



ePortfolio: Testen von Webapplikationen mit Cucumber und Selenium in Eclipse

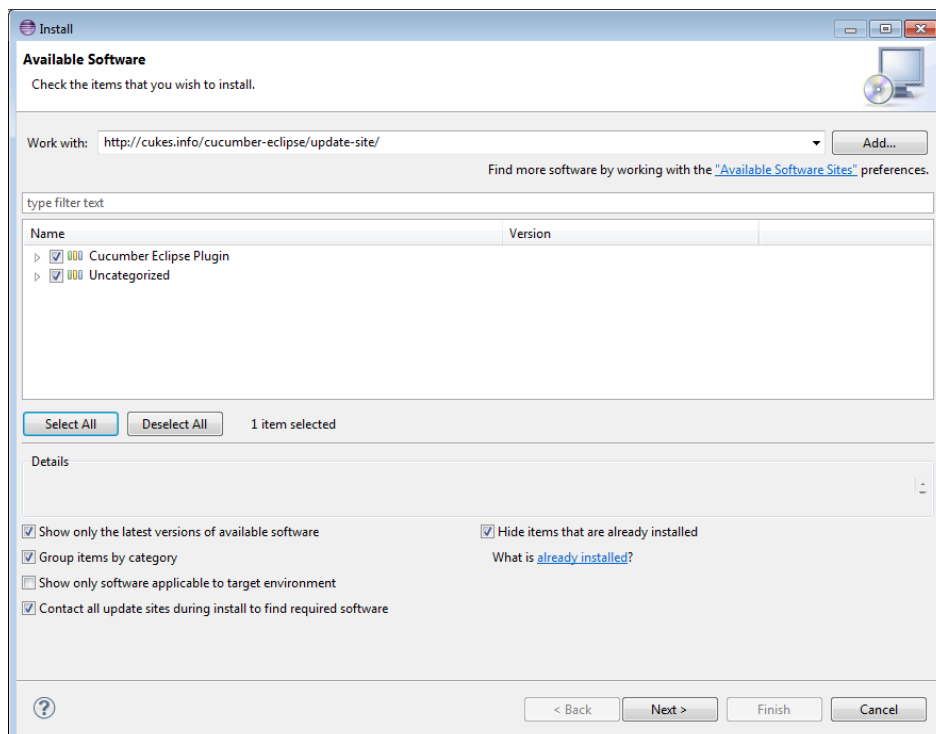
Vorbereitung

Installieren des Selenium WebDrivers

Die WebDriver für die jeweiligen Browser eurer Wahl stehen unter <http://www.seleniumhq.org/download/> bereit. Weitere Informationen zu deren Nutzung sind unter den Selenium Wiki Links zu entnehmen. In diesem Tutorial werden die Tests unter Firefox durchgeführt; für diesen ist unter dem selben Link das Selenium IDE Plugin verfügbar, mit dem der Testcode automatisch generiert werden kann.

Installieren des Cucumber-Plugins in Eclipse

Für das Syntax Highlighting der .feature-Files wird das Cucumber-Plugin benötigt. Dies installiert man in Eclipse unter Help > Install New Software.

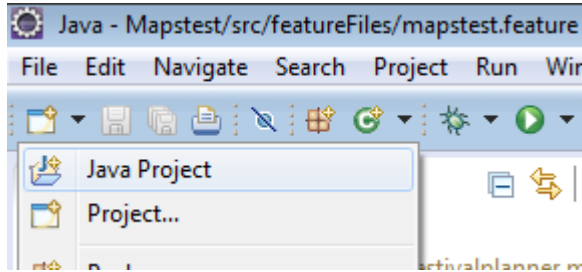


Im darauffolgenden Fenster gibt man unter „Work With“ den Link zur Plugin-Seite (<http://cukes.info/cucumber-eclipse/update-site/>) ein, drückt Enter, klickt auf „Select All“ und dann mehrfach auf „Next“. Zum Schluss werden die Lizenzvereinbarungen akzeptiert und der Installationsvorgang mit „Finish“ beendet.

