MENDIX MEETUP

UI Testing with Robot Framework in Mendix

Sebastiaan den Boer
Solution Architect

Agenda

01 Testing in the Mendix Ecosystem

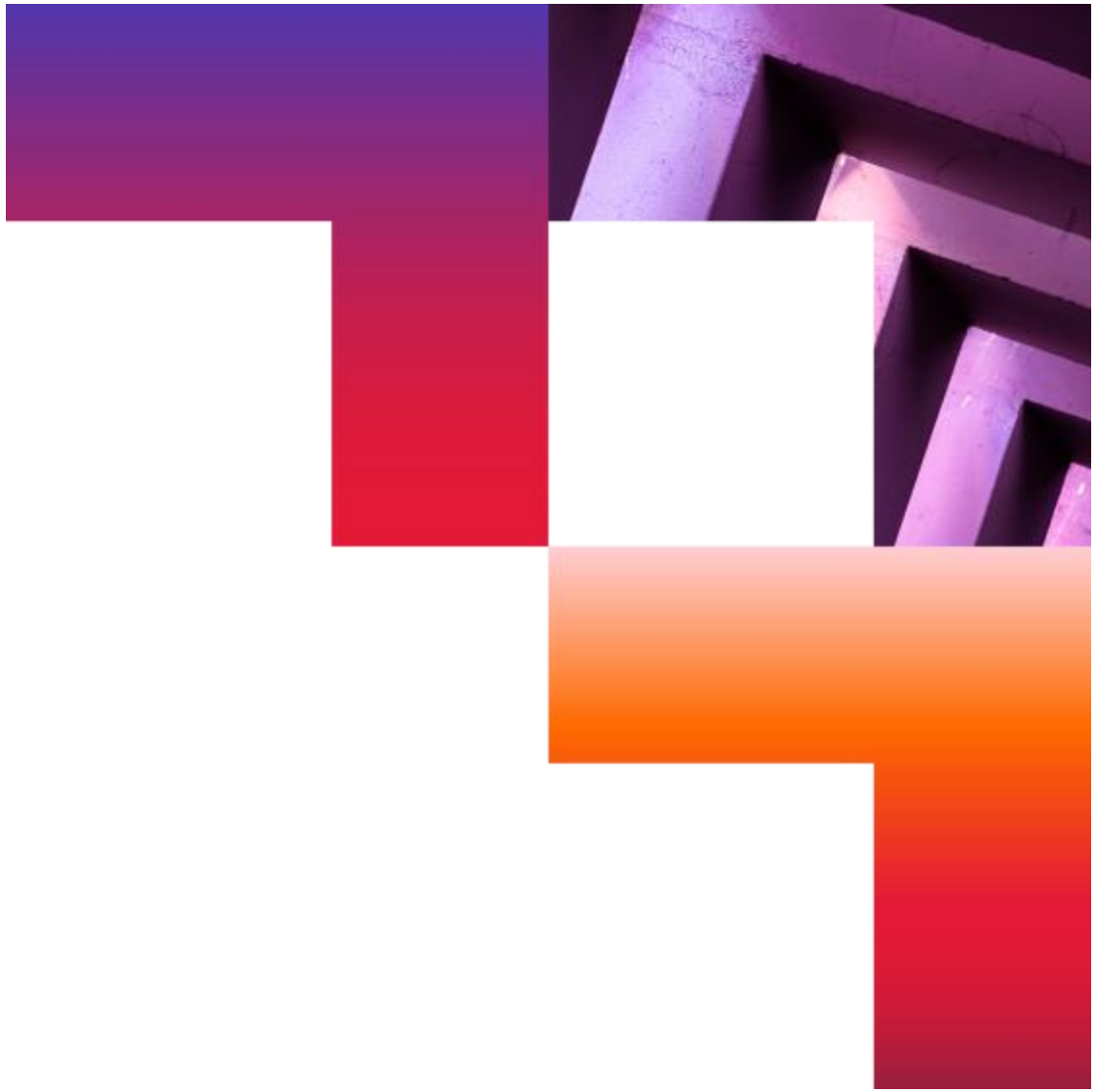
02 Introduction to Robot Framework

03 The Anatomy of a Test Case

04 Quick Tips for building Test Cases

05 Hands-on Session / Workshop

06 Recap & Questions



Testing in the Mendix Ecosystem

Testing in the Mendix ecosystem

Most-used methods of testing

1 **Manual Validation**
Manually checking the system to ensure it behaves as expected

