Sélénium Web Driver

*Oppen Browser*

*Localisation des éléments et action*

[*DragAndDrope*](#_Méthode_DragAndDrop)

[*Technique login @parametres*](#_Paramétrage_Niveau_de)

[*Sélectionner une valeur dans DropDown*](#_Méthode_DragAndDrop)

[*Méthode DragAndDrop*](#_Méthode_DragAndDrop)

*Vérifier l’info-bulle*

*Page POM*

*Login page avec Page factory*

*Gestion des alertes*

*Capture du message d'alerte.*

*Gestion des alertes*

*Capture d’écran*

[*Gérer plusieurs fenêtres dans une application Web*](#_gérer_toutes_les)

[*Parametre niveau de test TESTNG*](#_Paramétrage_Niveau_de)

*Lire des données à partir d’un fichier Excel*

*Types d’attente dans selenium*

|  |  |  |
| --- | --- | --- |
| Une image contenant texte, Police, reçu, algèbre  Description générée automatiquement | **Position du fichier test** | |
| **Public class ExempleImplicitWait(){**  **public String url = "https://demoqa.com/dynamic-properties";**  **public WebDriver driver;**  **@BeforeMethod**  *//lance le setUp avant chaque méthode de test (test indépendant l'un de l'autre)* **public void setUp()** {  System.*out*.println("setUp"); --**1**🡺 **WebDriverManager.*chromedriver*().setup();**  *//met en place le bon driver  //Instancie le driver : pour utiliser le chrome --***2**🡺**driver = new ChromeDriver();**  *//Globalement : tant que la session n'est pas terminé (driver.quit)* **--3**🡺**driver.manage().timeouts().implicitlyWait(10, TimeUnit.*SECONDS*);**  *//On augmente la fenêtre --***4**🡺**driver.manage().window().maximize();**  --**5**🡺 **driver.get(url);***//on va sur le site* | **@BeforeMethod****public void setUp()**  **{**  **}** | **Préparer le navigateur et initialiser l’url** |
|  |
|  | @test  **Public void test001()**  **{**  **}** |
| **@AfterMethod**  **public void tearDown()**  **{**  driver.quit();  **}** | @AfterMethod  **public void tearDown(){**  **}** |
| 🡺driver.findelement(By.name(‘’username’’)).sendkeys(“fethi”)**;** | **Localiser un element et saisir un texte** | |
| 🡺 If (exptitle.equals(acttitle)) | **exception** | |
| 🡺 Assert.assertEquals(actualTitle,exceptedTitele)**;** | **Vérifie un titre** | |
| 🡺driver.quit()**;** |  | |
| 🡺boolean flag = driver.findElement(by.xpath(“xpathElement”)).isDisplayed()**;**  Assert.asserttrue(flag)**;** | **Vérifier la présence d’un element** | |
| 🡺WebElement element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("visibleAfter")))**;**  Assert.assertTrue(element.isDisplayed())**;** | **Attendre l’apparition d’un element et Vérifier la présence d’un element** | |
| **Localisation des éléments et action** | | |
|  |  | |

