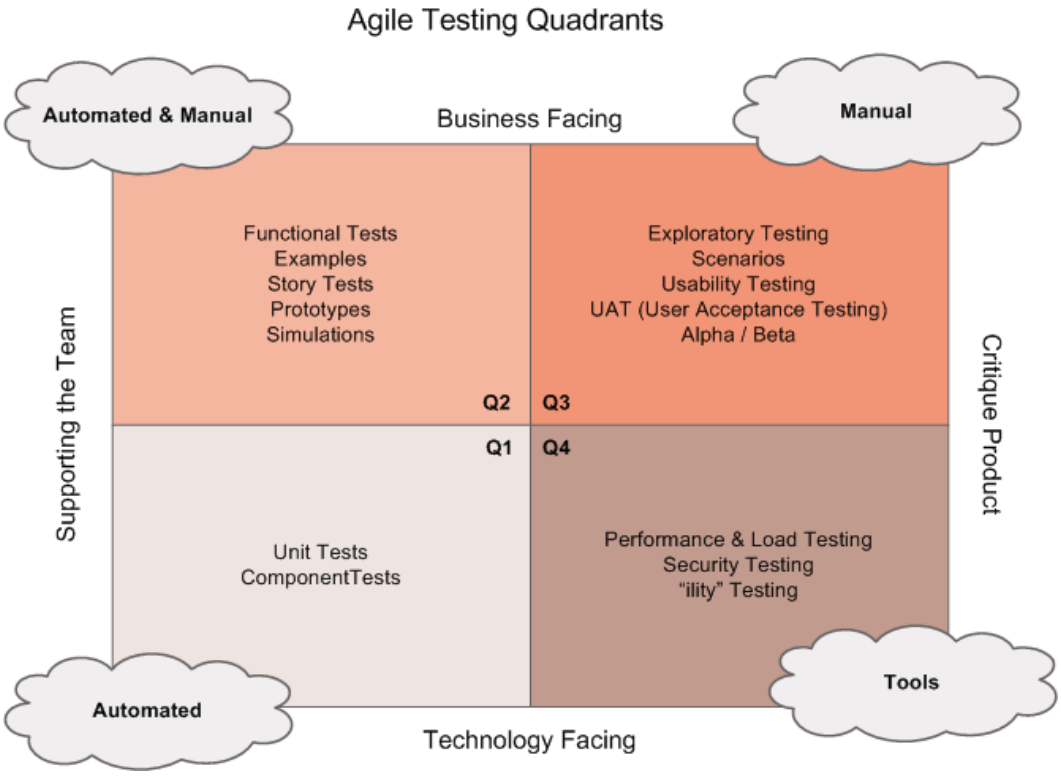
Test Automation

Contents

1 Waarom Test Automation?

- Sneller resultaat (meestal)
- Automate deel van het test plan (recurring parts)
  - Niet het hele test plan…
  - Maak een verstandige selectie
  - Regression testing
  - Smoke testing
- Bouw test suites geleidelijk op en gebruik wat je al hebt onmiddellijk
- Not everything can be automated!



- zie https://agiletester.ca/

Note:

- automated testing is een doorlopende investering
- automated tests moeten continu runnen (en niet 1 keer per release of 1 keer per week)
- automated tests vragen onderhoud (maintenance)
- 1 automated test is beter dan geen automated test…
- Test environment om automated tests te runnen
- Wat doe je met test data??
- Wat doe je als de test faalt?????