2 Unit Testing
Individual units of software are tested to validate component behavior

3 User Interface Testing
Automate user interface interactions and check if the application behaves as expected

Benefits

- Shorter release time for features, since no test code is needed

Disadvantages

- Testing becomes incredibly time-intensive with extensive features or a growing project scope
- Error-prone due to human nature of repeating manual steps
- Performance testing results are difficult to reproduce
- Cannot be run automatically in CI/CD environments

Testing in the Mendix ecosystem

Most-used methods of testing

1 **Manual Validation**
Manually checking the system to ensure it behaves as expected

2 **Unit Testing**
Individual units of software are tested to validate component behavior

3 **User Interface Testing**
Automate user interface interactions and check if the application behaves as expected

Testing in the Mendix ecosystem

Most-used methods of testing

1 **Manual Validation**
Manually checking the system to ensure it behaves as expected

2 **Unit Testing**
Individual units of software are tested to validate component behavior

3 **User Interface Testing**
Automate user interface interactions and check if the application behaves as expected

Popular UI Testing tools in the Mendix ecosystem

Selenium

- Older technology with a robust community and documentation
- Established and extensive coverage of web technology
- Stable platform, but performance lacks compared to alternatives
- Lacking modern features, creating customization overhead

Playwright

- Newer technology with a growing community and documentation
- Features and capabilities aligned with the **modern web**
- Stable platform with performance fitting complex web apps
- **Auto-wait feature**, ensuring more accurate results

Introduction to Robot Framework

Introduction to Robot Framework

"Robot Framework is a generic open-source automation framework. It can be used for **test automation** and **robotic process automation** (RPA)."

"Robot Framework has an **easy syntax**, utilizing **human-readable keywords**."

