Finding Web Elements & Working with Links: A Beginner's Guide in Selenium

👋 Introduction

After setting up Selenium successfully, the real fun begins interacting **with web pages like a human!**In real-world scenarios, we search for something, click links, upload/download files, fill out forms, and much more. But before we can perform any of these actions, we need to first **find the right element** on the page.

In this blog, we'll explore **how to locate elements** and **work with hyperlinks** using Selenium with **Java** – explained simply, with examples.

We'll be using this great free practice site:
https://www.leafground.com/dashboard.xhtml

🔍 What are Web Elements?

Web elements are the **building blocks of a webpage**. These include:

- Input boxes
- Buttons
- Links
- Dropdowns
- Images
- Checkboxes, and more

To interact with them, we must first **locate** them in our Selenium test.

☆ Selenium Element Locators – The 8 Strategies

Selenium provides **8 built-in locators** to find web elements:

Locator Type	What it Does	Example Code Snippet
id	Finds element by ID attribute	driver.findElement(By.id("username"));
name	Finds element by name attribute	driver.findElement(By.name("email"));
className	Finds element by class name	driver.findElement(By.className("form-control"));
tagName	Finds element by HTML tag	driver.findElement(By.tagName("input"));
cssSelector	Uses CSS syntax to find elements	driver.findElement(By.cssSelector("input[type='text']"));
xpath	Finds element using XML path	driver.findElement(By.xpath("//input[@id='email']"));
linkText	Finds anchor link by exact text	driver.findElement(By.linkText("Go to Home Page"));
partialLinkText	Finds anchor link by partial text	driver.findElement(By.partialLinkText("Home"));

Tip: id and name are the easiest for beginners. If those don't work, try xpath or CSS Selector.

Sample Practice: Search Google Using name Locator

Let's try sending a search term to Google using the name locator.

```
import org.openqa.selenium.By; import
org.openqa.selenium.WebDriver; import
org.openqa.selenium.chrome.ChromeDriver;

public class GoogleSearchExample {
   public static void main(String[] args) {
WebDriver driver = new ChromeDriver();
driver.get("https://www.google.com");

      // Locate the search box by name and enter a search term
driver.findElement(By.name("q")).sendKeys("Selenium WebDriver");

      // Submit the search
driver.findElement(By.name("q")).submit();
   }
}
```

🔗 Working with Links in Selenium

Hyperlinks are common. Let's understand some useful operations:

1. **Find the total number of links on a page**

```
List<WebElement> links = driver.findElements(By.tagName("a")); System.out.println("Total links: " + links.size());
```

2. **Read link destination without clicking**

```
WebElement link = driver.findElement(By.linkText("Go to Dashboard"));
System.out.println("Link goes to: " + link.getAttribute("href"));
```

3. **Click a link using partial text**

```
driver.findElement(By.partialLinkText("Dashboard")).click();
```

4. **Verify if a link is broken (Basic check)**

You can read the href and try accessing it using Java's HTTP connection:

```
String url = link.getAttribute("href");

HttpURLConnection connection = (HttpURLConnection) new URL(url).openConnection();
connection.setRequestMethod("GET"); connection.connect();

int statusCode = connection.getResponseCode();
System.out.println("Link status code: " + statusCode);
```

If the status code is 400 or above, the link might be broken.

5. Handle Stale Element Reference Exception

Sometimes elements disappear and reappear due to page refreshes. To handle this:

```
try {
   WebElement element = driver.findElement(By.id("someId"));
element.click();
} catch (StaleElementReferenceException e) {
   // Re-find the element
   WebElement element = driver.findElement(By.id("someId"));
element.click();
}
```

6. **Navigate Between Pages**

```
driver.navigate().to("https://www.leafground.com/dashboard.xhtml");
driver.navigate().back(); driver.navigate().forward();
driver.navigate().refresh();
```

Practice Website: LeafGround

This site is perfect for trying out everything from **forms, links, tables, checkboxes, file uploads** and more.

To explore links:

1. Click the **"Elements"** section on the sidebar.
2. Then choose **"Links"** → Start experimenting with the concepts shared above!

Final Thoughts

Finding and interacting with web elements is the **core of Selenium automation**. In this blog, we learned:

- How to locate elements using 8 different strategies
- How to handle links smartly
- How to deal with broken links and exceptions
- How to practice everything step-by-step on LeafGround

Next, we'll learn how to handle **buttons, dropdowns, checkboxes**, and even how to **upload/download files** using Selenium.

Coming Up Next

- Handling Buttons, Forms & Dropdowns in Selenium – Beginner to Intermediate Guide
- File Uploads & Downloads with Selenium WebDriver

Stay tuned, keep learning, and happy testing!

Let's learn together. Let's build together. And let's grow through **learning and sharing**.

Thanks for reading!

Follow me for the next part of the series!