2 Tool

2.1 Wat gaat we gebruiken

Robot Framework

- Generic **test automation** framework
- Keyword/behaviour driven
- Open source
- Written in python
- .robot files

Selenium

- Library for **browser automation**
- Interacts with elements on a web page
- Robot Framework provides easy syntax
- Must be installed via pip install
- http://robotframework.org/SeleniumLibrary/SeleniumLibrary.html

2.2 Installeer

- Installeer Python https://www.python.org/downloads/
- Zet volgende folders in de PATH variable
    - Python\Python38-32\ (install path **waar python.exe staat**)
    - Python\Python38-32\Scripts\
    - Python\Python38-32\Lib
- Test dit: open een cmd (of git-bash) en typ "python"
- Open een **elevated** cmd box
- Installeer Robot Framework met het volgende commando in cmd:
    - pip install robotframework
- Test dit: pip list -> dan moet je robotframework daartussen zien staan
- Voor deze cursus hebben we nog bijkomende libraries nodig:
    - *pip install requests*
    - *pip install DateTime*
    - *pip install robotframework-requests*
    - *pip install robotframework-jsonlibrary*
    - *pip install robotframework-seleniumlibrary*
- Install Chromedriver (nodig voor selenium)
    - https://chromedriver.chromium.org/
    - Afhankelijk van de **versie** van chrome die je geinstalleerd hebt
    - Download, zet ergens (bvb c:/chromedriver)
    - Zet deze dir in je path
- Installeer een IDE naar keuze
    - Noot: dit lukt ook in IntelliJ met plugin Robot Framework Support
    - Visual Studio Code met de volgende extensions: Robot Framework Intellisense FORK en robot framework language
    - PyCharm (van Jetbrains), Sublime, Notepad++ en RIDE

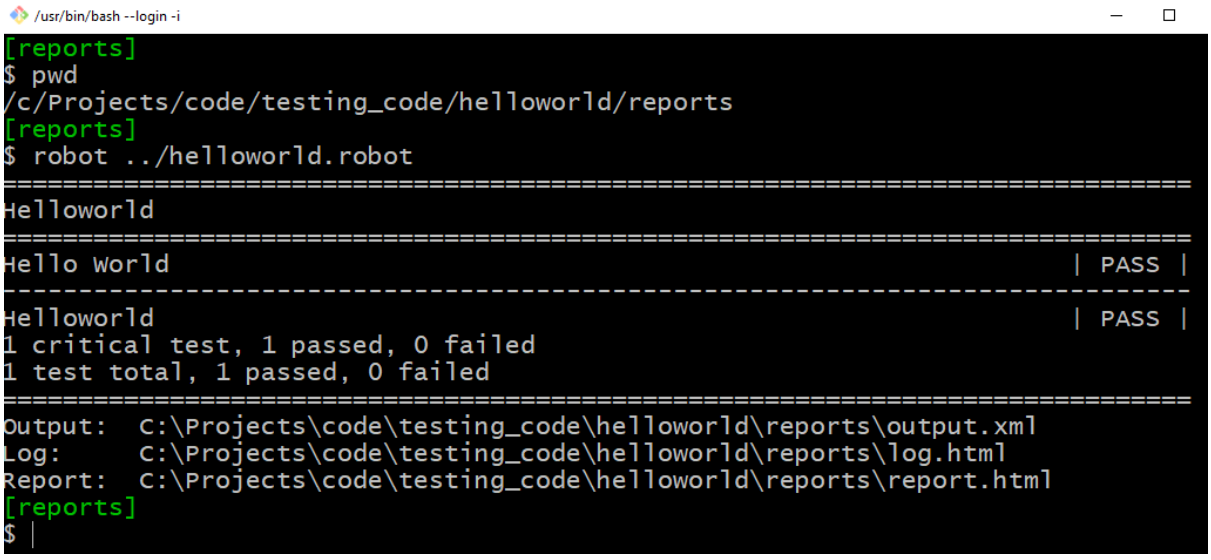
2.3 Uitproberen: Hello World

We voeren een simpele robot file uit (bvb in IntelliJ)

- Maak een folder “helloworld”
- Naak een file “helloworld.robot”
- Voeg deze 3 lijnen toe aan de file:

```
*** Test Cases ***
Hello World
    Log  Hello World!
```

- Noot: er staan 2 spaties tussen “Log” en “Hello World!”
- Execute:
  - Maak in folder helloworld een folder “reports”
  - Open **command line tool** in deze folder reports
  - Noot: je kan dit ook vanuit het Terminal tool-venster in IntelliJ
  - robot ..\helloworld.robot
- Resultaat: 3 files in . (reports):
  - **report.html**: samenvatting van het resultaat (groen of rood)
  - **log.html**: gedetailleerde log van de test execution
  - **output.xml**: dezelfde data die je ook ziet in report.html in xml formaat
- Noot: je kan ook een ganse directory met test files uitvoeren:
  - robot ..