|  |  |  |
| --- | --- | --- |
| **DragAndDrope** | |  |
| **@Test** *//a compliter et à couriger* public void TestDragandDropDropShoulsBeInAmount() {  **String** **xpathsource1** = "//a[@class='button button-orange' and ./parent::li[@id='credit']]";  **String** **xpathDestination1** = "//li[@class='placeholder']/parent::div";  **System.*out*.println("Le test");** 1🡺  **Actions action = new Actions(driver);**  ----------------------------------------------------------------2🡺  **WebDriverWait** wait = new WebDriverWait (driver, Duration.*ofSeconds*(20));  -------------------------------------------------------------------------------  3🡺  Position source element /  **WebElement sourceBank** =  **wait.until**(ExpectedConditions.*elementToBeClickable*(**By.*xpath***(xpathsource1)));  -------------------------------------------------------------------------------- 4🡺  destination element /  **WebElement destinationBank** =  **wait.until**(**ExpectedConditions.presenceOfElementLocated**(**By.xpath**(xpathDestination1)));  5🡺  Action/  **action.dragAndDrop(sourceBank,destinationBank).pause(Duration.ofSeconds(2)).build().perform();**  *//cela exécute le drag and drop  //action.dragAndDrop(sourceBank, destinationBank)*  *.pause(Duration.ofSeconds(2)).build().perform();  //Assertion \*\*Vérifie qu'on a bien le texte Bank écrit*  *---------------------------------------------------------------------*  *Vérification de l’element* 🡺String textActual = **driver.findElement**(**By.*xpath***(xpathDestination1)).**getText();** 🡺**Assert.assertEquals**(textActual, "-5000"); | |  |
| **Technique login @parametres** | |  |
| 31  21 | | [Clic ici avec la touche ctrl pour voir la vidéo](cours%20video/TestNG%20Parameters%20Annotation%20_%20Parameterizing%20from%20TestNG%20xml%20file%20_%20TestNG%20Parameterization.mp4) |
| 1 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sélectionner une valeur dans DropDown** | | | |
| *Importez-le package* ***« Select ».*** | import org.openqa.selenium.support.ui.Select; | | |
| *Déclarez l'élément déroulant comme instance de la classe Select* | Select drpCountry = new Select(driver.findElement(By.name("country"))); | | |
| *Execution* | drpCountry.selectByVisibleText("ANTARCTICA"); | | |
| drpCountry.selectByValue("234"); | | |
| drpCountry.selectByIndex(0); | | |
| drpCountry.deselectAll(); | | |
| *Plusieurs sélections à la fois.* |  | | |
| Vérifier la possibilité de faire plusieurs sélections | if (drpCountry.isMultiple())  {  //do something here  } | | |
| **Méthode DragAndDrop** | | | |
| *Élément qui doit être déplacé.* | WebElement From=driver.findElement(By.xpath("//\*[@id='credit2']/a")); | | |
| *Élément sur lequel il faut déposer.* | WebElement To=driver.findElement(By.xpath("//\*[@id='bank']/li")); | | |
| *Utilisation de la classe Action pour le glisser-déposer.* | Actions act=new Actions(driver); | | |
| *Glissé et déposé.* | act.dragAndDrop(From, To).build().perform(); | | |
| *Vérification du parfait ! message.* | if(driver.findElement(By.xpath("//a[contains(text(),'Perfect')]")).isDisplayed()) {  System.out.println("Parfait affiché !!!)} | | |
| **Vérifier l’info-bulle** | | | |
| ***Option 1 :*** |  | | |
| ***Option 2:***  *Méthode d'implémentation des info-bulles « jQuery Tools Tooltip »* |  | | |
|  | | | |
| *rgrgr* | **Page POM**  **package pages;**  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  public class Guru99Login {  WebDriver driver;  By user99GuruName = By.name("uid");  By password99Guru = By.name("password");  By login = By.name("btnLogin");  By titleText =By.className("barone");  **------------------------------------------------------------**  **public Guru99Login(WebDriver driver){**  **this.driver = driver;**  **}**  **------------------------------------------------------------**  **//Set user name in textbox**  **public void setUserName(String strUserName){ driver.findElement(user99GuruName).sendKeys(strUserName);**  **}**  **------------------------------------------------------------**  **//Set password in password textbox**  public void setPassword(String strPassword){ driver.findElement(password99Guru).sendKeys(strPassword);  }  **------------------------------------------------------------**  **//Click on login button**  public void clickLogin(){  driver.findElement(login).click();  }  **//Obtenir le titre de la page de connexion**  public String getLoginTitle(){  **return** driver.findElement(titleText).getText();  }  **----------------------Constructeur--------------------------**  /\*\*  \* Cette méthode POM sera exposée dans le scénario de test pour se connecter à l'application  \* @param strUserName  \* @param strPasword  \* @return  \*/  **public void loginToGuru99(String strUserName,String strPasword){**  //Remplissez le nom d'utilisateur  this.setUserName(strUserName);  //Remplissez password  this.setPassword(strPasword);  //Click Login button  this.clickLogin();  }  } | |
