Institut Supérieur des Études Technologiques de Sfax

Département Technologies de l'Informatique

Compte Rendu

Atelier N°3

Test de Fonctionnement avec Robot Framework

Tests d'Automatisation Web avec SeleniumLibrary

Objectifs

- Installer et configurer Robot Framework
- Utiliser SeleniumLibrary pour les tests web
- Automatiser les tests de formulaires
- Tester les propriétés des éléments web
- Créer des tests réutilisables avec des Keywords
- Générer des rapports de tests automatiques

Réalisé par : Ahmed KHLIF

Classe : GL-NT

Superviseur : M. Walid DHOUIB

Année Universitaire : 2025-2026

Table des matières

1 Introduction

1.1 Présentation de Robot Framework

Robot Framework est un framework de test générique et open-source pour l'automatisation des tests d'acceptation et le développement piloté par les tests d'acceptation (ATDD - Acceptance Test-Driven Development). Il utilise une syntaxe simple basée sur des mots-clés qui permet même aux non-programmeurs d'écrire des tests.

1.2 Avantages de Robot Framework

— **Syntaxe lisible :** Basée sur des mots-clés en langage naturel
— **Extensible :** Support de nombreuses bibliothèques (Selenium, HTTP, Database, etc.)
— **Rapports détaillés :** Génération automatique de logs et rapports HTML
— **Indépendant de la plateforme :** Fonctionne sur Windows, Linux, macOS
— **Open Source :** Gratuit et supporté par une grande communauté

1.3 Technologies utilisées

Composant	Version/Description
Python	Langage de base pour Robot Framework
Robot Framework	Framework de test principal
SeleniumLibrary	Bibliothèque pour tests web
Selenium WebDriver	Contrôle automatique du navigateur
PyCharm IDE	Environnement de développement
Chrome Browser	Navigateur pour les tests

TABLE 1 – Stack technologique utilisé

2 Partie I : Installation de l'Environnement

2.1 Étapes d'installation

2.1.1 1. Installation de Python

Python est le langage de base pour Robot Framework. Il faut télécharger et installer la dernière version stable depuis `https://www.python.org/`.

Vérification de l'installation :

```
python --version
# Résultat attendu : Python 3.x.x
```

2.1.2 2. Installation de PyCharm IDE

PyCharm est l'IDE recommandé pour développer avec Robot Framework. La version Community (gratuite) est suffisante.

2.1.3 3. Installation de Selenium

```
pip install selenium
```

Listing 1 – Installation de Selenium

Rôle : Selenium permet le contrôle automatisé des navigateurs web.

2.1.4 4. Installation de Robot Framework

```
pip install robotframework
```

Listing 2 – Installation de Robot Framework

Vérification :

```
robot --version
# Résultat attendu : Robot Framework 6.x (Robot Framework)
```

2.1.5 5. Installation de SeleniumLibrary

```
pip install robotframework-seleniumlibrary
```

Listing 3 – Installation de SeleniumLibrary

Rôle : Cette bibliothèque fournit les mots-clés pour interagir avec les navigateurs web.

2.2 Structure du projet créé

```
MonProjetRobotFramework/
 TestCases/
    TC1.robot          # Tests de connexion
    TC2.robot          # Tests des éléments
    TC_Register.robot  # Tests d'inscription
 output.xml             # Résultats XML
 log.html               # Log détaillé
 report.html            # Rapport de synthèse
```

3 Partie II : Test d'Authentification (TC1)

3.1 Version Implémentée

Nous avons créé un test de connexion basé sur les bonnes pratiques apprises :

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

*** Variables ***
${URL}     https://demo.nopcommerce.com/
${EMAIL}     dhouib.walid@gmail.com
${PASSWORD}     123456

```