Erstellen unseres Testprojektes

Erstellen eines neuen Projektes und Einbinden der benötigten Bibliotheken

1. Zunächst erstellen wir ein neues Java Projekt:

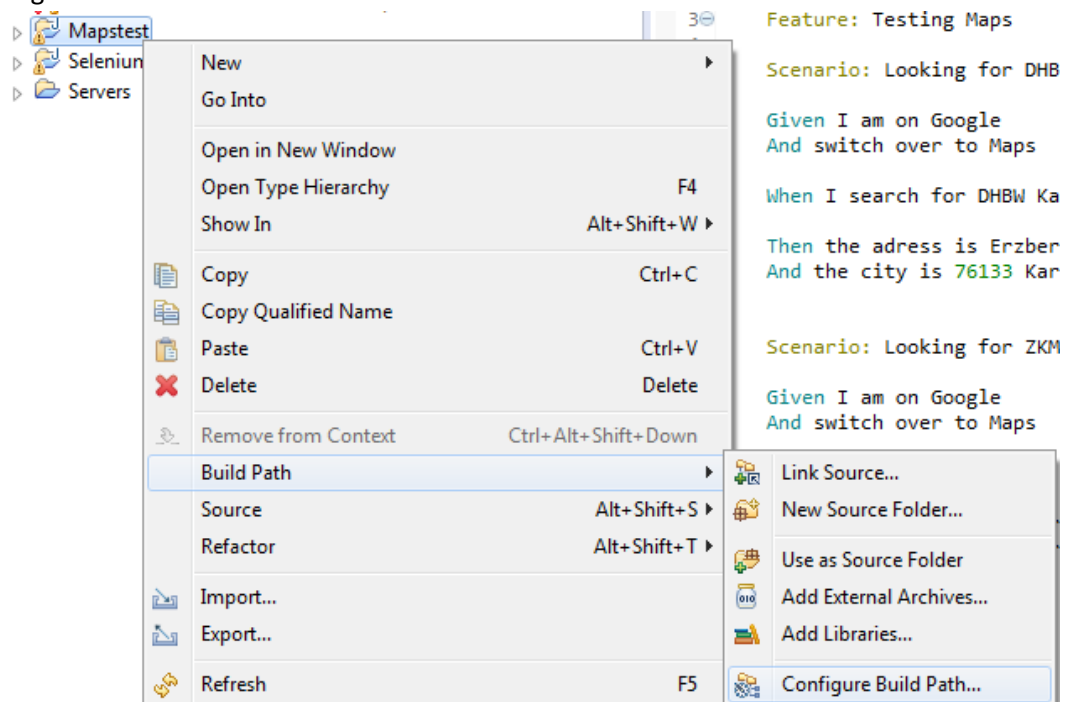


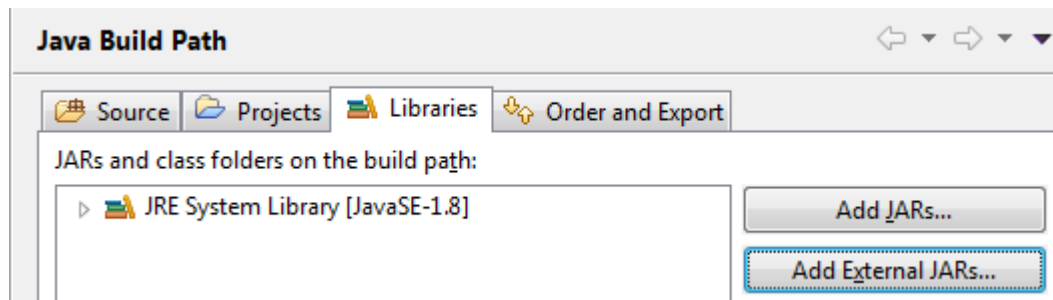
Nach Eingabe des Projektnamens können wir bereits auf „Finish“ klicken.

2. Um Cucumber und Selenium in unserem Projekt zu nutzen, benötigen wir einige .jar-Dateien. Diese können etwa unter <http://mvnrepository.com/> heruntergeladen werden. Es ist darauf zu achten, dass es sich hierbei stets um die aktuellen Versionen handelt.

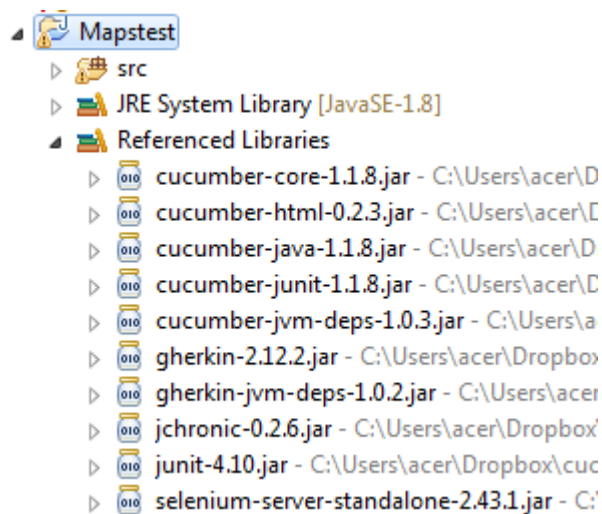
- cucumber-core-<versionsnummer>.jar
- cucumber-html-<versionsnummer>.jar
- cucumber-java-<versionsnummer>.jar
- cucumber-junit-<versionsnummer>.jar
- cucumber-jvm-deps-<versionsnummer>.jar
- gherkin-<versionsnummer>.jar
- gherkin-jvm-deps-<versionsnummer>.jar
- junit-<versionsnummer>.jar
- selenium-server-standalone-<versionsnummer>.jar

