Guide d'installation, d'intégration et de développement des tests automatisés avec Robot Framework, Cucumber, Page Object Model et Jenkins

Zied Hannachi

Sommaire

---

1. Introduction

Ce guide a pour objectif de détailler le processus d'installation de Robot Framework et Cucumber, la configuration de Visual Studio Code pour les tests automatisés, et l'intégration de ces outils avec Jenkins pour l'intégration continue. Il inclut également des exemples d'implémentation du design pattern Page Object Model (POM) pour structurer le code de test de manière plus maintenable et modulable.

2. Exemple de User Story et rédaction des cas de test manuels

2.1 Exemple de User Story

En tant qu'utilisateur connecté, je souhaite pouvoir me déconnecter de l'application afin de sécuriser mon compte lorsque je n'utilise plus le système.

2.2 Cas de test manuel

Cas de Test 1 : Connexion avec des identifiants valides

- **Étape 1** : Lancer l'application.
- **Étape 2** : Saisir un nom d'utilisateur valide.
- **Étape 3** : Saisir un mot de passe valide.
- **Étape 4** : Cliquer sur "Se connecter".
- **Résultat attendu** : L'utilisateur est connecté et redirigé vers la page d'accueil.

Cas de Test 2 : Déconnexion de l'utilisateur

- **Étape 1** : Cliquer sur le menu utilisateur.
- **Étape 2** : Sélectionner l'option "Déconnexion".
- **Résultat attendu** : L'utilisateur est redirigé vers la page de connexion.

Cas de Test 3 : Connexion avec des identifiants invalides

- **Étape 1** : Lancer l'application.
- **Étape 2** : Saisir un nom d'utilisateur invalide.
- **Étape 3** : Saisir un mot de passe invalide.
- **Étape 4** : Cliquer sur "Se connecter".
- **Résultat attendu** : Un message d'erreur indiquant que les identifiants sont incorrects est affiché.

3. Installation des outils

3.1 Installation de Visual Studio Code

1. Télécharger Visual Studio Code depuis le site officiel : https://code.visualstudio.com/.
2. Suivre les instructions d'installation pour votre système d'exploitation.

3.2 Installation de Robot Framework

1. Installer Python (si ce n'est pas déjà fait) :

```
sudo apt-get install python3
```

2. Installer Robot Framework avec la commande :

```
pip install robotframework
```

3. Installer SeleniumLibrary pour Robot Framework:

```
pip install robotframework-seleniumlibrary
```

4. Installer l'extension Visual Studio Code "Robot Framework Language Server".

3.3 Installation de Cucumber

1. Installer Java (JDK) :

```
sudo apt-get install openjdk-11-jdk
```

2. Installer Maven :

```
sudo apt-get install maven
```

3. Installer l'extension "Cucumber (Gherkin) Full Support" pour Visual Studio Code.

4. Design Pattern: Page Object Model (POM)

4.1 Présentation du POM

Le Page Object Model est un design pattern pour organiser le code des tests automatisés. Il consiste à créer une classe pour chaque page de l'application, contenant les localisations des éléments de la page ainsi que les méthodes pour interagir avec ces éléments.

4.2 Implémentation du POM avec des exemples détaillés

4.2.1 Exemple de structure du projet

```
automation_project/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   ├── pages/
│   │   │   │   ├── LoginPage.java
│   │   │   │   └── HomePage.java
│   │   │   ├── steps/
│   │   │   │   └── LoginSteps.java
│   │   │   └── utils/
│   │   │       └── BrowserFactory.java
├── robot_tests/
│   ├── login_tests.robot
├── cucumber_tests/
│   ├── login.feature
└── pom.xml
```

4.2.2 Développement des classes de pages

LoginPage.java

```java
package pages;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;

public class LoginPage {
    WebDriver driver;

    // Localisation des éléments
    By usernameField = By.id("username");
    By passwordField = By.id("password");
    By loginButton = By.id("login-button");

    // Constructeur
    public LoginPage(WebDriver driver) {
        this.driver = driver;
    }

    // Méthodes pour interagir avec la page
    public void enterUsername(String username) {
        driver.findElement(usernameField).sendKeys(username);
    }

    public void enterPassword(String password) {
        driver.findElement(passwordField).sendKeys(password);
    }

    public void clickLoginButton() {
        driver.findElement(loginButton).click();
    }
}
```