2.4 Pass/Fail

- Je ziet
  - we hebben 1 Test Case uitgevoerd
  - het resultaat is PASS
  - het totale resultaat is dus ook PASS
  - in report.html en log.html zie je ook dat dit resultaat groen is
- Voeg nu een Test case toe:

```
*** Test Cases ***
Hello World
    Log  Hello World!

Failing Hello World
    Log  Hello World!
    Fail
```

- Het resultaat van de eerste Test Case is PASS, maar dat van de tweede Test Case is FAIL.
- Noot: Het algemene resultaat is dus ook FAIL
- Bekijk ook de files in de reports folder

```
/usr/bin/bash --login -i
[reports]
$ robot ../helloworld.robot
==============================================================================
Helloworld
==============================================================================
Hello World                                                           | PASS |
------------------------------------------------------------------------------
Failing Hello World                                                   | FAIL |
AssertionError
------------------------------------------------------------------------------
Helloworld                                                            | FAIL |
2 critical tests, 1 passed, 1 failed
2 tests total, 1 passed, 1 failed
==============================================================================
Output:  C:\Projects\code\testing_code\helloworld\reports\output.xml
Log:     C:\Projects\code\testing_code\helloworld\reports\log.html
Report:  C:\Projects\code\testing_code\helloworld\reports\report.html
[reports]
$
```

3 Robot framework

3.1 Voorbeelden

Zie github: https://github.com/SeppVC-Brightest/Robot-Framework-Training

3.2 Keyword / Behaviour Driven

- **Keywords:** hergebruiken van code
- **Let op: Dubbele spatie** (!) tussen keywords
- Layer on top of programming syntax
- We maken test cases vanuit **behaviour perspective**
- "Easier" to create and read – bedoeling was dat ook niet-developers deze testen kunnen **lezen**

```
*** Test Cases ***
Login correct username and password
    Login    tomsmith    SuperSecretPassword!
    ${message}=  Get message
    Should Contain  ${message}  You logged into a secure area!
    Logout
```

3.3 Robot File Structuur

In een robot file heb je verschillende sections:

*** Settings ***
This section contains references to libraries & resources available in the script
Suite Setup & Suite Teardown is also used here

*** Variables ***
Initialize any global variables to be used in the script

*** Keywords ***
Create custom keywords to be used in the script

*** Test Cases ***
Collection of the test cases to be executed when running the script

```
*** Settings ***
Resource  ./resources/resource.robot

*** Variables ***
${MESSAGE}=  Hello World!

*** Keywords ***
Multiply numbers
    [Documentation]  This keywork multiplies 2 numbers, logs and returns the result
    [Arguments]  ${number1}  ${number2}
    ${result}=  Evaluate  ${number1} * ${number2}
    Log  ${result}
    [Return]  ${result}

*** Test Cases ***
Test case 1
    [Tags]  Sanity
    ${result}=  Multiply numbers    ${4}    ${9}
    Log  ${result}
    # This line is a comment
    ${result}=  Add two strings  Robot  Framework
    Log  ${result}
```

3.4 Syntax

3.4.1 Variabelen

- **Let op: Dubbele spatie** (!) tussen keywords
- Variables not bound to one datatype
- 3 types:
    - Scalar: $
    - List: @
    - Dictionary: &

```
*** Variables ***
${GLOBALVARIABLE}=  Hello World!

*** Test Cases ***
Assigning Variables
    ${number}=  Set Variable  ${3}
    ${string}=  Set Variable  String
    ${boolean}=  Set Variable  ${TRUE}
    ${empty}=  Set Variable  ${EMPTY}

Using Lists
    @{list}=  Create List  1  2  3

Using Dictionaries
    &{dictionary}=  Create Dictionary  first=1  second=${2}  ${3}=third
```

