Automation on Pet Perfect website using Java Selenium

Done By

Neveen Jarrar

Razan AL Far

Hanan Abu Raed

Ahmad Abu Yahya

Supervisor

Abdel Al Raheem AL Saqaa

This document was submitted in fulfillment of the requirements for the Quality Assurance course

Al Hussein Technical University

May, 2023

Aknowledgement

We would like to express our sincere gratitude to Abdel Al Raheem Al Saqaa, our project supervisor, for his invaluable and tireless efforts in reading, reviewing, guiding, encouraging, and, most importantly, for his patience throughout the entire process.

Also, we express my sincere gratitude to Hussein University and the esteemed sponsors of this course, namely GIZ (German Corporation for International Cooperation) and BMZ (Federal Ministry for Economic Cooperation and Development). I am profoundly grateful for the invaluable opportunity extended to us to become part of this distinguished program, which has facilitated the acquisition of profound knowledge and experience in the field of quality assurance

.

List of Contents

Chapter One

Introduction

1.1 About Demo Pet Store

is an e-commerce website that specializes in selling pet products. The website offers a wide range of products such as pet food, toys, accessories, and grooming tools for different types of pets. Our goal as a testing team using Java Selenium is to make sure the website works correctly by performing different tests.

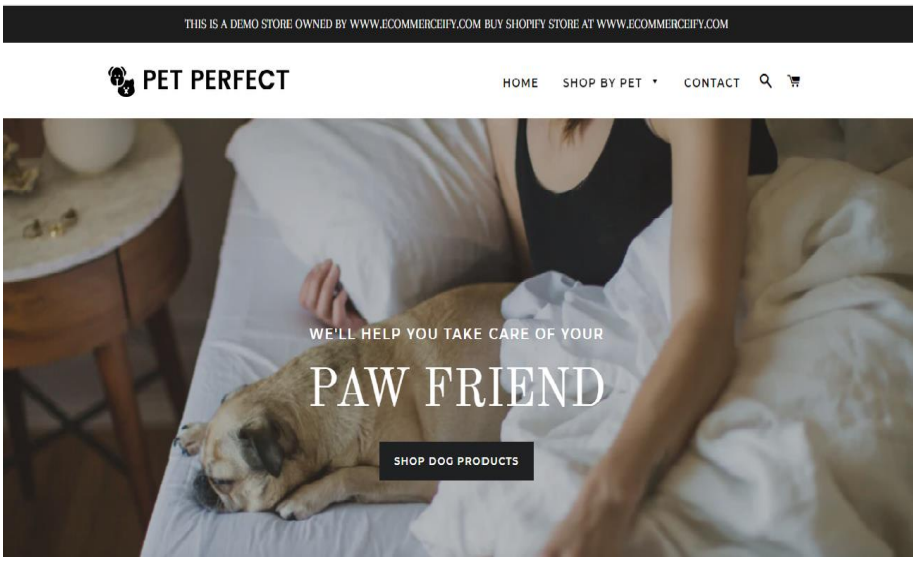

Fig. Pet Perfect Website

Chapter Two

Test Cases

2.1 Test Case:
it refers to a specific scenario or functionality that is automated and executed using the Selenium WebDriver framework. It represents a unit of testing that focuses on verifying the behavior of a web application.

2.2 Test Cases

1. Select randomly one item from the dog products and choose the color and size randomly.

2. Complete the checkout process then capture the screen.

3. Search for "cat" and verify that all results contain the word "cat".

4. select randomly between ("dogs,"cats,"fish", RABBITS") then count the items inside the page.

5. Filling the information required in the contact form.

Chapter Three

Dependencies

3.1 About Dependences

In the context of Java Selenium, dependencies refer to external libraries or modules that are required for the proper functioning of the Selenium WebDriver framework and its related components.

Dependencies in Java Selenium are managed using build automation tools like Apache Maven or Gradle. These tools handle the download and management of the required libraries, making it easier to integrate Selenium into your Java project.

When working with Java Selenium, you need to include specific dependencies in your project to utilize the Selenium WebDriver API and its functionalities. These dependencies typically include:

1. Selenium WebDriver: This is the core library that provides the programming interface to interact with web elements and control browsers programmatically.

2. Selenium WebDriver Browser Drivers: These are the browser-specific executables or drivers that facilitate communication between the Selenium WebDriver and the respective web browsers (e.g., Chrome Driver, Firefox, etc.).

3. Testing Frameworks: Depending on your preference, you may also include testing frameworks like TestNG or JUnit as dependencies. These frameworks offer additional features and annotations for organizing and executing test cases.

To include dependencies in your Java Selenium project, you need to configure the build automation tool (Maven) and specify the required dependencies in the project's configuration file (e.g., pom.xml for Maven). Once the dependencies are defined, the build tool will automatically download and include them in your project's class path during the build process.

By managing dependencies in Java Selenium, you ensure that all the necessary libraries are available for your project, enabling you to leverage the features and capabilities of Selenium WebDriver effectively.