HomePage.java

```java
package pages;

import org.openqa.selenium.WebDriver;

public class HomePage {
    WebDriver driver;

    // Constructeur
    public HomePage(WebDriver driver) {
        this.driver = driver;
    }

    // Méthode pour vérifier si l'utilisateur est sur la page d'accueil
    public boolean isUserOnHomePage() {
        return driver.getTitle().contains("Accueil");
    }
}
```

4.2.3 Développement des classes de tests

LoginSteps.java

```java
package steps;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import pages.LoginPage;
import pages.HomePage;
import io.cucumber.java.en.*;

public class LoginSteps {
    WebDriver driver;
    LoginPage loginPage;
    HomePage homePage;

    @Given("The user is on the login page")
    public void userOnLoginPage() {
        driver = new ChromeDriver();
        driver.get("https://example.com/login");
        loginPage = new LoginPage(driver);
    }

    @When("The user enters a valid username and password")
    public void userEntersValidCredentials() {
        loginPage.enterUsername("valid_user");
        loginPage.enterPassword("valid_password");
        loginPage.clickLoginButton();
    }

    @Then("The user should be redirected to the homepage")
    public void userIsRedirectedToHomepage() {
        homePage = new HomePage(driver);
        assert homePage.isUserOnHomePage();
        driver.quit();
    }

    @When("The user enters an invalid username or password")
    public void userEntersInvalidCredentials() {
        loginPage.enterUsername("invalid_user");
        loginPage.enterPassword("invalid_password");
        loginPage.clickLoginButton();
    }

    @Then("An error message should be displayed")
    public void errorMessageIsDisplayed() {
        // Code pour vérifier que le message d'erreur est affiché
        driver.quit();
    }
}
```

5. Développement des cas de test automatisés

Cas de test 1 : Connexion avec des identifiants valides

- Implémentation du test dans le fichier `login.feature`.

Cas de test 2 : Déconnexion de l'utilisateur

- Ajouter une nouvelle méthode dans la classe `HomePage` pour gérer la déconnexion.

Cas de test 3 : Connexion avec des identifiants invalides

- Vérification de l'affichage du message d'erreur.

6. Intégration continue avec Jenkins

6.1 Installation et configuration de Jenkins

1. **Télécharger et installer Jenkins** :
   - Sur Ubuntu :

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/source
sudo apt update
sudo apt install jenkins
```

   - Pour Windows et Mac, télécharger l'installateur depuis https://www.jenkins.io/download/ .
2. **Démarrer Jenkins** :

```
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

3. **Accéder à Jenkins** :
   - Ouvrir le navigateur et accéder à `http://localhost:8080` .
4. **Déverrouiller Jenkins** :
   - Le mot de passe initial est dans le fichier `/var/lib/jenkins/secrets/initialAdminPassword`.
5. **Installer les plugins recommandés** :
   - Pendant l'installation initiale, choisir "Install suggested plugins".

6.2 Configuration de Jenkins pour exécuter les tests

1. **Installer les plugins nécessaires** :

- Installer le plugin "Robot Framework Plugin" pour les rapports Robot Framework.
- Installer le plugin "Cucumber Reports" pour les rapports Cucumber.
- Installer "Git Plugin" pour la gestion de code source.
- Installer "Maven Integration Plugin" si les tests utilisent Maven.

2. **Créer un nouveau projet Jenkins** :
   - Cliquer sur "New Item" > "Freestyle project".
   - Nommer le projet et sélectionner "Freestyle project".
3. **Configurer la gestion de code source** :
   - Sélectionner "Git" et entrer l'URL du dépôt.
   - Configurer les identifiants si nécessaire.
4. **Ajouter une étape de build pour exécuter les tests** :
   - Ajouter un "Build Step" > "Execute Shell" (ou "Windows Batch Command" si vous utilisez Windows).
   - Commande pour exécuter les tests Robot Framework :

```
robot -d results/ tests/
```

   - Commande pour exécuter les tests Cucumber avec Maven :