3. Diese Dateien werden jetzt nach einem Rechtsklick auf unser Projekt in dessen Build-Path eingebunden:



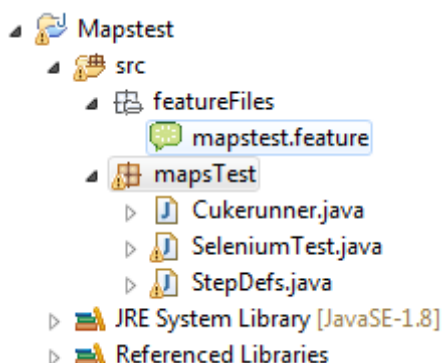


Nachdem im folgenden Dialogfenster die oben genannten Dateien ausgewählt wurden und nochmals auf OK geklickt wurde, sind diese nun in unserem Projekt unter „Referenced Libraries“ zu finden:



Erstellen der benötigten Klassen

Nun erstellen wir die beiden Packages `mapsTest` und `featureFiles`; in `mapsTest` erstellen wir wiederum die Klassen `Cukerunner.java`, `SeleniumTest.java` und `StepDefs.java`; während `featureFiles` eine `mapstest.feature` Datei enthalten soll:



`Cukerunner.java` enthält einige Konfigurationen, die das Verzeichnis der `.feature`-Files sowie das Ausgabeformat der Reports festlegen. Nähere Informationen zu möglichen CucumberOptions erhaltet ihr hier: <http://xebbie.xebbie.in/index.php/2014/07/31/cucumber-jvm-all-you-need-to-know-about-cucumeroptions/>

```

package mapsTest;

import org.junit.runner.RunWith;

import cucumber.api.CucumberOptions;

import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)

@CucumberOptions(

    features = {"src/"},

    format = {"pretty",

        "html:target/cucumber-html-report",

        "junit:target/cucumber-junit-report/allcukes.xml"}

    )

public class Cukerunner {

}

```

In SeleniumTest.java wird der WebTreiber definiert: Wir legen beispielsweise fest, welchen Browser wir verwenden und welche URL getestet werden soll.

```

package mapsTest;

import java.util.concurrent.TimeUnit;

import static org.junit.Assert.*;

import org.openqa.selenium.*;

//import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

//import org.openqa.selenium.htmlunit.HtmlUnitDriver;

//import org.openqa.selenium.ie.InternetExplorerDriver;

import org.openqa.selenium.remote.RemoteWebDriver;

//import org.openqa.selenium.safari.SafariDriver;

public class SeleniumTest {

    private WebDriver driver;

    private String baseUrl,browserName,browserVersion;

    public void setUp() throws Exception {

        //driver = new HtmlUnitDriver();
    }
}

```

```

driver = new FirefoxDriver();

//driver = new ChromeDriver();

//driver = new InternetExplorerDriver ();

//driver = new SafariDriver();

baseUrl = "http://www.google.de/";

driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

driver.manage().window().maximize();

Capabilities caps = ((RemoteWebDriver) driver).getCapabilities();

browserName = caps.getBrowserName();

browserVersion = caps.getVersion();

System.out.println("Automated test run. We're running on "+browserName+"
"+browserVersion);

}

public void tearDown() {

driver.quit();

}

public void goToHomePage(){

driver.get(baseUrl);

}

}

```

Die StepDefs.java wird später festlegen, was bei den im .featureFile definierten Schritten genau passieren soll.

```

package mapsTest;

import cucumber.api.PendingException;
import cucumber.api.java.en.*;
import cucumber.api.java.After;

public class StepDefs {

    private SeleniumTest script;

    public void setUpWebDriver() throws Exception {

        script = new SeleniumTest();

        script.setUp();

        script.goToHomePage();
    }
}

```

```

    }

    public void tidyUp() {

        script.tearDown();

    }

}

```

Kommen wir nun zu der mapstest.feature-Datei, in der wir nun in der in der Vorlesung behandelten Gherkin-Syntax unsere Testszenarien definieren. In unserem Test möchten wir gerne überprüfen, ob Google Maps zur DHBW Karlsruhe die richtige Adresse liefert:

```

Feature: Testing Maps

Scenario: Searching for DHBW Karlsruhe

Given I am on Google
And switch over to Maps

When I search for DHBW Karlsruhe

Then the address is Erzbergerstrasse 121
And the city is 76133 Karlsruhe

```

Mit Rechtsklick auf diese Datei können wir nun unter Run As > Cucumber Feature den Test durchlaufen lassen:

```

Feature: Testing Maps

  Scenario: Searching for DHBW Karlsruhe    # Z:/studium/dh/software
engineering/Mapstest/src/featureFiles/mapstest.feature:5
    Given I am on Google
    And switch over to Maps
    When I search for DHBW Karlsruhe
    Then the address is Erzbergerstrasse 121
    And the city is 76133 Karlsruhe

1 Scenarios (1 undefined)
5 Steps (5 undefined)

0m0,000s

```

Cucumber erkennt also, dass noch nicht definiert wurde, was in den einzelnen Schritten zu tun ist. Weiterhin greift es uns in der Konsolenausgabe mit bereits mit folgenden Codeschnipseln unter die Arme. Anmerkung: Zahlen im .feature-File werden als Parameter interpretiert, was bei den zu definierenden Methoden berücksichtigt werden muss.

