**Xpath**

Every element in a web page does not have static id, unique name, unique link text. For those elements we have to build Xpath to find the element and then perform actions on them.

Xpath is defined as XML path and is a query language used to traverse through xml or html documents. It is used commonly to search particular elements or attributes with matching patterns using HTML DOM structure.

**Types of Xpath:**

There are two types of xpath:

1. Absolute Xpath
2. Relative Xpath

**Absolute Xpath:**

if Xpath starts with root node or with ‘/’ then it is an absolute xpath. It is the direct way of find the element.

Eg: html/body/div[1]/section/div[1]/ul/li[2]/a

**Relative Xpath:**

If xpath starts with the node that we’ve selected or with ‘//’ then it is a relative xpath.

Eg: //input[@name='uid']

Syntax to write basic relative Xpath:

*//tagname[@attributename = ‘value’]*

**Difference between single ‘/’ or double ‘//’**

Single slash ‘/’ anywhere in xpath signifies to look for the element **immediately** inside the parent element.

Double slash ‘//’ signifies to look for **any child or nested – child** element inside the parent element.

**Xpath Axes:**

Xpath axes search different nodes in HTML document from current context node. Xpath axes are the methods used to find dynamic elements, which otherwise not possible by normal xpath method having no ID, Classname, Name etc. Xpath axes are used to identify elements by their relationship like parent, child, sibling, etc.

Syntax to write Xpath Axes :- **axes-name :: node-name**

Different Xpath axes:

|  |  |
| --- | --- |
| ***Axes name*** | ***Result*** |
| self | Represents the current node. |
| parent | Represents the parent of the current node. |
| child | Represents the children of the current node. |
| ancestor | Represents all ancestors (parent , grandparent etc) of the current node. |
| ancestor-or-self | Represents all ancestors (parent , grandparent etc) of the current node and current node itself. |
| descendant | Represents all descendants (child, grandchild etc) of the current node. |
| descendant-or-self | Represents all descendants (child, grandchild etc) of the current node and current node itself. |
| following | Represents all the nodes in the webpage after the current node. |
| following-sibling | Represents all the siblings after the current node |
| preceding | Represents all the nodes that appear before the current node in the document, |
| preceding-sibling | Represents all the siblings before the current node. |
| attribute | Represent all attributes of the current node. |
| namespace | Represents all namespace nodes of the current node. |

**Xpath methods:**

starts-with( @attribute, ‘value’)

Returns true if the attribute value starts with the specified value.

contains(@attribute, ‘value’)

Returns true if the attribute value contains the specified value.

text()

Returns true if element text is equal to the specified text.

last()

Returns the last element among all the similar objects.

position()

Returns any element based on their position using xpath

**CSS selector**

Syntax:

tagname[attribute='value']

“#” -­‐> Id

“.” -­‐> Class

Example:

input[id=displayed-text] or

#displayed-text or

input#displayed-text

input[class=displayed-class] or

.displayed-class or

input.displayed-class

We can append the classes like below until we find the unique element

.class1.class2.class3

Using wildcards

“^” -­‐> Represents the starting text

“$” -­‐> Represents the ending text

“\*” -­‐> Represents the text contained

Syntax:

tagname[attribute<special character>=’value’]

Finding child elements

“>” represents the immediate child of the current node

“ “(SPACE) represents the sub child of the current node