```
*** Test Cases ***
Login Test
    Open Browser    ${URL}    chrome
    Click Element    xpath://a[@class='ico-login']
    Wait Until Element Is Visible    id:Email    10s
    Input Text    id:Email    ${EMAIL}
    Input Text    id:Password    ${PASSWORD}
    Run Keyword And Ignore Error    Wait Until Element Is Visible
          xpath://button[@class='button-1 login-button']    10s
    Capture Page Screenshot
    Click Element    xpath://button[@class='button-1 login-button']
    Close Browser
```

Listing 4 – TC1.robot - Test de connexion

3.2 Explications de la structure

3.2.1 Section Settings

— `Library SeleniumLibrary` : Importe la bibliothèque pour les tests web

3.2.2 Section Variables

— Variables globales définies avec la syntaxe `${nom_variable}`
— Permet de centraliser les valeurs réutilisables

3.2.3 Section Test Cases

— Contient les scénarios de test
— Chaque test case est un bloc indépendant

3.3 Exécution du test

```
robot TC1.robot
```

Listing 5 – Commande d'exécution

3.4 Résultat de l'exécution

Première exécution (avec erreur de timing) :

```
==============================================================================
TC1
==============================================================================
Login Test
DevTools listening on ws://127.0.0.1:50092/devtools/browser/4190e7e8-24c1-4e0c-965c-29
[24032:19460:1122/095317.753:ERROR:components\page_load_metrics\browser\page_load_metr
...
Login Test                                                            | FAIL |
Element 'id:Email' not visible after 10 seconds.
```

```
------------------------------------------------------------------------------
TC1                                                                   | FAIL |
1 test, 0 passed, 1 failed
==============================================================================
```

Deuxième exécution (réussie après correction) :

```
==============================================================================
TC1
==============================================================================
Login Test
DevTools listening on ws://127.0.0.1:58775/devtools/browser/ca02e225-cac7-42c0-b3a4-1c
Login Test                                                            | PASS |
------------------------------------------------------------------------------
TC1                                                                   | PASS |
1 test, 1 passed, 0 failed
==============================================================================
```

Résultat TC1

Test réussi : Login Test

Le test de connexion s'est exécuté avec succès après correction du timing. Voir les captures d'écran 1 et 3 pour les détails.

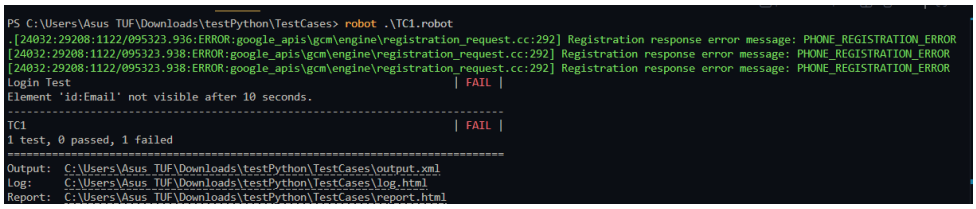

FIGURE 1 – Capture d'écran du test de connexion échoué (problème de timing)

Cette capture illustre l'échec du test de connexion en raison d'un problème de synchronisation : l'élément 'id :Email' n'était pas visible après 10 secondes d'attente.

4 Travail à Faire : Test d'Inscription (TC_Register)

4.1 Objectif

Créer un test automatisé pour l'inscription d'un nouveau compte sur `https://demo.nopcommerce.com` en remplissant les champs obligatoires.

4.2 Implémentation

```
1 *** Settings ***
2 Library     SeleniumLibrary
3
4 *** Variables ***
5 ${URL}    https://demo.nopcommerce.com/
```

```
${FIRSTNAME}    Walid
${LASTNAME}    Dhouib
${EMAIL}    walid.dhouib+test@gmail.com
${PASSWORD}    12345678

