Selenium

Class 3

Agenda

Xpath and CSS Locators

Xpath

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.

Types of Xpath:

1. Native Xpath(**Absolute)**

2. Relative Xpath

Native Xpath(Absolute)

Also called complete or full xpath. Absolute xpath starts from <html> tag or it starts from single slash (/).

it is like directing the xpath to go in direct way. like
html/body/div/div[5]/div/div/div/div/div/div/div/div[2]/div/form/fieldset/div[8]/input[@name='firstname']
Example: firstname field
toolsqa

Here the advantage of specifying native path is, finding an element is very easy as we are mention the direct path.

But if there is any change in the path (if something has been added/removed) then that xpath will break.

Relative Xpath

In relative xpath we will provide the relative path, it is like we will tell the xpath to find an element by telling the path in between.

A relative xpath is one where the path starts from the node of your choice - it doesn't need to start from the root node. It starts with Double forward slash (//).

Advantage here is, if at all there is any change in the html that works fine, until unless that particular path has changed. Finding address will be quite difficult as it need to check each and every node to find that path.
//table/tr/td

Create Customized xpaths

- To customize the xpath. It's very easy to create the **customized xpath**. Xpath has an fix structure like

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

- **Tag** is our HTML tag(like - div, input, name etc), **attribute** is our HTML attribute (like - Id, class etc), and **Value** is the value of that HTML attribute.

- XPath locators are very powerful and flexible. Any element on the page can be located via one or more XPaths and most other locators can be expressed as an XPath.

- Except CSS Selectors, no other locators share this feature. A well-written XPath can be very robust, but a poor XPath can be fragile, and may break when the application changes.

Construct partial Xpath by using XPath functions

contains() : It is used when the value of any attribute changes dynamically,The contain feature has an ability to find the element with partial text

```
//img[contains(@src,' imdbpro')]
```

to use it with a text:

```
//img[contains( text(),' imdbpro')]
```

'starts-with() : Start-with function finds the element whose attribute value changes on refresh or any operation on the webpage. In this expression, match the starting text of the attribute is used to find the element whose attribute changes dynamically.

```
//img[starts-with(@alt,' IMDbPro')]
```

to use it with a text:

```
//img[starts-with(text(), 'IMDbPro')]
```

CSS Selector

- When we don't have an option to choose Id or Name, we should prefer using CSS selector as the best alternative.
- CSS is "Cascading Style Sheets" and it is defined to display HTML in structured and colorful styles are applied to webpage.
- CSS has more Advantage than Xpath
- CSS is much more faster and simpler than the Xpath.
- In IE Xpath works very slow, whereas Css works faster when compared to Xpath.

CSS Selector

CSS Selectors have many formats, but we will only focus on the most common ones.

Tag and ID
Tag and class
Tag and attribute

Locating by CSS Selector - Tag and ID

Syntax	Description
css=*tag#id*	• tag = the HTML tag of the element being accessed • # = the hash sign. This should always be present when using a CSS Selector with ID • id = the ID of the element being accessed

url

WebElement element = driver.findElement(By.cssSelector("**a[id='current time']**"));

Or

WebElement element = driver.findElement(By.cssSelector("**a#current time**"));

Locating by CSS Selector - Tag and Class

Syntax	Description
css=*tag.class*	• tag = the HTML tag of the element being accessed • . = the dot sign. This should always be present when using a CSS Selector with class • class = the class of the element being accessed

url

WebElement element = driver.findElement(By.cssSelector("**a[class='current time']**"));

Or

WebElement element = driver.findElement(By.cssSelector("**a.current time**"));

Locating by CSS Selector - Tag and Attribute

Syntax	Description
css= *tag*[*attribute*=’*value*’]	• tag = the HTML tag of the element being accessed • [and] = square brackets within which a specific attribute and its corresponding value will be placed • attribute = the attribute to be used. It is advisable to use an attribute that is unique to the element such as a name or ID. • value = the corresponding value of the chosen attribute.

<input class="name" id="name" type="text" placeholder="Enter name..." />

WebElement element = driver.findElement(By.cssSelector(“**input[placeholder=‘Enter name...’]**"));

There are there important special characters:

1. '^' symbol, represents the starting text in a string.
2. '$' symbol represents the ending text in a string.
3. '*' symbol represents contains text in a string.

CSS Locators for substring matches(Start, end and containing text) in selenium

It will find input tag which contains 'id' attribute starting with 'ema' text. Email starts with 'ema'

```
css=input[id^='ema']
```

It will find input tag which contains 'id' attribute ending with 'ail' text. