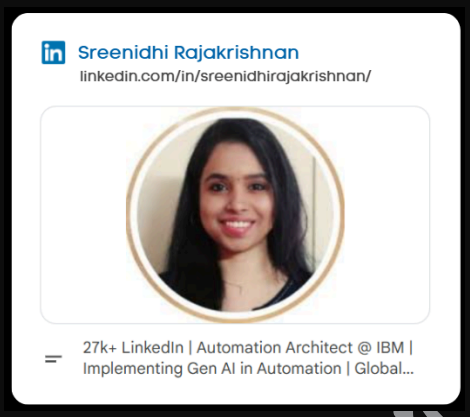


Selenium Locators (Java)

By Sreenidhi Rajakrishnan

This document provides a summary of the most critical locator strategies utilized in automated user interface (UI) testing, encompassing Simple Locators, XPath expressions, and CSS Selectors.

1. Simple Locators (Fundamental Methods)

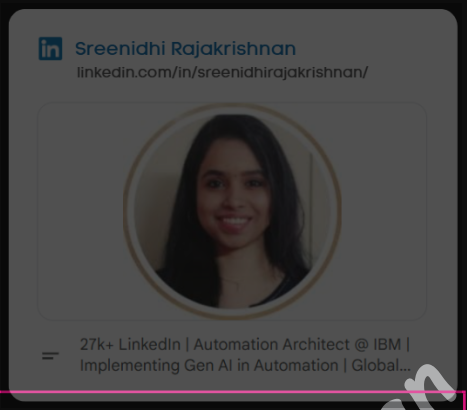
Strategy	Example	One-Line Explanation
ID	`By.id("username")`	Locates an element via its unique **ID** attribute, representing the most reliable method.
Name	`By.name("password")`	Locates an element based on the **name** attribute, typically utilized in form structures.
Class Name	`By.className("btn-primary")`	Identifies elements utilizing the designated **class** attribute value.
Tag Name	`By.tagName("a")`	Retrieves all elements defined by a specific HTML **tag** (e.g., anchor tags).
Link Text	`By.linkText("Login")`	Identifies an anchor link (<a>) based on its **exact** visible text content.
Partial Link Text	`By.partialLinkText("Forgot")`	Identifies an anchor link (<a>) whose text content partially **contains** the specified value.

WebElement element = driver.findElement(By.id(“username”));

element.click();

element.sendKeys(“Hello”);

String elText = element.getText();



2. XPath Selectors (Navigational Expressions)

Strategy	Example	One-Line Explanation
Relative Path	`//div`	Selects all `<div>` elements regardless of their position within the document structure.
Attribute Match	`//input[@id="username"]`	Selects `<input>` elements possessing the specified **id** attribute value.
Specific Child	`//ul/li[3]`	Selects the third `<li>` element, provided it is a direct child of a `<ul>` tag.
Any Element	`//*[@id="login"]`	Selects any element type that contains the `id="login"` attribute.
Text Match (Exact)	`//button[text()="Login"]`	Selects the `<button>` element matching the **exact text** content "Login".
Text Contains	`//a[contains(text(),"Forgot")]`	Selects the `<a>` element whose text content **contains** the string "Forgot".
Attribute Contains	`//div[contains(@class, "error")]`	Selects `<div>` elements whose **class** attribute value **contains** the specified substring "error".
Multiple Conditions	`//input[@type="text" and @name="user"]`	Selects `<input>` elements that satisfy **both** provided attribute conditions concurrently.

Following Sibling	`//h2/following-sibling::p[1]`	Selects the initial <p> element immediately following the current <h2> as a sibling.
Ancestor	`//button/ancestor::form`	Selects the closest `<form>` element positioned as an **ancestor** (parent or higher) of the `<button>`.

3. CSS Selectors (Style-Based Identification)

Strategy	Example	One-Line Explanation
ID	`#username`	Identifies the element corresponding to the **ID** selector.
Class	`.btn-primary`	Identifies elements assigned the **class** attribute value.
Tag + Class	`button.submit`	Selects `<button>` elements that possess the defined **class** attribute.
Attribute (Exact)	`[type="submit"]`	Locates elements matching the **exact attribute** and value specified.
Starts-with (^)	`[id^="msg-"]`	Identifies elements whose **ID** attribute value **commences with** the prefix "msg-".
Ends-with ($)	`[href$=".pdf"]`	Identifies links whose **href** attribute value **terminates with** the suffix ".pdf".
Contains (*)	`[class*="error"]`	Identifies elements whose **class** attribute value **contains** the substring "error".

Descendant	`div p`	Selects all `<p>` elements located **within** a `<div>` container, irrespective of nested depth.
Direct Child	`ul > li`	Selects `<li>` elements that serve as **immediate children** of a `<ul>` element.
Adjacent Sibling	`h2 + p`	Selects the `<p>` **adjacent sibling**
nth-child(n)	`tr:nth-child(3)`	Selects the elem within its immed
Negation	`input:not([disabled])`	Selects all `<input>` elements that **do not** possess the `disabled` attribute.