

TP : Automatisation des tests avec Selenium WebDriver

L'automatisation des tests est devenue une composante essentielle du développement logiciel moderne. Elle permet d'accélérer les processus de test, d'améliorer la couverture fonctionnelle, et de réduire les erreurs humaines dans les processus de vérification. Dans ce contexte, **Selenium WebDriver** est l'un des outils les plus populaires pour automatiser les tests d'applications web.

Catégorie	Commande Selenium	Description	Exemple
Initialisation	<code>WebDriver driver = new ChromeDriver();</code>	Initialise le navigateur Chrome.	<code>WebDriver driver = new ChromeDriver();</code>
Navigation	<code>driver.get(url);</code>	Ouvre une URL dans le navigateur.	<code>driver.get("https://www.google.com");</code>
	<code>driver.navigate().to(url);</code>	Navigue vers une URL spécifique.	<code>driver.navigate().to("https://www.example.com");</code>
	<code>driver.navigate().back();</code>	Revient à la page précédente.	<code>driver.navigate().back();</code>
	<code>driver.navigate().forward();</code>	Avance vers la page suivante.	<code>driver.navigate().forward();</code>
	<code>driver.navigate().refresh();</code>	Recharge la page actuelle.	<code>driver.navigate().refresh();</code>
Recherche d'éléments	<code>driver.findElement(By.id("id"));</code>	Trouve un élément en utilisant son ID.	<code>WebElement element = driver.findElement(By.id("username"));</code>
	<code>driver.findElement(By.name("name"));</code>	Trouve un élément par son attribut name.	<code>WebElement element = driver.findElement(By.name("password"));</code>

	<code>driver.findElement(By.className("class"));</code>	Trouve un élément par son attribut class.	<code>WebElement element = driver.findElement(By.className("button"));</code>
	<code>driver.findElement(By.cssSelector("selector"));</code>	Trouve un élément en utilisant un sélecteur CSS.	<code>WebElement element = driver.findElement(By.cssSelector(".login-form input[type='text']"));</code>
	<code>driver.findElement(By.xpath("xpath"));</code>	Trouve un élément en utilisant une expression XPath.	<code>WebElement element = driver.findElement(By.xpath("//input[@id='email']"));</code>
	<code>driver.findElements(By.tagName("tag"));</code>	Trouve une liste d'éléments avec une balise spécifique.	<code>List<WebElement> links = driver.findElements(By.tagName("a"));</code>
Interactions	<code>element.click();</code>	Clique sur un élément.	<code>element.click();</code>
	<code>element.sendKeys("text");</code>	Envoie du texte à un champ de formulaire.	<code>element.sendKeys("Selenium WebDriver");</code>
	<code>element.clear();</code>	Efface le contenu d'un champ de texte.	<code>element.clear();</code>
	<code>element.submit();</code>	Soumet un formulaire.	<code>element.submit();</code>
Gestion des fenêtres	<code>driver.getTitle();</code>	Récupère le titre de la page actuelle.	<code>String title = driver.getTitle();</code>

	<code>driver.getCurrentUrl();</code>	Récupère l'URL actuelle.	<code>String url = driver.getCurrentUrl();</code>
	<code>driver.getWindowHandle();</code>	Récupère l'ID de la fenêtre actuelle.	<code>String windowHandle = driver.getWindowHandle();</code>
	<code>driver.switchTo().window(handle);</code>	Bascule vers une fenêtre spécifique.	<code>driver.switchTo() .window(windo wHandle);</code>

Ce TP a pour but de familiariser les étudiants avec l'utilisation de Selenium WebDriver pour automatiser les tests fonctionnels d'une application web. Il introduit les concepts de base de l'automatisation, les éléments-clés pour interagir avec une interface utilisateur web, et les bonnes pratiques pour concevoir des scripts de test robustes.

Exercice 1 :

Objectif : Automatiser une recherche sur Google.

Étapes :

- Accédez à <https://www.google.com>.
- Saisissez une recherche dans la barre de recherche.
- Cliquez sur le bouton "Recherche Google".
- Vérifiez que les résultats s'affichent.

Exercice 2 : Test de formulaire

Objectif : Tester un formulaire de connexion (connexion.html)

Étapes :

- Accédez à la page.
- Remplissez les champs "username" et "password".
- Soumettez le formulaire.
- Validez l'apparition d'un message de succès ou d'erreur.

Exercice 3 : comparateur de prix

Créer une application Java qui automatise la recherche du prix d'un produit sur plusieurs sites de commerce électronique (par exemple, Amazon et eBay). Le programme doit afficher les résultats dans la console en comparant les prix récupérés.

Exercice 4 : Recherche d'entreprises pour PFE dans une région et un secteur

Créer un script Java qui automatise la recherche d'entreprises dans un secteur d'activité et une région spécifiques en utilisant le web scraping. Le programme permettra aux étudiants de trouver facilement des entreprises susceptibles de proposer des stages de fin d'études (PFE).