You can implement missing steps with the snippets below:

```

@Given("^I am on Google$")
public void i_am_on_Google() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Given("^switch over to Maps$")
public void switch_over_to_Maps() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
}

```

```

        throw new PendingException();
    }

    @When("^I search for DHBW Karlsruhe$")
    public void i_search_for_DHBW_Karlsruhe() throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @Then("^the adress is Erzbergerstrasse (\\d+)$")
    public void the_adress_is_Erzbergerstrasse(int arg1) throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @Then("^the city is (\\d+) Karlsruhe$")
    public void the_city_is_Karlsruhe(int arg1) throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }
}

```

Diese können wir nun in unsere StepDefs.java übernehmen. Den Rumpf der Methode `i_am_on_Google()` können wir durch die `setUpWebDriver()` Methode bereits implementieren:

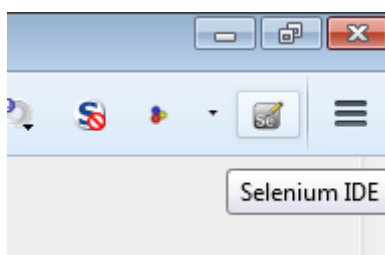
```

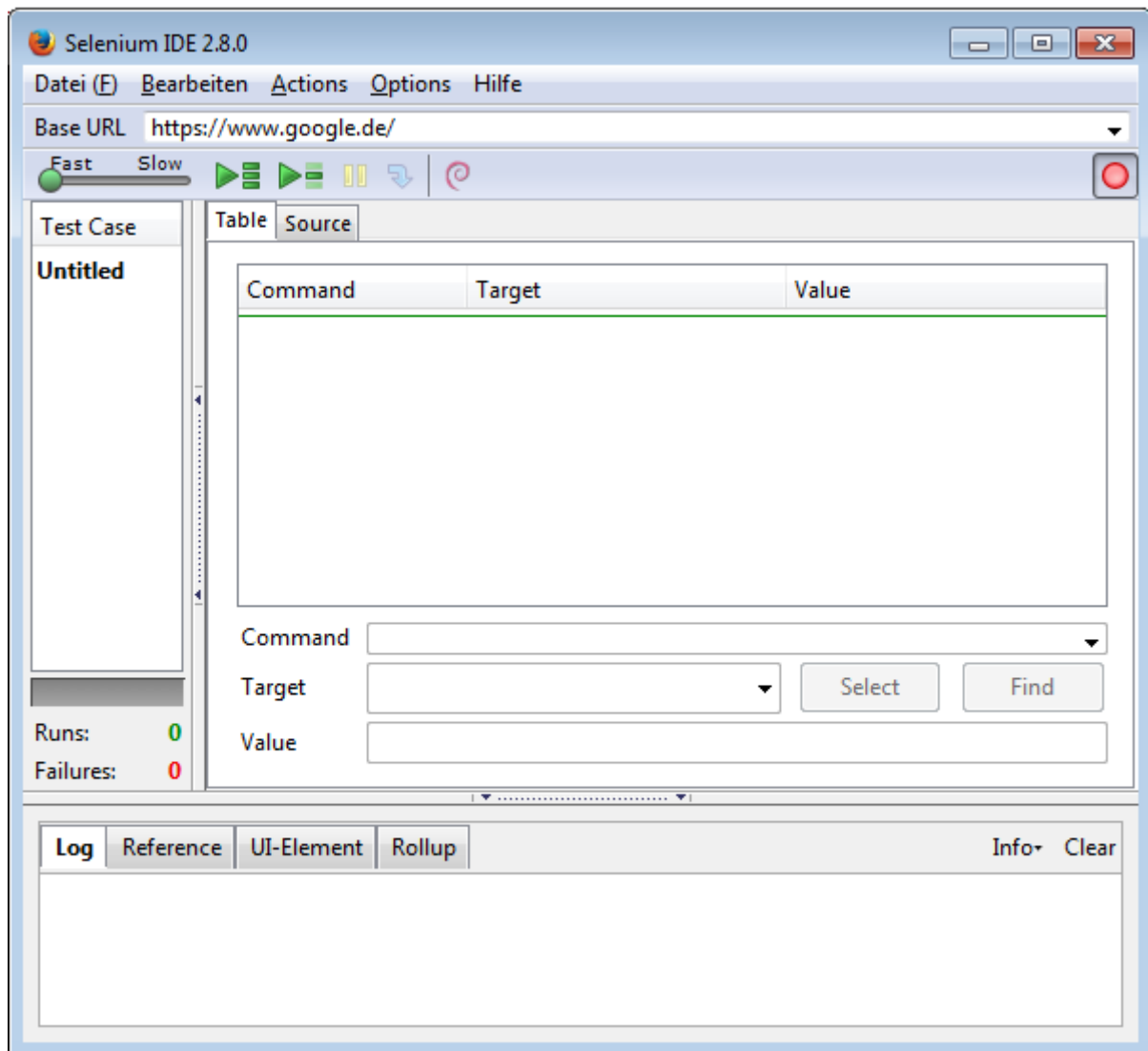
@Given("^I am on Google$")
public void i_am_on_Google() throws Throwable {
    setUpWebDriver();
}


```

Generieren des restlichen benötigten Codes

Mit der Selenium IDE in Firefox können wir den restlichen Code ganz einfach generieren lassen. Dazu gehen wir auf die Google-Startseite und öffnen die Selenium-IDE, welche oben rechts in Firefox zu finden ist:





Um unseren Code zu generieren, muss die Selenium IDE im Aufnahmefokus sein; dies erkennt man am hellroten Punkt oben rechts (siehe Bild). Ist der Aufnahmefokus nicht aktiviert, ist dieser Punkt dunkelrot .