|  | | | |
|  | | | |
| ***Login page avec Page factory*** | | | |
|  | | | |
| Code page POM avec page factory | package PageFactory;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.support.FindBy;  import org.openqa.selenium.support.PageFactory;  public class Guru99Login {  /\*\*  \* All WebElements are identified by @FindBy annotation  \*/  WebDriver driver;  @FindBy(name="uid")  WebElement user99GuruName;  @FindBy(name="password")  WebElement password99Guru;  @FindBy(className="barone")  WebElement titleText;  @FindBy(name="btnLogin")  WebElement login;  public Guru99Login(WebDriver driver){  this.driver = driver;  // Cette méthode initElements créera tous les WebElements  **PageFactory.initElements(driver, this);**  }  //Set user name in textbox  public void setUserName(String strUserName){  user99GuruName.sendKeys(strUserName);  }  //Set password in password textbox  public void setPassword(String strPassword){  password99Guru.sendKeys(strPassword);  }  //Click on login button  public void clickLogin(){  login.click();  }  //Get the title of Login Page  public String getLoginTitle(){  return titleText.getText();  }  /\*\*  \* This POM method will be exposed in test case to login in the application  \* @param strUserName  \* @param strPasword  \* @return  \*/  public void loginToGuru99**(String strUserName,String strPasword){**  //Fill user name  this.setUserName(strUserName);  //Fill password  this.setPassword(strPasword);  //Click Login button  this.clickLogin();  }} | | |
| Test case avec le concept page factory | package test;  import java.util.concurrent.TimeUnit;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;  import org.testng.Assert;  import org.testng.annotations.BeforeTest;  import org.testng.annotations.Test;  import PageFactory.Guru99HomePage;  import PageFactory.Guru99Login;  public class Test99GuruLoginWithPageFactory {  String driverPath = "C:\\geckodriver.exe";    WebDriver driver;  Guru99Login objLogin;  Guru99HomePage objHomePage;    **@BeforeTest**  public void setup(){  System.setProperty("webdriver.gecko.driver", driverPath);  driver = new FirefoxDriver();  driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);  driver.get("http://demo.guru99.com/V4/");  }  /\*\*  \* Ce test va sur http://demo.guru99.com/V4/  \* Vérifiez le titre de la page de connexion en tant que banque guru99  \* Connectez-vous à l'application  \* Vérifiez la page d'accueil à l'aide du message du tableau de bord  \*/  **@Test(priority=0)**  public void test\_Home\_Page\_Appear\_Correct(){  **//Create Login Page object**  objLogin = new Guru99Login(driver);    **//Verify login page title**  String loginPageTitle = objLogin.getLoginTitle(); Assert.assertTrue(loginPageTitle.toLowerCase().contains("guru99 bank"));  **//login to application**  objLogin.loginToGuru99("mgr123", "mgr!23");  **// go the next page**  **objHomePage = new Guru99HomePage(driver);**  **//Verify home page** Assert.assertTrue(objHomePage.getHomePageDashboardUserName().toLowerCase().contains("manger id : mgr123"));  }  } | | |
| Gestion des alertes08/01/2024 | | |
| **void dismiss()** // Cliquer sur le bouton 'Annuler' de l'alerte. | | **driver.switchTo().alert().dismiss();** |
| **void accept()** // Cliquer sur le bouton 'OK' de l'alerte. | | **driver.switchTo().alert().accept();** |
| **String getText()** // Pour capturer le message d'alerte. | | **driver.switchTo().alert().getText();** |
| **void sendKeys(String stringToSend)** // Pour envoyer des données à la boîte d'alerte. | | **driver.switchTo().alert().sendKeys("Texte");** |
| Exemple de programme | | **// Gestion des messages d'alerte**  **driver.findElement(By.name("cusid")).sendKeys("53920");**  **driver.findElement(By.name("submit")).submit();**    **// Passage à l'alerte**  **Alerte alerte = driver.switchTo().alert();**  **// Capture du message d'alerte.**  **String alertMessage= driver.switchTo().alert().getText();**    **// Affichage du message d'alerte**  **System.out.println(alertMessage);**  **Thread.sleep(5000);**    **// Acceptation de l'alerte**  **alert.accepter();** |

