# Testing and Continuous Integration

## Fontys University of Applied Sciences

Exam year course

Mentor: Erik Schriek

Students: Bilger Yahov, Petya Mihova, Shristy Karki

Eindhoven, the Netherlands

13.12.2018

REST API built with Java

Technologies: Java, Gradle, CI/CD with Jenkins, Apache Tomcat, Artifactory, Git(Hub), IntelliJ IDEA, Mockito, Junit, Jsoup

The current document is meant to act as a guidance for the mentor while assessing students’ course work. It contains important information about the project work, the technologies used, the implementation methods and environment settings.

Any feedback for improvements is welcome. Enjoy!

Contents

[Testing and Continuous Integration 1](#_Toc503701782)

[Fontys University of Applied Sciences 1](#_Toc503701783)

[Assignment Description 4](#_Toc503701784)

[Used tools description 4](#_Toc503701785)

[Application functionality description 6](#_Toc503701786)

# Assignment Description

The students need to build a project using Java programming language under IntelliJ IDEA development environment. The project has to be a RESTful API, which runs a website crawler. The software piece accepts a query parameter as a URI to the website to be crawled, returning a JSON response with the data. The request should be GET.

Crucial steps while preparing and working on the assignment are to make sure that tests for the code are written as well as Continuous Integration is achieved using Jenkins and Gradle.

# Used tools description

In order to achieve the final result the following tools have been used:

* Jenkins, Gradle, Artifactory and Apache Tomcat

Those tools have been used to achieve the Continuous Integration. In order to complete this part of the course, the students got 2 working virtual machines from Amazon Web Services. On one of the machines they successfully installed Jenkins. On the other machine they installed Apache Tomcat and Artifactory.

Inside Jenkins a project has been configured. The configured project is connected with the GitHub repository of the software application package. It contains configuration details about the build, which gets executed every time a new commit is made to “master” branch on GitHub. In order to complete the build steps, the project inside Jenkins uses a “Gradle” package, which makes sure that all the tasks of the build are successfully executed. This means that whenever the build tasks get executed successfully the software application gets deployed to an environment. However if there is something failing during the build process, the app does not get deployed further. Stoppage is done with an error message.

When the success case is executed, the artifacts of the current deploy are published to Artifactory app, which as mentioned above is hosted on one of the Virtual Machines on AWS.

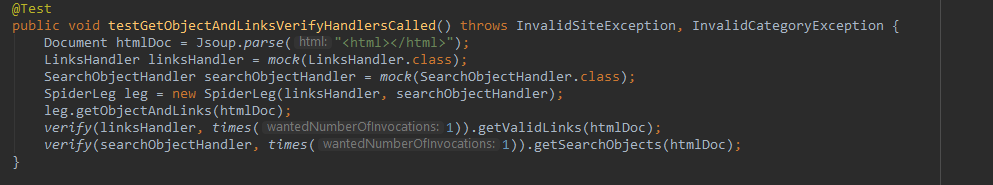
Artifactory is a single place, where all information about a certain build can be stored. Since the students did not decide to actually upload and store the app somewhere, they have made the Continuous Integration step with actually publishing the deploy artifacts. After executing a small research, and diving into a course about Jenkins with Gradle, they found out that the best tooling to be used is Artifactory, combined with Apache Tomcat.

More information about the deploy and build procedure with Continuous Integration can be seen at the video which is shared together with this deliverable. Also do not hesitate to contact the group to retrieve more information on this part of the project.

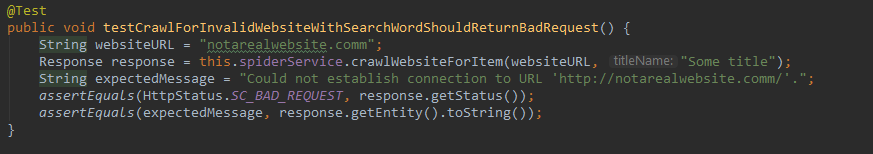
* Integration and Unit tests with Junit, Mockito, Jsoup

As already mentioned the other part of the course material is to cover testing. During the following course the students managed to work on two kinds of testing – unit testing and integration testing. Below a small information about those two types of testing can be found.

Unit testing – unit testing is all about testing small components of the code. It can be testing of a single function or it can be testing of just a small component. It does not require execution of the whole app in order to provide any results. Example unit test of the app can be seen below.



Integration testing – the integration testing is actually about making sure that the app makes the right decisions while running. It’s about testing the whole pieces working together. An integration test can only be run as long as the software application is running. An example integration test of the app can be seen below.



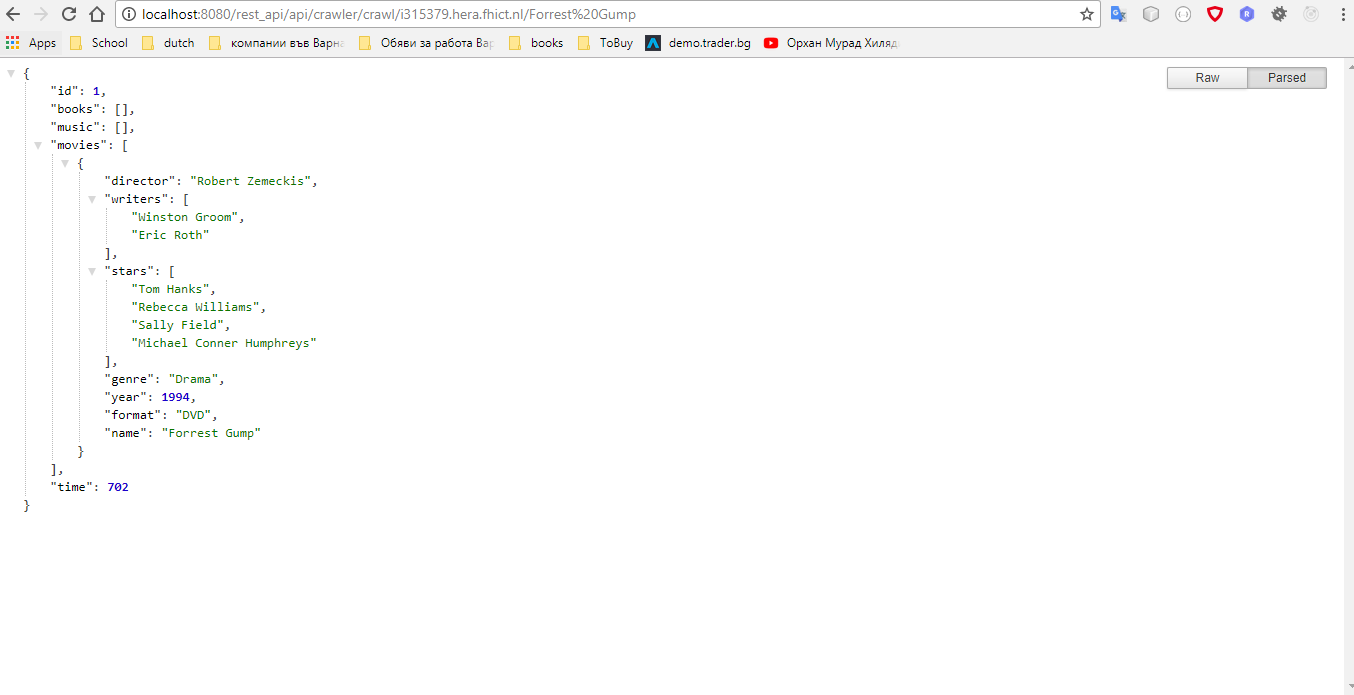
The tests of the software application can be run via executing the “gradle test” command. It automatically runs all the unit tests as well as the integration tests. While running the “gradle build” command, it makes sure that all the build steps are executed, before success or failure is reached. If there are any failing tests, the build fails, and the proper message is displayed on the console window. This description is also valid on a Jenkins CI/CD environment. The attached video to the current file will be demonstrating all those functions in details - what happens if the build fails and what happens if the build is successful.

