4.3 Multiple Ways to Locate Elements

This section will guide you to:

* Locate elements in Multiple ways using selenium web driver

This guide has mainly seven subsections, namely :

4.3.1 Using ID as a Locator

4.3.2Using class name as a Locator

4.3.3 Using name as a Locator

4.3.4 Using Link Text as a Locator

4.3.5 Using Xpath as a Locator

4.3.6 Using CSS Selector as a Locator

4.3.7 Using XPath for handling complex and dynamic elements

**Step 4.3.1:** Using ID as a Locator

* Open Eclipse
* Find a web element using Locator **ID**

1. Syntax: id = id of the element
2. Example: driver.findElement(By.id(“Email”));

**Step 4.3.2** Using class name as a Locator

* Find a web element using Locator **ClassName**
  1. Syntax: class = Class Name of the element
  2. Example: driver.findElement(By.class(“classname”));

**Step 4.3.3** Using Name as a Locator

* Find a web element using Locator **Name**
  1. Syntax: name = Name of the element
  2. Example: driver.findElement(By.name(“name”));

**Step 4.3.4** Using LinkText as a Locator

* Find a web element using Locator **Link Text**
  1. Syntax: link = partialLink of the element
  2. Example: driver.findElement(By.partialLinkText(“plink”));

**Step 4.3.5** Using Xpath as a Locator

* Find a web element using Locator **Xpath**
* Xpath can be created in two ways
  1. **Relative Xpath**
* Syntax: relativeXpath : //\*[@class=’relativexapath’]
* Example: driver.findElement(By.xpath(“//\*[@class=’relativexapath’]”));
  1. **Absolute Xpath**
* Syntax: absoluteXpath : 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”));

**Step 4.3.6** Using Xpath as a **CSS Selector**

* CSS Selector have 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]”));
5. **Inner text**
   * Syntax: ”css = tag.contains(“innertext”)”
   * Example: driver.findElement(By.cssSelector(font:contains(“Boston”)));

**Step 4.3.7** Using Xpath for handling complex and dynamic elements

* Dynamic Xpath has many formats, namely

1. **Contains();**
   * Syntax: “xpath = //\*[contains(text(),’text’)]
   * Example: driver.findElement(By.xpath(”//\*[contains(text(),’sub’]”));
2. **Using OR & AND**
   * Syntax: xpath=//\*[@type=’submit’ or @name=’btnReset’]
   * Example:

driver.findElement (By.xpath(”=//\*[@type=’submit’ or @name=’btnReset’]”));

1. **Start-with function**
   * Syntax: xpath= //label[starts-with(@id,’message’)]
   * Example:

driver.findElement (By.xpath(”//label[starts-with(@id,’message’)]”));

1. **Text();**
   * Syntax: xpath=//td[text()=’UserID’]
   * Example: : driver.findElement (By.xpath(”=//td[text()=’UserID’]”));
2. **Following**
   * Syntax: xpath=//\*[@type=’text’]//following::input
   * Example: driver.findElement(By.xpath(”=//\*[@type=’text’]//following::input”));
3. **Preceding**
   * Syntax: xpath=//\*[@type=’text’]//preceding::input
   * Example: driver.findElement(By.xpath(”//\*[@type=’text’]//preceding::input”));
4. **Following - sibling**
   * Syntax: xpath=//\*[@type=’submit’]//preceding::input
   * Example:

driver.findElement (By.xpath (”//\*[@type=’text’]//following-sibling::input”));

The code for the above steps 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 LocatorExample {

    public static void main(String[] args) {

        // Set the path to the ChromeDriver executable

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

        // Create a new instance of the ChromeDriver

        WebDriver driver = new ChromeDriver();

        // Launch the website

        driver.get("https://www.seleniumhq.org");

        // Step 4.3.1: Using ID as a Locator

        WebElement downloadTab = driver.findElement(By.id("menu\_download"));

        downloadTab.click();

        // Step 4.3.2: Using class name as a Locator

        WebElement versionsTab = driver.findElement(By.className("version"));

        versionsTab.click();

        // Step 4.3.3: Using Name as a Locator

        WebElement searchField = driver.findElement(By.name("q"));

        searchField.sendKeys("Selenium WebDriver");

        // Step 4.3.4: Using LinkText as a Locator

        WebElement aboutTab = driver.findElement(By.linkText("About"));

        aboutTab.click();

        // Step 4.3.5: Using Xpath as a Locator

        // Relative Xpath

        WebElement relativeElement = driver.findElement(By.xpath("//a[@class='navbar-brand']"));

        relativeElement.click();

        // Absolute Xpath

        WebElement absoluteElement = driver.findElement(By.xpath("/html/body/div[1]/div[1]/a"));

        absoluteElement.click();

        // Step 4.3.6: Using CSS Selector as a Locator

        // Tag and ID

        WebElement downloadButton = driver.findElement(By.cssSelector("a#downloadSeleniumBtn"));

        downloadButton.click();

        // Tag and Class

        WebElement projectName = driver.findElement(By.cssSelector("h1.project-name"));

        System.out.println("Project Name: " + projectName.getText());

        // Step 4.3.7: Using Xpath for handling complex and dynamic elements

        // Contains()

        WebElement projectsLink = driver.findElement(By.xpath("//\*[contains(text(),'Projects')]"));

        projectsLink.click();

        // Close the browser

        driver.quit();

    }

}

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