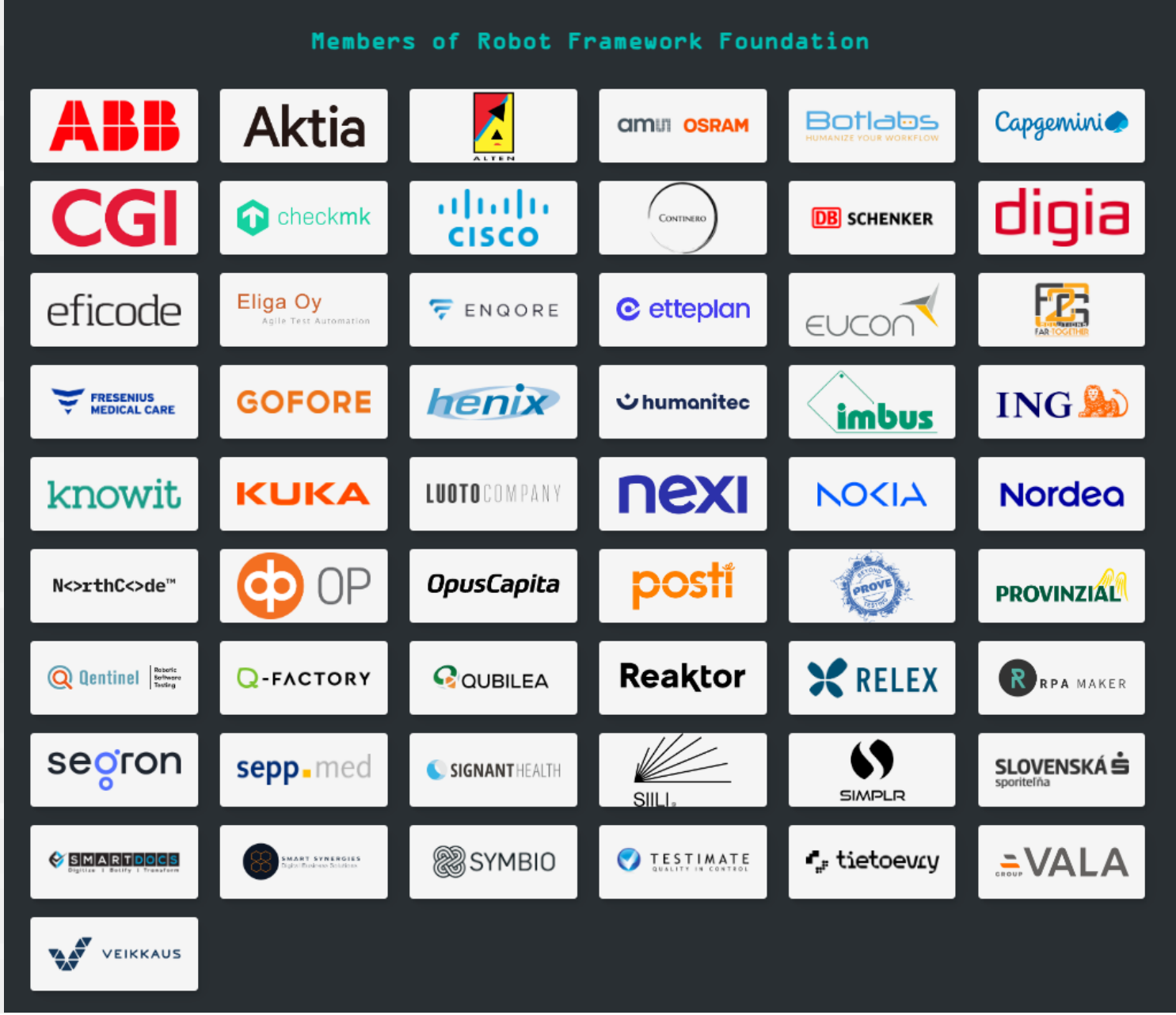
Robot Framework Foundation
https://robotframework.org/

Foundation

Development of Robot Framework is funded by the non-profit Robot Framework Foundation.

It consists of companies and organizations that want to ensure the continuity of Robot Framework now and in the future.

