Falling in love with BDD

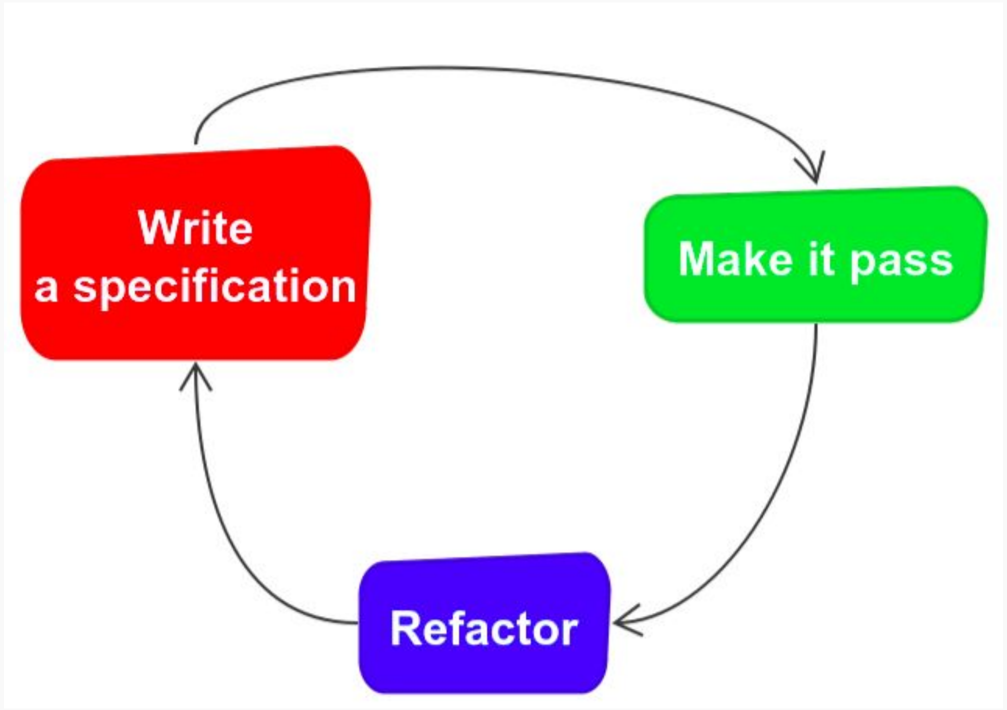
by Vaida Valukonytė

Let's start with a definition

Behaviour driven development - software development method that focuses on creating test using real-life examples.

IT'S ALL ABOUT
BEHAVIOUR
imgflip.com

TestFirst. Code second



Pros & Cons

+ Focus on the end user
+ Living documentation
+ Collaboration between users, developers and testers
+ Automated test creation from the beginning

- Time overhead
- Feature files management
- Writing tests up front is more difficult

