Afbeelding met tekst, Lettertype, logo, Graphics

Automatisch gegenereerde beschrijving

Version 1, 22-08-2023

Jorne van Zanden

Valori Quality Assurance & Testing B.V | Orteliuslaan 1000, 3528 BD, Utrecht

**Valori Automation Practice Shop Robot Framework Learning Project**

# **Introductie**

Deze opdracht dient ter kennismaking met Robot FW. We hebben allereerst de RobotFW-training gevolgd, georganiseerd door de Academy, en/of de online training Python en RobotFW op TAU (<https://testautomationu.applitools.com>). Met deze opgedane kennis willen we ons eigen testautomatiseringsframework opzetten.

Valori heeft ons de taak gegeven om een webshop te testen op risico's die verband houden met de functionaliteit ervan. De eigenaar van de webshop maakt zich zorgen over de juiste werking ervan. Deze webshop staat bekend als de Valori Webshop (<https://webshop.mobiletestautomation.nl>).

Om gericht te werk te gaan, hebben we met de eigenaar afgesproken dat we eerst een analyse zullen maken van de productrisico's. Hierop zullen we later onze test- en automatiseringsinspanningen baseren. De resultaten van de PRA zijn ook te vinden in de repository. Hierin hebben we verschillende functionaliteiten binnen de webshop onderscheiden.

Als team hebben we eerst de meest opvallende functionaliteiten geautomatiseerd om van elkaars ervaringen te leren. Vervolgens zijn we individueel aan de slag gegaan met het automatiseren van diverse functionaliteiten op aparte branches. Na onderling overleg worden de 'best practices' samengevoegd in de hoofdbranch.

Opdracht: Er zijn nog enkele functionaliteiten over die je zelfstandig kunt automatiseren. Om echter vertrouwd te raken met RobotFW en de website, is het verstandig om eerst de eerder geautomatiseerde functionaliteiten opnieuw aan te pakken. Zo heb je een 'voorbeeld' om op terug te vallen. Pas daarna kun je nieuwe functionaliteiten beter automatiseren. Onze ervaring leert dat dit samen met anderen wordt gedaan voor een groter leereffect.

Clinton Wong & Murat Ozben

# **Installatie en configuratie benodigde software**

* Deze instructies zijn ook terug te vinden in de README.md die in de “Documentation” folder van het Robot Framework project staat.
* **Robot-Framework**

**Installation**

The installation instructions are mainly for setting up the framework on windows.

**Download Python**

https://www.python.org/downloads/

Download and install

**Check pip version**

Run the following command:

pip --version

**Install Robot Framework**

Run the following command:

pip install robotframework

**Install Robot Selenium Library**

Run the following command:

pip install --upgrade robotframework-seleniumlibrary

**Install webdrivermanager**

Run the following command:

pip install webdrivermanager

**Download the repo**

Go to the folder where you want to setup up your framework.

For example:

C:\Projects

Repository location:

https://github.com/JanWouterLok/Robot-Framework.git

* **Optional**

**Install git**

https://github.com/git-guides/install-git

Download and install

**Install robotframework-requests library**

pip install robotframework-requests

**There are many more libraries to choose from, suited for different testing needs**

Visit https://robotframework.org/ and look under “Resources” for the libraries you need in your projects

**Instead of using the WebDriver Manager it is also possible to download the webdrivers manually for more hands on control.**

Chrome: https://chromedriver.chromium.org/downloads

Edge: https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/

Firefox: https://github.com/mozilla/geckodriver/releases

Download and unzip in your preferred directory.

For example:

C:\bin

Make sure to add the system path like this:

Step 1: Open System Properties (Systeemeigenschappen)

1. Press the Windows key on your keyboard to open the Start Menu.
2. Type "System Properties" and select "Edit the system environment variables" from the search results.

Step 2: Access Environment Variables

1. In the "System Properties" window, click the "Environment Variables (Omgevingsvariabelen)" button. This will open the "Environment Variables" window.

