FR\_AUTO\_TESTS

Javier Ventas

1. INTRODUCTION

The present document provides more information about the technical test performed by Javier Ventas.

The aim of the test is to create an automated test for a booking up to a decline payment. The language use has been java with junit and maven.

1. CLARIFICATIONS

The Selenium Page Object Pattern has been used. The page objects java classes are located in the package *<<com.ryanair.jvselenium.pageobjects>>.* The organization of the different elements into the java page objects classes is the following:

* **BookinHomePage**: it contains all the elements needed to perform a flight search.
* **FlightSelectionPage**: it contains all the elements needed to select one of the flight results from the search and check it out.
* **LoginPage**: it contains all the elements to allow the user to login into the system.
* **PaymentPassengerDetailsPage**: it contains all the elements needed to the user to fill the passenger information.
* **PaymentContactDetailsPage**: it contains all the elements needed to the user to fill the contact information.
* **PaymentMethodPage**: it contains all the elements needed to the user to fill the payment method and information.
* **PaymentBillingAddressPage**: it contains all the elements needed to the user to fill the payment billing address.
* **PaymentConfirmationPage**: it contains all the elements needed to confirm the payment and collect the authorized payment error.

Only the web elements needed for the test has been created. The distribution of elements and the page objects would be different in case of validating the whole system.

To locate the different web elements xpath has been used. Some people say that css is faster and more efficient though.

In some cases I had to use JavascriptExecutor in order to interact with these web elements.

I have decided to user random created security code (CVV) in the test to avoid errors from the system recognizing the booking attempt.

To select the date of the flight I have used only the day, letting the system to fill the month and year automatically, otherwise the test will not be valid after the day selected for the flight.

The test has been implemented to be executed against Firefox.

The project has been executed on Windows, but it should work on MAC.

1. CONFIGURATION

To be able to execute the test the following components must be installed and configured:

Java: <https://www.java.com/en/download/help/download_options.xml>

Maven : <https://maven.apache.org/install.html>

Geckodriver : <https://github.com/mozilla/geckodriver/releases>

There is a *<<config.properties>>* file that must be configured before building and executing the test. The file is located in << /jvselenium/src/main/resources>>. It contains the following properties

#Sun Jan 28 15:00:00 2018

driver=webdriver.gecko.driver

driver\_location=C:\\geckodriver-v0.19.1-win64\\geckodriver.exe

ryanairurl=https://www.ryanair.com/ie/en/

username=jventas@gmail.com

userpsw=RynairTest1

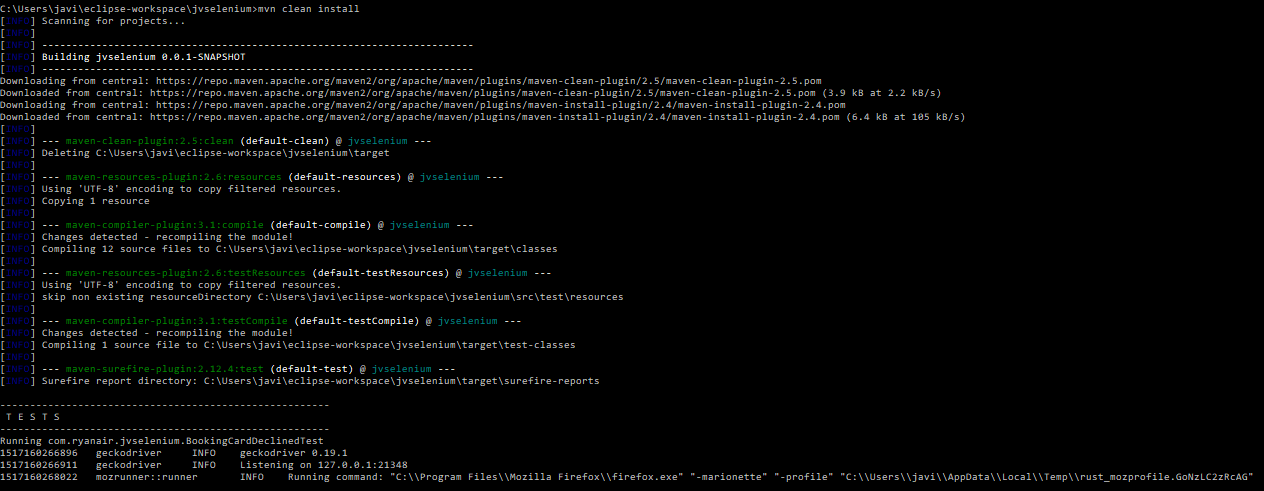
The property driver and driver\_location must contain the information of the driver installed in the target system.

1. BUILD AND EXECUTION

The source code can be found in the following url: <https://github.com/jventasg/selenium.git>

Once downloaded the code an updated the config.properties file the code will need to be built with maven. To build the project the following command should be executed on the project folder:

*mvn clean install*



Once build, the test case can be executed with the following command:

*mvn test*

1. REPORTING

We can generate automatically the surefire-report adding to the maven command “surefire-report:report”.

*mvn clean install test surefire-report:report*

It will generate an html report located on << jvselenium\target\site>>.

Attached to this document you will find an example. The style could be improved by using maven skins.

