieszy przechodzi przez jezdnię

    Pieszy powinien bezpiecznie przejść przez jezdnię -
    musimy się upewnić, że gdy świeci mu się zielone, nie
    jedzie żaden samochód

Steps - Scenario

Scenario - definicja testu, opisujemy dany przypadek testowy

```
Scenario: Pieszy podchodzi do przejścia i ma zielone światło
```

```
Scenario: I want to play one of my favourite songs
```

Synonimem słowa Scenario jest **Example**

Steps

Scenariusz testu jest poukładany w kroki. Każdy z kroków zaczyna się od jednego ze słów kluczowych

Given - ma za zadanie opisać inicjalny stan systemu przed testem. W szczególności opis zdarzeń w przeszłości

```
Given: Pieszy podszedł do przejścia
```

```
Given: I am logged in
```

Steps

When - opisuje zdarzenie lub akcję, którą wykonujemy. Może to być również zdarzenie wykonane przez system zewnętrzny

When `Pieszy widzi zielone światło`

When `zaświeciło się zielone światło`

When `I click the MyFavorities button`

Dobre praktyki: w teście powinna się znaleźć tylko jedna klauzula When. Jeżeli czujemy, że chętnie dodalibyśmy kolejną, to zwykle oznacza, że warto rozważyć napisanie kolejnego testu

Steps

Then - używamy, żeby opisać spodziewany rezultat.

Then `Światło dla samochodów jest czerwone`

Then `I see most often played songs`

Then stanowi opis asercji w teście

Steps - grupowanie

Background - grupowanie kroków wspólnych dla każdego testu w pakiecie feature

Jeżeli w każdym ze scenariuszy powtarza się ta sama sekwencja kroków Given, warto wydzielić część wspólną i wpisać ją do sekcji Backgroud. Zostaną wykonane przed krokami konkretnego scenariusza

Background:

```
    Given Pieszy podchodzi do przejścia i obserwuje światło
```

Background:

```
    Given I am logged into the system
```

Steps - łączniki

And, **But** - słowa łączące kroki o tym samym poziomie

```
Given Pieszy podszedł do przejścia
Given Pieszy widzi zielone światło

Then Pieszy ma zielone światło
Then: Samochody mają czerwone światło

Then Samochód widzi zielone światło
Then nie widzi żółtego
```

```
Given Pieszy podszedł do przejścia
  And Pieszy widzi zielone światło

Then Pieszy ma zielone światło
 And Samochody mają czerwone

 Then Samochód widzi zielone światło
  But Nie widzi żółtego
```

Argumenty - Doc strings

Doc strings przydają się do przekazywania większych bloków tekstów jako parametr

```
Given a blog post named "Random" with Markdown body
  """
  Some Title, Eh?
  ===============
  Here is the first paragraph of my blog post. Lorem ipsum dolor sit
  consectetur adipiscing elit.
  """
```

Argumenty - tabele z danymi

Tabele służą do przekazywania list wartości do definicji testu

```
Given a list of cyclist and their ridden distance
  | cyclist    | distance |
  | johnybravo | 11       |
  | rambo      | 30       |
```

Gherkin 6 - Rule

Słowo kluczowe Rule pozwala na grupowanie scenariuszy wg wspólnej reguły biznesowej

```
Feature: Pieszy przechodzi przez jezdnię
  Rule: przypadki dla zielonego światła
    Scenario: …
    Scenario: …
  Rule: przypadki dla czerwonego światła
    Scenario: ...
```

Grupowanie kroków i ich definicji

Cucumber potrafi znaleźć kroki w dowolnym pliku .java ze ścieżki glue, pod warunkiem zastosowania odpowiednich znaczników. Natomiast ze względu na dobre praktyki wskazane jest pogrupowanie kroków wg ich przeznaczenia i umieszczenie w osobnych plikach.

Np:

- EmployeeSteps.java
- EducationSteps.java
- ExperienceSteps.java
- AuthenticationSteps.java

Kleimy - jak połączyć feature file z kodem?

Aby poprawnie znaleźć definicje testów w kodzie java, Cucumber potrzebuje zdefiniowania ścieżki za pomocą tzw. glue

```
@CucumberOptions(glue = {"<package>", "<package>", "<etc>"})

 public class RunCucumberTest{}
```

Feature file - dobre praktyki

- Scenariusze powinny być proste i nie powinno ich być zbyt dużo w jednym pliku
  - Jeżeli opis staje się zbyt skomplikowany, należy rozważyć użycie wyższego poziomu abstrakcji
  - Jeżeli zaczyna być zbyt wiele scenariuszy, należy pogrupować je po części wspólnej i wydzielić do osobnych plików .feature
- Nie używamy Background do opisu skomplikowanych stanów, posługujemy się prostymi definicjami
  - Zamiast “Given: wchodzę na stronę z aplikacją, wpisuję dane i naciskam przycisk logowania” lepiej: “Given: I am logged in”
- Background powinien być krótki
  - Jeżeli sekcja Background zawiera więcej niż 4 linie, rozważamy utworzenie kroku o wyższym poziomie abstrakcji
- Używamy barwnych ale znaczących opisów
  - Mamy opowiedzieć historię. Zamiast: User1 lepiej: Pieszy, który chce przejść przez jezdnię