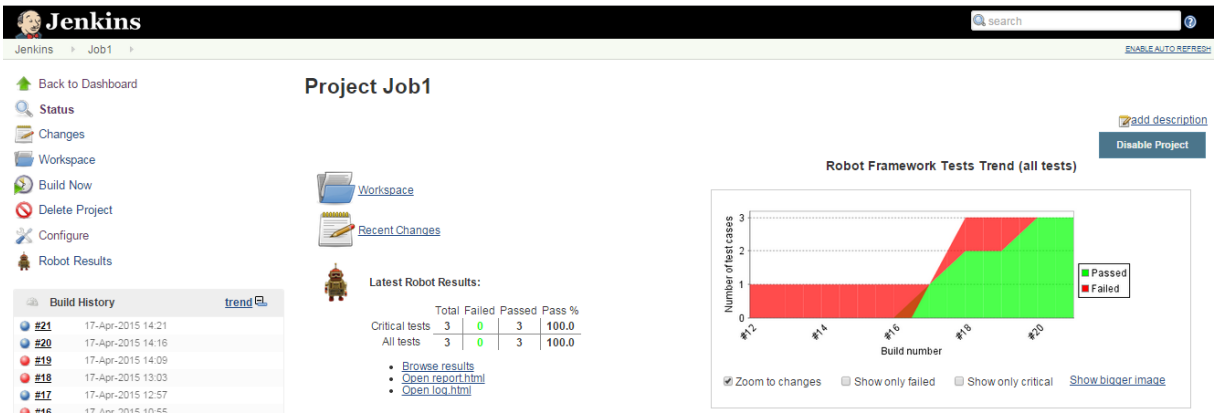
```
mvn test
```

5. **Publier les rapports** :
   - Pour les rapports Robot Framework, ajouter "Publish Robot Framework test results".
   - Pour les rapports Cucumber, ajouter "Publish Cucumber test result report" et spécifier le chemin vers les fichiers de résultats (par exemple, `target/cucumber-reports/*.json`).

7. Création d'un dashboard dans Jenkins

1. **Installer le plugin "Dashboard View"** :
   - Aller dans "Manage Jenkins" > "Manage Plugins" > "Available" et chercher "Dashboard View".
   - Installer le plugin.
2. **Créer un tableau de bord Jenkins** :
   - Aller à l'écran d'accueil Jenkins, cliquer sur "New View".
   - Choisir "Dashboard" comme type de vue.
   - Configurer les widgets à afficher sur le tableau de bord :
     - Ajouter des widgets pour les "Test Statistics Charts".
     - Ajouter "Build History" pour visualiser les builds précédents.
     - Utiliser "Test Trend Chart" pour afficher les tendances des résultats de tests au fil du temps.

Exemple de Dashboard Jenkins :



8. Visualisation des résultats avec graphiques et diagrammes

8.1 Génération de rapports Cucumber

1. **Configurer le rapport Cucumber** :
   - Ajouter l'étape "Publish Cucumber test result report" dans les configurations Jenkins.
   - Spécifier le chemin des fichiers de rapport JSON (par exemple, `target/cucumber-reports/*.json`).
2. **Afficher les tendances des résultats de tests** :
   - Utiliser les options "Trend Chart" pour visualiser les succès/échecs des tests.

8.2 Génération de rapports Robot Framework

1. **Configurer le rapport Robot Framework** :
   - Ajouter "Publish Robot Framework test results" dans les configurations Jenkins.
   - Spécifier le dossier où les rapports sont générés (par exemple, `results/`).
2. **Visualisation avancée avec le plugin "Robot Dashboard"** :
   - Installer le plugin "Robot Dashboard" pour avoir un tableau de bord dédié aux rapports Robot Framework.

8.3 Création de graphiques avec le plugin "Plot" de Jenkins

1. **Installer le plugin "Plot"** :
   - Aller dans "Manage Jenkins" > "Manage Plugins" et chercher "Plot Plugin".
   - Installer le plugin.
2. **Configurer le plugin pour créer des graphiques personnalisés** :
   - Dans les configurations du projet, ajouter une "Post-build Action" > "Plot build data".
   - Spécifier le fichier de données (ex. fichier CSV) à partir duquel les graphiques seront générés.
   - Configurer les axes et les types de graphiques (ligne, barre, etc.).