**Locators**

**What are Locators?**

**Locator is a command that tells Selenium which GUI elements ( Ex. Text Box, Buttons, Check Boxes etc) its needs to operate on. Identification of correct GUI elements is a prerequisite to creating an automation script. But accurate identification of GUI elements is more difficult than it sounds. Sometimes, you end up working with incorrect GUI elements or no elements at all! Hence, Selenium provides a number of Locators to precisely locate a GUI element.**

**1.Locating by ID**

**This is the most common way of locating elements since ID's are supposed to be unique for each element.**

**Target Format: id=id of the element**

**driver.findElements(By.id(“email”));**

**2.Locating by Name**

**Locating elements by name are very similar to locating by ID, except that we use the "name=" prefix instead.**

**Target Format: name=*name of the element***

## 3. Locating by Link Text

**This type of locator applies only to hyperlink texts. We access the link by prefixing our target with "link=" and then followed by the hyperlink text.**

**Target Format: link=link\_text**

## 4. . Locating by Partial Link Text

**In some situations, we may need to find links by a portion of the text in a Link Text element. it contains. In such situations, we use Partial Link Text to locate elements.**

|  |  |
| --- | --- |
| **1** | **findElement(By.partialLinkText("partialLinkText"))** |

**5. Tag Name Locator:**

**Tag Name locator is used to find the elements matching the specified Tag Name. It is very helpful when we want to extract the content within a Tag.**

|  |  |
| --- | --- |
| **findElement(By.tagName("HTML Tag Name"))** |  |

**6. Class Name Locator:**

**Class Name locator gives the element which matches the values specified in the attribute name “class”.**

**Syntax:**

**findElement(By.className("Element Class")**

### ****7.CSS Selectors Locator:****

**There is a debate on the performance of CSS Locator and XPath locator and the debate on the performance of CSS and XPath locator is out of scope. Most of the automation testers believe that using CSS selectors makes the execution of script faster compared to XPath locator. CSS Selector locator is always the best way to locate elements on the page. CSS is always same irrespective of browsers.**

**Following are some of the mainly used formats of CSS Selectors.**

* **Tag and ID css=tag#id css=input#Email**

**Tag and Class css=tag.class css=input.inputtext**

* **Tag and Attribute**

|  |  |
| --- | --- |
|  | **css=tag[attribute=value]** |

* **css=input[name=Email]**
* **Tag, Class, and Attribute css=tag.class[attribute=value] css=input.inputtext[name=email]**

**input.form-control[id=email]**

* **Sub-String Matches**
  + **Starts With (^)**

**To select the element, we would use ^ which means ‘starts with’**

**css=input[id^='Em']**

* + **Ends With ($)**

**To select the element, we would use $ which means ‘ends with’**

**css=input[id$='001']**

* + **Contains (\*)**

**To select the element, we would use *\** which means ‘sub-string’**

**css=input[id\*='id']**

* **Child Elements**
  + **Direct Child**
* **Syntax: parentLocator>childLocator**
* **CSS Locator: div#buttonDiv>button**
* **Explanation: ‘div#buttonDiv>button’ will first go to div element with id ‘buttonDiv’ and then select its child element – ‘button’**
  + **Sub-child**

**Syntax: parentLocator>locator1 locator2**

**CSS Locator: div#buttonDiv button**

**Explanation: ‘div#buttonDiv button’ will first go to div element with id ‘buttonDiv’ and then select ‘button’ element inside it (which may be its child or sub child)**

* + **nth-child**

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5** | **<ul id="automation">**  **<li>Selenium</li>**  **<li>QTP</li>**  **<li>Sikuli</li>**  **</ul>** |

**css="ul#automation li:nth-of-type(2)"**

**XPATH**

**In Selenium automation, if the elements are not found by the general locators like id, class, name, etc. then XPath is used to find an element on the web page .**

**XPath is the language used when locating XML (Extensible Markup Language) nodes. Since HTML can be thought of as an implementation of XML, we can also use xpath in locating HTML elements.**

**Advantage: It can access almost any element, even those without class, name, or id attributes.**

**Disadvantage: It is the most complicated method of identifying elements because of too many different rules and considerations.**

**Absolute Xpath**

**html/body/div[1]/section/div/div/div/div/div[3]/div[1]/div/h4**

**Relative Xpath**

**//input[@id=’email’]**

**Syntax**

**//tagname[@Attribute\_Name=’Attribute\_Value’]**

* **// : Select current node.**
* **Tagname: Tagname of the particular node.**
* **@: Select attribute.**
* **Attribute: Attribute name of the node.**
* **Value: Value of the attribute.**
* **What are XPath axes.**
* **XPath axes search different nodes in XML 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.**
* **Axes methods are used to find those elements, which dynamically change on refresh or any other operations. There are few axes methods commonly used in Selenium Webdriver like child, parent, ancestor, sibling, preceding, self, etc.**

[**Contains()**](https://www.guru99.com/xpath-selenium.html#6)

**Xpath=//tagName[contains(@attribute, 'value')]**

**Xpath=//tagName[contains(text(), ‘ki’)]**

[**Using OR & AND**](https://www.guru99.com/xpath-selenium.html#7)

**Xpath=// tagName[@ attribute ='value' or @ attribute ='value']**

**Xpath=// tagName[@ attribute ='value' and @ attribute ='value']**

[**Starts-with function**](https://www.guru99.com/xpath-selenium.html#8)

**Xpath=//tagName[starts-with(@attribute, 'value')]**

[**Text()**](https://www.guru99.com/xpath-selenium.html#9)

**//tagName[text()=’value’]**

[**XPath axes methods**](https://www.guru99.com/xpath-selenium.html#10)

[**Following**](https://www.guru99.com/xpath-selenium.html#11)

**Selects all elements in the document of the current node( )**

**change the XPath according to the requirement by putting [1],[2]…………and so on**

**Xpath=//tagName[@attribute=’Value’]//following::tagName[1]**

[**Ancestor**](https://www.guru99.com/xpath-selenium.html#12)

**The ancestor axis selects all ancestors element (grandparent, parent, etc.) of the current node.**

**Xpath=//tagName[text()='value']//ancestor::tagName**

[**Child**](https://www.guru99.com/xpath-selenium.html#13)

**Selects all children elements of the current node.**

**Xpath=//tagName[@attribute=’value’]//child::tagName**

[**Preceding**](https://www.guru99.com/xpath-selenium.html#14)

**Select all nodes that come before the current node.**

**Xpath=//\*[@atrribute='value']//preceding::tagName**

**Xpath=//\*[@atrribute='value']//preceding::tagName[1,2,3,....]**

[**Following-sibling**](https://www.guru99.com/xpath-selenium.html#15)

**Select the following siblings of the current node. Siblings are at the same level of the current node. It will find the element after the current node.**

**xpath=//tagName[@attribute='value']//following-sibling::tagName**

**xpath=//tagName[@attribute='value']//following-sibling::tagName[1,2,3....]**

[**Parent**](https://www.guru99.com/xpath-selenium.html#16)

**Selects the parent of the current node.**

**Xpath=//tagName[@attribute='value']//parent::tagName**

[**Self**](https://www.guru99.com/xpath-selenium.html#17)

**Selects the current node or 'self' means it indicates the node itself.**

**One node matching by using "self " axis. It always finds only one node as it represents self-element.**

**Xpath =//tagName[@attribute='value']//self::tagName**

[**Descendant**](https://www.guru99.com/xpath-selenium.html#18)

**Selects the descendants of the current node as.**

**In the below expression, it identifies all the element descendants to current element which means down under the node (child node , grandchild node, etc.).**

**Xpath=//tagName[@atribute='value']//descendant::tagName**