HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

Final Countdown

Finishing the project & the course

AMT 2019

Olivier Liechti

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Understand how to use dependency injection when implementing your BDD tests with Cucumber

Comments about project requirements

Dates

Q&A

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

BDD for REST APIs with CucumberJVM

Linking the specs with the system

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Executable Specifications

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

Test Fixtures

```
public class TraderSteps { // look, Ma, I'm a POJO!!

    private Stock stock;

    @Given("a stock of symbol $symbol and a threshold
of $threshold")
    public void aStock(String symbol, double threshold)
{
        stock = new Stock(symbol, threshold);
    }

    @When("the stock is traded at $price")
    public void theStockIsTradedAt(double price) {
        stock.tradeAt(price);
    }

    @Then("the alert status should be $status")
    public void theAlertStatusShouldBe(String status) {
        ensureThat(stock.getStatus().name(),
equalTo(status));
    }

}
```

System Under Test (SUT)

BDD Kickstart - Boston, US - August 2017

An open-source tool for executable specifications

A vibrant community

An ingenious company

Dependency

If you are going to use the lambda expressions API to write the Step Definitions, you need:

```
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-java8</artifactId>
    <version>1.2.5</version>
    <scope>test</scope>
</dependency>
```

Otherwise, to write them using annotated methods, you need:

```
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>1.2.5</version>
    <scope>test</scope>
</dependency>
```

While it's not required, we strongly recommend you include one of the Dependency Injection modules as well. This allows you to share state between Step Definitions without resorting to static variables (a common source of flickering scenarios).

PicoContainer

Dependency

```
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-picocontainer</artifactId>
    <version>1.2.5</version>
    <scope>test</scope>
</dependency>
```

Step dependencies

The picocontainer will create singleton instances of any Step class dependencies which are constructor parameters and inject them into the Step class instances when constructing them.

Step scope and lifecycle

All step classes and their dependencies will be recreated fresh for each scenario, even if the scenario in question does not use any steps from that particular class.

If any step classes or dependencies use expensive resources (such as database connections), you should create them lazily on-demand, rather than eagerly, to improve performance.

Step classes or their dependencies which own resources which need cleanup should implement org.picocontainer.Disposable as described at http://picocontainer.com/lifecycle.html . These callbacks will run after any cucumber.api.java.After callbacks.

```
public class CreationSteps {

    private Environment environment;
    private DefaultApi api;
    private ApiResponse lastApiResponse;
    private ApiException lastApiException;
    private boolean lastApiCallThrewException;
    private int lastStatusCode;
    Fruit fruit;

    public CreationSteps(Environment environment) {
        this.environment = environment;
        this.api = environment.getApi();
    }

    @Given("^there is a Fruits server$")
    public void there_is_a_Fruits_server() throws Throwable
{
        assertNotNull(api);
    }

@Given("^I have a fruit payload$")
public void i_have_a_fruit_payload() throws Throwable {
    fruit = new io.avalia.fruits.api.dto.Fruit();
}
```

```
@When("^I POST it to the /fruits endpoint$")
public void i_POST_it_to_the_fruits_endpoint() throws Throwable {
    try {
        lastApiResponse = api.createFruitWithHttpInfo(fruit);
        lastApiCallThrewException = false;
        lastApiException = null;
        lastStatusCode = lastApiResponse.getStatusCode();
    } catch (ApiException e) {
        lastApiCallThrewException = true;
        lastApiResponse = null;
        lastApiException = e;
        lastStatusCode = lastApiException.getCode();
    }

}

@Then("^I receive a (\\d+) status code$")
public void i_receive_a_status_code(int arg1) throws Throwable {
    assertEquals(201, lastStatusCode);
}

}
```

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running io.avalia.fruits.api.spec.SpecificationTest
Feature: Creation of fruits

  Background:                          # creation.feature:3
    Given there is a Fruits server # CreationSteps.there_is_a_Fruits_server()

  Scenario: create a fruit                    # creation.feature:6
    Given I have a fruit payload              # CreationSteps.i_have_a_fruit_payload()
    When I POST it to the /fruits endpoint #
CreationSteps.i_POST_it_to_the_fruits_endpoint()
    Then I receive a 201 status code          # CreationSteps.i_receive_a_status_code(int)

1 Scenarios (1 passed)
4 Steps (4 passed)
0m0.496s

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.824 sec
```

Question

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

How can I reuse steps across multiple feature files?

In other words, how can I share state between different Step classes?

The key is to understand that:

- You use arguments in the Step constructor to inject dependency on the shared state. Pico Container manages these objects.
- The Step classes and shared state classes are re-instantiated for every scenario.
- Therefore, if you inject the same dependency in different Step classes involved in the same scenario, you effectively share state between them.

Question

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

How can I reuse steps across multiple feature files?

In other words, how can I share state between different Step classes?

http://www.thinkcode.se/blog/2017/04/01/sharing-state-between-steps-in-cucumberjvm-using-picocontainer

Question

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Ok… but in the example project (Fruits API), the lastApiResponse, lastApiException, lastApiCallThrewException and lastStatusCode are defined in the CreationStep class.

How could I reuse the step definition @Then("^I receive a (\\d+) status code$") that depends on them?

Refactor and move these fields in the Environment class, or in another dependency that you inject in the Step classes.

Question

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

What happens if I have the same Step definition in multiple classes? For instance, what happens if I have "I receive the status code" in multiple classes.

Nothing. Cucumber will not let you define ambiguous regular expressions and you will get a warning on the console when running the tests.

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Comments about project guidelines

- **Swagger** (**Open API**) to create a formal documentation of the REST APIs (this formal documentation has to be used in the development cycle);

Warning: I have a project that did not the openapi (swagger) plugin in maven. As a result, it did not implement the top-down + bottom-up process (which it has to). The easiest thing to do is to use our project skeleton to start your project.

- **CucumberJVM** to implement BDD tests.

Warning: writing scenarios to test HTTP status codes is a good start… but is not enough. Think about more meaningful tests and assertions to validate the behaviour of your server.

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Dates

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Project

Deadline for submission: Sunday, January 19th, 23h.

Final test

Monday, January 20th