Ist die IDE im Aufnahmefokus gehen wir ganz einfach die Schritte manuell durch; wir klicken also auf das Maps Symbol, geben dann „DHBW Karlsruhe“ ein, klicken auf den Suchbutton und lassen uns das Ergebnis anzeigen:

+Ich Gmail Bilder



Anmelden



+Ich



Suche



YouTube



Maps



Play



News



Gmail



Drive



Kalender

Mehr

dhbw karlsruhe



**DHBW Karlsruhe (ehem.
BA Karlsruhe)**

Erzbergerstraße 121
76133 Karlsruhe



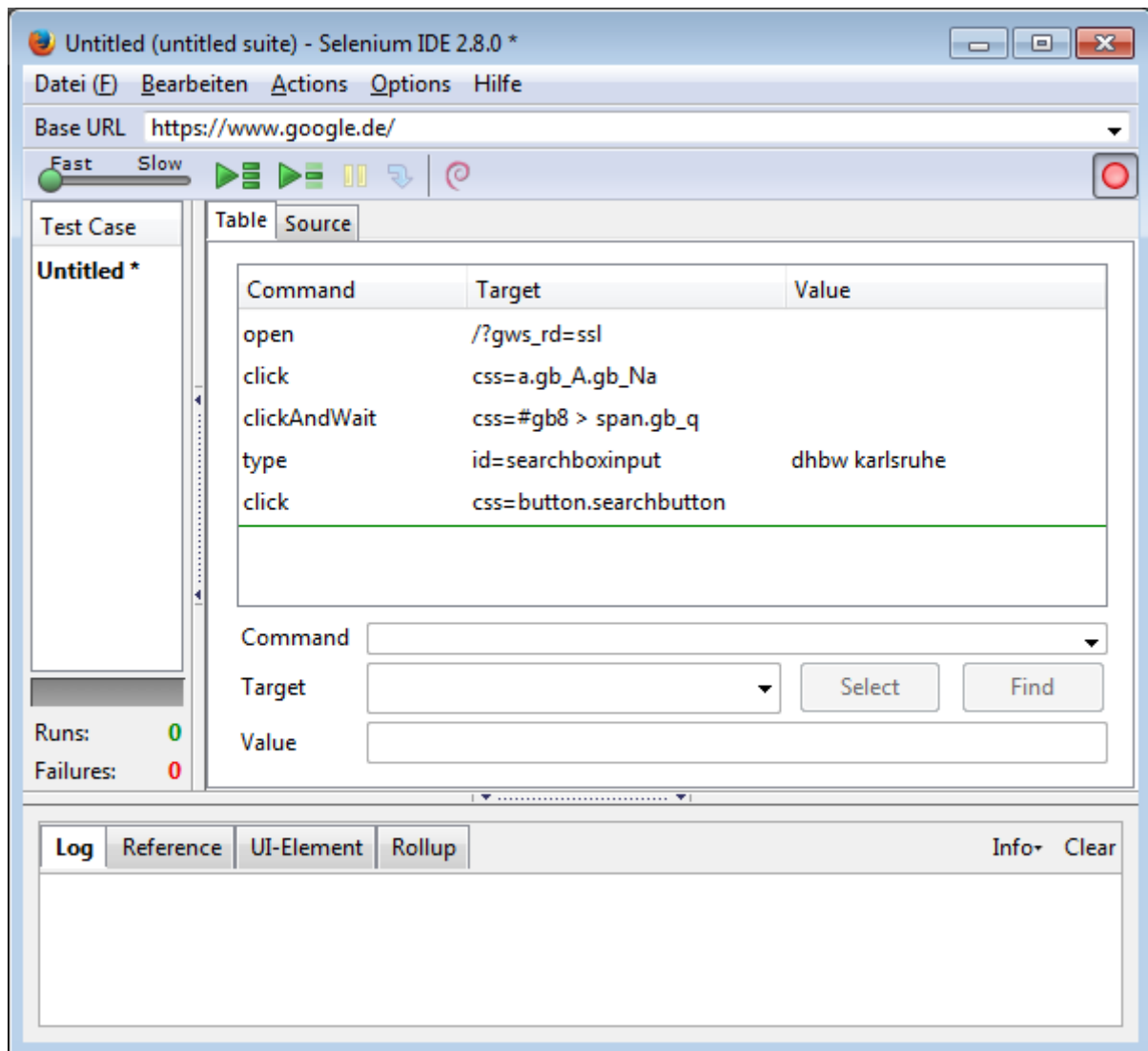
Routenplaner



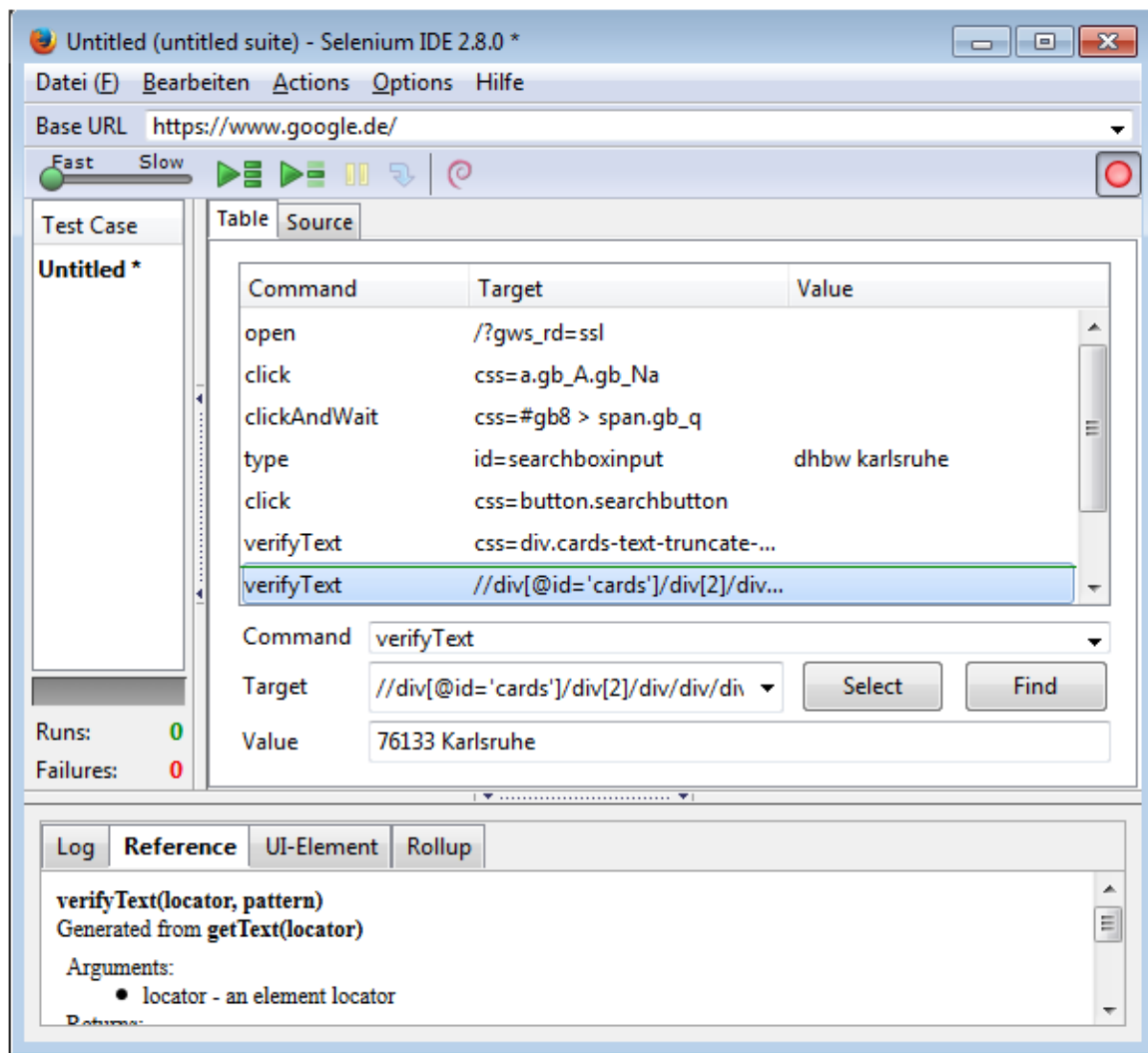
Speichern

dhbw-karlsruhe.de

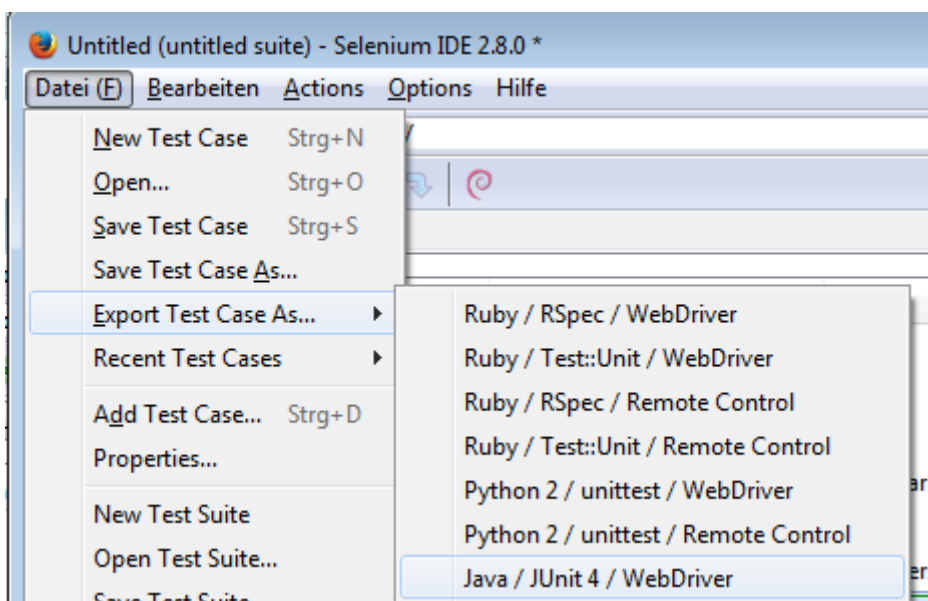
0721 97355



Um nun die Adresse überprüfen zu lassen, müssen wir ein wenig nachhelfen: Wir klicken in das freie Feld unter dem grünen Strich und wählen unter „Command“ „verifyText“ aus. Danach klicken wir neben „Target“ auf „Select“ und klicken dann in unserem Suchergebnis auf die Adresse. Dasselbe wiederholen wir nochmal mit der Stadt.



Danach können wir den Aufnahmemodus abschließen und unseren Code exportieren:



Danach können wir die Selenium IDE und den Firefox schließen. Die eben erstellte Datei können wir nun in Eclipse unter File > Open File öffnen:

```
SeleniumTest.java  mapstest.feature  Cukrunner.java  StepDefs.java  mapstest.java
1 package com.example.tests;
2
3 import java.util.regex.Pattern;
11
12 public class Mapstest {
13     private WebDriver driver;
14     private String baseUrl;
15     private boolean acceptNextAlert = true;
16     private StringBuffer verificationErrors = new StringBuffer();