|  |  |  |
| --- | --- | --- |
| Gérer plusieurs fenêtres dans une application Web | | |
| gérer toutes les fenêtres ouvertes | | Driver.getWindowHandles(); |
| gérer la fenêtre principale | | Driver.getWindowHandles();08/01/2024 |
| Exemple programme | **Connection à un site et cliquer sur un bouton pour afficher la première fenêtre**    String MainWindow=driver.getWindowHandle();(page principale)  **// Pour récupérer id de toutes les nouvelles fenêtres ouvertes.**    **Set<String> s1=driver.getWindowHandles();**  **Iterator<String> i1=s1.iterator();**    while(i1.hasNext())  {  String ChildWindow=i1.next();  if(!MainWindow.equalsIgnoreCase(ChildWindow))  {**// Switching to Child window**  **driver.switchTo().window(ChildWindow);**  **driver.findElement(By.name("emailid"))**  **.sendKeys("gaurav.3n@gmail.com");**  driver.findElement(By.name("btnLogin")).click();  **// Closing the Child Window.**  **driver.close();**  }  }  **// Switching to Parent window i.e Main Window.**  **driver.switchTo().window(MainWindow);**  }  } | |

|  |  |  |
| --- | --- | --- |
| **Fournisseur de données TestNG.XML** | | |
|  |  | |
| Paramétrage Niveau de test TestNG.xml | <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">  <suite name="TestSuite" thread-count=" 3" >  <parameter name="author" value="Guru99" />  <parameter name="searchKey" value="India" />  <test name="testGuru">  <parameter name="searchKey" value="UK " />  <classes>  <class name="parameters.ParameterWithTestNGXML">  </class>  </classes>  </test>  </suite> | |
| Exemple de programme | **@Test**  **@Parameters({"author","searchKey"})**  public void testParameterWithXML( @Optional("Abc") **String author,String searchKey**) throws InterruptedException{  WebElement searchText = driver.findElement(By.name("q"));  //Recherche de texte dans la zone de texte Google  searchText.sendKeys(**searchKey**);  System.out.println("Bienvenue ->"+author+" Votre clé de recherche est->"+**searchKey**);  System.out.println("Le thread va dormir maintenant");  Thread.sleep(3000);  System.out.println("Valeur dans la zone de recherche Google = "+searchText.getAttribute("value") +" ::: Valeur donnée par input = "+**searchKey**);  //vérification de la valeur dans le champ de recherche Google  AssertJUnit.assertTrue(searchText.getAttribute("value").equalsIgnoreCase(**searchKey**)); | |
| **Problème n° 1** | La valeur du paramètre dans testng.xml ne peut pas être transtypée dans le paramètre de la méthode de test correspondante, cela générera une erreur.  **string** ==**x**=🡺 **int** | |
| **Problème n°2** | Vos @Parameters n'ont pas de valeur correspondante dans testing.xml. Vous pouvez résoudre cette situation en ajoutant **l'annotation @optional** dans le paramètre correspondant dans la méthode de test. | |
| **Problème n°3** | vous souhaitez tester plusieurs valeurs du même paramètre à l'aide de Testng.xml,La réponse simple est que cela n’est pas possible ! Vous pouvez avoir plusieurs paramètres différents, mais chaque paramètre ne peut avoir qu'une seule valeur. Cela permet d'éviter le codage en dur des valeurs dans le script. Cela rend le code réutilisable. Considérez-le comme des fichiers de configuration pour votre script. Si vous souhaitez utiliser plusieurs valeurs pour un paramètre, **utilisez DataProviders** | |
| **Dataprovider** | | |
|  | | |
| **dataProviderClass** | |  |
| **Exemple de code** | | **@BeforeTest{}**  @Test(**dataProvider="SearchProvider",dataProviderClass=DataproviderClass.class**)    public void testMethod(**String author**,**String searchKey**) throws InterruptedException{  WebElement searchText = driver.findElement(By.name("q"));  //Rechercher du texte dans la zone de texte Google  searchText.sendKeys(searchKey);  System.out.println("Bienvenue ->"+author+" Votre clé de recherche est->"+searchKey);  Thread.sleep(3000);  //récupère le texte de la zone de recherche  String testValue = searchText.getAttribute("value");  System.out.println(testValue +"::::"+searchKey);  searchText.clear();  // vérifier si la zone de recherche a une valeur correcte  Assert.assertTrue(testValue.equalsIgnoreCase(searchKey));  } |