3.4.2 Keywords

- Vergelijk met functies
- We maken de keywords in een robot file in section *** Keywords ***
- We gebruiken de Keywords in de test cases
- **Let op: Dubbele spatie** (!) tussen keywords

- Return value: 0, 1 of meer return
- Meerdere arguments – ook default arguments mogelijk
- Robot Framework BuiltIn Keywords:
  zie http://robotframework.org/robotframework/latest/libraries/BuiltIn.html
- Bvb: **Log**

```
*** Keywords ***
One Argument With Default Value
    [Arguments]  ${arg}=default value
    [Documentation]  This keyword takes 0-1 arguments
    Log  Got argument ${arg}

One Required And One With Default
    [Arguments]  ${required}  ${optional}=default
    [Documentation]  This keyword takes 1-2 arguments
    Log  Required: ${required}
    Log  Optional: ${optional}

Single Return Keyword
    [Documentation]  This keyword returns 1 value
    [Return]  Return value

Multiple Return Keyword
    [Documentation]  This keyword returns 2 values
    [Return]  value1  ${2}
```

3.4.3 Pass / Fail

- Als een script tot het einde komt dan is het resultaat PASS
- Keywords kunnen het resultaat van de Test Case veranderen: zie builtIn library
  - http://robotframework.org/robotframework/latest/libraries/BuiltIn.html
  - zoek naar Should
- Als een "**Should**" conditie faalt dan eindigt de uitvoering van deze Test Case
- De keywords "**Fail**" en "**Pass Execution**" stoppen ook de verdere uitvoering van de Test Case

Shortcuts

List style: Compact Expanded

Call Method · Catenate · Comment · Continue For Loop · Continue For Loop If · Convert To Binary · Convert To Boolean · Convert To Bytes · Convert To Hex · Convert To Integer · Convert To Number · Convert To Octal · Convert To String · Create Dictionary · Create List · Evaluate · Exit For Loop · Exit For Loop If · Fail · Fatal Error · Get Count · Get Length · Get Library Instance · Get Time · Get Variable Value · Get Variables · Import Library · Import Resource · Import Variables · Keyword Should Exist · Length Should Be · Log · Log Many · Log To Console · Log Variables · No Operation · Pass Execution · Pass Execution If · Regexp Escape · Reload Library · Remove Tags · Repeat Keyword · Replace Variables · Return From Keyword · Return From Keyword If · Run Keyword · Run Keyword And Continue On Failure · Run Keyword And Expect Error · Run Keyword And Ignore Error · Run Keyword And Return · Run Keyword And Return If · Run Keyword And Return Status · Run Keyword If · Run Keyword If All Critical Tests Passed · Run Keyword If All Tests Passed · Run Keyword If Any Critical Tests Failed · Run Keyword If Any Tests Failed · Run Keyword If Test Failed · Run Keyword If Test Passed · Run Keyword If Timeout Occurred · Run Keyword Unless · Run Keywords · Set Global Variable · Set Library Search Order · Set Local Variable · Set Log Level · Set Suite Documentation · Set Suite Metadata · Set Suite Variable · Set Tags · Set Task Variable · Set Test Documentation · Set Test Message · Set Test Variable · Set Variable · Set Variable If · Should Be Empty · Should Be Equal · Should Be Equal As Integers · Should Be Equal As Numbers · Should Be Equal As Strings · Should Be True · Should Contain · Should Contain Any · Should Contain X Times · Should End With · Should Match · Should Match Regexp · Should Not Be Empty · Should Not Be Equal · Should Not Be Equal As Integers · Should Not Be Equal As Numbers · Should Not Be Equal As Strings · Should Not Be True · Should Not Contain · Should Not Contain Any · Should Not End With · Should Not Match · Should Not Match Regexp · Should Not Start With · Should Start With · Sleep · Variable Should Exist · Variable Should Not Exist · Wait Until Keyword Succeeds

