4.4 Locating Elements through CSS and XPath

This section will guide you to:

* Locate elements on the web page.

This guide has mainly three subsections, namely:

4.4.1 Finding the element present on the page by using CSS Selector.

4.4.2 Finding the element present on the page by using XPath.

4.4.3 Pushing the code to your GitHub repositories

**Step 4.4.1:** Finding the element present on the page using CSS Selector.

* Using CSS Selectors in Selenium. As we all know, CSS stands for Cascading Style Sheets. By using CSS selectors, we can find or select HTML elements on the basis of their id, class, or other attributes. CSS is faster and simpler than XPath, particularly in case of IE browser where Path works very slowly.
* Open Eclipse
* Using Path as a CSS Selector
* CSS Selector has many formats, namely:

1. **Tag and ID**
   * Syntax: “css = tag#id”
   * Example: driver.findElement(By.cssSelector(“input#email”));
2. **Tag and Class**
   * Syntax: “css = tag.class”
   * Example: driver.findElement(By.cssSelector(”input.inputtext”));
3. **Tag and Attribute**
   * Syntax: “css = tag[attribute=value]”
   * Example: driver.findElement(By.cssSelector(“input[name=lastName]”));
4. **Tag, Class, and Attribute**
   * Syntax: “tag.class[attribute=value]”
   * Example:

driver. findElement(By.cssSelector(“input.inputtext[tabindex=1]”));

1. **Inner text**
   * Syntax: “css = tag.contains(“innertext”)”
   * Example: driver.findElement(By.cssSelector(font:contains(“Boston”)));

**Step 4.4.2:** Finding the element present on the page using Path.

* In Selenium automation, if the elements are not found by the general locators like id, class, name, etc., then XPath is used to find an element on the web page.
* XPath contains the path of the element situated at the web page. Standard syntax for creating XPath is:

XPath=//tagname[@attribute='value']

* **//:** Select current node.
* **Tagname:**Tagname of the particular node.
* **@:** Select attribute.
* **Attribute:** Attribute name of the node.
* **Value:** Value of the attribute.
* Types of XPath:

There are two types of XPath:

1. **Absolute XPath**

* It is a direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element, then that XPath fails.
* The key characteristic of XPath is that it begins with the single forward slash (/), which means you can select the element from the root node.
* Syntax for absolute Path: html/body/div[1]/div[1]/div/h4[1]/b
* Example: driver.findElement(By.xpath(“html/body/div[1]/div[1]/div/h4[1]/b”));
* Writing absolute XPath on the elements which are present in the web page will be very lengthy. To reduce the length, we use relative XPath.

1. **Relative XPath**

* For relative XPath, the path starts from the middle of the HTML DOM structure. It starts with the double forward-slash (//), which means it can search the element anywhere on the web page.
* You can start from the middle of the HTML DOM structure and you don’t need to write long XPath.
* Syntax for relativeXPath: //\*[@class=’relativexapath’]
* Example: driver.findElement(By.xpath(“//\*[@class=’relativexapath’]”))

The code for the above is as follows:

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class ElementLocatorExample {

    public static void main(String[] args) {

        // Set the path to the ChromeDriver executable

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

        // Launch the Chrome browser

        WebDriver driver = new ChromeDriver();

        // Navigate to the W3Schools HTML Examples page

        driver.get("https://www.w3schools.com/html/html\_examples.asp");

        // Locating elements using CSS selectors

        // Example 1: Tag and ID

        WebElement element1 = driver.findElement(By.cssSelector("a#topnavbtn\_references"));

        // Example 2: Tag and Class

        WebElement element2 = driver.findElement(By.cssSelector("h2.w3-container.w3-red"));

        // Example 3: Tag and Attribute

        WebElement element3 = driver.findElement(By.cssSelector("img[alt='W3Schools.com']"));

        // Example 4: Tag, Class, and Attribute

        WebElement element4 = driver.findElement(By.cssSelector("div.w3-panel.w3-leftbar.w3-sand.w3-padding"));

        // Example 5: Inner text

        WebElement element5 = driver.findElement(By.cssSelector("a:contains('W3Schools')"));

        // Locating elements using XPath

        // Example 1: Absolute XPath

        WebElement element6 = driver.findElement(By.xpath("/html/body/div[5]/div[1]/div[1]/div[4]/h2"));

        // Example 2: Relative XPath

        WebElement element7 = driver.findElement(By.xpath("//\*[@class='w3-sidebar w3-bar-block w3-light-grey w3-card']//a[contains(text(),'Try it Yourself')]"));

        // Perform actions on the located elements

        // ...

        // Close the browser

        driver.quit();

    }

}

Note: Replace the "**path/to/chromedriver**" with the actual path to the chromedriver executable on your system.

**Step 4.4.3:** Pushing the code to GitHub repositories

Open your command prompt and navigate to the folder where you have created your files cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you initially created using the following command:

git push -u origin master