

Web 2

Les 6b: Selenium

AGENDA

□ Automatisatie



OP HET EXAMEN...



- ☑ Je kunt Selenium gebruiken vanuit een gewone Java klasse of vanuit een JUnit testklasse. Wat is het voordeel van de JUnit testklasse?
- ☑ Waartoe dient Selenium?
- ☑ Wat is het grote voordeel van Selenium testen te maken?
- ☑ Je wil de werking van een formulier testen met Selenium. Wat ga je allemaal testen?
- ☑ ...



Formulieren steeds opnieuw testen wordt snel saai en tijdrovend!
Beter automatiseren...

SELENIUM

- Selenium WebDriver
- Tool voor testen van websites.
- Bootst echte gebruiker na

Selenium automates browsers. That's it!
What you do with that power is entirely
up to you.



Selenium heeft verschillende tools
Selenium WebDriver is de tool die we zullen
gebruiken

SELENIUM API

- API
 - Application Programming Interface
 - Library van klassen, ... die je kan gebruiken
 - <https://seleniumhq.github.io/selenium/docs/api/java/>
- Voorbeeld:
 - `WebDriver`: browser
 - `WebElement`: HTML element.

Selenium heeft een aantal klassen en interfaces die we kunnen gebruiken. Belangrijk voor ons zijn:

- De interface `WebDriver` stelt een browser voor
- De interface `WebElement` stelt een HTML element voor

SELENIUM DEPENDENCY

```
<dependency>  
<groupId>org.seleniumhq.selenium</groupId>  
<artifactId>selenium-java</artifactId>  
<version>3.141.59</version>  
</dependency>
```

```
<dependency>  
  <groupId>org.seleniumhq.selenium</groupId>  
  <artifactId>selenium-java</artifactId>  
  <version>3.141.59</version>  
</dependency>
```

1. DRIVER

- Maak driver object om browser te simuleren

```
WebDriver driver;
```

- Setup webdriver en initieer webdriver

```
WebDriverManager.chromedriver().setup();  
driver = new ChromeDriver();
```

```
WebDriverManager.chromedriver().setup();  
WebDriverManager.firefoxdriver().setup();  
WebDriverManager.operadriver().setup();  
WebDriverManager.phantomjs().setup();  
WebDriverManager.edgedriver().setup();  
WebDriverManager.iedriver().setup();
```

→ Er is een `webDriver`
voor iedere browser!

De “driver server” stelt Selenium in staat om met de browser te communiceren.

DRIVER DEPENDENCY

```
<dependency>  
  <groupId>io.github.bonigarcia</groupId>  
  <artifactId>webdrivermanager</artifactId>  
  <version>5.0.0</version>  
</dependency>
```

De “driver server” stelt Selenium in staat om met de browser te communiceren.

```
<dependency>  
  <groupId>io.github.bonigarcia</groupId>  
  <artifactId>webdrivermanager</artifactId>  
  <version>5.0.0</version>  
</dependency>
```

2. START URL

- `get ()` : navigeer naar URL

- Voorbeeld:

```
driver.get("http://en.wikipedia.org/wiki/Main_Page");
```

3. ELEMENTEN VINDEN

- `findElement()` : HTML element zoeken
- `By`: Zoeken op id, name, css selector, ...
- Voorbeelden:

```
WebElement searchField =  
    driver.findElement(By.id("searchinput"));  
  
WebElement welcomeParagraph =  
    driver.findElement(By.tagName("p"));  
  
WebElement errorDiv =  
    driver.findElement(By.cssSelector(".error"));
```

WATCH OUT NO Separation Of Concerns =>
change className or change the css class => tests need to be rewritten

LET OP NO Separation Of Concerns:
als je een css klasse later zou hernoemen, zal je
test niet meer werken!

4. USER INPUT

- `sendKeys()` : typen
`searchField.sendKeys("KHL");`
- `click()` : klikken
`goButton.click();`
- `clear(), ...`

OPDRACHT

- Schrijf een test die:
 - navigeert naar voegToe.jsp
 - op de knop klikt
- Het resultaat zou moeten zijn:
 - nog steeds op dezelfde pagina
 - foutmelding voor niet-ingevulde velden

- Vul een naam in.
- Vul een soort in.
- Vul een nummer in voor voedsel.