# Application functionality description

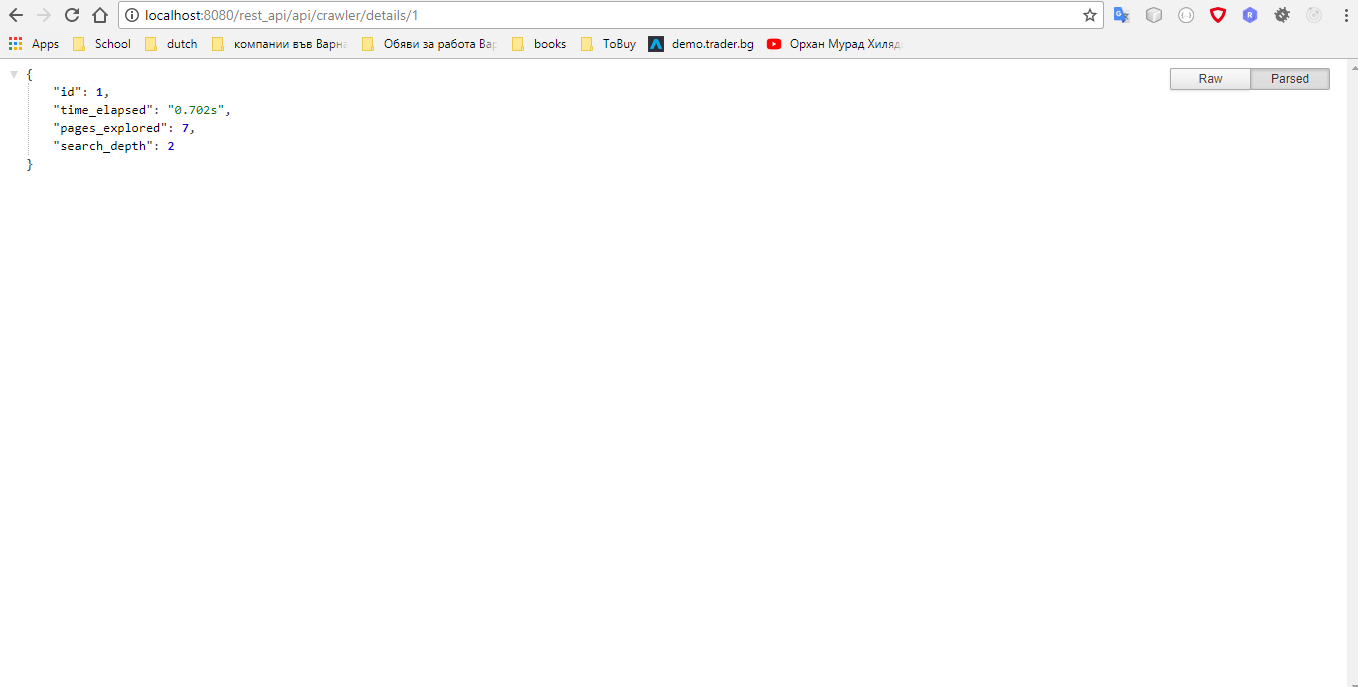
The application built is a RESTful API, which crawls a given website. The API has a GET method handler, which accepts a query parameter, holding the website URI to crawl. The expected result is to return a JSON formatted response, which holds the information, retrieved from the website. Image below illustrates the described actions.



This functionality crawls the whole website, making sure to retrieve information about books, music and movies. The application also can receive a continuation of the query parameter with an expression to search for. Let’s say we want to add the name “Forest Gump” to the query parameter.



In this case we can see the crawler taking a search action based on the parameters provided by the user. Another interesting case to mention is that the application also can give proper information about the crawls that have been previously done. Let’s say we want to see the first crawling action results and performance information.



This piece of information shows the details about the first crawling action. Starting from the ID, showing the time, pages and search depth performed.