Automation Machine Test

This project is a basic example of using Selenium WebDriver with Java, TestNG, and the Page Object Model (POM) design pattern to automate the login process for a WooCommerce site.

Prerequisites

- Java JDK 8 or higher: Make sure Java is installed. You can download it from https://www.oracle.com/java/technologies/javase-jdk8-downloads.html .
- Eclipse IDE: You can download it from https://www.eclipse.org/downloads/ .
- Google Chrome: The project uses ChromeDriver for browser automation.
- ChromeDriver: Download the ChromeDriver executable from https://sites.google.com/chromium.org/driver/ .

Setup Instructions

1. Create a Maven Project:
    - Open Eclipse.
    - Go to File > New > Maven Project > Click Next, then select Create a simple project (skip) and click Next > Enter your Group Id (automation) and Artifact Id (Website_Auto), then click Finish.
2. Set Up ChromeDriver:
    - Download the ChromeDriver executable that matches your Chrome browser version.
    - Extract the downloaded file to a directory of your choice.
    - Update the path to the ChromeDriver executable in the `LoginTest.java` file:

      ```
      System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
      ```

    - Replace `"path/to/chromedriver"` with the actual path where you extracted ChromeDriver.

3. Add Dependencies:

   Open the `pom.xml` file in your project and add the following dependencies:

```
<dependencies>
<dependency> <groupId>org.seleniumhq.selenium</groupId>
<artifactId>selenium-java</artifactId> <version>4.0.0</version>
      </dependency>
      <dependency>
         <groupId>org.testng</groupId>
         <artifactId>testng</artifactId>
         <version>7.4.0</version>
         <scope>test</scope>
      </dependency>
   </dependencies>
```

4. Create two packages:

   - Right-click on src/main/java folder> select New > Package and name it (eg: website_auto).
   - Right-click on src/test/java > select New > Package, and name it (eg: website_test).

1 - Login test

1. Create Page Object Model (POM) Class for Login Page:

   Create LoginPage Class:

   - Right-click on website_auto > select New > Class, and name it LoginPage.
   - Copy and paste the following code into LoginPage.java (replace with your package name).

```
package website_auto;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

public class LoginPage {
        private WebDriver driver;
```

```
// Locators
        private By usernameField = By.id("username");
        private By passwordField = By.id("password");
        private By loginButton = By.xpath("//button[@name='login']");

 // Constructor

         public LoginPage(WebDriver driver) {
        this.driver = driver;

                    }


// Methods

        public void enterUsername(String username) {
        WebElement usernameElem = driver.findElement(usernameField);
        usernameElem.sendKeys(username);
        }

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

        public void clickLoginButton() {
        WebElement loginBtn = driver.findElement(loginButton);
        loginBtn.click();
        }
        public void login(String username, String password) {
        enterUsername(username);
        enterPassword(password);
        clickLoginButton();

                    }

                }
```

Create LoginTest Class:

- Right-click on website_test > select New > Class, and name it LoginTest.
- Copy and paste the following code into LoginTest.java((replace with your package name).

```
package website_test;
import loginpage_auto.LoginPage;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class LoginTest {

    private WebDriver driver;

    private LoginPage loginPage;

@BeforeClass

    public void setUp() {

                    // Set path to your ChromeDriver executable
            System.setProperty("webdriver.chrome.driver","C:\\Users\\jesee\\Desktop\\Testing\\A
            utomation\\Chapters\\Selenium\\chromedriver-win32\\chromedriver.exe"); // Update
            the path with your chromedriver file.

                driver = new ChromeDriver();
            driver.get("https://woocommerce-850415-2933260.cloudwaysapps.com/my-
            account");
            driver.manage().window().maximize();
            loginPage = new LoginPage(driver);

    }

    @Test

    public void testLogin() {
            loginPage.login("test_customer", "password");

    }

    @AfterClass

    public void tearDown() {
             //driver.quit();

    }

}
```

2. Run Tests Using TestNG:

    - Right-click on the test class (LoginTest.java) in Eclipse.
    - Select Run As > TestNG Test.
    - TestNG will execute the test, and you will see the results in the Eclipse console.

3. Viewing the Test Results:

    - The results will be displayed in the TestNG console.
    - You can also view detailed reports in the `test-output` folder generated by TestNG in your project directory.