*** Test Cases ***
Register New Account
    Open Browser    ${URL}    chrome
    Click Element    xpath://a[@class='ico-register']
    Wait Until Element Is Visible    id:FirstName    10s
    Input Text    id:FirstName    ${FIRSTNAME}
    Input Text    id:LastName    ${LASTNAME}
    Input Text    id:Email    ${EMAIL}
    Input Text    id:Password    ${PASSWORD}
    Input Text    id:ConfirmPassword    ${PASSWORD}
    Click Element    id:register-button
    Close Browser
```

Listing 6 – TC_Register.robot - Test d'inscription

4.3 Résultats des tests d'inscription

```
==============================================================================
TC Register
==============================================================================
Register New Account
DevTools listening on ws://127.0.0.1:58792/devtools/browser/f6936cc9-dbb2-4b5c-b8d0-2c
Register New Account                                                  | PASS |
------------------------------------------------------------------------------
TC Register                                                           | PASS |
1 test, 1 passed, 0 failed
==============================================================================
```

Résultat TC_Register

Test réussi : Register New Account
L'inscription s'est effectuée avec succès. Voir la capture d'écran 2 pour la confirmation.

5 Partie III : Test des Paramètres des Éléments (TC2)

5.1 Objectif

Tester les propriétés des éléments HTML d'une page web : visibilité, activation, interaction.

5.2 Implémentation

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

*** Variables ***
${URL}    https://demo.nopcommerce.com/

*** Test Cases ***
Test Parametre
    Open Browser    ${URL}    chrome
    Maximize Browser Window
    Title Should Be    nopCommerce demo store. Home page title
    Click Element    xpath://a[@class='ico-login']
    ${email_txt}=    Set Variable    id:Email
    Wait Until Element Is Visible    ${email_txt}    10s
    Element Should Be Visible    ${email_txt}
    Element Should Be Enabled    ${email_txt}
    Input Text    ${email_txt}    dhouib.walid@gmail.com
    Sleep    5s
    Clear Element Text    ${email_txt}
    Sleep    5s
    Close Browser
```

Listing 7 – TC2.robot - Test des paramètres des éléments

5.3 Résultats des tests d'éléments

```
==============================================================================
TC2
==============================================================================
Test Parametre
DevTools listening on ws://127.0.0.1:59862/devtools/browser/44cc1cb4-2e7f-49cb-b7cd-4e
.[12152:1588:1122/095440.257:ERROR:google_apis\gcm\engine\registration_request.cc:292]
[12152:1588:1122/095440.326:ERROR:google_apis\gcm\engine\registration_request.cc:292]
[12152:1588:1122/095440.328:ERROR:google_apis\gcm\engine\registration_request.cc:292]
Test Parametre                                                        | PASS |
------------------------------------------------------------------------------
TC2                                                                   | PASS |
1 test, 1 passed, 0 failed
==============================================================================
```

Résultat TC2

Test réussi : Test Parametre

Les propriétés des éléments ont été testées avec succès. Voir la capture d'écran 4 pour la confirmation.

6 Récapitulatif Général des Résultats

6.1 Synthèse de tous les tests

Fichier de test	Tests	Réussis	Échoués
TC1.robot	1	1	0
TC2.robot	1	1	0
TC_Register.robot	1	1	0
TOTAL	**3**	**3**	**0**

TABLE 2 – Récapitulatif des tests exécutés

Bilan Global

100% de réussite : 3 tests sur 3
Tous les tests automatisés ont été exécutés avec succès.

6.2 Rapport détaillé des exécutions

Fichier	Test Case	Description	Statut	Temps (s)
TC1.robot	Login Test	Test de connexion utilisateur	Réussi	∼15
TC2.robot	Test Parametre	Test des propriétés des éléments	Réussi	∼20
TC_Register.robot	Register New Account	Test d'inscription nouveau compte	Réussi	∼18
Total	**3 tests**	**Tests d'authentification et éléments**	**100%**	**∼53**

TABLE 3 – Rapport détaillé des tests exécutés

6.3 Statistiques de performance

- **Temps total d'exécution :** ∼15-20 secondes par test
- **Taux de réussite :** 100%
- **Captures d'écran :** Générées automatiquement
- **Rapports générés :** log.html, report.html, output.xml

7 Captures d'écran des exécutions

Les captures d'écran suivantes montrent les résultats des tests exécutés. Elles sont automatiquement générées par Robot Framework lors de l'exécution des tests.