3.4.4 Logical operators

- Run Keyword If …
- Run Keyword Unless …
- Run Keyword If … else …
- Keywords scheiden met 2 spaties

```
*** Test Cases ***
If Condition Is True
    ${boolean}=  Set Variable  ${TRUE}
    Run Keyword If  ${boolean}  Log  This boolean is true  ELSE  Log  This boolean is false

If Condition Is False
    ${number}=  Set Variable  ${7}
    Run Keyword Unless  ${number}==7  Log  The number was 7

Multiple Conditions
    ${colour}=  Set Variable  red
    Run Keyword If  '${colour}'=='red' or '${colour}'=='purple'  Log  That's a nice colour

Setting Variables With If
    ${boolean}=  Set Variable  ${TRUE}
    ${value}=  Run Keyword If  ${boolean}  Set Variable  Variable Value 1  ELSE  Set Variable  Variable Value 2
    Log To Console  ${value}
```

3.4.5 Exercise 1: logical operators

Create a test script which does the following

1. Define two number variables
2. Log the largest number

3.4.6 Loops

- FOR .. IN RANGE ..
    - Loops over several values
    - Step size can also be given
- FOR .. IN ..
    - Loops over dictionaries and lists
- Nested for loops
    - Nested for loops are not supported directly
    - Can be achieved by putting a for loop into a custom keyword

```
Only upper limit
    [Documentation]  Loops over values from 0 to 9
    FOR  ${index}  IN RANGE  10
        Log  ${index}
    END
```

```
Looping Over Lists
    @{list}=  Create List  1  2  3
    FOR  ${entry}  IN  @{list}
        Log  ${entry}
    END

Looping Over Dictionaries
    &{dictionary}=  Create Dictionary  first=1  second=${2}  ${3}=third
    FOR  ${key}  ${value}  IN  &{dictionary}
        Log  ${key}
        Log  ${value}
    END
```

```
*** Keywords ***
Nested For
    [Documentation]  Keyword which contains a for loop used in a nested for loop
    [Arguments]  ${list}
    Log  Logging items in list:
    FOR  ${item}  IN  @{list}
        Log  ${item}
    END

*** Test Cases ***
Nested For Loop
    [Documentation]  For loops cannot be nested with Robot Framework,
    ...              instead a new keyword needs to be defined
    ${list1}=  Create list  String1-1  String1-2  String1-3  String1-4
    ${list2}=  Create list  String2-1  String2-2  String2-3  String2-4
    ${list3}=  Create list  String3-1  String3-2  String3-3  String3-4
    ${list4}=  Create list  ${list1}  ${list2}  ${list3}

    FOR  ${item}  IN  @{list4}
        Log  Logging list
        Log  ${item}
        Nested for  ${item}
    END
```

3.4.7 Exercise 2: loops

Create a test script which prints out all odd numbers from 1 to 99

3.4.8 Exercise 3: loops

Print out all numbers between 1 and 100 which are:

1. Divisible by 3
2. Divisible by 5
3. Divisible by both

3.5 Resource files

- Resource files zitten in folder resources
- Bevatten keywords
- Bevatten geen test cases (je krijgt een error)

3.6 Libraries

Robot Framework Libraries

- Imported automatically:
  - BuiltIn
  - http://robotframework.org/robotframework/latest/libraries/BuiltIn.html
- Standard libraries
  - Collections, DateTime, OperatingSystem…
  - http://robotframework.org/robotframework/
- External libraries
  - AppiumLibrary, SeleniumLibrary, SikuliLibrary…
  - Import in the robot file under *** Settings ***
  - after installing via pip install
- Custom libraries
  - = Robot Framework file without *** Test Cases *** header
  - Can be imported via *** Settings ***

Python Libraries

- Python class:
  - Import keyword module

```
from robot.api.deco import keyword
```

  - Bind Python functions via @Keyword

```
@keyword("Custom made keyword")
def custom_made_keyword(self):
```

- Robot file:
  - Add python class as library
  - Collections, DateTime, OperatingSystem…
  - http://robotframework.org/robotframework/

4 Selenium

4.1 What is Selenium

- Library for browser automation
- http://robotframework.org/SeleniumLibrary/SeleniumLibrary.html

