

Werken met



Selenium Webdriver

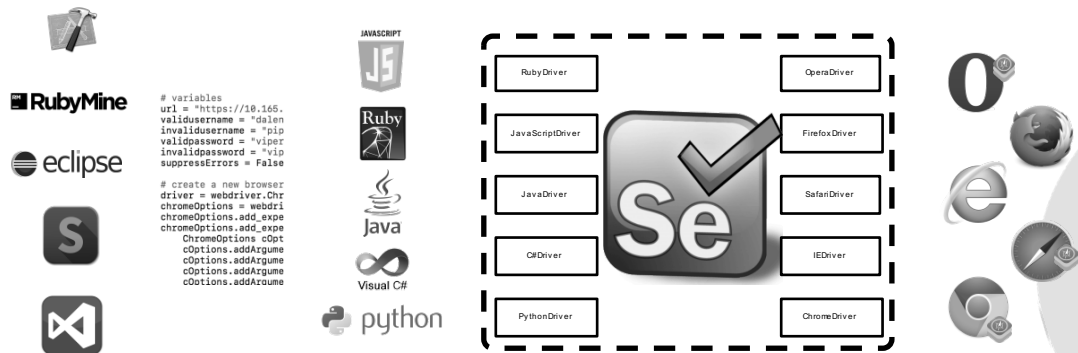
 Testconsultancy Groep

Werken met Selenium Webdriver

- Onderdelen van de 'stack'
- Installatie van de Selenium Webdriver stack
- Python programmeertaal
- Van start met Selenium Webdriver

 Testconsultancy Groep

Onderdelen van de 'stack'



Editor / IDE

- Maken en wijzigen van de scriptbestanden waarin de *testcases* en *testfixtures* worden opgenomen
- Syntaxkleuring en syntaxondersteuning (autocomplete)
- Opstarten van de testen
- Hulp bij het vinden van fouten
- Versiebeheer
- Vele opties: Eclipse, Visual Studio Code, Ruby Mine, Sublime, Xcode

Programmeertaal

- Persoonlijke smaak, eigen ervaring, teamvoorkeur, aansluiting bij gebruikte programmeertaal in project, eisen vanuit andere tools (Cucumber: Ruby)
- Of Python, met ingebouwde unittest library
- Editor moet de taal goed ondersteunen, bij voorkeur met *stepping*, *breakpoints*, *variabelen inspectie* en *version controle*
- Iedere programmeertaal heeft een eigen Selenium driver

Selenium Webdriver

- Download voor Linux, Windows of Mac OS X
- Houd een werkende directorystructuur aan

Browsers

- Voor iedere browser is een Selenium webdriver nodig
- Safari heeft de driver ingebouwd
- Google Chrome heeft een solide ondersteuning
- De interne werking van de browsers verschilt (Webkit, Gecko, ..)
- Eenvoudig switchen tussen browsers is mogelijk

Toegang tot websites

- Selenium Webdriver is bedoeld en geschikt voor websites
- Matig tot geen ondersteuning voor het werken met (HTTP) API's
- Testomgeving..
- Internet (om te Googelen)..

Installatie van de Selenium Webdriver stack



Onze keuzes..

- Windows platform
- Sublime editor
- Python programmeertaal
- Python driver
- Selenium Webdriver voor Windows
- Google Chrome browser
- Chromedriver



Sublime editor installeren

- Volg de instructies op <https://www.sublimetext.com/3>

Alternatieven

- Visual Studio Code van <https://code.visualstudio.com/download>

Selenium Webdriver installeren

- Installeer Python 3.6 met behulp van de installatie op <https://www.python.org/downloads/windows>
- Start cmd.exe en start de installatie van Selenium met `pip install selenium` (C:\Python27\Scripts\pip.exe)
- Installeer de ChromeDriver 2.3 <https://sites.google.com/a/chromium.org/chromedriver/downloads>
Noteer het installatiepad van de driver
- Installeer Google Chrome, indien nodig



Controle Python development stack

Test de installatie

- Type de tekst `print ("Hello, world!")` in Sublime, WebStorm of Visual Studio Code
- Bewaar dit als `hello.py`
- Druk op Ctrl-B
- Als alles goed werkt verschijnt de tekst Hello, world! op het scherm

<https://git.io/vxEU8>

Python programmeertaal



Print

```
print ("Hello, World!")
ndays = 365
number_of_days = 365
number_ofDays = 365
print ("Hello, World! There are" + str(ndays) + "days ahead")
```

Probeer het zelf!