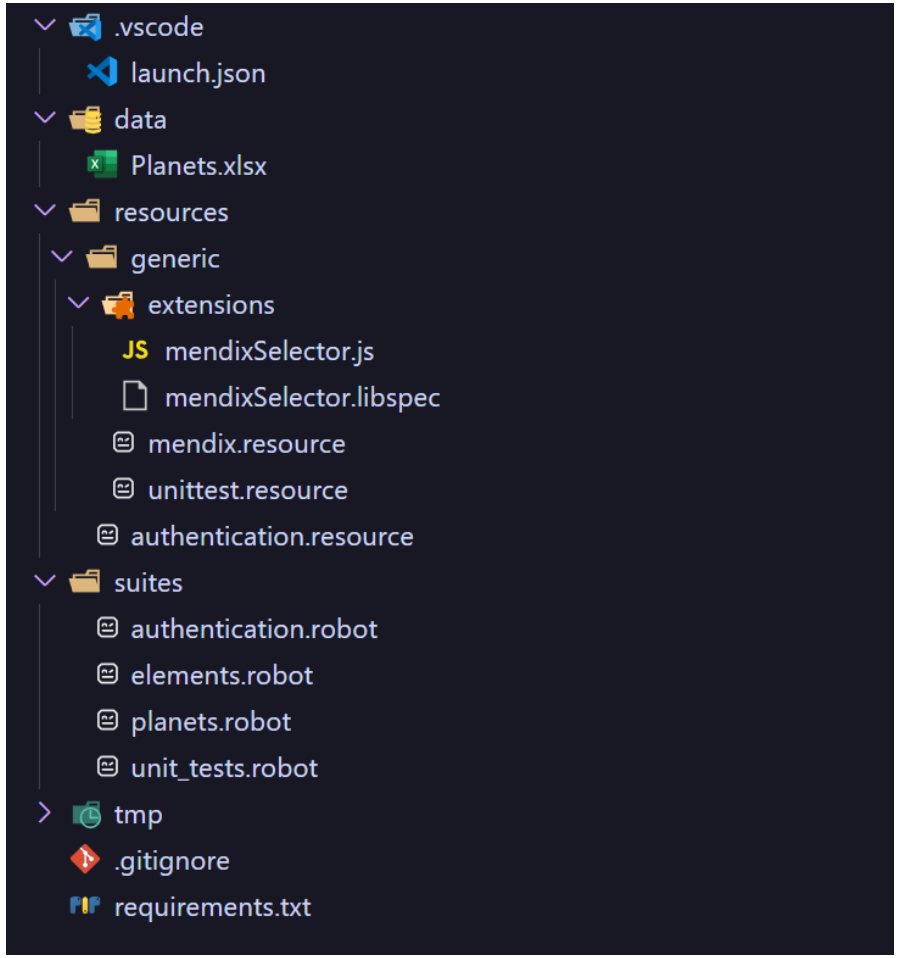
CGI Representative: Pieter Wesseling



How to Get Started with Robot Framework

Add a folder called **“test”** to the root directory of your Mendix application repository with the following optional directory structure.

Folder/File	Purpose
.vscode/	Visual Studio Code configuration
data/	data files for use in test cases
resources/	**.resource files and library extensions**
suites/	**.robot files with test suites & test cases**
tmp/	specified in .vscode/launch.json as default output directory for result files
.gitignore	Git file defining ignored files in the directory
requirements.txt	can be used to install all Python packages using “pip install -r requirements.txt”



Finding Elements

Robot Framework uses **“selectors”** to find and obtain elements on a web page. Each selector has its own strategy:

Strategy	Example
CSS	css=.class > \#login_btn
XPath	xpath=//input[@id="login_btn"]
Text	text=Login
ID	id=login_btn

Tip: Using the **jsextension** argument whilst importing the Browser library, you can create your own custom selector strategy!

```
Load in Interactive Console
Login
    [Documentation]    Authenticate with an Account, given its username and password.
    [Arguments]    ${username}    ${password}

    # Navigate to the Login page
    Go to Mendix Path    login
    Wait Until Network Is Idle

    Fill Text    mx=loginIdTextBox >> input    ${username}
    Fill Secret    mx=passwordTextBox >> input    ${password}

    ${login_btn}=    Get Element    mx=signInButton
    Click    ${login_btn}
    Wait For Elements State    ${login_btn}    detached    timeout=5s

Load in Interactive Console
Logout
    [Documentation]    Logout from the current Mendix application.
    ${logout_btn}=    Get Element    css=.mx-navbar-item >> a[title='Logout']
    Click    ${logout_btn}
    Wait For Elements State    ${logout_btn}    detached    timeout=5s
```

mendix.resource

resources > generic > mendix.resource > ...

```
Load in Interactive Console
*** Settings ***
Documentation       Useful generic Keywords for testing Mendix applications.

Library             BuiltIn
Library             Browser    jsextension=${CURDIR}/extensions/mendixSelector.js
```

For this workshop, we have added a “**mx**” selector, allowing you to directly specify the Name property of a Mendix page element

The Anatomy of a Test Case

The Anatomy of a Test Case

Settings contain imported resources and libraries

Test Cases contains test definitions for this “suite”