BDD frameworks

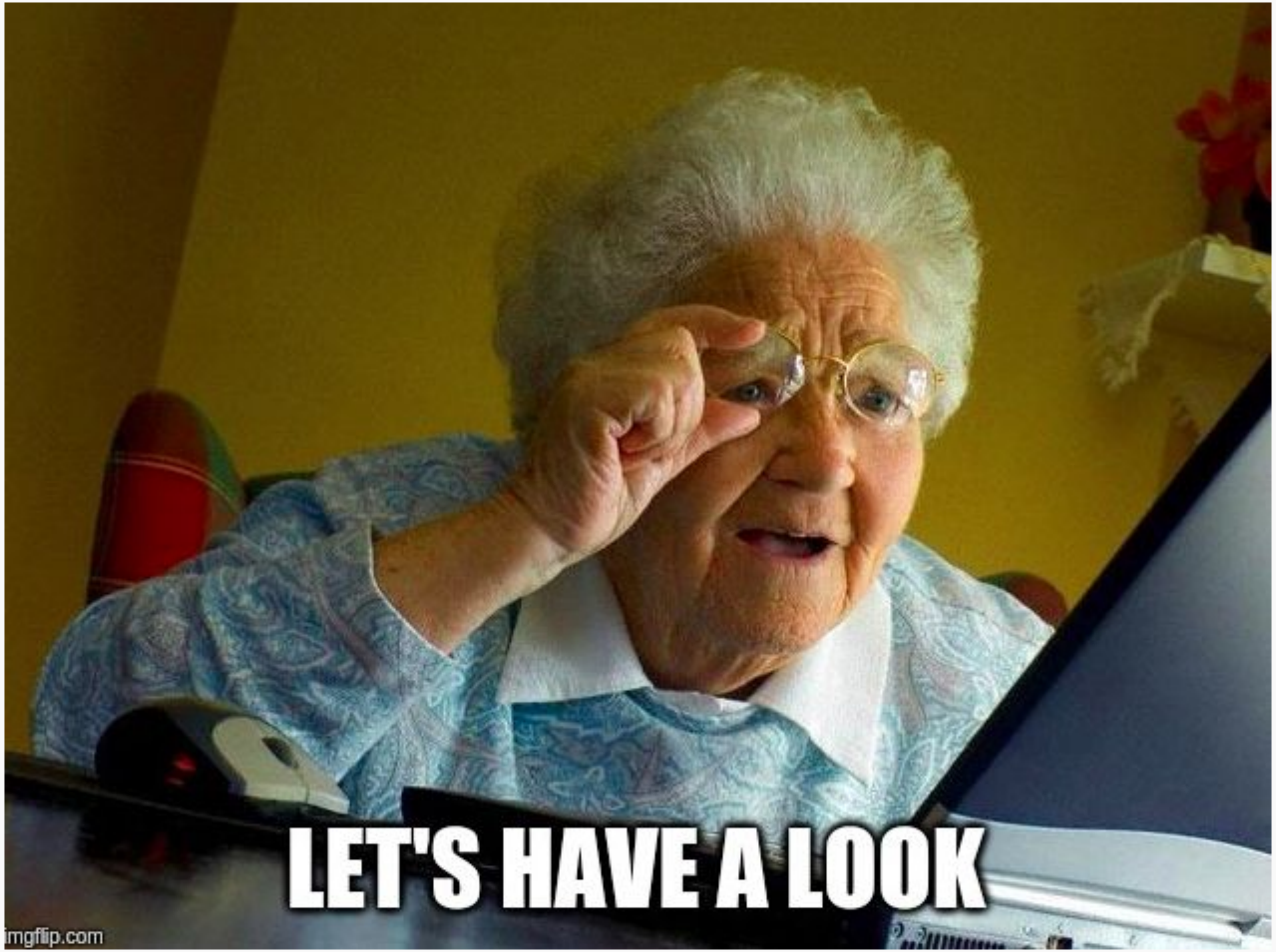
cucumber

Jasmine

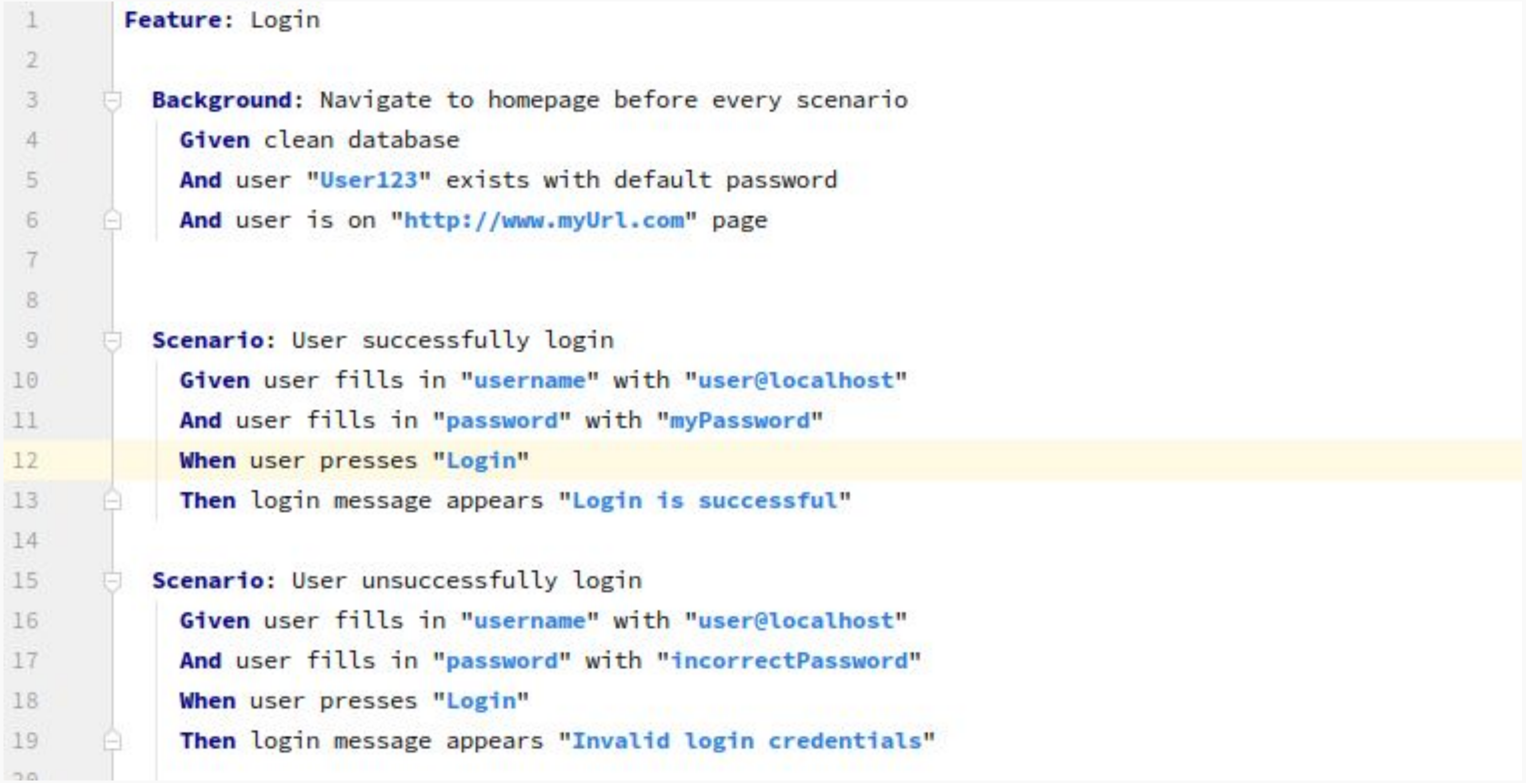
specflow

behat

https://codoid.com/bdd-framework-list/


LET'S HAVE A LOOK
imgflip.com

SATALIA



```
Feature: Login

  Background: Navigate to homepage before every scenario
    Given clean database
    And user "User123" exists with default password
    And user is on "http://www.myUrl.com" page


  Scenario: User successfully login
    Given user fills in "username" with "user@localhost"
    And user fills in "password" with "myPassword"
    When user presses "Login"
    Then login message appears "Login is successful"

  Scenario: User unsuccessfully login
    Given user fills in "username" with "user@localhost"
    And user fills in "password" with "incorrectPassword"
    When user presses "Login"
    Then login message appears "Invalid login credentials"
```

Feature-Background-Scenario

Feature - provides a high-level description of a software feature, and to a group related scenarios. Every *.feature file consists of a single feature.

Background - allows to add some context in the feature and runs before each scenario.

