Mastering in writing xpath and css Selectors

PART-1

Why Should I learn xpath and css?

- ✓ Mastering XPath or CSS is **essential** for the Selenium test automation engineers to locate dynamic web elements.
- ✓ It is noted that new Selenium automation engineers do not pay much attention to master location strategies.
- ✓ This leads to failure of test automation efforts when web pages with dynamic contents are automated.
- ✓ Most of the testers rely on extracting the XPaths from Firebug/Firepath like tools. The tools cannot be used directly for dynamic web elements.

We will discuss XPath in detail with examples

Why do we need to master many XPath syntaxes ?

- ✓ We may not be able to locate the elements directly using their unique attributes as some element do not have unique attributes.
- ✓ Some attributes are dynamically changed.[/box] Hence we may have to locate them differently than the static elements.
- ✓ Locate elements with respective a known element

- ✓ Locate elements with partial fixed attribute values
- ✓ Locate element without attributes or without unique attributes XPath can do bidirectional navigation (Going forward and backward)
- ✓ XPath is one of most flexible and strongest location strategy

Types of XPath

Absolute Xpath

- ✓ Absolute XPaths starts with the root of the html pages.
- ✓ Absolute XPaths are not advisable for most of the time due to following reasons
- ✓ Absolute XPaths are lengthier and not readable
- ✓ Absolute XPaths are brittle when minor structural changes are done to the web pages
- ✓ Absolute XPaths shall be used only when a relative a XPath cannot be constructed. (highly unlikely).
- ✓ Absolute XPaths tends to break as web pages/content is changed. Hence it not recommended to use absolute XPath in Selenium.

Most of the beginners tend use absolute XPaths when they copy them from tools like FireBug. Theses XPaths are tend to break (brittle) most of the time.

Syntax: Absolute XPaths start with **/html**

Example : /html/body/div[1]/div/div[2]/form/div[2]/input

Continue….

Relative XPath

- ✓ Relative XPaths is used for locating an element from a known element. The element of your choice is referred relative to a known element.
- ✓ **Syntax :** Relative XPaths are started with two forward slashes. **//**

  Example : //div[@id='divUsername']/input
- ✓ Absolute XPaths are faster than the relative XPaths

IF there are multiple elements for an XPath?

- ✓ Selenium will select the first element in the path if there are multiple candidates for a given XPath. If you want to specify the location of the element square brackets with the sequence number shall be used.

Examples :

- ✓ To select the third **input** element : //form/input[3]
- ✓ To select the last **input** element : //form/input[last()]

- ✓ Locating elements by position is discussed further in a separate section

What should be considered when choosing a XPath?

- ✓ It is important to consider following while choosing a XPath from available options.

A good locator is,

- ✓ Unique
- ✓ Descriptive
- ✓ Unlikely to change

Locating Elements with Known Attributes

Locating Elements with known Element and Attributes

- ✓ Function Notated Xpath
- ✓ Text()
- ✓ Last()
- ✓ First()
- ✓ Starts-with()
- ✓ Contains()
- ✓ Not()
- ✓ Normalize-space()
- ✓ Text()—>Exact Match

- ✓ Contains()
- ✓ Starts-with
- ✓ Logical Xpath
- ✓ And Or Xpath axes

Locating a parent element

- ✓ XPath of the known element : **//div[@id='divUsername']**
- ✓ Now we can locate the parent **form** element with respect to the **div** element
- ✓ //div[@id='divUsername']/**parent::form**
- ✓ Also//div[@id='divUsername']/**parent::***This can be written as //div[@id='divUsername']**/..** alsoThere can be only one parent to a context (known) element. Hence specifying the element name is optional.