Keywords contain reusable methods (either in .robot or .resource files)

suites > authentication.robo

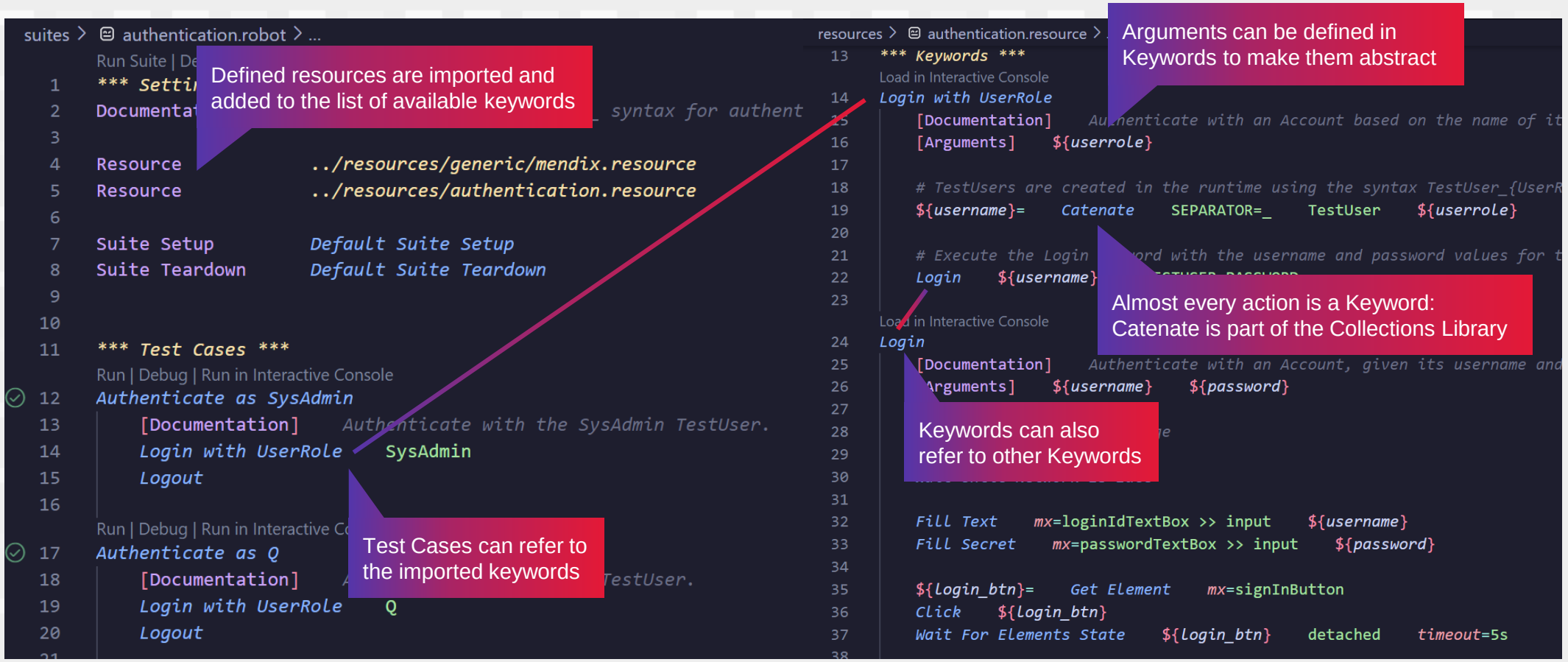
Run Suite | Debug Suite | Load in Interactive Console

```
1  *** Settings ***
2  Documentation       Verifies that the TestUser_ syntax for authent
3
4  Resource            ../resources/generic/mendix.resource
5  Resource            ../resources/authentication.resource
6
7  Suite Setup         Default Suite Setup
8  Suite Teardown      Default Suite Teardown
9
10
11 *** Test Cases ***
   Run | Debug | Run in Interactive Console
12 Authenticate as SysAdmin
13     [Documentation]    Authenticate with the SysAdmin TestUser.
14     Login with UserRole    SysAdmin
15     Logout
16
   Run | Debug | Run in Interactive Console
17 Authenticate as Q
18     [Documentation]    Authenticate with the Q TestUser.
19     Login with UserRole    Q
20     Logout
```