Functie-aanroep.
Spaties zijn hier
optioneel

Tekst (strings)

```
print('perfect')      # Tussen enkele aanhalingstekens
print("showcase")    # Tussen dubbele aanhalingstekens
print('''Hello,
World!''')           # Over meerdere regels
print('One, ' + ' two, three') # Strings concateneren
print('Echo.. '*4)    # Herhaling van strings
print(len('Hello'))   # Lengte van de string
print(int('365'))     # Vertaal naar een getal
print(str(365))       # Vertaal naar een string
print ("123" + "456")
```

'len' is een
functie

Wat komt
hier uit?

Variabelen

```
aantalPogingen = 0
print (aantalPogingen)
aantalPogingen = aantalPogingen + 1
gemiddelde = 1422 / aantalPogingen
tekst = "Het antwoord is " + str(gemiddelde)
print (tekst)
```

Wat gebeurt hier?

Kies om een blok code uit te voeren (of niet)

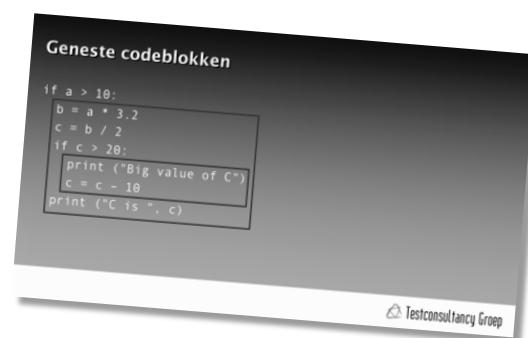
```
x = 3
if x == 3:
    print ("x is 3")
```

Twee keer =
teken

Dubbele punt

Inspringen is
verplicht!!

Code 'blok': regels met dezelfde
inspring (plus met een hogere
inspring)



Probeer dit uit. Verander de eerste 3 in andere waarden. Stel vast dat het blok code onder de if dan niet wordt uitgevoerd

If .. else

```
mark = 80
if mark > 50:
    print ("Pass")
else:
    print ("Fail")
```

Let op de dubbele
punten en op de
inspring

Elif

```
quality = 3
if quality == 1:
    print ("First choice")
elif quality == 2:
    print ("Second choice")
elif quality == 3:
    print ("Third choice")
else:
    print ("Pig's food")
```

Er kunnen een
onbeperkt aantal
elifs voorkomen

Herhaal een blok 10 keer

```
for i in range(10):  
    print(i)
```

```
for i in range(5, 15):  
    print(i)
```

```
for i in range(15, 5, -1):  
    print(i)
```

Probeer deze
constructies uit!

Tekst manipulatie (1)

```
msg = 'hello'  
if msg == 'hello':  
    print('howdy')  
  
if 'ell' in msg:  
    print('has ell')  
  
if 'lo' not in msg:  
    print('no lo in message found')
```

Tekst manipulatie (2)

```
msg2 = msg.replace('e','a')  
msg3 = msg.replace('o','a').replace('ll','l').replace('e','o')  
print(msg, msg2, msg3)
```

Funcities

```
def multiply(a, b):  
    c = a * b  
    return c  
  
result = multiply(3, 6)  
print result
```