Step 3: Add a System Path

1. In the "Environment Variables" window, under the "System variables" section, scroll down and find the variable named "Path".
2. Select the "Path" variable and click the "Edit..." button.

Step 4: Add a New Path

1. In the "Edit Environment Variable" window, click the "New" button to add a new path.
2. Enter the path you want to add. Each path should be on a new line. For example, if you want to add "C:\bin", simply type that in.
3. Click "OK" to close the "Edit Environment Variable" window.

Step 5: Verify the Changes

1. Click "OK" on the "Environment Variables" window to close it.
2. Click "OK" on the "System Properties" window to close it as well.

Step 6: Restart or Apply Changes

For the changes to take effect, you might need to restart your computer or any relevant applications that use the system path.

* **Editors**

**Install VS Code**

https://code.visualstudio.com/download

Download and install

**Install PyCharm (Optional)**

https://www.jetbrains.com/pycharm/download/#section=windows

Download and install

**Install RIDE (Optional)**

Run the following commands:

pip install -U wxPython

pip install --pre --upgrade robotframework-ride==2.0b1

**Risk Analysis & Examples**

***Important Note****: These risk analysis examples are general examples and not specific for the Valori Automation Practice Shop. For educational purposes it is requested you make your own risk analysis for the Robot Framework Project.*

“Risk analysis is a crucial part of software development and testing. It involves identifying potential risks or issues that could impact the quality, functionality, security, or usability of a software application. The identified risks are then categorized into different levels based on their severity and potential impact.”

## **Possible Risk Factors**

1. **No Risk (1)**:
   * A spelling mistake in a user interface text.
   * A minor styling issue on a less-visited page.
2. **Low Risk (2-3)**:
   * A button color slightly deviates from the brand guidelines.
   * A rarely used feature has a small alignment issue.
3. **Low to Moderate Risk (4-5)**:
   * A form validation error that might lead to incorrect data being saved.
   * An intermittent issue where a user's session is occasionally timed out unexpectedly.
4. **Moderate Risk (6)**:
   * A navigation link that sometimes doesn't work, affecting user experience.
   * An error in a payment processing module leading to occasional payment failures.
5. **Moderate to High Risk (7-8)**:
   * Security vulnerability that requires specific conditions to exploit, but could lead to unauthorized access.
   * Inconsistent behavior of a critical feature, resulting in data discrepancies.
6. **High Risk (9)**:
   * A security vulnerability that can be exploited remotely and potentially lead to a data breach.
   * A core functionality consistently failing, impacting a large portion of users.
7. **Critical Risk (10)**:
   * A security breach that exposes sensitive user data, affecting compliance with regulations.
   * The entire application crashing upon startup for all users.

**Possible User Stories**:

1. **No Risk (1):**
   * User Story: As a user, I want to see UI text without spelling or grammatical errors, and visual elements that match the design guidelines consistently.
2. **Low Risk (2-3):**
   * User Story: As a user, I want UI elements to have consistent colors across different sections of the website and proper alignment in less prominent areas.
3. **Low to Moderate Risk (4-5):**
   * User Story: As a user, I want images on product pages to load correctly and reflect accurate product prices from the database.
4. **Moderate Risk (6):**
   * User Story: As a user, I want to access the payment processing form without encountering failures and complete the checkout process successfully on various devices and browsers.
5. **Moderate to High Risk (7-8):**
   * User Story: As a user, I want to log in securely, ensuring that the login form is not vulnerable to unauthorized access. I also want my data to remain secure during my entire session on the website.
6. **High Risk (9):**
   * User Story: As a user, I want to provide sensitive customer data securely, trusting that it won't be exposed. Additionally, I want to trust that encryption and decryption of my sensitive data work as intended.
7. **Critical Risk (10):**
   * User Story: As a user, I want to rely on robust security measures that prevent widespread breaches and experience effective incident response procedures in handling critical security situations.

