Guide Selenium

Etape 0 - Les prérequis

IDE :	IntelliJ IDEA, Eclipse...
Dépendances Maven :	selenium-java, selenium-chrome-driver selenium-edge-driver testng
fichiers nécessaires :	msedgedriver.exe, chromedriver.exe

Remarque:

Ce projet sera fait en Java. Si vous voulez utiliser Python, vous pouvez installer Selenium avec pip:

```
C:Users\Anass Soulimani> pip install selenium
```

Liens des drivers:

Chrome:

https://sites.google.com/chromium.org/driver/

Edge:

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

Firefox:

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

Safari:

https://webkit.org/blog/6900/webdriver-support-in-safari-10/

Installer JAVA

- Télécharger Java.(https://www.oracle.com/java/technologies/downloads/)
- Installer Java.
- Définir le chemin de l'environnement Java.(Control Panel -> System -> Advanced System Setting -> Environment Variables)
- Vérifier l'installation de Java.(java -version)

Etape 1 - Introduction

Qu'est ce que Selenium:

Selenium WebDriver est un framework web qui vous permet d'exécuter des tests multi-navigateurs. Cet outil est utilisé pour automatiser les tests d'applications Web pour vérifier qu'il fonctionnent correctement. Selenium WebDriver vous permet de choisir un langage de programmation de votre choix pour créer des scripts de test.

Architecture de Selenium:

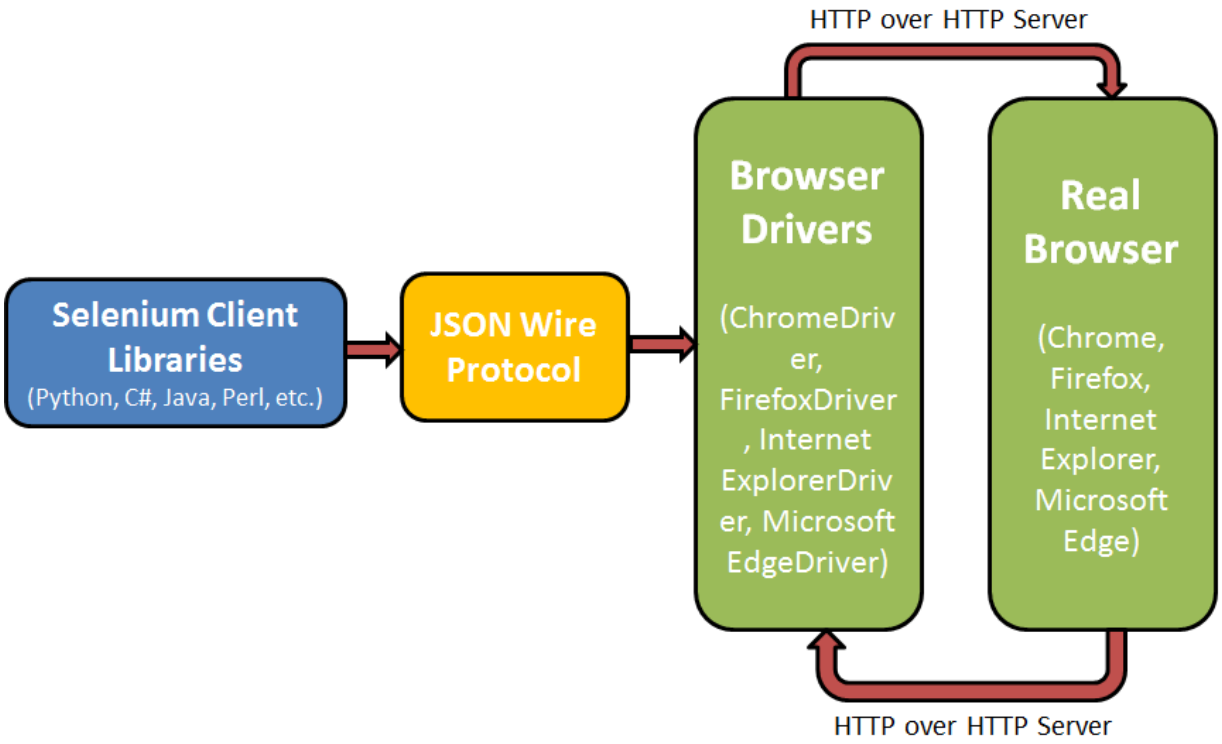

Figure 1: ArchitectureSelenium

Travail à faire:

Dans ce guide, nous verrons comment ouvrir votre navigateur préféré. Ensuite, nous essaierons de tester la commande "waits",Dans ce guide, nous verrons comment ouvrir votre navigateur préféré. Ensuite, nous essaierons de tester la commande "waits",localisation des éléments en HTML,les clics et la création d'une ActionChain.