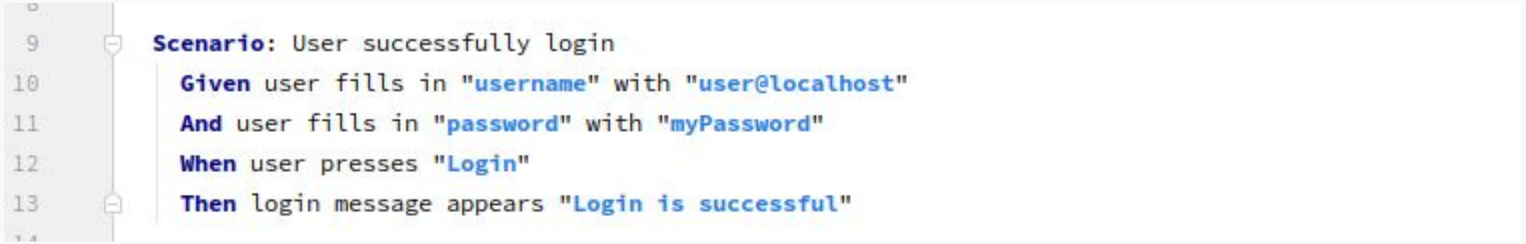
Scenario - description of project behaviour from one or more users perspectives.

Given-When-Then

Given Preconditions or Initial Context

When Event or Trigger

Then Expected output

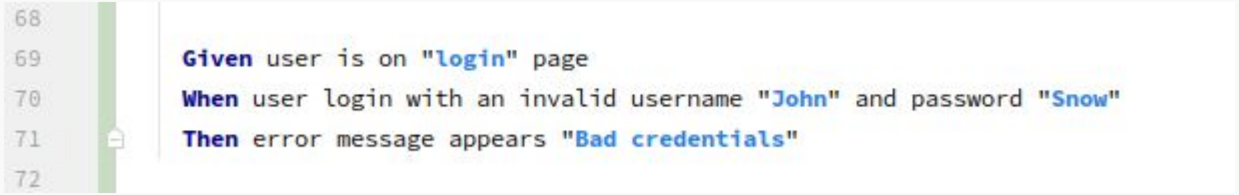


```
Scenario: User successfully login
  Given user fills in "username" with "user@localhost"
  And user fills in "password" with "myPassword"
  When user presses "Login"
  Then login message appears "Login is successful"
```