### **Urgency**

1. **No Risk (1):**
   * Urgency: No immediate action required.
   * Use Case: Ensuring correct spelling and grammar in UI text, consistent styling on all pages.
2. **Low Risk (2-3):**
   * Urgency: Low urgency, can be addressed in routine maintenance.
   * Use Case: Maintaining consistent button colors, addressing minor alignment issues.
3. **Low to Moderate Risk (4-5):**
   * Urgency: Attention needed, but not critical.
   * Use Case: Correcting form validation errors, addressing intermittent session timeout issues.
4. **Moderate Risk (6):**
   * Urgency: Moderate urgency, needs prompt attention.
   * Use Case: Fixing navigation link issues, resolving occasional payment failures.
5. **Moderate to High Risk (7-8):**
   * Urgency: High urgency, requires swift action.
   * Use Case: Fixing security vulnerabilities, ensuring consistent critical feature behavior.
6. **High Risk (9):**
   * Urgency: Extremely high urgency, demands immediate action.
   * Use Case: Addressing remote security vulnerabilities, core functionality failures.
7. **Critical Risk (10):**
   * Urgency: Critical urgency, requires immediate and focused attention.
   * Use Case: Responding to security breaches exposing user data, handling application crashes.

# **Test Scenarios & Test Cases**

Afbeelding met tekst, schermopname, nummer, Parallel

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, lijn, Parallel

Automatisch gegenereerde beschrijving

Credits: Murat Ozben

# **Setting up a Robot Framework Test Suite according to the Page Object Model(POM)**

***Important Note****: Setting up a Robot Framework Project doesn't come with a fixed rulebook. The examples provided are gentle nudges on how one might begin their journey with a Robot Framework project. By browsing through the Valori Automation Practice Shop Robot Framework Learning Project repository, you'll see that contributors have explored various approaches. It's worth noting that even the Page Object Model (POM) can be left behind if circumstances require or preferences align as such.*

**Step 1: Create the Page Object Classes**

1.1. Create a folder structure for your project:

markdown

- tests

- page\_objects

- test\_suites

1.2. Inside the page\_objects folder, create a .robot file for each page of your webshop. For example, you might have HomePage.resource, ProductPage.resource, and CheckoutPage.resource.

1.3. Define page object keywords in these .resource files. These keywords will use SeleniumLibrary keywords to interact with elements on the page.

Example HomePage.resource:

robot

\*\*\* Settings \*\*\*

Library SeleniumLibrary

\*\*\* Keywords \*\*\*

Open Webshop Homepage

Go To ${URL}

Input Search Keyword

[Arguments] ${keyword}

Input Text id=search-box ${keyword}

Submit Search

Click Button name=submit-button

Repeat the process for other page object files (ProductPage.resource, CheckoutPage.resource, etc.).

**Step 2: Create the Test Suite**

2.1. Inside the test\_suites folder, create a .robot file. For example, you could create a file named webshop\_tests.robot.

2.2. Define your test suite and test cases in the .robot file. Import the page object keywords from the page object files and use them to interact with the web application.

Example webshop\_tests.robot:

robot

\*\*\* Settings \*\*\*

Library SeleniumLibrary

Resource ../page\_objects/HomePage.resource

Resource ../page\_objects/ProductPage.resource

Resource ../page\_objects/CheckoutPage.resource

\*\*\* Variables \*\*\*

${BROWSER} Chrome

${URL} http://your-webshop-url.com

\*\*\* Test Cases \*\*\*

Search and Add to Cart

Open Browser ${URL} ${BROWSER}

Open Webshop Homepage

${product} Set Variable Laptop

${quantity} Set Variable 2

Input Search Keyword ${product}

Submit Search

Click Link ${product}

# Perform actions on the Product page

# ...

# Perform actions on the Checkout page

# ...

Close Browser

**Step 3: Run the Test Suite**

3.1. Open a terminal and navigate to your project's root directory.

3.2. Run the test suite using the Robot Framework command:

robot tests/test\_suites/webshop\_tests.robot

This will execute the test cases defined in the webshop\_tests.robot file.