2 - Product page fields test (This test is to check whether the fields add to the product page is working properly)

1. Create Page Object Model (POM) Class for Product Page:

Create a ProductPage class:

- Right-click on website_auto, select New > Class, and name it ProductPage.
- Copy and paste the following code into ProductPage.java:

```
package website_auto;
import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.Select;
import org.openqa.selenium.support.ui.WebDriverWait;

public class ProductPage {
        private WebDriver driver;

// Locators

        private By Select_color= By.xpath("//*[@id=\"product-
        145\"]/div[3]/form/table/tbody/tr[1]/td/div/ul/li[1]/span");
```

```
private By Select_orientation  = By.xpath("//*[@id=\"product-
145\"]/div[3]/form/table/tbody/tr[2]/td/div/ul/li[2]/span");

 private By profileDescriptionField = By.id("profile_desc");

private By phoneNumberCheckbox =
By.xpath("//*[@id=\"phone_number_checkbox\"]");

private By phoneNumberField = By.id("phone_number_field");

private By Select_idType = By.xpath("//*[@id=\"type\"]");

private By additionalElements = By.xpath("//*[@id=\"product-
145\"]/div[3]/form/div/div[2]/div[2]/div[1]/h3");

private By uploadLogoField = By.xpath("//*[@id=\"logo\"]");

private By Select_Border1 =By.xpath("//*[@id=\"product-
145\"]/div[3]/form/div/div[2]/div[2]/div[3]/div[2]/div[1]/label/div/img");

private By Select_Border2 =By.xpath("//*[@id=\"product-
145\"]/div[3]/form/div/div[2]/div[2]/div[3]/div[2]/div[3]/label/div/img
");

private By addToCartButton = By.xpath("//*[@id=\"product-
145\"]/div[3]/form/div/div[2]/button");

 // Constructor

 public ProductPage(WebDriver driver) {
this.driver = driver;

}

 // Methods

public void selectColor(String color) {
driver.findElement(Select_color).click();

 }

public void selectOrientation(String orientation) {
driver.findElement(Select_orientation).click();
 }

public void enterProfileDescription(String description) {
WebElement descriptionElem =
driver.findElement(profileDescriptionField);
descriptionElem.sendKeys(description);
```

```
}


public void checkPhoneNumberCheckbox() {
driver.findElement(phoneNumberCheckbox).click();
}
public void enterPhoneNumber(String phoneNumber) {
WebElement phoneNumberElem =
driver.findElement(phoneNumberField);
phoneNumberElem.sendKeys(phoneNumber);

}
public void selectIDType(String idType) {
driver.findElement(Select_idType).click();
Select idTypeSelect = new Select(driver.findElement(Select_idType));
idTypeSelect.selectByVisibleText(idType);

}
public void uploadLogo(String filePath) {
WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(10));
WebElement uploadElem =
wait.until(ExpectedConditions.elementToBeClickable(uploadLogoFiel
d));
uploadElem.sendKeys(filePath);
System.out.println("File uploaded: " + filePath);

 }
public void selectBorder1() {
driver.findElement(Select_Border1 ).click();

}
public void selectBorder2() {
driver.findElement(Select_Border2 ).click();

}
public void clickAddToCart() {
WebElement addToCartBtn = driver.findElement(addToCartButton);
addToCartBtn.click();

}

}
```

2. Create Test Class

Create a ProductPageTest class:

- Right-click on website_test, select New > Class, and name it ProductPageTest.
- Copy and paste the following code into ProductPageTest.java:

```
package website_test;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
import website_auto.ProductPage;

        public class ProductPageTest {
        private WebDriver driver;
        private ProductPage productPage;

@BeforeClass

   public void setUp() {

      // Set path to your ChromeDriver executable

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\jesee\\Desktop\\Testing\\Automation\\Chapters\\Selenium\\chromedriver-
win32\\chromedriver.exe"); // Update path to chromedriver

        driver = new ChromeDriver();
        driver.get("https://woocommerce-850415-2933260.cloudwaysapps.com/product/rf-id-
        card");
        driver.manage().window().maximize();
        productPage = new ProductPage(driver);

   }

@Test

        public void testProductPageFields() {
        productPage.selectColor("Red");
        productPage.selectOrientation("Landscape");
```

```
        productPage.enterProfileDescription("Happy pdt!");
        productPage.checkPhoneNumberCheckbox();
        productPage.enterPhoneNumber("9876543210");
        productPage.selectIDType("Premium ($50.00)");
        productPage.selectBorder1();
        productPage.selectBorder2();

        productPage.uploadLogo("C:\\Users\\jesee\\Desktop\\Zennode\\image.jpg"); // Update
        with the actual file path
        productPage.clickAddToCart();


        // Add assertion to verify product added to cart successfully

        String currentURL = driver.getCurrentUrl();
        Assert.assertTrue(currentURL.contains("rf-id-card"), "View cart“RF ID Card” has
        been added to your cart.");

}
@AfterClass
public void tearDown() {
 // driver.quit();
}
}
```

3. Run Your Test

    - Right-click on the ProductPageTest class in Eclipse.
    - Select Run As > TestNG Test.