17
18     @Before
19     public void setUp() throws Exception {
20         driver = new FirefoxDriver();
21         baseUrl = "https://www.google.de/";
22         driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
23     }
24
25     @Test
26     public void testMaps() throws Exception {
27         driver.get(baseUrl + "?gws_rd=ssl");
28         driver.findElement(By.cssSelector("a.gb_A.gb_Na")).click();
29         driver.findElement(By.cssSelector("#gb8 > span.gb_q")).click();
30         driver.findElement(By.id("searchboxinput")).click();
31         driver.findElement(By.id("searchboxinput")).clear();
32         driver.findElement(By.id("searchboxinput")).sendKeys("dhbw karlsruhe");
33         driver.findElement(By.cssSelector("button.searchbutton")).click();
34         try {
35             assertEquals("", driver.findElement(By.cssSelector("div.cards-text-truncate-and-wrap > span")).getText());
36         } catch (Error e) {
37             verificationErrors.append(e.toString());
38         }
39     }
40 }
```

Alles, was wir nun brauchen, befindet sich in der Methode „testMaps“. Nun können wir die StepDefs.java erweitern, dazu greifen wir auf Methoden zurück, die wir in SeleniumTest.java implementieren.

Die fehlenden Methoden der StepDefs.java sehen nun also folgendermaßen aus:

```
@Given("^switch over to Maps$")
public void switch_over_to_Maps() throws Throwable {
    script.switchToMaps();
}

@When("^I search for DHBW Karlsruhe$")