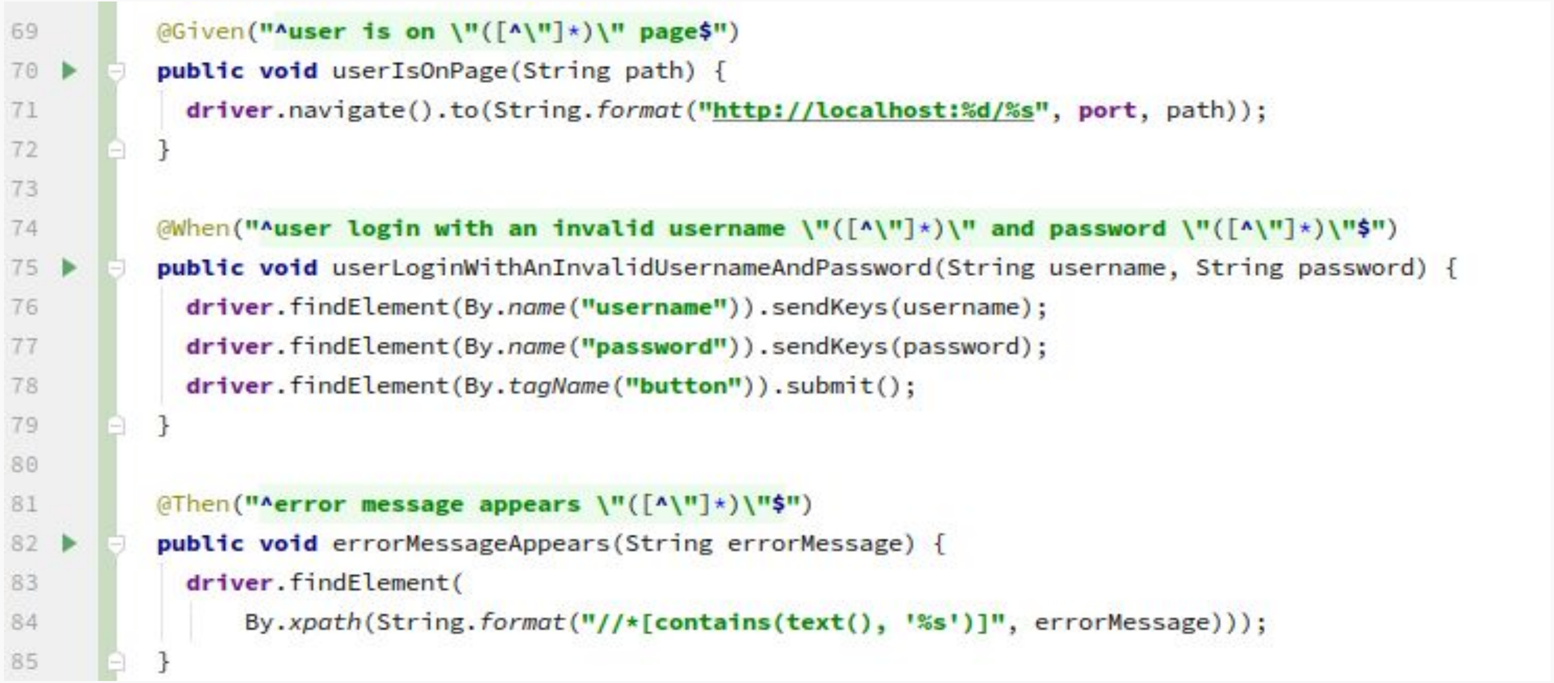
Step definitions

Step definition maps the Scenario Steps in the feature files (introduced by Given/When/Then) into code.

SATALIA



```

Given user is on "login" page
When user login with an invalid username "John" and password "Snow"
Then error message appears "Bad credentials"

```

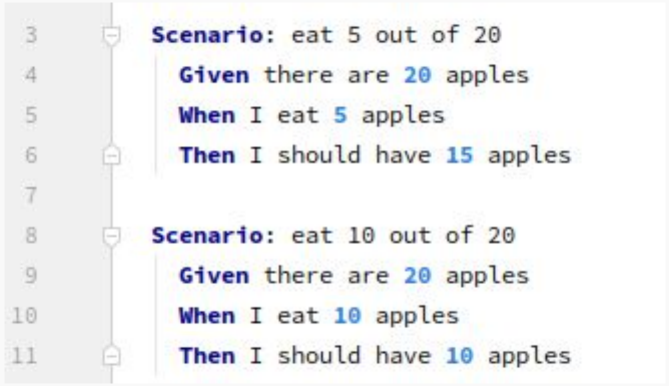


```java
@Given("^user is on \"([^\"]*)\" page$")
public void userIsOnPage(String path) {
  driver.navigate().to(String.format("http://localhost:%d/%s", port, path));
}

@When("^user login with an invalid username \"([^\"]*)\" and password \"([^\"]*)\"$")
public void userLoginWithAnInvalidUsernameAndPassword(String username, String password) {
  driver.findElement(By.name("username")).sendKeys(username);
  driver.findElement(By.name("password")).sendKeys(password);
  driver.findElement(By.tagName("button")).submit();
}

@Then("^error message appears \"([^\"]*)\"$")
public void errorMessageAppears(String errorMessage) {
  driver.findElement(
      By.xpath(String.format("//*[contains(text(), '%s')]", errorMessage)));
}
```

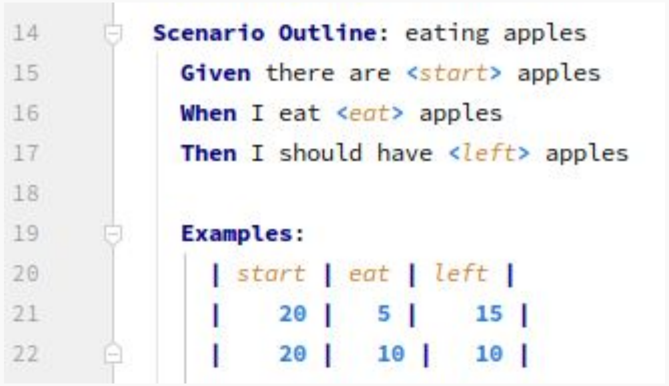
Parametrization

SATALIA

Parameterization - allows to run the same scenario for two or more different input data.



