Selenium Testing Tool

Carlos García González (alu0100898026)

Prácticas Externas Universidad de La Laguna
CodeBay

Abstract

In this article we will discuss about the Selenium testing tool, we will do a little introduction; talk about its advantages and disadvantages and then we will explain more in detail its possibilities and how this tool works.

Contents

1. Introduction

Selenium is a portable framework for testing web applications. Selenium provides a playback (formerly also recording) tool for authoring functional tests without the need to learn a test scripting language (Selenium IDE). It also provides a test domain-specific language (Selenese) to write tests in a number of popular programming languages, including C#, Groovy, Java, Perl, PHP, Python, Ruby and Scala. The tests can then run against most modern web browsers. Selenium deploys on Windows, Linux, and macOS platforms. It is open-source software, released under the Apache 2.0 license: web developers can download and use it without charge.

2. Why choose Selenium?

Nowadays there are many Testing Tools available so, why choose Selenium then? What differentiates Selenium from the other tools? For reply to this questions we will enumerate its advantages and disadvantages.

2.1. Advantages

- Its supported in the three more common operative system: Windows, Linux and OS X.
- Selenium has more supported languages than the most of the other testing tools: C#, Groovy, Java, Perl, PHP, Python, Ruby, Scala and JavaScript.
- It's a free and an open source tool.
- Selenium has numerous plugins and applications from external companies than greatly increases its capabilities.
- Has an IDEA (Selenium IDEA) with a playback option, that allows you to create and execute tests without knowing test scripting language and let you do it more quickly because you can instantiate the test recording the movement of the mouse in the screen.

- Its also compatible with other tools that the team are already using: Maven, IntelliJ, Jira, etc.
- Its widely used and its a standardized tool because of its versatility.
- And others of its advantages are synthesized here: Can drive a browser natively either locally or on remote machines using almost any programming language and testing framework (WebDriver and Remote Control) and can running test on parallel on many server at the same time (Grid).

2.2. Disadvantages

- Has a more hard learning curve than the others because it have more options and you have to configure it by yourself.
- Most of the external applications are payment applications and only some of them offers a little trial version.

3. How Selenium works?

In this section we will focus on the steps for setting up the Selenium test environment in the supported language Java, using the tool Maven and the IDEA IntelliJ. Also we will see an example and the documentation for a better understanding.

3.1. Setting up Selenium tests in IntelliJ

Before we can start setting up the environment we have to make sure that we have some tools already installed:

1. Maven.
2. IntelliJ.
3. Selenium WebDrivers (IEDriver, GeckoDriver, ChromeDriver, etc).
4. Java SE Development Kit.
5. Selenium IDEA (Optional).

Once you have all of these you can start to create the project following the next steps:

1. Open IntelliJ and create a new Maven project (in the fields "GroupId" and "ArtifacId" put the name of the project).
2. Configure the pom.xml file for include the dependencies that you will need for the Selenium tests (at least you have to import the dependencies *selenium-api* and *selenium-java*. You can find the dependencies you need here).
3. Save the file and run in the IntelliJ terminal the following command: *mvn clean install* (if you need more information while it execute try: *mvn -e clean install*).
4. Create a new Java Module in the project root and create a new Java class inside of the module. This class will contain the test that you want to execute.
5. Program all the Selenium Java test as you want. Here you have an example of a simple test that open the browser, navigate to Google, search the string *Cheese!* in it and checks the title of the resultant page, if it starts with *cheese!* the test pass. Here you have another example more complex using an AEM web page.

4. Bobcat

Now that we have already introduce Selenium Java tests its time to present Bobcat. Bobcat is a framework that wrap Selenium and greatly increases its functionality. Moreover is a free and an open source tool that is designed specifically to do tests with AEM web pages using Selenium.

4.1. Setting up Bobcat tests in IntelliJ

Before we can start setting up the environment we have to make sure that we have all the tools that we needed before and a few new tools already installed like:

1. Gradle.
2. Android SDK.

Once we have all of this we can develop a Bobcat test:

1. Go to this repository and clone it in a folder of your choice. It have all the programs necessary for easily instantiate a Bobcat project.
2. Follow the instructions of the repository for generate a Bobcat project.
3. After it is done we open Intellij and open the Bobcat project for start to develop the tests.

5. Documentation and useful links

This sections has the porpoise of increase your knowledge through the officials documentations and links of interests:

- Maven official site.
- Maven repository.
- Maven dependencies search.
- IntelliJ official site.
- Download Selenium WebDrivers.
- Download Selenium WebDrivers (command-line).
- Download Java SE Develoment Kit.
- Selenium Documentation.
- More Selenium Documentation.
- Selenium Wiki.
- Selenium Java API Documentation.
- Download Selenium IDE.
- Selenium IDE Documentation.
- Official Gradle page.

- Install Gradle.
- Gradle Documentation.
- Gradle Tutorials and Guides.
- Download Android Studio or Android SDK only.
- Bobcat GitHub.
- Getting started in Bobcat.
- Bobcat Wiki.
- Bobcat API Documentation.
- How to start a Bobcat project.