BDD & Cucumber

Artur Panek

PGS SOFTWARE

AGENDA

- BDD – Behaviour Driven Development
- Cucumber - implementacja dla języka Java
- Pierwszy projekt w użyciem Cucumber’a
- Feature, Scenario, Scenario Outline, Background
- @Given, @When, @Then, @And
- Junit Runner, @tags, maven profile
- @Before, @After
- picocontainer

BDD – Behaviour Driven Development

BDD to podejście oparte na TDD (Test Driven Development). W BDD, tak jak w TDD, zanim napisze się kod, tworzy się test. Różnica jest taka, że w TDD jest to zwykle test jednostkowy. W BDD są to testy akceptacyjne, związane z logiką biznesową programu.

Testowanie akceptacyjne jest przeprowadzane w celu określenia, czy dany moduł lub system spełnia określone potrzeby i realizuje procesy biznesowe.

Implementacje testów akceptacyjnych muszą stosować język dziedzinowy i nie zawierać szczegółów dotyczących metod współdziałania z aplikacją.

Utrzymanie testów akceptacyjnych na przestrzeni dłuższego czasu wymaga dyscypliny. Należy zwracać uwagę na utrzymanie wydajności testów oraz refaktoryzację. Zestawy testów akceptacyjnych muszą być refaktoryzowane w miarę dodawania nowych kryteriów akceptacyjnych by zapewnić niezakłóconą z nimi spójność.

BDD – Behaviour Driven Development

Stosowanie BDD poprzez wykorzystanie języka naturalnego do opisu funkcjonalności stanowi żyjącą, na bieżąco aktualizowaną dokumentację, którą można wykorzystać w dalszych etapach budowy systemu. Przy odpowiednim nacisku na testy, będą one zawsze aktualne, gdyż w przypadku konieczności wprowadzenia modyfikacji do funkcjonalności modyfikacja musi obejmować również testy.

Język dziedzinowy *(Gherkin)* pozwalający na przedstawienie kryteriów akceptacji został zaproponowany w następujący sposób:

Given ***Zakładając*** *pewien kontekst początkowy*
When ***Jeśli*** *wystąpi dane zdarzenie*
Then ***To*** *otrzymamy pewne rezultaty*

Cucumber

Cucumber jest narzędziem, które umożliwia definiowanie scenariuszy testowych za pomocą języka Gherkin oraz ich wykonywanie i raportowanie wyników.

Scenariusze testów zdefiniowane są wewnątrz plików typu *.feature

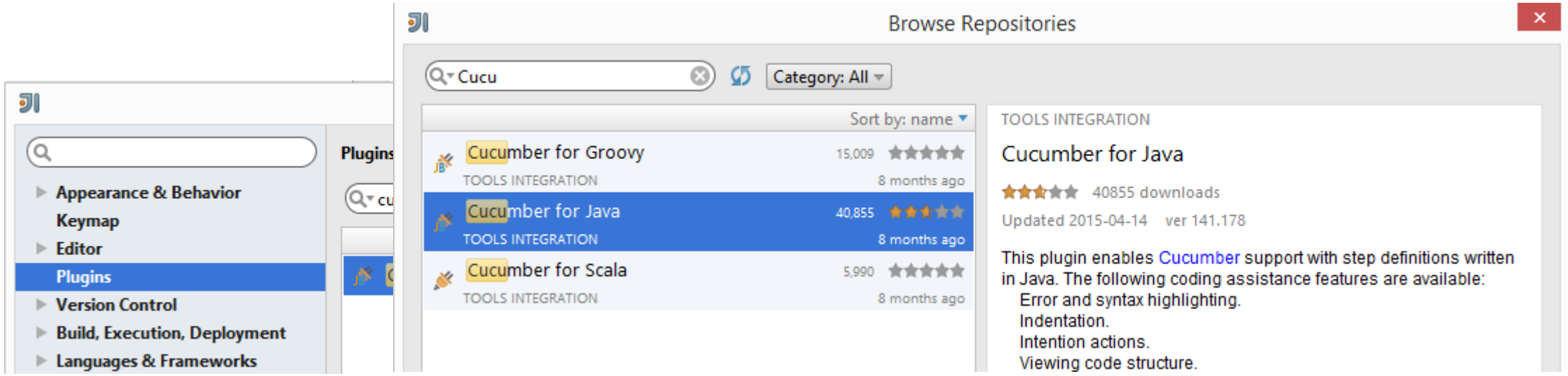
```
@search
Feature: Enter a search term into Google and view results

  Scenario: Submit search term
    Given I am on the website 'http://www.google.co.uk'
    When I submit the search term 'opencredo'
    And accept the first search result
    Then I should be on the page 'http://www.opencredo.com/'
```

Cucumber & IntelliJ

IntelliJ w wersji community wspiera pracę z Cucumber'em za pomocą dodatkowo instalowanego pluginu.

Instalacja pluginów:

Z głównego menu wybieramy: **File/Settings** i zaznaczamy na liście **Plugins**
Następenie wyszukujemy według klucza Cucumber i klikamy link „Browse".
Wybieramy „Cucumber for Java" i instalujemy.



Cucumber & IntelliJ

Bezpośredenie uruchamianie scenariuszy testów może wymagać określenia położenia definicji stepów oraz innych klas wykorzystywanych podczas uruchamiania:

W tym celu w IntelliJ wybieramy z głównego menu „Run/Edit Configurations”

Przechodzimy do „Defaults” na liście i wybieramy „Cucumber java”

Następnie w polu edycyjnym „Glue” wpisujemy nazwy package’y w których zdefiniowane są stepy oraz inne funkcje pomocnicze wykorzytujące adnotacje cucumber’a

Dla naszych przykładów wpisujemy: `steps hooks`

Cucumber & IntelliJ

Dalsze wyjaśnienia wymagają użycia przykładowej aplikacji oraz projektu testów:

1) Aplikacja do testowania:

   https://github.com/AnalyzeAppPerformance/spring-boot-example.git

2) Projekt zawierający testy:

   https://github.com/AnalyzeAppPerformance/cucumber-testing.git

PGS SOFTWARE

Dzięki! ☺