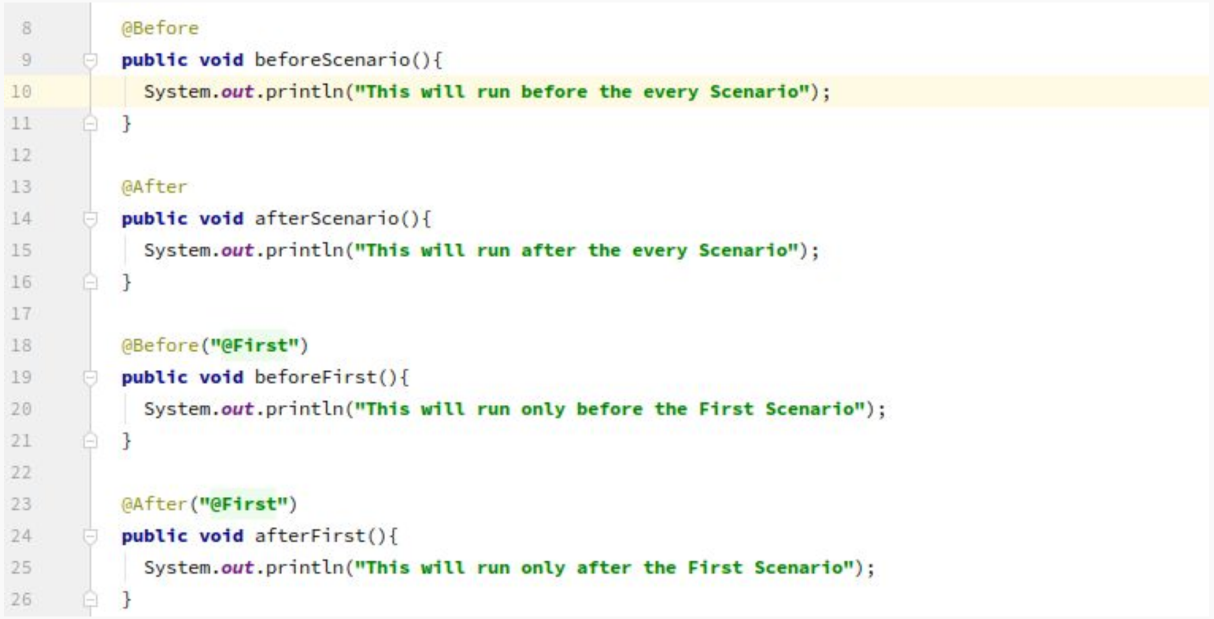
```
Scenario: eat 5 out of 20
  Given there are 20 apples
  When I eat 5 apples
  Then I should have 15 apples

Scenario: eat 10 out of 20
  Given there are 20 apples
  When I eat 10 apples
  Then I should have 10 apples
```



```
Scenario Outline: eating apples
  Given there are <start> apples
  When I eat <eat> apples
  Then I should have <left> apples

  Examples:
    | start | eat | left |
    |    20 |   5 |   15 |
    |    20 |  10 |   10 |
```

Hooks

Hooks - blocks of code that run ***before*** or ***after*** each scenario.



```
@Before
public void beforeScenario(){
    System.out.println("This will run before the every Scenario");
}

@After
public void afterScenario(){
    System.out.println("This will run after the every Scenario");
}

@Before("@First")
public void beforeFirst(){
    System.out.println("This will run only before the First Scenario");
}

@After("@First")
public void afterFirst(){
    System.out.println("This will run only after the First Scenario");
}
```

The purpose of hooks

- Starting webdriver
- Setting DB connections
- Setting test data
- Setting up browser cookies
- Anything else **before** the test

- Killing webdriver
- Closing DB connections
- Clearing test data
- Clearing up browser cookies
- Anything else **after** the test

Tips & Tricks

- Understand BDD process
- Know the code-base
- Be active and knowledgeable
- Write meaningful titles
- One test per scenario
- Use tables/tags
- Remember that missing scenarios may be bugs

Thank you!

vaida@satalia.com

Any questions?