LINGI2255

Software Development Project

Tests suites description - Team 06

Professor :
Kim Mens

Teaching Assistant :
Benoît Duhoux

Authors :
Ilias Boutchichi : 7693-12-00 - ilias.boutchichi@student.uclouvain.be,
Sophie Madessis : 2593-12-00 - sophie.madessis@student.uclouvain.be,
Tristan Moers : 2734-16-00 - tristan.moers@student.uclouvain.be
Samuel Monroe : 0467-10-00 - samuel.monroe@student.uclouvain.be,
Youri Mouton : 0746-16-00 - youri.mouton@student.uclouvain.be,
Antoine Rime : 3624-13-00 - antoine.rime@student.uclouvain.be,
Nicola Romano : 0182-17-00 - nicola.romano@student.uclouvain.be,
Rémy Voet : 6642-16-00 - remy.voet@student.uclouvain.be

2017-2018

1 Introduction

About our module within this project, we decided to back our work up with a solid testing suite. In order to accomplish this, we decided to use a combination of unit testing and feature testing. We will explain all of those in the following sections.

2 Feature Testing - Behaviour Driven Development

As the Unit Testing targets smaller and precise pieces of code to ensure a method/class behaviour, the feature testing is a higher-level way to test an application.

Indeed, a feature test will usually target a user story and test multiple scenarios about that user story. The test will automatically make sequences of actions, just as a real user would do, and check if the application reacts like expected.

2.1 BDD Environment

The Behaviour Driven Development Environment relies on the framework **Behave**, using the webdriver **Selenium** and the headless browser **PhantomJS**.

The main directory for this test is called **features**, located at the root of the app, and contains :

- The **environment.py** file, defining how the tests are executed.
  It is first composed of settings concerning the WebDriver, about the database used for testing, etc...
  You find also here definitions of actions to do before or after, for each or for all, steps or scenarios.

- ***.feature** files, which consist of explicit and english written scenarios belonging to a specific feature.
  Each scenario is composed of small steps written in the Gherkin syntax.

- A folder **steps** containing the definition of our stdeps written in the features definition.
  Each step is a method interacting with the DOM when the tests are launched.

In order to populate the test database with mock entries, we also use the package **FactoryBoy**. Defining factories allows us to inject entries and create relations satisfying constraints in the database, for using these in the test right after that. These factories are located in *./test/factories*.

3 Unit Tests

We are testing the behavior of our model in normal situation (no exceptions). This behaviors means to generate one or more solutions with different domain/image parameters. We do also test whether the model is able to handle unexpected behaviors like unknown parameters or not, etc...

Our testing framwork for the Unit Tests is called **Hamcrest**. For the moment, the tests are just located on the *tests.py* file in our module folder, although we will have to change this to have multiple test files and to avoid having thousands of lines in a single and too common testing file.