Remember that this is a simplified example. You'll need to complete the keywords for the Product and Checkout pages with the actual interactions and validations required for your webshop.

**Extra pointers**

It is a good programming practice to keep different functions separated from each other in their own files. In a Robot Framework project, you might want to use separate directories such as 'Keywords', 'Locators', 'Variables', 'PO' (Page Objects), 'Results', 'Tests', and others. Each of these directories would contain their respective files.

For example:

The file containing the tests:

ValoriWebShopTests.robot

\*\*\* Settings \*\*\*  
  
Library *OperatingSystem*Library *Dialogs*Library *String*Resource *../Resources/ValoriWebShop.resource*Resource *../Resources/Common.resource*Resource *../Resources/Variables/TestCasesVariables.resource*Test Setup Start web test  
Test Teardown End web test  
  
\*\*\* Test Cases \*\*\*  
  
Should be able to open webshop  
 ***[Documentation]*** *As a user I want to be able to open the webshop* ***[Tags]*** *TC-001 Smoke Open Webshop* ValoriWebShop.Go to Valori Webshop landing page  
  
Should be able to run tests in different browsers  
 ***[Documentation]*** *As a user I want to be able to open the webshop and run the tests in different browsers* ***[Tags]*** *TC-002 Smoke Open Webshop* ${new\_browser} = Get Selection From User *Choose a browser to run this suite with chrome headlesschrome firefox headlessfirefox edge* Set Global Variable ${BROWSER} ${new\_browser}  
  
Should be able to sign in webshop  
 ***[Documentation]*** *Enter a valid username&password and sign in* ***[Tags]*** *TC\_WS\_001 Smoke Sign in Webshop* ValoriWebShop.Go to Valori Webshop landing page  
 ValoriWebShop.Sign in to Valori Webshop  
  
Should not be able to login with invalid email and valid password  
 ***[Documentation]*** *Enter an invalid username&password and try to sign in* ***[Tags]*** *TC\_WS\_002 Functional Sign in Webshop* ValoriWebshop.Go to Valori Webshop landing page  
 ValoriWebshop.Sign in to Valori Webshop with invalid email  
  
Should not be able to login with valid email and invalid password  
 ***[Documentation]*** *Enter an invalid username&password and try to sign in* ***[Tags]*** *TC\_WS\_003 Functional Sign in Webshop* ValoriWebshop.Go to Valori Webshop landing page  
 ValoriWebshop.Sign in to Valori Webshop with invalid password  
  
Should not be able to login with invalid username and invalid password  
 ***[Documentation]*** *Enter an invalid username&password and try to sign in* ***[Tags]*** *TC\_WS\_004 Functional Sign in Webshop* ValoriWebshop.Go to Valori Webshop landing page  
 ValoriWebshop.Sign in to Valori Webshop with invalid email and password  
  
Should be able to create an account and confirm account has been created  
 ***[Documentation]*** *Confirm that the account has been successfully created by verifying that the confirmation page or message states that the account has been created* ***[Tags]*** *TC\_WS\_005 Functional Create Account* ValoriWebShop.Go to Valori Webshop landing page  
 ValoriWebShop.Create a "Mr." account on Valori Webshop

The resource file from the system under test:

ValoriWebShop.resource

\*\*\* Settings \*\*\*  
Resource *../Resources/PO/AccountPage.resource*Resource *../Resources/PO/FooterNav.resource*Resource *../Resources/PO/LandingPage.resource*Resource *../Resources/PO/SignInPage.resource*Resource *../Resources/PO/TopNav.resource*Resource *../Resources/PO/CreateAccountPage.resource*Resource *../Resources/PO/SingleProductPage.resource*Resource *../Resources/PO/OrderModal.resource*Resource *../Resources/PO/CartPage.resource*Resource *../Resources/PO/AddressPage.resource*Resource *../Resources/Keywords/GenerateAndUseMockData.resource*\*\*\* Keywords \*\*\*  
  