| DataproviderClass.java | | paramètres du package ;  importer org.testng.annotations.DataProvider;  public class DataproviderClass {  **@DataProvider(name="SearchProvider")**  public static Object[][] getDataFromDataprovider(){  return new Object[][] {  { "Guru99", "India" },  { "Krishna", "UK" },  { "Bhupesh", "États-Unis" }  } ;  }} |
| **Option** | |  |
| Dans l'exemple suivant :  Nous vérifions si le nom de la méthode est testMethodA.  Si oui, renvoie un ensemble de valeurs  Sinon, renvoie un autre ensemble de valeurs | | @Test(dataProvider="SearchProvider")  public void **testMethodA**(String author,String searchKey) throws InterruptedException{    WebElement searchText = driver.findElement(By.name("q"));  //Rechercher du texte dans la zone de recherche  searchText.sendKeys(searchKey);  //Imprimer l'auteur et la chaîne de recherche  System.out.println("Welcome ->"+author+" Votre clé de recherche est->"+searchKey);  Thread.sleep(3000);  Chaîne testValue = searchText.getAttribute("value");  System.out.println(testValue +"::::"+searchKey);  searchText.clear();  // Vérifiez si la zone de texte Google affiche la valeur correcte  Assert.assertTrue(testValue.equalsIgnoreCase(searchKey));  }  @Test(dataProvider="SearchProvider")  public void **testMethodB(**String searchKey) throws InterruptedException{  {  WebElement searchText = driver.findElement(By.name("q"));  //Rechercher du texte dans la zone de recherche  searchText.sendKeys(searchKey);  //Imprimer uniquement la chaîne de recherche  System.out.println("Bienvenue ->Utilisateur inconnu Votre clé de recherche est->"+searchKey);  Thread.sleep(3000);  Chaîne testValue = searchText.getAttribute("value");  System.out.println(testValue +"::::"+searchKey);  searchText.clear();  // Vérifiez si la zone de texte Google affiche la valeur correcte  Assert.assertTrue(testValue. equalsIgnoreCase(searchKey));  } }  /  \*\*  \* @param m  @DataProvider(name="SearchProvider")  public Object[][] getDataFromDataprovider(Method m){  if(m.getName().equalsIgnoreCase**("testMethodA")){**  return new Object[][] {  { "Guru99", "India" },  { "Krishna" , "Royaume-Uni" },  { "Bhupesh", "USA" }  };}  else{  return new Object[][] {  { "Canada" },  { "Russie" },  { "Japon" }  };}  }  } |
| Résumé | | * **Le paramétrage** est nécessaire pour créer **des tests basés sur les données** . * TestNG prend en charge deux types de paramétrage, en utilisant **@Parameter+TestNG.xml** et en utilisant **@DataProvider** * Dans **@Parameter+TestNG.xml,** les paramètres peuvent être placés au niveau de la suite et au niveau de test. Si   Le même nom de paramètre est déclaré aux deux endroits ; Le paramètre de niveau de test aura la préférence sur le paramètre de niveau de combinaison.   * en utilisant @Parameter+TestNG.xml, une seule valeur peut être définie à la fois, mais @DataProvider renvoie **un tableau 2D de Object** . * Si DataProvider est présent dans une classe différente, alors dans la classe où réside la méthode de test, **DataProvider** doit être **une méthode statique** . * Il existe deux paramètres pris en charge par **DataProvider** : **Method** et **ITestContex** |
|  | | * **Lire et écrire des données à partir d'un fichier Excel dans Selenium Webdriver : POI et JXL31/12/2023** * Exécution parallèle dans Selenium : gestion de session et dépendance à TestNG * TestNG : Comment exécuter plusieurs suites de tests dans Selenium * Génération de rapports TestNG dans Selenium WebDriver * Personnaliser, PDF et envoyer par courrier électronique les rapports TestNG dans Selenium WebDriver. |
|  | |  |

