WebTesting using Cucumber with Selenium

Tutorial

Denis Reibel

ePortfolio 31. October 2018

Software Engineering

Prof. K. M. Berkling

Table of Contents

1. Introduction

This Tutorial shows you how to write a Webtest using Cucumber and Selenium.

You will learn how to Setup your Project. Further you will get a basic understand of Cucumber and how it works together with Selenium.

In the End you will write your own Webtest and have the ability to test your own Websites.

2. Setup

The easiest way to use Cucumber with Selenium is to create a Maven Project and include the dependencies.

You can download Maven here: https://maven.apache.org/download.cgi .

If you dont know how to install Maven you can follo this guide: https://www.mkyong.com/maven/how-to-install-maven-in-windows/ .

If your Maven is running correctly, create a new Project. You can do this by using the Cucumber-Archtype in your Command Line.

1. Open your Command Line
2. Go tot he location where you want to create the Project
3. Run the following Command in your Command Line:

```
mvn archetype:generate                               \
    -DarchetypeGroupId=io.cucumber                   \
    -DarchetypeArtifactId=cucumber-archetype         \
    -DarchetypeVersion=2.3.1.2                       \
    -DgroupId=hellocucumber                          \
    -DartifactId=hellocucumber                       \
    -Dpackage=hellocucumber                          \
    -Dversion=1.0.0-SNAPSHOT                         \
    -DinteractiveMode=false                          \
```

4. Afterwards run: „mvn clean install“

When you created your Project open it with your IDE.

Locate the „pom.xml“-file and open it.

Include the following Code into your <dependencies>-Tag:

```
<dependencies>
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>2.3.1</version>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-junit</artifactId>
        <version>2.3.1</version>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>


    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.13.0</version>
    </dependency>
</dependencies>
```

Now you have a running Project which includes Cucumber and Selenium.

3. Cucumber

Cucumber is a Technology to write Testcases. The main reason to use Cucumber is, that you can write Testcases which are readable for everyone.

Cucumber is build by using .feature-Files to specify your tests.

Typical .feature-Files will look like this:

```
Feature: Does the Calculator work?
  I want to know if my Calculator adds and multiplicates correctly

  Scenario: Can my Calculator add two numbers?
    Given I type in 3
    When I add the number 4
    Then The Calculator should add and return 7


  Scenario Outline: Can my Calculator multiplicate numbers with 5?
    Given I type in 5
    When I type in the number "<firstNumber>"
    Then The Calculator should multiplicate and return "<answer>"

    Examples:
      | firstNumber | answer |
      | 2 | 10 |
      | 5 | 25 |
      | -6 | -30 |
```

Each Keywords describe different things:

- Feature: describes what the whole file is used to test
- Scenario: describes what the exact testcase tests
- Given: describes what ist he starting point fort he test
- When: describes a user action
- Then: describes the expected output

You can implemented the .feature Files by using every possible type of Source Code. For this you can generate „StepDefinitions“-Files. The StepDefinition for the .feature File from above look like this:

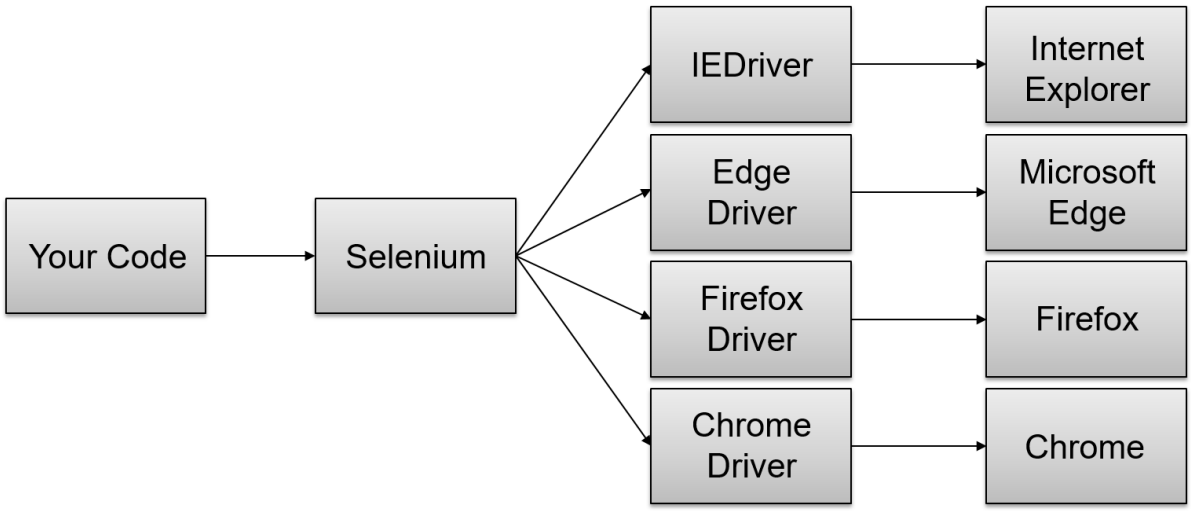
```java
public class calculatorStepDefinitions {

    private Calculator calc = new Calculator();
    private int firstNumber;
    private int secondNumber;

    @Given("^I type in (\\d+)$")
    public void iTypeIn(int first) { firstNumber=first; }

    @When("^I add the number (\\d+)$")
    public void iAddTheNumber(int second) { secondNumber=second; }

    @Then("^The Calculator should add and return (\\d+)$")
    public void theCalculatorShouldAddAndReturn(int sum) {
        assertEquals(sum, calc.add(firstNumber,secondNumber));
    }

    @When("^I type in the number \"([^\"]*)\"$")
    public void iTypeInTheNumber(int second) { secondNumber=second; }

    @Then("^The Calculator should multiplicate and return \"([^\"]*)\"$")
    public void theCalculatorShouldReturn(int answer) throws Throwable {
        assertEquals(answer, calc.multiplicate(firstNumber,secondNumber));
    }
}
```

So each Cucumber Line gets matched to an own Java method, which you can implement as you want.

4. Selenium

Selenium is a Framework to automatically control a webbrowser Instance. It simulates all the most common Browsers on the Market.

How Selenium works is shown on this picture:



Selenium includes a huge variety of Commands, which i cannot completly include here. You can look them up here: https://www.seleniumhq.org/docs/ .

5. Example Test

This Section shows an example of how to use Cucumber with Selenium for automatetd Webtests. The example opens the google home page, searchs for „katzenbabys“ and checks if the page title iss et correctly.

.feature File:

```
Feature: Does Google sets the Page Title correctly?

  Scenario: I search a keyword in google
    Given I open the Chrome Browser and go to "http://www.google.com"
    When I write "katzenbabys" into the search bar
    And I klick on the serach button
    Then the page title should be set correctly
```

StepDefinitions:

```
public class googleSearchStepDefinitions {

    private WebDriver driver;
    private String pathToChromeDriver= "..\\hellocucumber\\driver\\chromedriver.exe";
    private String pathToFirefoxDriver= "..\\hellocucumber\\\\driver\\\\geckodriver.exe";

    @Given("^I open the Chrome Browser and go to \"([^\"]*)\"$")
    public void iOpenTheChromeBrowserAndGoTo(String url) throws Throwable {
        System.setProperty("webdriver.chrome.driver",pathToChromeDriver);
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get(url);
    }

    @When("^I write \"([^\"]*)\" into the search bar$")
    public void iWriteIntoTheSearchBar(String keyword) throws Throwable {
        driver.findElement(By.id("lst-ib")).sendKeys(keyword);
        Thread.sleep( millis: 1000);
    }

    @And("^I klick on the serach button$")
    public void iKlickOnTheSerachButton() throws Throwable {
        driver.findElement(By.className("sbgs_c")).click();
    }

    @Then("^the page title should be set correctly$")
    public void thePageTitleShouldBeSetCorrectly() throws Throwable {
        assertEquals( expected: "katzenbabys - Google-Suche", driver.getTitle());
    }

}
```

6. Conclusion

After this Tutorial you should be able to use Cucumber and Selenium together to automate your Webtests.

Be sure to test every part of your Website carefully so there won't be any Bugs.

For more Information about this topic you can checkout the Doumentation of Cucumber(https://docs.cucumber.io/guides/10-minute-tutorial/) and Selenium(https://www.seleniumhq.org/docs/ .). Also you can checkout my github repository (https://github.com/DRiXD/CucumberSelenium) to see more Webtest examples.