Voeg je huisdier toe

Naam:

Soort:

Aantal keer eten per dag:

Voeg dier toe

TEST NOK

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class TestNietAutomatisch {
    public static void main(String[] args) {
        WebDriver driver;
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();

        driver.get("http://localhost:8080/add.jsp");
        WebElement button = driver.findElement(By.id("submit"));
        button.click();

        driver.close();
    }
}
```

→ in commentaar zetten om browser open te laten

- Vul een naam in.
- Vul een soort in.
- Vul een nummer in voor voedsel.

Voeg je huisdier toe

Naam:

Soort:

Aantal keer eten per dag:

`driver.close();` → `//driver.close();`

OPDRACHT

- Schrijf een test die:
 - navigeert naar voegToe.jsp
 - alle velden correct invult
 - op de knop klikt
- Het resultaat zou moeten zijn:
 - overzichtspagina met nieuwe dier toegevoegd

Bekijk alle dieren

Naam	Soort	Voedsel
Lex	hond	2
Nijn	kat	5
Bruintje	kip	1
Mia	kat	7

```

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class TestNietAutomatischFormulierValidCaseSubmitten {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "/Applications/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("http://localhost:8080/add.jsp");

        WebElement naam = driver.findElement(By.id("naam"));
        naam.clear();
        naam.sendKeys("Mia");

        WebElement soort = driver.findElement(By.id("soort"));
        soort.clear();
        soort.sendKeys("kat");


        WebElement voedsel = driver.findElement(By.id("voedsel"));
        voedsel.clear();
        voedsel.sendKeys("7");

        WebElement button = driver.findElement(By.id("submit"));
        button.click();
    }
}

```

Bekijk alle dieren

Naam	Soort	Voedsel
Lex	hond	2
Nijn	kat	5
Bruintje	kip	1
Mia	kat	7



Nog steeds manuele controle !

Resultaat zou ook automatisch gecontroleerd moeten worden

Hoe pakken we dit best aan:

- if statement schrijven —> omslachtig
- JUnit testcase schrijven —> veel beter!

JUNIT

```
@Test
public void test_Form_is_shown_again_if_all_fields_are_empty() {
    WebDriver driver;
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();
    driver.get("http://localhost:8080/add.jsp");

    driver.findElement(By.id("submit")).click();

    assertEquals("Voeg een huisdier toe", driver.getTitle());

    driver.close();
}
```

titel van de pagina
vragen aan driver

• Vul een naam in.
• Vul een soort in.
• Vul een nummer in voor voedsel.

Voeg je huisdier toe

Naam:

Soort:

Aantal keer eten per dag:

Voeg dier toe

Voordeel:

- assert-methodes om resultaat te controleren
- automatische controle
- testrapport

WAT ALLEMAAL TESTEN?



RISICO'S

```
public class TestAddDier {  
  
    @Test  
    public void test_Result_is_shown_if_all_fields_are_filled()  
    { ... }  
  
    @Test  
    public void test_Result_is_shown_if_name_is_empty()  
    { ... }  
  
    @Test  
    public void test_Same_page_is_shown_if_soort_is_empty()  
    { ... }  
  
    @Test  
    public void test_Same_page_is_shown_if_voedsel_is_empty()  
    { ... }  
  
}
```

Voordeel: alle testen ineens uitvoeren

ANDERE?

- hoeveelheid voedsel
 - minimum
 - maximum
- eenzelfde dier mag maar één keer toegevoegd worden
- ...

Controleren of aantal hoeveelheid voedsel niet te hoog of te laag is, ...

Verder uitwerken wanneer we server-side validatie gezien hebben

VOORBEELD

```
public class TestExample {  
  
    WebDriver driver;  
    String url = "http://localhost:8080/";  
  
    @Before  
    public void setUp() throws Exception {  
        WebDriverManager.chromedriver().setup();  
        driver = new ChromeDriver();  
    }  
  
    @After  
    public void clean() {  
        driver.quit();  
    }  
}
```

Don't repeat yourself... gemeenschappelijke code isoleren.

@Before: wordt altijd herhaald **VOOR** de uitvoering van **iedere** testmethode

@After: wordt altijd herhaald **NA** de uitvoering van **iedere** testmethode

```
@Test
public void test_Form_is_shown_again_with_error_messages_if_name_field_is_left_empty() {
    WebElement naamInput = driver.findElement(By.id("naam"));
    naamInput.clear();
    naamInput.sendKeys("");