3.2 Dependencies in our project

A. Selenium: to interact with web elements and perform actions on the pet perfect website.
B. WebDriverManager: to ensure if the correct version of the WebDriver is downloaded and set up for testing.
C. TestNG: is used to structure and run the code as a test, It provides features like test sequencing, parallel execution(run multiple tests simultaneously), and detailed reporting.
D. Common.io: is a library in Java that provides classes for input and output operations.

Chapter four

Solution Methodology

In this chapter, we will discuss the methodology of solving test cases using Java Selenium.

4.1 Parameters Class

```
WebDriver driver = new EdgeDriver();
        Random rand= new Random();
        SoftAssert Assert = new SoftAssert();
        static String url = "https://ecom-pet-store.myshopify.com/";
        static String dogs_page = "https://ecom-pet-store.myshopify.com/collections/frontpage";
        WebDriverWait =new WebDriverWait(driver, Duration.ofMinutes(2));
        String Cart_Url="https://ecom-pet-store.myshopify.com/cart";
         static String CONTACT_URL = "https://ecom-pet-store.myshopify.com/pages/contact";
         static String EXPECTED_MSG = "Thanks for contacting us. We'll get back to you as
soon as possible.";
```

4.2 Before Test

```
@BeforeTest
        public void BeforeTest() {
                driver.get(url);

        }
```

4.3 Select randomly one item from the dog products and choose the color and size randomly

```
@Test(priority = 1)
        public void search_for_dog() throws InterruptedException {
                driver.get(dogs_page);
//create a list to select the products randomly
                List<WebElement> Products =
driver.findElements(By.className("grid-product__meta"));
                int productRandInd = rand.nextInt(Products.size());
                Products.get(productRandInd).click();
```

```
//  Create a list to store all available options for sizes and colors of the products

                List<WebElement> colors =
driver.findElements(By.cssSelector("#ProductSelect-option-0 label"));

                List<WebElement> sizes =
driver.findElements(By.cssSelector("#ProductSelect-option-1 label"));

                int colorRandInd = rand.nextInt(colors.size());

                int sizeRandInd = rand.nextInt(sizes.size());

                // select color & size of products randomly

                colors.get(colorRandInd).click();
                sizes.get(sizeRandInd).click();
                driver.findElement(By.className("btn__text")).click();
        }
```

4.4 Complete the check-out process then capture the screen

```
@Test(priority = 2)
        public void verify_checkout_process_and_screenshot_payment_page()
throws IOException {
                // Starting the checkout process by clicking on the 'Checkout' button
                driver.get(Cart_Url);
                WebElement checkout_Button =
driver.findElement(By.className("cart__checkout"));
                checkout_Button.click();

        webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(By.id(
"email")));
                // Declaring a WebElement to fill the required information
                WebElement email_field = driver.findElement(By.id("email"));
                WebElement last_field = driver.findElement(By.name("lastName"));
                WebElement address_field =
driver.findElement(By.name("address1"));
                WebElement city_field = driver.findElement(By.name("city"));
```

```
            WebElement postal_field =
driver.findElement(By.name("postalCode"));
            WebElement submit_button = driver.findElement(By.xpath(

        "/html/body/div[1]/div/div/div/div[1]/div/div[2]/div/div/div/div[2]/div/div/di
v/main/form/div[1]/div/div[2]/div[2]/div[1]/button"));
            // Filling the information
            email_field.sendKeys("asdfsfa@gmail.com");
            last_field.sendKeys("johnson");
            address_field.sendKeys("Groove Street");
            city_field.sendKeys("San Andreas");
            postal_field.sendKeys("95249");

            // submit
            submit_button.click();

        webDriverWait.until(ExpectedConditions.elementToBeClickable(By.xpath(

        "/html/body/div[1]/div/div/div/div[1]/div/div[2]/div/div/div/div[2]/div/div/di
v/main/form/div[1]/div/div/div[2]/div[1]/button")));
            WebElement continue_to_payment = driver.findElement(By.xpath(

        "/html/body/div[1]/div/div/div/div[1]/div/div[2]/div/div/div/div[2]/div/div/di
v/main/form/div[1]/div/div/div[2]/div[1]/button"));
            continue_to_payment.click();

        webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(By.xp
ath(

        "/html/body/div[1]/div[1]/div/div/div[1]/div/div[2]/div/div/div/div[2]/div/div
/div/main/div/form/div[1]/div/div[1]/section/div/div[1]/p")));

            // steps to take a screenshot

            TakesScreenshot screenshotDriver = (TakesScreenshot) driver;
            String dateTimeString =
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMMdd_HHmm
ss"));
            File screenshotFile = new File("D:\\projects\\Pet-Store\\screenshot_"
+ dateTimeString + ".png");
```

```
        FileUtils.copyFile(screenshotDriver.getScreenshotAs(OutputType.FILE),
screenshotFile).
                // Performing an assertion to ensure whether the screenshot was taken
or not
                Assert.assertTrue(driver.findElement(By.id("step-section-primary-
header")).isDisplayed());
        }
```