Van start met Selenium Webdriver



Simple Usage

- Selenium Python Bindings ('SPB') hoofdstuk 2, pagina 7
- Kopieer de tekst uit de PDF van SPB in Sublime
- Let erop dat de tekst precies overkomt in Sublime ☺
- Vervang `webdriver.Firefox()` door `webdriver.Chrome(installatiepad van de driver)`
- Bewaar het bestand als `python_org_search.py`
- Start het op met Ctrl-B
- Na 5 seconden is de test afgerond



```
1 from selenium import webdriver
2 from selenium.webdriver.common.keys import Keys
3 driver = webdriver.Chrome('chromedriver path')
4 driver.get("http://www.python.org")
5 assert "Python" in driver.title
6 elem = driver.find_element_by_name("q")
7 elem.clear()
8 elem.send_keys("pycon")
9 elem.send_keys(Keys.RETURN)
10 assert "No results found." not in driver.page_source
11 driver.close()
```

<https://git.io/vxEU8>

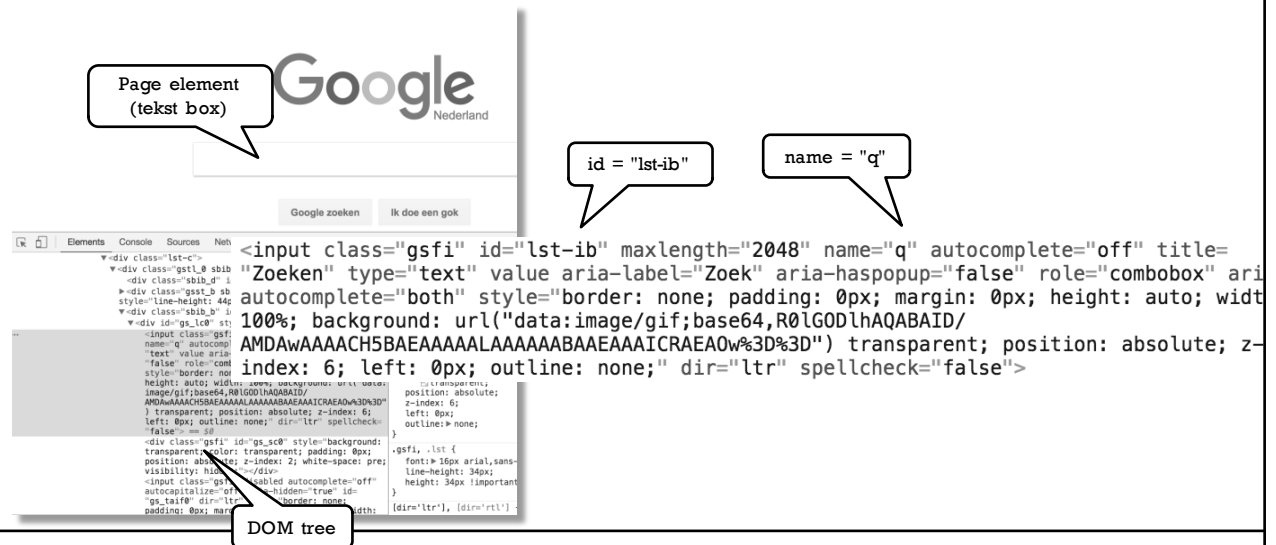
Problemen?

- Controleer of je de tekst uit SPB precies zo hebt overgenomen
- Let op `driver = webdriver.Chrome('c:\ ... ')`
- Controleer of je de chromedriver, python en selenium goed hebt geïnstalleerd

Stap voor stap door het voorbeeld

- Example Explained, SPB pagina 7

Interactie met de pagina



The image shows a screenshot of the Google search page with several annotations:

- A callout box labeled "Page element (tekst box)" points to the search input field.
- A callout box labeled "id = 'lst-ib'" points to the search input field.
- A callout box labeled "name = 'q'" points to the search input field.
- A callout box labeled "DOM tree" points to the DOM tree view in the browser's developer tools.