|  |  |  |
| --- | --- | --- |
| **Comment générer un rapport dans selenium** | | |
| Méthode 1 : emailable-report.html | | * 1-Cliquez sur l'option « emailable-report.html * 2-Cliquez sur l'option navigateur Web |
| Méthode 2 : index.html | | * Faites un clic droit sur index.html dans le répertoire du projet. * Sélectionnez l’option Ouvrir avec l’option du navigateur Web. Il affichera le résultat dans l’ordre suivant. |
| * Méthode 3 : classe de journaliste | | * Reporter.log dans Selenium est une classe présente dans TestNG pour les rapports Selenium. Il propose 4 méthodes différentes pour stocker les informations du journal : * Reporter.log(String s); * Reporter.log(String s, Boolean logToStandardOut); * Reporter.log (String s, niveau int); * Reporter.log(String s, niveau int, Boolean logToStandardOut) ; |
| **Lire des données à partir d’un fichier Excel** | | |
|  | [Exemple de programme](lecture%20des%20données%20a%20partir%20d'un%20fichier%20excelExportAndFileIO.docx) | |
|
|
|

.

|  |  |  |
| --- | --- | --- |
| **Type d’attente dans Selenium** | | |
| Implicite Wait | ***Attente à chaque action*** | *driver.manage().timeouts().implicityWait(10,TimeUnit.SECONDS) ;* |
| Explicit wait | ***Attente spécifique à l’élément*** | ***WebDriverWait wait= WebDriver(driver,Duration.ofseconds(10));***  ***Webelement aboutlink = wait.until(ExêctedConditions.elementtobeclicable(By.xpath(“-------“)));***  **Assert.*assertTrue*(element.isDisplayed());*//l'élément est présent***  ***----------------------------------Options----------------------------------------------***  *.invisibilityof(By.xpath(“-------“)));*  *.presenceOfElementLocated(By.xpath(“-------“))); .testToBepresentInElemnt(By.xpath(“-------“))); .visibilityOf(By.xpath(“-------“)));* |
| Fluent wait | ***Attente jusqu’à la présence de l’élément et ignorer l’exception*** | ***@Test***  *Public void testButtonBecomesClickable() {*  *Fluentwait<Webdriver> wait = new FluentWait<>(driver)*  *.withTimeout(Duration.ofseconds(30)) // Max wait time*  *.pollingEvery(duration.ofseconds(1)) // frequency of checking*  *.ignoring(nosuchElementexception.class);*  *WebElement boutton = wait.until(new Function<webdriver, WebElement>() {*  *Public WebElement apply(WebDriver driver){*  *Return driver.findElement(By.xpath(“---------------“));*  *}*  *});*  *aboutLink.click();*  *WebElement botton2= wait.until(ExpectedConditions.visibilityOfElementLocated(by.xpath(“-----“)));* |
|  |  | @BeforeMethod != @beforeClasse  @Aftermethode @Test(priority = 0) ou @Test(priority = 1) |
| 1. NoSuchElementException : Lancée lorsque l'élément ciblé n'est pas trouvé sur la page.  2. StaleElementReferenceException : Lancée lorsque l'élément ciblé a été modifié ou n'est plus attaché au DOM.  3. TimeoutException : Lancée lorsque le délai d'attente spécifié pour une action est écoulé.  4. ElementNotVisibleException : Lancée lorsque l'élément ciblé est présent sur la page mais n'est pas visible.  5. ElementNotSelectableException : Lancée lorsque l'élément ciblé ne peut pas être sélectionné, par exemple pour les cases à cocher ou les boutons radio.  6. InvalidElementStateException : Lancée lorsque l'état de l'élément ciblé est invalide pour effectuer une action.  7. UnexpectedAlertPresentException : Lancée lorsqu'une alerte inattendue est présente sur la page.  8. WebDriverException : Lancée lorsque quelque chose de grave se produit avec le WebDriver lui-même.  9. NoSuchWindowException : Lancée lorsque la fenêtre demandée n'a pas été trouvée.  10. NoSuchFrameException : Lancée lorsque le cadre demandé n'a pas été trouvé. | | |