4.5 Search for "cat" and verify that all results contain the word "cat"

```
@Test(priority = 3)
        public void search_for_cat() {
                //Selecting the Search icon
                WebElement searchIcon =
driver.findElement(By.cssSelector("#AccessibleNav > li:nth-child(4) > a"));
                //Click Search icon
                searchIcon.click();
                //Select the search input field
                WebElement searchBar = driver.findElement(By.className("input-
group-field"));
                //Send the word Cat
                searchBar.sendKeys("cat" + Keys.ENTER);
                //Get title of each  element
                List<WebElement> searchResulTitles =
driver.findElements(By.className("grid-product__title"));
                //Define a boolean flag to  track the condition
                boolean flagIfTitleContainsCat = false;
//Creating a for loop to pass across all titles
                for (int i = 0; i < searchResulTitles.size(); i++) {
                        //Define a string called "title" to store the formatted title
                        String title = searchResulTitles.get(i).getText().toLowerCase();
                        System.out.println(title);
                        //Building an if condition structure to verify the existence of the
word cat then apply changes on the boolean flag, using the method "contains"
                        if (title.contains("cat")) {
                                flagIfTitleContainsCat = true;
                        } else {
```

```
                    flagIfTitleContainsCat = false;
                    break;
                }
            }
            //Doing an assertion for the flag to be true, which means all titles
contains the word "cat"
            Assert.assertEquals(flagIfTitleContainsCat, true).
        }
```

4.6 select randomly between ("dogs, "cats ,“fish", RABBITS") then count the items inside the page

```
@Test(priority = 4)
    public void verify_selecting_random_animal() {
            // Navigating to the specified URL
            driver.get(url);
        // Finding a list of pet types on the webpage
            List<WebElement> pet_type =
driver.findElement(By.cssSelector("div.grid:nth-child(2)"))
                    .findElements(By.className("collection-grid__item-
link"));
            // Generating a random index to select a pet type
            int random_pet_type = rand.nextInt(0, pet_type.size() - 2);
            // Clicking on a randomly selected pet type
            pet_type.get(random_pet_type).click();
        // Finding the grid of pet items
            WebElement items_grid = driver.findElement(By.className("grid-
collage"));
            // Finding all the pet items in the grid
            List<WebElement> pet_items =
items_grid.findElements(By.className("product--image"));
            // Asserting that the number of pet items is either 10 or 12
            Assert.assertTrue(pet_items. size() == 10 || pet_items.size() == 12,
                    "The actual number of items is not as expected");
    }
```

4.7 filling the information required in contact form

```
@Test(priority = 5)
	public void verify_filling_contact_form_and_submit() {

	   // Navigating to the required information

		driver.get(CONTACT_URL);
		driver.findElement(By.id("ContactFormName")).sendKeys("Carl");

	driver.findElement(By.id("ContactFormEmail")).sendKeys("a@vmb.com");

	driver.findElement(By.id("ContactFormPhone")).sendKeys("55292500611")
;
		driver.findElement(By.id("ContactFormMessage")).sendKeys("I want
my order wrapped as a gift");

		// Clicking the submit button on the Contact Form
		driver.findElement(By.className("btn")).click();

	   // Waiting for the success message element to be visible
		WebElement success_msg = webDriverWait
                                .
until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"contact
_form\"]/p")));

		// Asserting that the success message matches the expected message
		Assert.assertEquals(EXPECTED_MSG, success_msg.getText());
	}
```

4.8 After Test

```
@AfterTest

	public void AfterTest() {
	   // Asserting all the assertions made in the test
		Assert.assertAll();

	   // Quitting the driver
		 driver. quit();}}
```

Chapter five

5.1 Discussion and Conclusion

Through the upskilling program at HTUC, we have gained valuable knowledge and expertise in the field of Quality Assurance (QA). The program has equipped us with essential skills, including manual testing, automation testing, test case creation, website and mobile app testing. In manual testing, we have learned how to examine software and systems, identifying defects, and ensuring they meet the desired functionality. Automation testing has empowered us to leverage tools and frameworks to efficiently execute test scripts, saving time and improving test coverage as example (selenium,appium,postman). Additionally, we have acquired proficiency in test case creation, enabling us to develop comprehensive and precise test cases to validate software behavior using black box techniques (Equivilance Class Partitioning ,Boundry Value Ananlysis ,Decision Table) . Lastly, our training in website and mobile app testing has provided us with the ability to evaluate the performance, functionality, and user experience of these platforms, ensuring optimal quality. Overall, this upskilling program has enhanced our expertise as QA professionals, enabling us to contribute effectively to the success of software development .projects

5.2 REFERENCES

- **Maven repository**
- **W3schools**
- **Geeks for geek**
- **Abdel Al Raheem Al Saqaa**