Etape 2 - Initialisation

Configurer Selenium WebDriver avec IntelliJ:

Pour configurer IntelliJ, nous devons effectuer les tâches suivantes:
- Lancez l'IDE Intellij.
- Créer un nouveau projet.
- Créer deux nouveaux packages.(un pour tester Chrome et l'autre pour Edge).
- Créer deux nouvelles classes.
- Ajouter les dépendances au projet.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
↪   http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>Selenium_java</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>4.1.1</version>
        </dependency>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-chrome-driver</artifactId>
            <version>4.1.1</version>
        </dependency>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-edge-driver</artifactId>
            <version>4.1.1</version>
        </dependency>
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>7.4.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

Configuration de Chrome:

Définition du pilote:

```
System.setProperty("webdriver.chrome.driver", "C:\\chromedriver.exe");
```

Initialisation du WebDriver:

```
WebDriver driver = new ChromeDriver();
```

Configuration de Edge:

Définition du pilote:

```
System.setProperty("webdriver.edge.driver", "C:\\msedgedriver.exe");
```

Initialisation du WebDriver:

```
WebDriver driver = new EdgeDriver();
```

Suppression des cookies:

```
driver.manage().deleteAllCookies();
```

Ouverture de l'URL souhaitée

```
driver.get("https://ulring.github.io/");
```

Waits:

Les commandes wait sont indispensables pour l'exécution des tests Selenium. Ils aident à observer et à résoudre les problèmes qui peuvent survenir en raison de la variation du décalage temporel.

```
WebDriverWait wait_for_details = new WebDriverWait(driver, Duration.ofSeconds(20));
wait_for_details.until(ExpectedConditions.elementToBeClickable(By.tagName("details")));
```

Localisation et Clic:

Un clic est l'action utilisateur la plus fondamentale effectuée par toute personne accédant à Internet. Il permet à l'utilisateur de naviguer sur des pages Web ou d'effectuer des tâches particulières en interagissant avec des liens, des boutons et d'autres éléments Web. Mais d'abord, l'interaction avec une page Web nécessite que le pilote localise l'élément Web pour déclencher son événement.

```
driver.findElement(By.tagName("details")).click();
```

ActionChains:

Les ActionChains sont un moyen d'automatiser les interactions de bas niveau telles que les mouvements de la souris, les actions des boutons de la souris, les touches ou bien les interactions du menu contextuel...

```
actions.click(filter);
//On cherche des liens sur notre éditeur de texte préféré:VIM!
actions.sendKeys(filter, "vim");
WebDriverWait wait_for_table = new WebDriverWait(driver, Duration.ofSeconds(20));
wait_for_search.until(ExpectedConditions.elementToBeClickable(By.tagName("td")));
actions.perform();
```

N'oublions pas de fermer le navigateur à la fin

```
driver.close();
```

Etape 2 - TestNG

Note:

Ce guide a été réalisé à l'aide du pdf builder de 42. Il est donc juste de vérifier si le repo fonctionne toujours tout en apprenant comment tester dans Selenium.

Initialisation:

```
public String baseUrl = "https://github.com/42-AI/42ai_pdf_builder";
public String gitUrl;
String driverPath = "C:\\chromedriver.exe";
public WebDriver driver ;

@BeforeTest
public void launchBrowser() {
        System.setProperty("webdriver.chrome.driver", driverPath);
    driver = new ChromeDriver();
    driver.get(baseUrl);
}
```

Exctraction du HTTPS:

Il est facile d'extraire l'URL de la barre d'adresse et de vérifier si le lien marche toujours avec HttpURLConnection. Alors on va pimenter un peu les choses : Nous allons extraire le lien du clibboard !

```
@Test
public void getUrl(){
    WebDriverWait wait_for_code = new WebDriverWait(driver, Duration.ofSeconds(20));
    wait_for_code.until(ExpectedConditions.elementToBeClickable(By.tagName("get-repo")));
    driver.findElement(By.tagName("get-repo")).click();
    WebDriverWait wait_for_git_ssh = new WebDriverWait(driver, Duration.ofSeconds(20));
    wait_for_git_ssh.until(ExpectedConditions.elementToBeClickable(
                            By.tagName("clipboard-copy")));
    driver.findElement(By.tagName("clipboard-copy")).click();
    try {
        gitUrl = (String)
↪   Toolkit.getDefaultToolkit().getSystemClipboard().getData(DataFlavor.stringFlavor);
        Assert.assertNotNull(gitUrl);
    } catch (UnsupportedFlavorException e) {
        e.printStackTrace();
    } catch (IOException e) {
                e.printStackTrace();
    }
}
```

Verification du HTTPS:

Une fois que nous avons un HTTPS valide, nous utilisons la ligne de commande au lieu de JGIT pour voir si tout va bien.

```
@Test
public void verifyUrl(){
    Process p;
    try {
        p = Runtime.getRuntime().exec("cmd /c wsl git ls-remote " + gitUrl);
        p.waitFor();
        BufferedReader reader=new BufferedReader(new InputStreamReader(p.getInputStream()));
        Assert.assertNotNull(reader.readLine());
    } catch (IOException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

(Encore une fois)N'oublions pas de fermer le navigateur à la fin

```
@AfterTest
public void terminateBrowser(){
    driver.close();
}
```

Remarque:

S'il vous plaît essayez de respecter autant que possible checkstyle dans JAVA ou pycodestyle pour PYTHON lorsque vous travaillez avec du Selenium