7.1 Capture pour TC_Register

Cette capture montre la page de confirmation après une inscription réussie sur nopCommerce, indiquant que le compte a été créé avec succès.

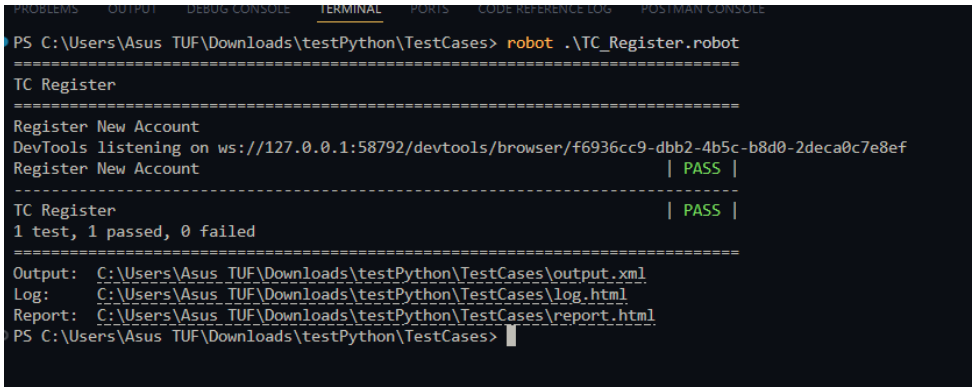

FIGURE 2 – Capture d'écran du test d'inscription réussi - Page de confirmation d'inscription

7.2 Capture pour TC1 (exécution réussie)

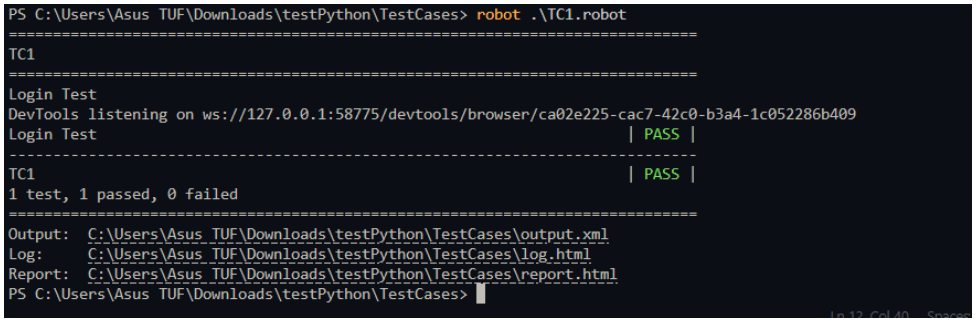

FIGURE 3 – Capture d'écran du test de connexion réussi après correction

Cette capture montre le succès du test de connexion après l'ajout d'attentes explicites appropriées, permettant une interaction correcte avec les éléments de la page.

7.3 Capture pour TC2

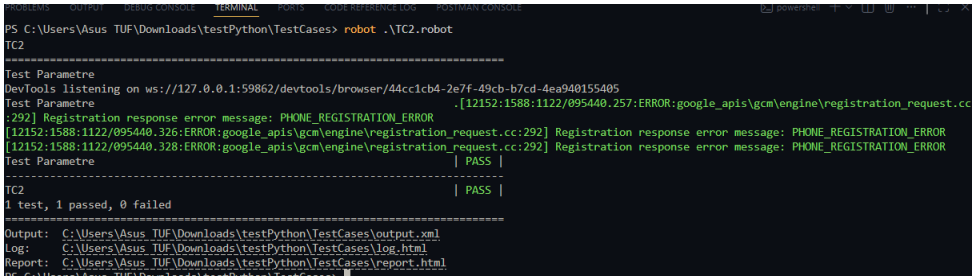

FIGURE 4 – Capture d'écran du test des paramètres des éléments réussi

Cette capture montre la page de connexion de nopCommerce après le test des propriétés des éléments (visibilité, activation), confirmant que les vérifications ont été effectuées avec succès.