public void i_search_for_DHBW_Karlsruhe() throws Throwable {
    script.searchDH();
}

@Then("^the adress is Erzbergerstrasse (\\d+)$")
public void the_adress_is_Erzbergerstrasse(int arg1) throws Throwable {
    script.checkAdress(arg1);
}

@Then("^the city is (\\d+) Karlsruhe$")
public void the_city_is_Karlsruhe(int arg1) throws Throwable {
    script.checkCity(arg1);
}

@After

public void tearDown() throws Exception {
    script.tearDown();
}
}
```

Anmerkung: Der Tag @After legt fest, was nach dem Test passiert; in diesem Fall wird also das Browserfenster geschlossen.

Den generierten Code verarbeiten wir jetzt in der SeleniumTest.java; der grau hinterlegte Code muss noch hinzugefügt werden.:

```
public void switchToMaps() {
    driver.findElement(By.cssSelector("a.gb_A.gb_Na")).click();
    driver.findElement(By.cssSelector("#gb8 > span.gb_q")).click();
}

public void searchDH() {
    driver.findElement(By.id("searchboxinput")).click();
    driver.findElement(By.id("searchboxinput")).clear();
    driver.findElement(By.id("searchboxinput")).sendKeys("dhbw karlsruhe");
    driver.findElement(By.cssSelector("button.searchbutton")).click();
}

public void checkAdress(int hnr) {
    assertEquals("Erzbergerstraße "+hnr,
driver.findElement(By.cssSelector("div.cards-text-truncate-and-wrap >
span")).getText());
}

public void checkCity(int plz) {
    assertEquals(plz+" Karlsruhe",
driver.findElement(By.xpath("//div[@id='cards']/div[2]/div/div/div/div/div/div/div/div[2]/span")).getText());
}
```