Go to Valori Webshop landing page  
 LandingPage.Load  
 LandingPage.Verify page loaded  
  
Sign in to Valori Webshop  
 TopNav.Click sign in link  
 SignInPage.Verify Sign In page loaded  
 SignInPage.Enter Email  
 SignInPage.Enter password  
 SignInPage.Click sign in  
 AccountPage.Verify Account page loaded  
  
Sign in to Valori Webshop with invalid email  
 TopNav.Click sign in link  
 SignInPage.Verify Sign In page loaded  
 SignInPage.Enter invalid Email  
 SignInPage.Enter password  
 SignInPage.Click sign in  
 SignInPage.Capture Page Screenshot after entering invalid email  
 SignInPage.Verify if still on sign in page after entering invalid email and/or password  
  
Sign in to Valori Webshop with invalid password  
 TopNav.Click sign in link  
 SignInPage.Verify Sign In page loaded  
 SignInPage.Enter Email  
 SignInPage.Enter invalid password  
 SignInPage.Click sign in  
 SignInPage.Verify authentication failed message displayed  
  
Sign in to Valori Webshop with invalid email and password  
 TopNav.Click sign in link  
 SignInPage.Verify Sign In page loaded  
 SignInPage.Enter unknown Email  
 SignInPage.Enter invalid password  
 SignInPage.Click sign in  
 SignInPage.Capture Page Screenshot after entering invalid email and password  
 SignInPage.Verify if still on sign in page after entering invalid email and/or password  
  
Create a "Mr." account on Valori Webshop  
 TopNav.Click sign in link  
 SignInPage.Verify Sign In page loaded  
 SignInPage.Click create an account link  
 CreateAccountPage.Verify Create Account page loaded  
 CreateAccountPage.Click radio button "Mr."  
 GenerateAndUseMockData.Generate and Use Valid "Create Account" Mock Data  
 CreateAccountPage.Enter date of birth  
 CreateAccountPage.Click "Receive offers" checkbox  
 CreateAccountPage.Click "Sign up for newsletter" checkbox  
 CreateAccountPage.Click "Save" new account button  
 TopNav.Click logged in user link  
 AccountPage.Verify Account page loaded

One of the Page Object resource files:

\*\*\* Settings \*\*\*  
Library *SeleniumLibrary*Resource *../Variables/TestCasesVariables.resource*Resource *../Locators/SignInPageLocators.resource*\*\*\* Keywords \*\*\*  
  
Verify Sign In Page Loaded  
  
 Wait Until Page Contains Element ${SIGN\_IN\_PAGE\_HEADER}  
 Page Should Contain Element ${SIGN\_IN\_PAGE\_HEADER}  
  
Enter email  
  
 Input Text ${SIGN\_IN\_PAGE\_EMAIL\_BOX} ${VALID\_EMAIL}  
  
Enter password  
  
 Input Text ${SIGN\_IN\_PAGE\_PASSWORD\_BOX} ${VALID\_PASSWORD}  
  
Enter invalid email  
  
 Input Text ${SIGN\_IN\_PAGE\_EMAIL\_BOX} ${INVALID\_EMAIL}  
  
Enter unknown email  
  
 Input Text ${SIGN\_IN\_PAGE\_EMAIL\_BOX} ${INVALID\_UNKNOWN\_EMAIL}  
  
Capture Page Screenshot after entering invalid email  
 Sleep *1s* Capture Page Screenshot ${SCREENSHOT\_INVALID\_EMAIL\_SIGN\_IN}  
  