|  |  |
| --- | --- |
| **Capture d’écran** | |
| **Étape 1)** Convertir l'objet pilote Web en TakeScreenshot | **TakesScreenshot scrShot =((TakesScreenshot)webdriver);** |
| **Étape 2) Appelez la méthode getScreenshotAs pour créer un fichier image** | **Fichier SrcFile=scrShot.getScreenshotAs(OutputType.FILE);** |
| Méthode | public static void takeSnapShot(WebDriver pageweb ,String “chemain du fichier”) throws Exception{  //Conversion de l’objet web driver vers un objet TakeScreenshot  TakesScreenshot scrShot =((TakesScreenshot) pageweb);  //Appelez la méthode getScreenshotAs pour créer un fichier image  File SrcFile = scrShot.getScreenshotAs(OutputType.FILE);  //Déplacer le fichier image vers une nouvelle destination  File DestFile=new File(chemain du fichier);  //Copy file at destination  FileUtils.copyFile(SrcFile, DestFile); |
| Déplacement vers un element | |
| Se déplacer vers un element web | Actions actions = new Actions(driver) ;  actions.moveToElement(**ELEMENTWEB.build().perform();** |
| Se déplacer vers un élément web et click | WebElement **elementInput** = wait.until(Expectedconditions.*presenceOfElementLocated*(By.id…………)));  Actions actions = new Actions(driver) ;  actions.moveToElement(**elementInput**).pause(Duration.ofseconds(1)).clickAndHold().pause(Duration.ofseconds(1)).sendKeys(“-------------“).build().perform(); |
| Exécution dans un browser spécifique  @parametre  **Etape1** | *<?*xml version="1.0" encoding="UTF-8"*?>* <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd"*>* <suite name="All Test Suite">  **<test verbose="2" preserve-order="true"  name="E:/selenium Web Driver/dragAndDrope/src/test/java/dragAndDrp/TPageDragDropTest.java">**  **<parameter name="Browser" value="Edge"/>**  **<classes>**  **\***  **\***  **\***  **\***  **\***  **\*** |
| Dans le setup script  **Etape2** | **@Parameters("Browser")** @BeforeMethod public void setUp(**String Browser**) {  if(**Browser.equals**("chrome")){  WebDriverManager.*chromedriver*().setup();  driver = new ChromeDriver();  }  else if(**Browser.equals**("firefox")){  WebDriverManager.*firefoxdriver*().setup();  driver = new FirefoxDriver();  }  driver.manage().window().maximize();  driver.get(url); } |
| Exécuter des tests avec plusieurs Browser | *<?*xml version="1.0" encoding="UTF-8"*?>* <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd"*>* **<suite name="All Test Suite">**  **<test** verbose="2" preserve-order="true"  name="E:/selenium Web Driver/dragAndDrope/src/test/java/dragAndDrp/TPageDragDropTest.java">  **<parameter name="Browser" value="chrome"/>**  <classes>  <class name="dragAndDrp.TPageDragDropTest">  <methods>  <include name="TestDragandDropDropShoulsBeInAccountColum"/>  <include name="TestDragandDropDropShoulsBeInAmount"/>  </methods>  </class>  </classes>  **</test>**  Exécution avec **chrome**  **<test** verbose="2" preserve-order="true"  name="E:/selenium Web Driver/dragAndDrope/src/test/java/dragAndDrp/TPageDragDropTest.java">  **<parameter name="Browser" value="Edge"/>**  <classes>  <class name="**dragAndDrp.TPageDragDropTest**">  <methods>  <include name="TestDragandDropDropShoulsBeInAccountColum"/>  <include name="TestDragandDropDropShoulsBeInAmount"/>  </methods>  </class>  </classes>  **</test>**  </suite>  Exécution avec **Edge** |
|  | **1-Analyse**  **2-Préorisation des test**  **3-Préorisation des fonctionnalité de base :**  **Test sanitaire (fonctionalité de bsae) ou test fonctionnel ou test de fumé.**  **4-Exploration des cas de test priorisés.**  **Test manuel :**  **5-Réalisation des tests manuel.**  **6-Exécutuion des tests manuel.**  **7-Géneration des rapport des tests manuel.**  **Automatisation :**  **6-Conseption et dévellopement de test automatisé.**  **8-Exécution des tests (exécution en testNG + exécution en POMXML + Exécution dans la commande DOS CMD).**  **9-Test de non régrission après déclaration d’un begue.**  **10-géneration des rapport des tests**  **11-Push dans gitHub.**  **12-Pul à partir de jinkins ou autre outills d’intergation.**  **13-création d’une nouvel verssion.**  **14-test de la nouvelle version.**  **15-push dans github**  **16-pul à partir de jinkins ou autre outills d’intergation.** |
|  |  |