Nun lassen wir unseren Test nochmals laufen: Der Firefox öffnet sich nun von selbst und führt alle definierten Schritte durch. Auch die Konsole meldet Erfolge:

```
Feature: Testing Maps
Automated test run. We're running on firefox 32.0.3

Scenario: Searching for DHBW Karlsruhe # Z:/studium/dh/software
engineering/Mapstest/src/featureFiles/mapstest.feature:5
  Given I am on Google # StepDefs.i_am_on_Google()
  And switch over to Maps # StepDefs.switch_over_to_Maps()
  When I search for DHBW Karlsruhe #
StepDefs.i_search_for_DHBW_Karlsruhe()
  Then the adress is Erzbergerstrasse 121 #
StepDefs.the_adress_is_Erzbergerstrasse(int)
  And the city is 76133 Karlsruhe # StepDefs.the_city_is_Karlsruhe(int)

1 Scenarios (1 passed)
5 Steps (5 passed)
0m33,443s
```

Um einen Fehlerfall zu simulieren, werden wir nun testen, ob das ZKM sich zufällig an derselben Adresse befindet wie die DHBW. Hierzu erweitern wir unsere Dateien folgendermaßen:

maptest.feature

```
Scenario: Searching for ZKM

Given I am on Google
And switch over to Maps

When I search for ZKM

Then the address is Erzbergerstrasse 121
And the city is 76133 Karlsruhe
```

StepDefs.java

```
@When("^I search for ZKM$")
public void i_search_for_ZKM() throws Throwable {
    script.searchZKM();
}
```

SeleniumTest.java

```
public void searchZKM() {
    driver.findElement(By.id("searchboxinput")).click();
    driver.findElement(By.id("searchboxinput")).clear();
    driver.findElement(By.id("searchboxinput")).sendKeys("zkm");
    driver.findElement(By.cssSelector("button.searchbutton")).click();
}
```

Lassen wir den Test ein letztes Mal durchlaufen; sehen wir, dass ein Szenario erfolgreich durchläuft, das andere jedoch nicht; unser Test arbeitet also korrekt:

```
Feature: Testing Maps
Automated test run. We're running on firefox 32.0.3

Scenario: Searching for DHBW Karlsruhe # Z:/studium/dh/software
engineering/Mapstest/src/featureFiles/mapstest.feature:5
  Given I am on Google # StepDefs.i_am_on_Google()
  And switch over to Maps # StepDefs.switch_over_to_Maps()
  When I search for DHBW Karlsruhe #
StepDefs.i_search_for_DHBW_Karlsruhe()
  Then the address is Erzbergerstrasse 121 #
StepDefs.the_adress_is_Erzbergerstrasse(int)
  And the city is 76133 Karlsruhe # StepDefs.the_city_is_Karlsruhe(int)
Automated test run. We're running on firefox 32.0.3

Scenario: Searching for ZKM # Z:/studium/dh/software
engineering/Mapstest/src/featureFiles/mapstest.feature:15
  Given I am on Google # StepDefs.i_am_on_Google()
  And switch over to Maps # StepDefs.switch_over_to_Maps()
  When I search for ZKM # StepDefs.i_search_for_ZKM()
  Then the address is Erzbergerstrasse 121 #
StepDefs.the_adress_is_Erzbergerstrasse(int)
  org.junit.ComparisonFailure: expected:<[Erzbergerstraße 121]> but
was:<[Lorenzstraße 19]>
    at org.junit.Assert.assertEquals(Assert.java:125)
    at org.junit.Assert.assertEquals(Assert.java:147)
```

```
at mapsTest.SeleniumTest.checkAdress(SeleniumTest.java:81)
at mapsTest.StepDefs.the_adress_is_Erzbergerstrasse(StepDefs.java:49)
at ?.Then the adress is Erzbergerstrasse 121(Z:/studium/dh/software
engineering/Mapstest/src/featureFiles/mapstest.feature:22)

And the city is 76133 Karlsruhe          # StepDefs.the_city_is_Karlsruhe(int)

2 Scenarios (1 failed, 1 passed)
10 Steps (1 failed, 1 skipped, 8 passed)
1m4,166s

org.junit.ComparisonFailure: expected:<[Erzbergerstraße 121]> but
was:<[Lorenzstraße 19]>
    at org.junit.Assert.assertEquals(Assert.java:125)
    at org.junit.Assert.assertEquals(Assert.java:147)
    at mapsTest.SeleniumTest.checkAdress(SeleniumTest.java:81)
    at mapsTest.StepDefs.the_adress_is_Erzbergerstrasse(StepDefs.java:49)
    at ?.Then the adress is Erzbergerstrasse 121(Z:/studium/dh/software
engineering/Mapstest/src/featureFiles/mapstest.feature:22)
```

Viel Spaß beim Nachprogrammieren!

Quelle: <http://hjrlive.wordpress.com/2014/02/15/test-post/>