Selenium with Java
Lecture 4

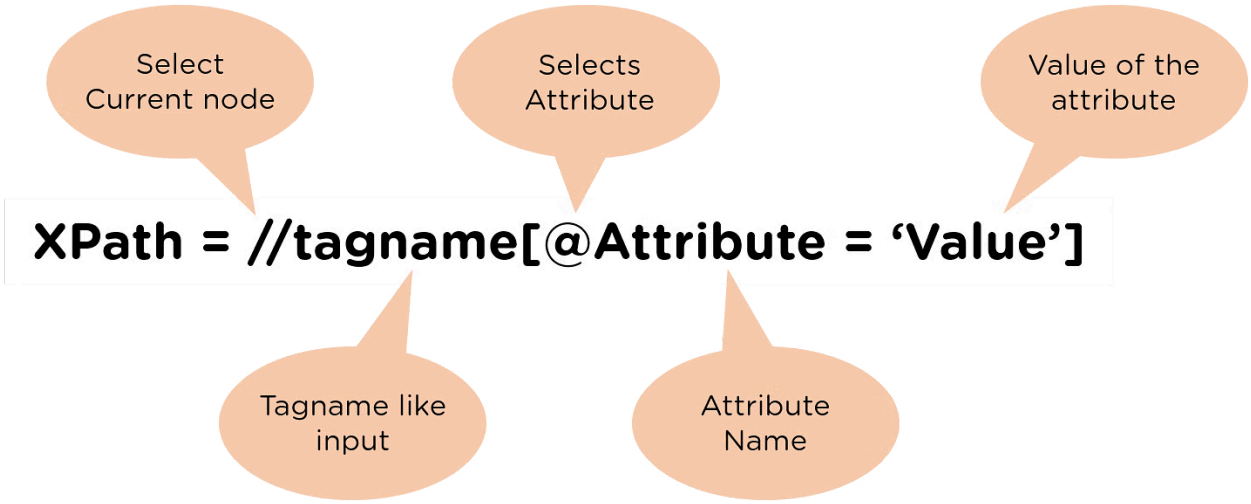
Xpath in Selenium :

XPath is a Selenium technique that is used to navigate through the HTML structure of a webpage. It is a syntax or language that makes finding elements on a webpage possible using XML path expression.

In Selenium automation, there may be times when elements cannot be found with general locators like ID, name, class, etc. And this is when XPath is used to locate those elements on the webpage. XPath in Selenium may be used on both XML and HTML documents.

Syntax of XPath

XPath in Selenium provides many essential XPath functions and axes to write effective XPaths of the web elements and define unique locators.



- // : to select the current node.
- tag name: tag name of a specific node.
- @: to select the attribute.

- attribute: it is the attribute name of the node.
- value: it is the value of the node.

Types of Xpath:

1. Absolute XPath
2. Relative XPath

Absolute XPath:

Absolute XPath refers to the direct way of finding an element. The major drawback of Absolute XPath is that if there are any changes in the element's path, then the XPath will fail.

The XPath begins with a single forward-slash (/), which states that the element can be selected from the root node.

E.g: Absolute XPath:/html/body/div[1]/div/div[2]/header/div/div[2]/a/img

Relative XPath:

In the case of Relative XPath, the path begins from the middle of the HTML DOM structure. Here, the structure starts with a double forward-slash (//) that states that the element can be searched anywhere on the webpage.

Relative XPath enables you to write from the middle of the HTML DOM structure without any need to write a long XPath.

Relative XPath://*[@id=”block-perfecto-main-menu”]/ul/li[6]/a

XPath Functions in Selenium:

There may be times that while locating a particular web element using general attributes, there are elements that have similar properties, like the same name or the same class name. This is when the simple XPath strategies are not efficient and the need for XPath Functions arises. XPath in Selenium provides XPath functions to write efficient XPaths to discover elements uniquely.

XPath Contains() function:

It is used if part of the value of any attribute changes dynamically, like login information, etc. The function can navigate to the web element with the partial text present.

```
//tag_name[contains(@attribute,'value_of_attribute')]
```

```
//input[contains(@name,'UserEmail')]
```

XPath Text() function:

The XPath Text() is a function used to locate the element on a web page using the web element's text. The function proves its worth if the element contains a text, like a label, etc.

```
//tag_name[text()= 'Text of the element']
```

```
//a[text()='Main Services Agreement — Developer Services']
```

XPath Starts-with() function:

The XPath Starts-with() function is used to find the element in which the attribute value starts with some specific character or a sequence of characters. The function plays a major role while working with the dynamic web pages.

//tag_name[starts-with(@attribute,’Part_of_Attribute_value’)]

//input[starts-with(@id,'UserLastName')]

AND & OR operators in Xpath:

These 2 operators very helpful if u want to use more than 2 attributes to find element on webpage.

Xpath=//tagname[@attribute=’value’ or @attribute=’value’]

Xpath=//tagname[@attribute=’value’ and@attribute=’value’]

//input[@type='text' and @name='UserLastName']

//input[@type='text' or @name='UserLastName']

XPath Axes:

All the XML DOM elements are in a hierarchical structure and can be either located using Absolute paths or Relative paths. For this, XPath provides specific attributes called "XPath Axis."

Here, an axis shows a relationship to the current node and helps locate the relative nodes concerning the tree's current node. So, the XPath Axis uses the relation between several nodes to find those nodes in the DOM structure.

Axis	Description
ancestor	To locate the ancestors of the current node, including the parent node and the root node.
ancestor-or-self	To locate the current node along with its ancestors.
attribute	To specify the attributes of the current node.
child	To locate the chid/children of the current node
descendant	To locate the children of the current node up to the leaf node.
descendant-or-self	To locate the current node with its descendants.
following	To locate all the nodes that come after the current node.
parent	To locate the parent of the current node.
self	To locate the current (present) node.

- Ancestor: Selects the current node's ancestor nodes (except the node itself).
    - E.g //div[@class='field error']//ancestor::div selects all div elements that are ancestors of div[@class='field error'].
- Ancestor-or-self: Selects all ancestors of the current node, including the node itself.
    - Example: //div[@class='field error']//ancestor-or-self::div selects the div element itself and all its ancestor div elements.
- Attribute: Selects attributes of the current node.
    - //input/attribute::type selects the type attribute of input elements.
- Child: Selects all child nodes of the current node.
    - Example: //div/child::* selects all child elements of div
- Descendant: Selects all descendant nodes (children, grandchildren, etc.) of the current node.
    - //div/descendant-or-self::span selects the span elements inside div,
    - Example, //div/descendant-or-self::span selects the span elements inside div, as well as div itself if it is a span.

- Following: Selects all nodes that come after the current node in the document, excluding any descendants.
    - Example. //h2/following::p selects all p elements that follow an h2 element.
- Following-sibling: Selects all siblings (have the same parent) that come after the current node.
    - Example //h2/following-sibling::p selects all p elements that are siblings following an h2.
- Parent: Selects the parent node of the current node.
    - Example //p/parent::div selects the div parent of a p element.
- Preceding: Selects all nodes that come before the current node in the document, excluding any ancestors.
    - Example. //p/preceding::h2 selects all h2 elements that precede p.
- Preceding-sibling: Selects all siblings that come before the current node.
    - Example: //p/preceding-sibling::h2 selects all h2 elements that are siblings preceding p.
- Self: Selects the current node itself. //div/self::div selects the div element itself.