4. Verify the Results

    - Check the TestNG console output in Eclipse to see if the test ran successfully.
    - The assertion in the test checks if the URL contains "rf-id-card", indicating that the product was successfully added to the cart

3 - Field validation test (To check the exception rising while an invalid value is entered in a field):

1. Create Test Class

    Create a FieldValidationTest class:

- Right-click on website_test, select New > Class, and name it FieldValidationTest.
- Copy and paste the following code into FieldValidationTest.java:

```
package website_test;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
import website_auto.ProductPage;

        public class FieldValidationTest {
        private WebDriver driver;
        private ProductPage productPage;

@BeforeClass

   public void setUp() {

      // Set path to your ChromeDriver executable

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\jesee\\Desktop\\Testing\\Automation\\Chapters\\Selenium\\chrome
driver-win32\\chromedriver.exe"); // Update path to chromedriver

        driver = new ChromeDriver();
        driver.get("https://woocommerce-850415-
        2933260.cloudwaysapps.com/product/rf-id-card");
        driver.manage().window().maximize();
        productPage = new ProductPage(driver);

   }

@Test

   public void testFieldvalidation() {
   productPage.selectColor("Red");
   productPage.selectOrientation("Landscape");
   productPage.enterProfileDescription("Happy pdt!");
   productPage.checkPhoneNumberCheckbox();
   productPage.enterPhoneNumber("abcdefgh");
   productPage.selectIDType("Premium ($50.00)");
   productPage.selectBorder1();
```

```
productPage.selectBorder2();
productPage.uploadLogo("C:\\Users\\jesee\\Desktop\\Zennode\\image.jpg"); // Update with the actual file path
productPage.clickAddToCart();

// Add assertion to verify product added to cart successfully

String currentURL = driver.getCurrentUrl();
Assert.assertTrue(currentURL.contains("rf-id-card"), "Phone Number (abcdefgh) is not a valid number.");
}

@AfterClass
public void tearDown() {
 // driver.quit();
}
}
```

2. Run Your Test

    - Right-click on the FieldValidationTest class in Eclipse.
    - Select Run As > TestNG Test.

3. Verify the Results

    - Check the TestNG console output in Eclipse to see if the test ran successfully.
    - The assertion in the test checks if the URL contains "rf-id-card", indicating that the product was successfully added to the cart.

.

4 - Minimum selection test (To check the minimum select condition for multi select field):

1. Create Test Class

    Create a MinimumSelectionTest class:

    - Right-click on website_test, select New > Class, and name it MinimumSelectionTest.
    - Copy and paste the following code into FieldValidationTest.java:

```
package website_test;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
import website_auto.ProductPage;

        public class MinimumSelectionTest {
        private WebDriver driver;
        private ProductPage productPage;

@BeforeClass

    public void setUp() {

        // Set path to your ChromeDriver executable

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\jesee\\Desktop\\Testing\\Automation\\Chapters\\Selenium\\chrome
driver-win32\\chromedriver.exe"); // Update path to chromedriver

        driver = new ChromeDriver();
        driver.get("https://woocommerce-850415-
        2933260.cloudwaysapps.com/product/rf-id-card");
        driver.manage().window().maximize();
        productPage = new ProductPage(driver);

    }

@Test

    public void testFieldvalidation() {
    productPage.selectColor("Red");
    productPage.selectOrientation("Landscape");
    productPage.enterProfileDescription("Happy pdt!");
    productPage.checkPhoneNumberCheckbox();
    productPage.enterPhoneNumber("abcdefgh");
    productPage.selectIDType("Premium ($50.00)");
    productPage.selectBorder1();
    productPage.uploadLogo("C:\\Users\\jesee\\Desktop\\Zennode\\image.jpg")
    ; // Update with the actual file path
    productPage.clickAddToCart();
```

```
// Add assertion to verify product added to cart successfully

String currentURL = driver.getCurrentUrl();
Assert.assertTrue(currentURL.contains("rf-id-card"), "Phone Number
(abcdefgh) is not a valid number.");
}

@AfterClass
public void tearDown() {
 // driver.quit();
}
}
```

4. Run Your Test

    - Right-click on the MinimumSelectionTest class in Eclipse.
    - Select Run As > TestNG Test.

5. Verify the Results

    - Check the TestNG console output in Eclipse to see if the test ran successfully.
    - The assertion in the test checks if the URL contains "rf-id-card", indicating that the product was successfully added to the cart.

--END----------