4.2 Example: Open a webpage

```
*** Settings ***
Library  SeleniumLibrary

*** Test Cases ***
The First Selenium Test
    Open Browser  http://www.brightest.be  Google Chrome
    Sleep  5s
    Close Browser
```

4.3 Locate and Interact

Manier van werken is altijd:

- Locate html element
- Interact with that html element
- Verification (Should…, Pass, Fail)

4.3.1 Locate html element

Zie:
http://robotframework.org/SeleniumLibrary/SeleniumLibrary.html#Locating%20elements

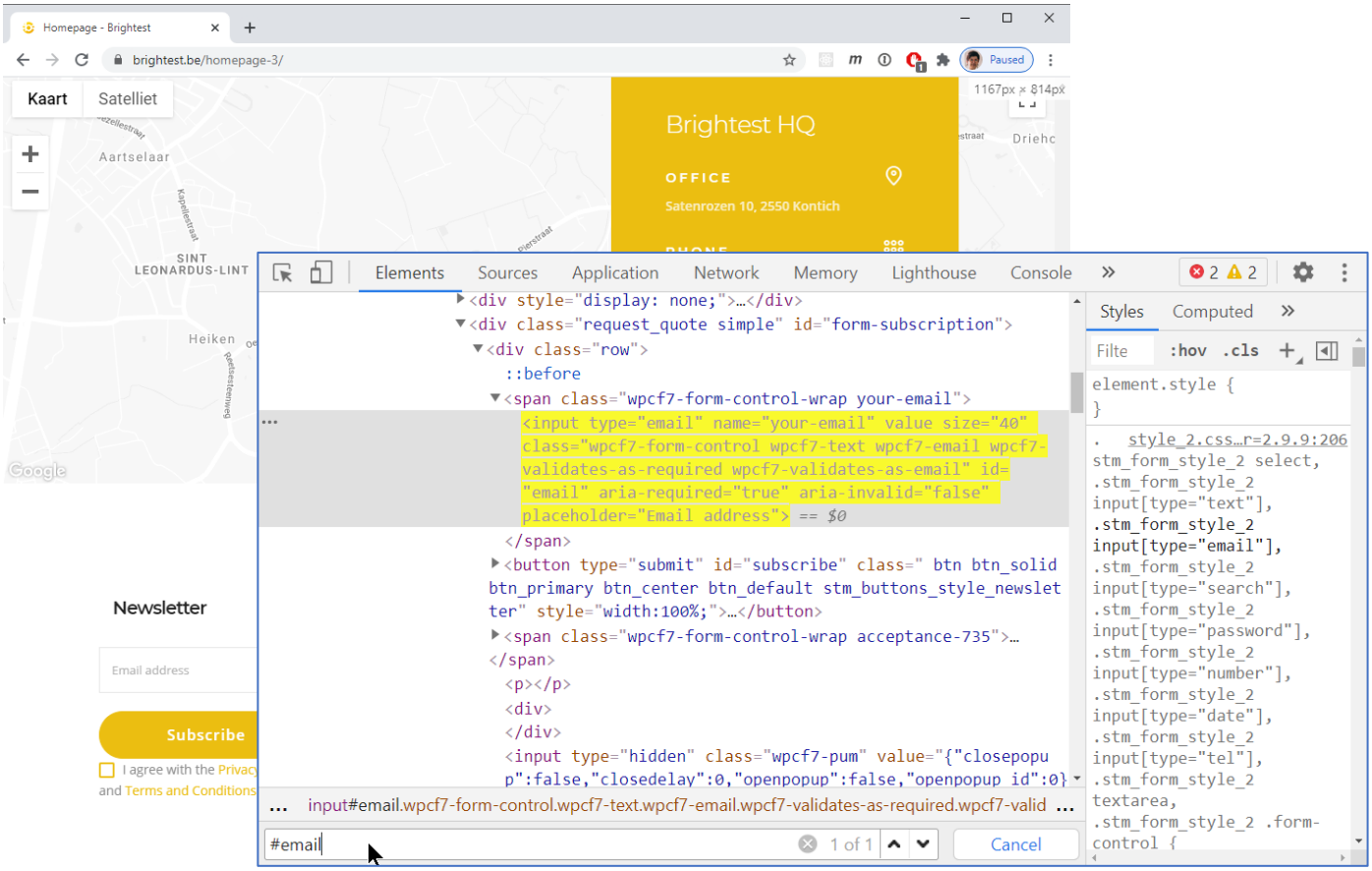
Locate html elements kan op verschillende manieren:

- **id**
  - het gemakkelijkste en **stabielste** maar alleen mogelijk als de developers willen meewerken…
  - #email
    - het (unieke) element met id email
- **css selector**
  - https://www.w3schools.com/cssref/css_selectors.asp
  - css:.header_user_info > a
    - zoek element met class “header_userinfo” en daarvan het kind van type a
  - css:[title='About us']
    - zoek element dat een attribute title heeft met waarde “About us”
- **class**
- **link**
- **xpath**
  - https://www.w3schools.com/xml/xpath_intro.asp
- …

Probleem is altijd: zal dit nog werken bij een nieuwe versie van de software?

Gebruik Chrome Dev Tools om te kijken wat de mogelijkheden zijn binnen de pagina die je wil testn.

Je kan hier ook heel wat uittesten in de element-locator (CTRL-F)



4.3.2 Interact with html element

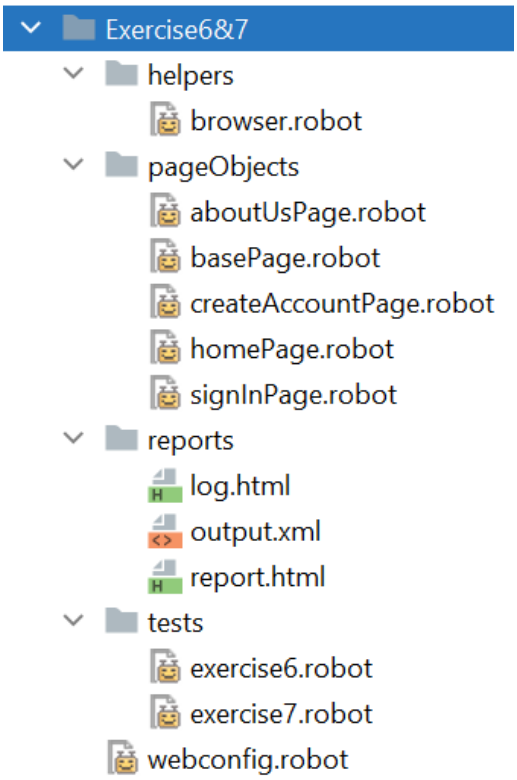
Er zijn met Selenium heel veel mogelijkheden om iets te doen met een html element.
Zie http://robotframework.org/SeleniumLibrary/SeleniumLibrary.html#Keywords

Bijvoorbeeld:

- Click Button
- Click Element
- Click Link
- Delete All Cookies
- Element Should Contain
- Element Text Should Be
- Get Text
- Get Title
- Input Text
- Page Should Contain
- Page Should Not Contain
- Reload Page

4.4 Project structure: Page Object Model

Gebruik deze project structuur:



In de pageObjects folder zet je per pagina een robot file.

- Een pageObject behandelt een deel van de UI als een object
  - Meestal maak je 1 object per pagina
  - Grote paginas worden opgebroken in sub-pagina's
- Een pageObject is verantwoordelijk voor:
  - Vinden van elementen
  - Functinaliteit voor deze elementen voorzien
  - Mappen van resultaten van acties
- pageObject bevat geen assertions
  - assertions (should) zitten in de tests

De Test cases in de tests folder gebruiken de keywords in de pageObjects folder
De file webconfig.robot bevat de urls als variabelen

4.5 Exercise: Login example

Eerste Test case

- Go to https://the-internet.herokuapp.com/login
- Enter tomsmith as username
- Enter incorrectPassword as password
- Click on login
- Validate the error message to be "Your password is invalid!"