The DOM tree view shows the following HTML structure:

```
<input class="gsfi" id="lst-ib" maxlength="2048" name="q" autocomplete="off" title="Zoeken" type="text" value aria-label="Zoek" aria-haspopup="false" role="combobox" aria-autocomplete="both" style="border: none; padding: 0px; margin: 0px; height: auto; width: 100%; background: url('data:image/gif;base64,R0lGODlhAQABAID/AMDAwAAACHSBAEAAAAALAAAAABAAEAAAIICRAEA0w%3D%3D') transparent; position: absolute; z-index: 6; left: 0px; outline: none; dir="ltr" spellcheck="false">
```

Pagina elementen

```
element = driver.find_element_by_id('lst-ib')
element = driver.find_element_by_name('q')
element = driver.find_element_by_xpath('//*[ @id="lst-ib"]')
```

```
element.clear()
element.send_keys("Dit wil ik weten")
element.submit()
```

Reguliere expressie die naar een element in de DOM refereert

Dit werkt natuurlijk alleen als de elementen ook echt op de pagina aanwezig zijn..

Asynchrone verwerking is *a pain*...

- Het laden van de pagina gaat minder snel dan Python instructies kan uitvoeren. Het element met de naam 'q' is er nog een hele tijd niet, nadat `driver.get("http://www.google.nl")` is uitgevoerd
- Kortgezegd: je programma is al afgelopen voordat de pagina in het geheel is geladen
- Een mogelijkheid is, om na iedere instructie waarvan je verwacht dat deze even tijd in beslag neemt, een expliciete pauze in te voegen met `import time` en `time.sleep(5)`
- Het gaat dan allemaal wel langer duren...

Opdracht Herhaal pagina opvragen

Zoek een webpagina en maak een nieuw script dat deze pagina herhaald opvraagt, controleert dat de pagina er (goed) is, er iets invult, ergens op klikt (`element.click()`), de pagina dan afsluit en herhaalt. Houd in een variabele bij hoe vaak de pagina is opgevraagd en print de waarde van deze variabele.



Interactie met de pagina

We hebben een aantal basis interacties met de DOM tot onze beschikking:

- Pagina-elementen zoeken
- Wachten tot een pagina-element beschikbaar of zichtbaar wordt
- Op een element klikken
- Een element van waarde veranderen
- En *if all else fails*, de `time.sleep(seconden)`



Pagina-elementen zoeken (SPB: locating elements)

- `find_element_by_name("the_name")`
- `find_element_by_id("the_id")`
- `find_element_by_xpath("the_xpath")`
- `find_element_by_link_text("the_link_text")`

Maar ook:

- `list_of_elements = find_elements_by_xpath("the_xpath")`
- Wat is een 'xpath'?

XPATH

- XML element selectie criterium
- Bedoeld om 1 of meerdere elementen in de DOM te selecteren
- Mogelijk om pagina-onafhankelijk te specificeren
- Brwosers bieden hulp, door een XPATH voor een pagina-element te definiëren
- Zie de interactieve tutorial op http://www.zvon.org/comp/r/tut-XPath_1.html#intro

Wachten tot een pagina-element beschikbaar of zichtbaar wordt ('waits', SPB)

- **Implicit waits** driver probeert een instelbare tijd een element te vinden, totdat hij 'm vindt, of het na de ingestelde tijd opgeeft (handig!)
- Stel 10 seconden in: `driver.implicitly_wait(10)`
- **Explicit waits** voorzieningen zoals dit voorbeeld:

```
result = True
try:
    element = EC.visibility_of_element_located((By.XPATH, '//*[@id="q"]'))
    WebDriverWait(self.driver, 60).until(element)
except TimeoutException:
    result = False
```

Op een element klikken

- `driver.find_element_by_id("submit").click()`

Exceptions

try:

`a = 1 / 0`

`print("It couldn't have worked!")`

except:

`print("Sure it didn't.")`

Delen door 0 levert
een exceptie op

File handling

- Python kan heel snel met bestanden werken. Je kunt regels inlezen en ook elementen op een regel splitsen
- Data in bestanden kan worden gebruikt om testen aan te sturen:
 - URL's die je wilt opvragen
 - Paginatitels die bij de URL's horen
 - Het resultaat van de opvraging
- In de repository vind je voorbeeld [fileHandling.py](#)

ForEach

```
for option in list_of_options:  
    print(option)
```

Met pagina-elementen:

```
select = driver.find_element_by_xpath("//select")  
all_options = select.find_elements_by_tag_name("option")  
for option in all_options:  
    print "Value is: %s" % option.getValue  
    option.setSelected()
```

Opdracht Interactie met een webpagina

Verbeter je script van de vorige opdracht door gebruik te maken van data die je in een tekstbestand hebt opgeslagen. Maak een tekstbestand aan met verschillende data en lees steeds een regel van dit bestand uit. Voer de data in de webpagina in.

p.s. Probeer de website niet te spammen 😊

Python unittest framework

- De Python unittest framework is bedoeld en geschikt voor testen die met Selenium worden opgebouwd
- Test Cases worden *gesubclassed* van `unittest.TestCase`
- Individuele testen binnen de Test Case die beginnen met de letters 'test' worden automatisch (in alfabetische volgorde) uitgevoerd
- Binnen de Test Case kan een *setup* en *teardown* worden gedefinieerd
- Alle fouten worden afgevangen en na het draaien getoond
- In de repository vind je het voorbeeld [tryUnitTest.py](#)

Asserts (SPB, 1)

- Asserts zijn de uitvoercontroles van de test. Er kunnen meerdere (vele) controles binnen één test worden uitgevoerd
- Asserts die falen breken de test, maar met de Python `unittest.TestCase` pas nadat alle andere testen doorlopen zijn

```
assertTrue(x, 'Melding als x niet gelijk is aan true')
```

Asserts (SPB, 2)

Assert	Controleert dat
<code>assertEqual(a, b, msg)</code>	<code>a == b</code> , <i>msg</i> is shown otherwise
<code>assertNotEqual(a, b, msg)</code>	<code>a != b</code> , <i>msg</i> is shown otherwise
<code>assertTrue(x, msg)</code>	<code>x</code> is True, <i>msg</i> is shown otherwise
<code>assertFalse(x, msg)</code>	<code>x</code> is False, <i>msg</i> is shown otherwise
<code>assertIsNone(x, msg)</code>	<code>x</code> is None, <i>msg</i> is shown otherwise
<code>assertIn(a,b, msg)</code>	<code>a</code> in <code>b</code> , <i>msg</i> is shown otherwise
<code>assertIs(a,b, msg)</code>	<code>a</code> is <code>b</code> , <i>msg</i> is shown otherwise

Opdracht Gebruik het unittest framework

Download het Python programma `testFrameWork.py` van de repository. Draai het en analyseer de werking. Merk op dat het aantal resultaten van een zoekactie van de pagina wordt opgepikt. Voeg verschillende asserts toe om de tests op de pagina te verbeteren.

Benodigde testen voor een website

- Welke pagina's zijn er
- Wordt de pagina tijdig geladen
- Is de pagina (en de inhoud van de pagina) volledig geladen. Wat moet je daarvoor controleren?
- Hoe zorg je ervoor dat de test nog steeds loopt als de pagina wijzigt?
- Hoe deel je je testen in? Alles in 1 bestand? Wat wordt data en wat wordt code?
- Hoeveel tijd gaat er zitten in het uitschrijven van de testen?

Opdracht Test de website Retro-Lab.nl

Bekijk de website www.retro-lab.nl. Maak een lijst van te testen pagina's: welke informatie is er? Bepaal op welke manier je kunt vaststellen dat pagina's correct zijn getoond. Maak verschillende klassen. Scheid de data van de testen van de code, door de data in een los (tekst) bestand te zetten. Denk er over na hoe je dat wilt doen en implementeer dit dan