resources > authentication.res > ...

```
13 *** Keywords ***
   Load in Interactive Console
14 Login with UserRole
15     [Documentation]    Authenticate with an Account based on the name of it
16     [Arguments]    ${userrole}
17
18     # TestUsers are created in the runtime using the syntax TestUser_{UserR
19     ${username}=    Catenate    SEPARATOR=_    TestUser    ${userrole}
20
21     # Execute the Login keyword with the username and password values for t
22     Login    ${username}    $TESTUSER_PASSWORD
23
   Load in Interactive Console
24 Login
25     [Documentation]    Authenticate with an Account, given its username and
26     [Arguments]    ${username}    ${password}
27
28     # Navigate to the Login page
29     Go to Mendix Path    login
30     Wait Until Network Is Idle
31
32     Fill Text    mx=loginIdTextBox >> input    ${username}
33     Fill Secret    mx=passwordTextBox >> input    ${password}
34
35     ${login_btn}=    Get Element    mx=signInButton
36     Click    ${login_btn}
37     Wait For Elements State    ${login_btn}    detached    timeout=5s
```

The Anatomy of a Test Case



The Anatomy of a Test Case

suites > authentication.robot > ...

Run Suite | Debug Suite | Load in Interactive Console

```
*** Settings ***
Documentat
Resource
Resource          ../resources/authentication.resource

Suite Setup       Default Suite Setup
Suite Teardown    Default Suite Teardown


*** Test Cases ***
Run | Debug | Run in Interactive Console
Authenticate as SysAdmin
    [Documentation]    Authenticate with the SysAdmin TestUser.
    Login with UserRole    SysAdmin

Authenticate as Q
    [Documentation]    Authenticate with the Q TestUser.
    Login with UserRole    Q
    Logout
```

Suite Setup and Teardown can be used for defining general actions to execute before and after Test Cases

Test Case results are displayed next to their definition in Visual Studio Code. It also can be clicked to run the test

resources > authentication.resource > ...

```
*** Keywords ***
Load in Interactive Console
Login with UserRole
    [Documentation]    Authenticate with an Account based on the name of it
    [Arguments]    ${userrole}

    # TestUsers are created in the runtime using the syntax TestUser_{UserR
    ${username}=    Catenate    SEPARATOR=_    TestUser    ${userrole}

    # Execute the Login keyword with the username and password values for t
    Login    ${username}    $TESTUSER_PASSWORD

Load in Interactive Console
Login
    [Documentation]    Authenticate with an Account, given its username and
    [Arguments]    ${username}    ${password}

    # Navigate to the Login page
    Go to Mendix
    Wait Until Ne

    Fill Text
    Fill Secret    =passwordTextBox >> input    ${password}

    ${login_btn}=    Get Element    mx=signInButton
    Click    ${login_btn}
    Wait For Elements State    ${login_btn}    detached    timeout=5s
```

You can store result values of certain keywords in variables

Quick Tips for building Test Cases

Quick tips for Mendix-specific UI testing scenario’s

- For **long-running, asynchronous actions and page navigation**, be aware that Mendix client performance can impact element presence and value-correctness.
  → Note that Playwright / Browser library has Implicit waiting mechanisms. Depending on the situation you might choose to explicitly state Wait for Elements State to modify the default auto-wait and retry behavior.

- **Be specific!** Use CSS or custom selectors to find specific Mendix elements using their Name property, since this is converted to a CSS class: **“.mx-name-[Name]”**

- **Clean up your mess!** Not all data can often be deleted from within the UI, such as generated log data. This can be solved by adding Unit Tests in a separate module which delete test data generated by Robot Framework (**see example project**).
  → Note that you can even run Unit Tests from within Robot Framework using the Remote API.