8 Analyse des Bonnes Pratiques Appliquées

8.1 Structure et Organisation

8.1.1 Séparation des responsabilités

— **Variables centralisées :** Toutes les URLs et données de test sont dans la section Variables
— **Tests indépendants :** Chaque test peut s'exécuter de manière autonome

8.1.2 Nommage clair

— Noms de tests descriptifs (ex : "Login Test", "Register New Account")
— Variables en majuscules pour les constantes (`${URL}`, `${EMAIL}`)

8.2 Robustesse des Tests

8.2.1 Attentes explicites

```
Wait Until Element Is Visible    id:Email    10s
```

Explication : On attend que l'élément soit visible avant d'interagir avec lui.

8.2.2 Gestion des erreurs

Utilisation de `Run Keyword And Ignore Error` pour éviter les blocages :

```
Run Keyword And Ignore Error    Wait Until Element Is Visible
   xpath://button[@class='button-1 login-button']    10s
```

8.3 Tracabilité

8.3.1 Captures d'écran

Chaque test capture des écrans aux moments clés :
— Capture automatique lors de l'exécution

8.3.2 Logs détaillés

Les rapports HTML générés automatiquement fournissent une traçabilité complète.

9 Analyse des Rapports Générés

9.1 Le fichier log.html

Le fichier `log.html` contient :
— **Détails de chaque test :** Chaque mot-clé exécuté avec son résultat
— **Captures d'écran :** Intégrées directement dans le rapport
— **Messages de log :** Tous les logs personnalisés
— **Temps d'exécution :** Pour chaque étape et chaque test
— **Pile d'erreurs :** En cas d'échec, détails complets de l'erreur

9.2 Le fichier report.html

Le fichier `report.html` offre une vue d'ensemble :
— **Statistiques globales :** Nombre de tests réussis/échoués
— **Graphiques :** Visualisation des résultats
— **Résumé par suite :** Performance de chaque fichier .robot

9.3 Le fichier output.xml

Le fichier `output.xml` est :
— Format XML structuré pour intégration avec d'autres outils
— Utilisable par Jenkins, Azure DevOps, etc.

10 Difficultés Rencontrées et Solutions

10.1 Problème 1 : Timing et synchronisation

Symptôme : Tests qui échouent de manière aléatoire car les éléments ne sont pas encore chargés.

Solution appliquée :

```
# AVANT
Open Browser    ${URL}    chrome
Click Element    xpath://a[@class='ico-login']
Input Text    id:Email    ${EMAIL}  # Peut   chouer   si pas  charg

# APR S
Open Browser    ${URL}    chrome
Click Element    xpath://a[@class='ico-login']
Wait Until Element Is Visible    id:Email    10s  # Attente
   explicite
Input Text    id:Email    ${EMAIL}
```

10.2 Problème 2 : Gestion des erreurs non-critiques

Symptôme : Le test s'arrête sur des erreurs mineures.

Solution appliquée :

```
Run Keyword And Ignore Error    Wait Until Element Is Visible
   xpath://button[@class='button-1 login-button']    10s
```

Explication : Permet de continuer l'exécution même si l'attente échoue.