- Close the browser

Tweede Test case

- Go to https://the-internet.herokuapp.com/login
- Enter tom as username
- Enter incorrectPassword as password
- Click on login
- Validate the error message to be “Your username is invalid!”
- Close the browser

4.6 Suite - Setup – Teardown

- **Suite** = Test Cases in 1 file
- **Test Setup**: wordt uitgevoerd voor elke Test Case van deze Suite voordat de Test uitgevoerd wordt
- **Test Teardown:** wordt uitgevoerd voor elke Test Case van deze Suite nadat de Test uitgevoerd is – ook als die gefaald is
- **Suite Setup**: wordt uitgevoerd voordat een test van een Suite uitgevoerd worden
- **Suite Teardown**: wordt uitgevoerd nadat alle tests van een Suite uitgevoerd zijn – zelfs al zijn ze gefaald

Dus:

1. Suite Setup
    a. Test Setup
        i. Test Case 1
    b. Test Teardown
    c. Test Setup
        i. Test Case 2
    d. Test Teardown
2. Suite Teardown

Zie

- https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#test-setup-and-teardown
- https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#suite-setup-and-teardown

4.7 Best practices

- Gebruik **variabelen** om duplicatie te vermijden (bvb urls)
- Gebruik **keywords** om duplicatie te vermijden
- **Verifieer** altijd iets zodat het duidelijk is wat je wil testen
- **Clean up** na de test: sluit browser, verwijder data die je aangemaakt hebt
- **Clean up** ook als de test mislukt

5 Conclusie

- automated testing is een doorlopende investering
- automated tests moeten continu runnen (en niet 1 keer per release of 1 keer per week)
- automated tests vragen onderhoud (maintenance)
- 1 automated test is beter dan geen automated test…
- Test environment om automated tests te runnen
- Wat doe je met test data??
    - Stel: een test insert data (bvb maakt een nieuw product aan – doet een bestelling)
    - Zorg ervoor dat deze data ook verwijderd wordt
    - …ook als de test faalt!!!
- Wat doe je als de test faalt?????
    - Analyse: waarom faalt de test?
        - Bug in de code
        - Bug in de test
        - Code is aangepast en de test niet
        - Test data is niet meer consistent
        - Probleem in de test environment
    - Maak een log van de problemen en de redenen
    - FIX de failing test!!!
    - Doe dit onmiddelllijk: als de tests eenmaal rood zijn zie je niet meer of er nog failing tests bijkomen

6 Opdracht

- Opdracht in Canvas "Test Automation"
- Automate 5 Test Cases
- Applicatie die je test: **MenuCard frontend**
    - https://verapeeters-thomasmore.github.io/menus/#/
    - https://verapeeters-thomasmore.github.io/menubackend/
    - Zie menuCardStories.pdf
    - Noot: Je moet menubackend niet testen maar zal het wel nodig hebben om functionaliteit van frontend te testen
- Let op: gebruik niet de class-names die gegenereerd worden door package StyledComponents want die veranderen bij elke nieuwe versie van de applicatie
- Deze Test Cases mogen uit je Test Plan komen – het mogen ook andere zijn
- De Test Cases moeten
    - voldoende complex zijn
    - bruikbaar zijn
    - de aanbevolen structuur met keywords gebruiken
- zet je test-project in github en invite de docent
- Geef in de Canvas opdracht de juiste info ivm je testen:
    - url van de github repository van je test-project

    - report van het resultaat van de automatische testen
- Bij de mondelinge eindevaluatie in week 7 toon je je automatische testen en je legt uit hoe ze werken en wat ze testen.

7 Referentie

- Robot Framework
    - http://robotframework.org/robotframework/
    - http://robotframework.org/robotframework/latest/libraries/BuiltIn.html
    -
- Selenium
    - http://robotframework.org/SeleniumLibrary/SeleniumLibrary.html
- Voorbeeld code:
    - https://github.com/SeppVC-Brightest/Robot-Framework-Training