Xpath Locators for Selenium

Element Selection Strategies

✳ **Id** - located elements whose ID attribute matches the search value.
✳ **Name** - locates elements whose NAME attribute matches the search value.
✳ **Class Name** - locates elements whose class name contains the search value. Easy to use, but there may be several class names on the page with the same attributes.
✳ **Link Text** - locates anchor elements whose visible text matches the search value. Applies only to hyperlink text.
✳ **Partial Link Text** - locates anchor elements whose visible text contains the search value. But, if many elements are matching, only the first one will be selected.
✳ **Tag Name** - locates elements whose tag name matches the search value.
✳ **CSS** - locates elements matching a CSS selector. Good balance between structure and attributes.
✳ **Xpath** - locates elements matching an Xpath expression. Xpath, along with CSS, are the more complex locator strategies.

What is Xpath?

✳ XML Path Language - a query language for selecting nodes from XML documents.
✳ May be used to compute values from the content of an XML document.
✳ When automating tests in Selenium, and there is no Id or Name that can be used to find an element on the page, Xpath can help locate elements without Id, Name or Class attributes. E.g., *driver.findElement(By.xpath('//a[name='rows']/parent::div")).*
✳ Xpath produces reliable locators, if used correctly. It's also the most complicated method of identifying elements.

Xpath Terminology

Nodes

✳ Element node
✳ Attribute node
✳ Text nodes
✳ Document nodes
✳ Others
- Also, there are public nodes without children or parents.

Xpath Locators for Selenium

Relationships of Nodes

- Parents
- Children
- Siblings
- Ancestors
- Descendants

Each element or attribute has 1 parent.
Element nodes can have 0,1, 2…or more children.
Siblings are nodes that have the same parent.
Ancestors are node's parent, parent's parent and so on.
Descendants are child(ren), child(ren)'s child(ren) and so on.

Locating Web Elements

- *driver.findElement(By.id("add_btn"));*
- *driver.findElement(By.name("Save"));*
- *driver.findElement(By.className("btn_round"));*
- *driver.findElement(By.linkText("Automation Tests"));*
- *driver.findElement(By.partialLinkText("Automation"));*
- *driver.findElement(By.tagName("input"));*

No special skills are required. Inspect element, review HTML used in the DOM, and there we have element tags, attributes and attribute values.

Location of elements is more complex with CSS and Xpath locators. Relative locators have been added to Selenium locator strategies.

Tools | Extensions | Plugins

- **Xpath Finder** - Firefox browser add-on.
- **Ranorex Selocity** - Chrome extension.
- **SelectorsHub** - Supports few browsers at a time. Provides shadow DOM support. Gives an error when Xpath is incorrect.

Xpath Locators for Selenium

We can use some tools if the selectors aren't too complex. But it's better to learn creating CSS and Xpath locators ourselves.
Nowadays, web browser tools are so advanced, you don't even need the add-ons. You can right-click on the element, Inspect element, go to the Elements tab in the console, right-click on the element and copy Xpath.

I use Chrome and create my own Xpath locators. I do not rely on auto-suggested paths from add-ons.

Types of Xpath

✳ Absolute
/html/body/div/div/section/section/div/div/div/input

- Direct way to locate element.
- Starts from the root node.
- Not reliable - as soon as there's a change to the UI or an element between our element and the root, Xpath will no longer work.

✳ Relative
//div[@id="row1]/input

- Starts from the node of your choice.
- Shorter than Absolute Xpath.
- More reliable.

// - can search element anywhere on the page

Xpath Syntax

//tag[@attribute = 'value']/child_tag

- *[@attribute = 'value']* is the placeholder for predicators.
- Make sure to use ' ' (single quotes) for attribute names.

Xpath Locators for Selenium

/ vs // vs ./ vs .//

✳ Single Forward Slash /

- Short for child node
- Absolute location path
- Selects a root element

✳ Double Forward Slash //

- Short for descendant or self node
- Relative location path
- Selects element anywhere on a page

//div[@id='rows]//input
//input - descendant, not direct child of *div*.