Enter invalid password  
  
 Input Text ${SIGN\_IN\_PAGE\_PASSWORD\_BOX} ${INVALID\_PASSWORD}  
  
Capture Page Screenshot after entering invalid email and password  
 Sleep *1s* Capture Page Screenshot ${SCREENSHOT\_INVALID\_EMAIL\_AND\_PASSWORD\_SIGN\_IN}  
  
click sign in  
  
 Click Button ${SIGN\_IN\_BUTTON}

One of the Locators resource files:

SignInPageLocators.resource

\*\*\* Variables \*\*\*  
  
${SIGN\_IN\_PAGE\_HEADER} = *xpath=//\*[@id="main"]/header/h1*${SIGN\_IN\_PAGE\_EMAIL\_BOX} = *name=email*${SIGN\_IN\_PAGE\_PASSWORD\_BOX} = *name=password*${SIGN\_IN\_BUTTON} = *id=submit-login*${SIGN\_IN\_PAGE\_INVALID\_PASSWORD\_ERROR} = *Authentication failed.*${SIGN\_IN\_PAGE\_AUTHENTICATION\_ERROR} = *Authentication failed.*${SIGN\_IN\_PAGE\_CREATE\_ACCOUNT\_LINK} = *css=.no-account > a*

And the TestCasesVariables.resource file containing the necessary Test Variables:

\*\*\* Variables \*\*\*  
${BROWSER} *chrome*${URL} *https://webshop.mobiletestautomation.nl*${VALID\_EMAIL} *dantester@jourrapide.com*${VALID\_PASSWORD} *1qazxsw2*${INVALID\_EMAIL} *fout.emailadres*${INVALID\_UNKNOWN\_EMAIL} *Onbekend@email.adres*${INVALID\_PASSWORD} *UnheelvoudPazzwort*${ITEM\_SEARCH} *hummingbird*${FIRST\_NAME\_NEW\_ACCOUNT} *Roy*${LAST\_NAME\_NEW\_ACCOUNT} *Batty*${EMAIL\_NEW\_ACCOUNT} *Replicant6@BladeRunner.com*${PASSWORD\_NEW\_ACCOUNT} *Priss123*${BIRTHDAY\_NEW\_ACCOUNT} *01/01/1993*# Screenshot paths  
  
# Sign in page  
${SCREENSHOT\_INVALID\_EMAIL\_SIGN\_IN}= *../Results/Screenshot\_Invalid\_Email\_Sign\_In.png*${SCREENSHOT\_INVALID\_EMAIL\_AND\_PASSWORD\_SIGN\_IN}= *../Results/Screenshot\_Invalid\_Credentials\_Sign\_In.png*# Create account page  
${SCREENSHOT\_INVALID\_EMAIL\_CREATE\_ACCOUNT}= *../Results/Screenshot\_Invalid\_Email\_Create\_Account.png*${SCREENSHOT\_INVALID\_PASSWORD\_CREATE\_ACCOUNT}= *../Results/Screenshot\_Invalid\_Password\_Create\_Account.png*

# **Further study**

For the study of Robot Framework there are some very good tutorials available on the internet.

* For starters there is the good, detailed, and easy to follow tutorial series made by Raghav Pal from Automation Step By Step on YouTube (I suggest playing it in Firefox with the YouTube Adblocker extension installed to prevent getting completely bonkers by all the ads YouTube throws at you).

Best of all, it is completely free!

https://www.youtube.com/watch?v=oY1JSpVk5ek

* Another very good (even better) one are the series of tutorial videos on LinkedIn Learning and Udemy, “Robot Framework Test Automation: Level 1” and Robot Framework Test Automation: Level 2 by Bryan Lamb.

Beware, these are paid courses!

https://www.linkedin.com/learning/robot-framework-test-automation-level-1-selenium-21986416/about-the-instructor?u=2132228

https://www.udemy.com/course/robot-framework-2/

* The last suggestion is the series of tutorial videos by Paul Merrill on Test Automation University.

https://testautomationu.applitools.com/robot-framework-refresh/

Also free, jolly 😊