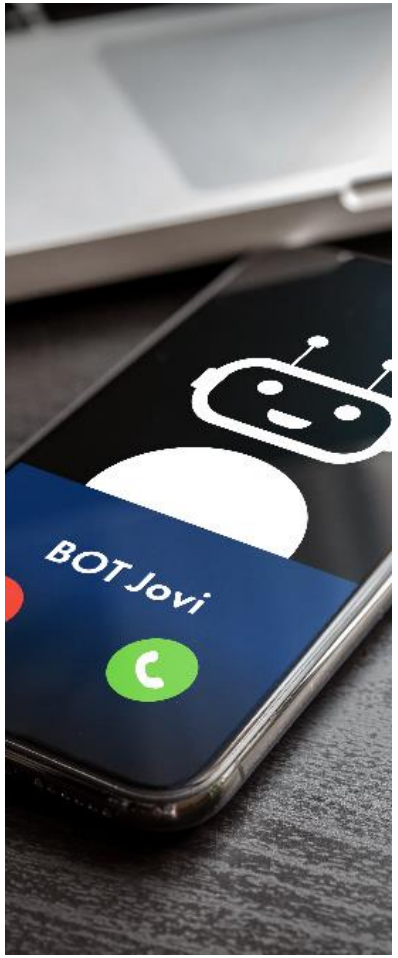
Want more tips & tricks? Ask our mentors during the workshop!

Quick tips for working with Robot Framework

- **Don’t reinvent the wheel!** There are libraries available for most use cases.
  An overview of these libraries can be found here: https://robotframework.org/#resources

- **Build first, optimize later!** This principle works the same as your microflows. There is a reason we “Extract to submicroflow” rather than copy actions every time we reuse them.
  → In Robot Framework, we can abstract to keywords in either “.robot” or “.resource” files. Depending on your level of reusability, consider adding resource files related to your Mendix modules. This will keep your repository nice and clean!

- **The Robot Framework and Browser library documentation are amazing resources** and often-overlooked by developers and testers. You can find them here:
  - Getting Started with Robot Framework
  - Introduction to Browser Library
  - Browser Library Keyword Documentation



Want more tips & tricks? Ask our mentors during the workshop!