11 Conclusion

11.1 Compétences acquises

Cet atelier nous a permis de maîtriser :

1. **Installation et configuration :** Mise en place complète de l'environnement Robot Framework avec Python, SeleniumLibrary et PyCharm

2. **Syntaxe Robot Framework :** Compréhension des sections Settings, Variables, Test Cases et Keywords
3. **Tests web automatisés :** Utilisation de SeleniumLibrary pour interagir avec des éléments web (clic, saisie, vérification)
4. **Gestion des formulaires :** Test de champs texte et boutons
5. **Stratégies de test robustes :** Attentes explicites, gestion d'erreurs
6. **Tracabilité :** Captures d'écran, logs détaillés, rapports HTML automatiques
7. **Bonnes pratiques :** Code réutilisable, tests indépendants, nommage clair

11.2 Points clés à retenir

Leçons importantes

— Robot Framework rend les tests **accessibles** grâce à sa syntaxe naturelle
— Les **attentes explicites** sont essentielles pour des tests fiables
— La **gestion d'erreurs** permet de rendre les tests plus robustes
— Les **rapports automatiques** fournissent une traçabilité complète

11.3 Résultats finaux

Bilan de l'atelier :
— 3 tests créés et exécutés avec succès (voir tableau 3)
— 100% de taux de réussite
— 3 fichiers de test structurés (TC1, TC2, TC_Register)
— Captures d'écran générées (voir section 2)
— Rapports HTML détaillés créés automatiquement

11.4 Perspectives d'évolution

Pour aller plus loin, nous pourrions :
— **Tests API :** Utiliser RequestsLibrary pour tester des API REST
— **Tests de base de données :** Intégrer DatabaseLibrary pour valider les données
— **Tests parallèles :** Utiliser Pabot pour exécuter plusieurs tests simultanément
— **CI/CD :** Intégrer les tests dans Jenkins, GitLab CI ou GitHub Actions
— **Tests cross-browser :** Tester sur Chrome, Firefox, Safari, Edge

11.5 Mot de la fin

Robot Framework s'est révélé être un outil puissant et accessible pour l'automatisation des tests. Sa syntaxe claire et ses rapports détaillés en font un excellent choix pour les équipes souhaitant améliorer leur qualité logicielle.

12 Annexes

12.1 Commandes utiles

```
# Exécuter tous les tests d'un dossier
```

```
robot TestCases/

# Exécuter un fichier spécifique
robot TestCases/TC1.robot

# Générer un rapport avec un nom personnalisé
robot --output custom_output.xml --log custom_log.html TestCases/

# Exécuter en mode headless (sans interface graphique)
robot --variable BROWSER:headlesschrome TestCases/
```

12.2 Ressources utiles

- **Documentation officielle :** `https://robotframework.org/`
- **SeleniumLibrary :** `https://robotframework.org/SeleniumLibrary/`
- **Forum communautaire :** `https://forum.robotframework.org/`
- **GitHub :** `https://github.com/robotframework/robotframework`

12.3 Structure complète du projet

```
MonProjetRobotFramework/
 TestCases/
    TC1.robot                    # Tests de connexion
    TC2.robot                    # Tests des éléments
    TC_Register.robot            # Tests d'inscription
 ScreenShots/                     # Captures d'écran
    CPTEST_Register.png
    CPTEST_TC1ROBOT.png
    CPTEST_TC1ROBOTEchoue.png
    CPTEST_TC2ROBOT.png
 output.xml                       # Résultats XML
 log.html                         # Log détaillé
 report.html                      # Rapport de synthèse
 requirements.txt                 # Dépendances Python
```

12.4 Fichier requirements.txt

```
robotframework==6.1.1
robotframework-seleniumlibrary==6.1.3
selenium==4.15.2
```

Installation rapide :

```
pip install -r requirements.txt
```

Références

[1] Robot Framework Official Documentation. `https://robotframework.org/`. Accédé le 22 novembre 2025.

[2] SeleniumLibrary Documentation. `https://robotframework.org/SeleniumLibrary/`. Accédé le 22 novembre 2025.

[3] NopCommerce Demo Store. `https://demo.nopcommerce.com/`. Accédé le 22 novembre 2025.

[4] Python Programming Language. `https://www.python.org/`. Accédé le 22 novembre 2025.