    WebElement soortInput = driver.findElement(By.id("soort"));
    soortInput.clear();
    soortInput.sendKeys("hond");

    WebElement voedselInput = driver.findElement(By.id("voedsel"));
    voedselInput.clear();
    voedselInput.sendKeys("5");

    driver.findElement(By.id("submit")).click();

    assertEquals("Voeg een huisdier toe", driver.getTitle());
    ArrayList<WebElement> lis =
        (ArrayList<WebElement>) driver.findElements(By.tagName("li"));
    assertTrue(containsWebElementsWithText(lis, "Vul een naam in."));
}

private boolean containsWebElementsWithText(ArrayList<WebElement> elements, String text) {
    for (int i = 0; i < elements.size(); i++) {
        if (elements.get(i).getText().equals(text)) {
            return true;
        }
    }
    return false;
}
```

- zoek invulveld (driver.findElement())
- maak invulveld leeg (input.clear())
- vul invulveld in (input.sendKeys("azerty"))
- zoek knop (driver.findElement())
- klik op de knop (button.click())
- zoek lijst met alle html elementen li (want foutboodschappen staan in ul)
- controleer of gezochte foutboodschap voorkomt in lijst (methode containsWebElementsWithText())


```
@Test
public void test_Overview_is_shown_If_all_fields_are_filled_out_correctly() {
    WebElement naamInput = driver.findElement(By.id("naam"));
    naamInput.clear();
    naamInput.sendKeys("Max");

    WebElement soortInput = driver.findElement(By.id("soort"));
    soortInput.clear();
    soortInput.sendKeys("hond");

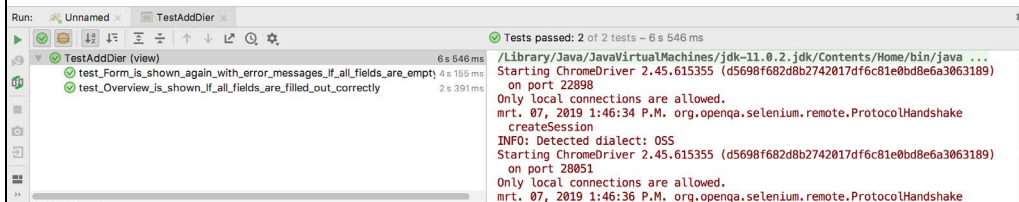
    WebElement voedselInput = driver.findElement(By.id("voedsel"));
    voedselInput.clear();
    voedselInput.sendKeys("9");

    driver.findElement(By.id("submit")).click();

    assertEquals("Bekijk alle dieren", driver.getTitle());

    ArrayList<WebElement> tds =
        (ArrayList<WebElement>) driver.findElements(By.tagName("td"));
    assertTrue(containsWebElementsWithText(tds, "Max"));
}
```


RESULTAAT



OPMERKING

```
...  
<div id="p-search" role="search">  
  <h3>  
    <label for="searchInput">Search</label>  
  </h3>  
  
  <form action="/w/index.php" id="searchform">  
    <div id="simpleSearch">  
      <input type="search" name="search" placeholder="Search"  
        title="Search Wikipedia [f]" accesskey="f" id="searchInput" />  
      <input type="hidden" value="Special:Search" name="title" />  
      <input type="submit" name="fulltext" value="Search"  
        title="Search Wikipedia for this text" id="mw-searchButton"  
        class="searchButton mw-fallbackSearchButton" />  
      <input type="submit" name="go" value="Go"  
        title="Go to a page with this exact name if one exists"  
        id="searchButton" class="searchButton" />  
    </div>  
  </form>  
</div>  
...
```

↓
maak elementen vindbaar met een id!

WAT CONTROLEREN?

- Controleer **alle** aspecten van de user story
 - pagina die de browser toont (<title>, hoofding)
 - foutboodschap
 - vooraf ingevulde waarden
 - tekst
 - ...

POM.XML

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.2</version>
</dependency>
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>3.141.59</version>
</dependency>
<dependency>
  <groupId>io.github.bonigarcia</groupId>
  <artifactId>webdrivermanager</artifactId>
  <version>5.0.0</version>
</dependency>
```

AGENDA

- ✓ Herhaling
- ✓ Automatisatie