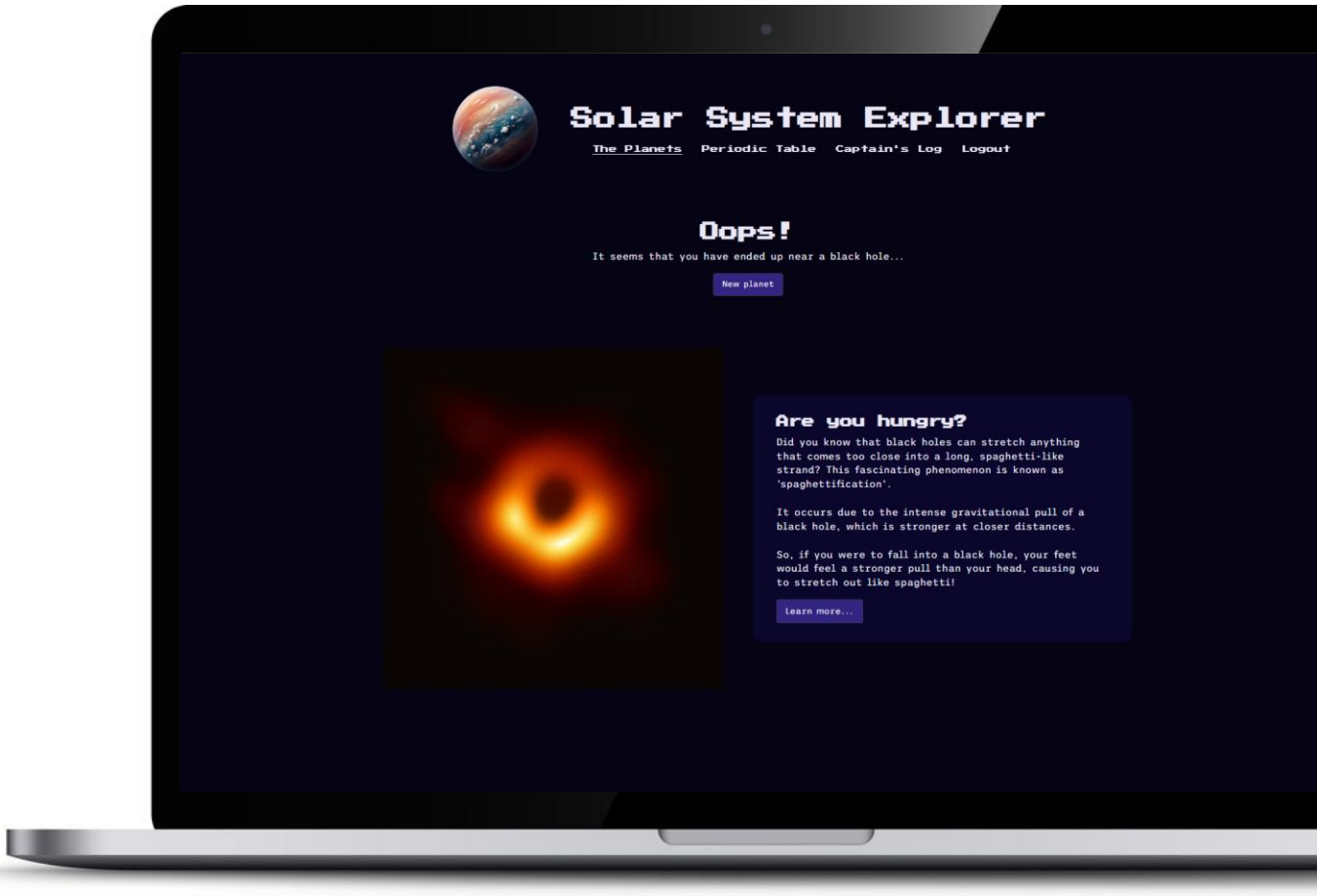
Hands-on Session / Workshop

Workshop – UI Testing with Robot Framework

Solar System Explorer

- **Explore the solar system** and find out all there is to know about our planetary neighbors!
- **Become an astronaut** and log all your interplanetary missions in your personal Captain’s Log!
- Feeling cheeky? **Want to make the universe your sandbox?** Become all-powerful and take the fate of the universe in your own hands!

Your mission: practice your UI testing skills with Robot Framework by completing stub test scenarios



Workshop – UI Testing with Robot Framework

Mentors

Robert Hertel
robert.hertel@cgi.com

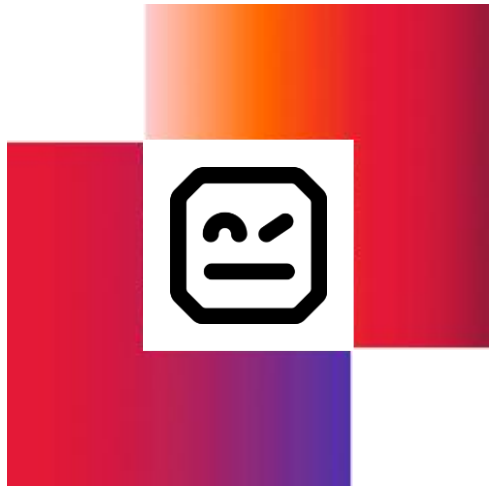
Darlene Hill
darlene.hill@cgi.com

Sebastiaan den Boer
sebastiaan.den.boer@cgi.com

Tim Bouma
t.bouma@cgi.com

Ricardo Adriaens
ricardo.adriaens@cgi.com

Workshop – UI Testing with Robot Framework
Until19.20



Break-out rooms available for this workshop:

- B10.01
- B10.02
- B10.03
- B10.04

All challenges are solvable with the listed dependencies

Feel free to experiment; multiple solutions are possible

Are you stuck with the challenges? **Ask one of our mentors!**

→ **Full source code available at https://github.com/SebastiaandenBoer/mendix-robot-framework-demo**

Or scan the QR code!

Solutions Walkthrough

Please fill out the survey

Thank you for coming!

Sebastiaan den Boer
Solution Architect

+31 6 82723452
sebastiaan.den.boer@cgi.com

Questions? Find me during the network drinks!

→ **We cannot wait to see you during our next meetups in collaboration with BlueGreen and Rijkswaterstaat!**