✳ Dot in Front of Xpath

.//input or *./input*

In automated tests, the use of the dot (.) ties to context.

.// - creates a relative location path, starting at the context node.

When using *WebDriver.findElement*, there is no need for Dot.
When using *row.findElement(by.xpath(".//input"))*, we only look under the row, not everywhere on the page.

Position and Index

//h5[position() = 3] returns the h5 element in position 3, e.g., Test Case 2.

With positions we can use =, !=, and other operators.

//h5[position() != 3] returns all h5 elements, except the one in position 3.

Xpath Locators for Selenium

Position Operators

= Equal	*//h5[position()=2]*
!= Not equal	*//h5[position()!=2]*
< Less than	*//h5[position()<2]*
<= Less than or equal to	*//h5[position()<=2]*
> Greater than	*//h5[position()>2]*
>= Greater than or equal to	*//h5[position()>=2]*

Positions can be used to select more than one element. Index address only selects one element.

Another function, similar to Position, is **Last**:

//h5[last()] - useful when we don't know how many elements we have, but we know for sure we need the last one.

Can also use Subtraction with the Last function:

//h5[last()-1] - gives us the one before the last.

Adding Index to Xpath:

(//div[@class='row'])[2] - put Xpath in ( ), then open another set of [ ] and add index.

E.g.:
(//div[@class='row'])[2]/button[3]
(//div[class='row']/button)[6]

Index Cheat Sheet

- *//tag[index]*
- *//tag1[index1]/tag2[index2]*
- (*//tag1[i@attribute='value']/tag2)[index]*

Q: What's the difference between these two Xpaths: *//h5[2]* and *//h5[position()=2]*?
A: No difference, they both select the 2nd element.

Xpath Locators for Selenium

Xpath Functions

- **position()**
- **last()**
- **text()**
- **contains()**
- **starts-with()**
- **not()**

If we need to find a text element using the text function, then we use this formula:

//tag[text()='Create list of your tests by priority']
//a[text()='Selenium Automation Smoke Testing']

Contains Function - useful when we have partially dynamic values. E.g.:

<button id = "login65">
<button id = "login11">
<button id = "login73">
<button id = "login55">

Part of the id attribute stays the same, part doesn't.

//tag[contains(@attribute, 'partial value')]
//button[contains(@id, 'login')]
//body[contains(@class, 'page-template-test_exceptions')]

Can also use **Text Function** with Contains:

//tag[contains(text(), 'partial value')]
//p[contains(text(), 'You have successfully logged in')]

Starts-With Function - similar to Contains, but is more specific. We also use part of the value, but only the beginning of the value.

//tag[starts-with(@attribute, 'beginning')]
//input[starts-with(@class, 'input')]

Can also use Contains with **Text Function**:

//tag[starts-with(text(), ‘beginning’)]
//p[starts-with(text(), ‘This page is created’)]

✳ **Not Function** - use the Not Function inside predicates. Then, inside the predicates, use any single attribute value.

//tag [not(anything we learned before)]

//tag[not(@attribute = ‘value’)]
//tag[not(text() = ‘value’)]
//tag[not(contains(@attribute, ‘partial value’))]
//tag[not(@id = ‘edit_btn’)]

Xpath Operators

+, -, *, div
=, !=, <, >
or, and

Q: Which one of the Xpath expressions is correct for this element: *<h5>Test case 1: NoSuchElementException</h5>*?
A: //h5{[text() = ‘Test case1: NoSuchElementException’]

Q: Which Xpath expression is correct for this element: *<button id=”edit_btn” class=”btn” name=”Edit”>Edit</button>*?
A: All four of the following will work:

1. *//button[text()=‘Edit’]*
2. *//button[contains(text(),‘Edit’)]*
3. *//button[contains(@id,’_btn’)]*
4. *//button[starts-with(@id,’edit’)]*

Q: Which one is a correct use of the not() function?
A: All four of the following are correct:

1. not attribute and value
2. not text
3. not contains
4. not starts-with