|  |  |
| --- | --- |
|  | **MoveToelement** |
|  | | **Méthode** | **Description** | | --- | --- | | **cliquezEtMaintenez()** | Clics (sans relâcher) à l'emplacement actuel de la souris. | | **contexteClic()** | Effectue un clic contextuel à l’emplacement actuel de la souris. (Action de clic droit de la souris) | | **doubleClic()** | Effectue un double-clic à l'emplacement actuel de la souris. | | **dragAndDrop (source, cible)** | Effectue un clic prolongé à l'emplacement de l'élément source, se déplace vers l'emplacement de l'élément cible, puis relâche la souris.  **Paramètres:**  élément source sur lequel émuler le bouton.  élément cible vers lequel se déplacer et relâcher la souris. | | **dragAndDropBy (source, décalage x, décalage y)** | Effectue un clic et maintient le doigt sur l'emplacement de l'élément source, se déplace d'un décalage donné, puis relâche la souris.  **Paramètres** :  élément source sur lequel émuler le bouton.  xOffset- décalage de déplacement horizontal.  yOffset- décalage de déplacement vertical. | | **keyDown(modifier\_key)** | Effectue une pression sur une touche de modification. Ne relâche pas la touche de modification : les interactions ultérieures peuvent supposer qu'elle est maintenue enfoncée.  **Paramètres** :  modifier\_key – l'une des touches de modification (Keys.ALT, Keys.SHIFT ou Keys.CONTROL) | | **keyUp(modificateur \_key)** | Effectue une libération de clé.  **Paramètres** :  modifier\_key – l'une des touches de modification (Keys.ALT, Keys.SHIFT ou Keys.CONTROL) | | **moveByOffset (décalage x, décalage y)** | Déplace la souris de sa position actuelle (ou 0,0) du décalage donné.  **Paramètres** :  x-offset- décalage horizontal. Une valeur négative signifie déplacer la souris vers la gauche.  y-offset- décalage vertical. Une valeur négative signifie déplacer la souris vers le bas. | | **moveToElement(versElement)** | Déplace la souris au milieu de l'élément.  **Paramètres** :  toElement- élément vers lequel se déplacer. | | **libérer()** | Relâche le bouton gauche de la souris enfoncé à l'emplacement actuel de la souris | | **sendKeys (onElement, séquence de caractères)** | Envoie une série de frappes sur l'élément.  **Paramètres** :  onElement – ​​élément qui recevra les frappes au clavier, généralement un champ de texte  charsequence – toute valeur de chaîne représentant la séquence de frappes à envoyer | |
|  |  |
|  |  |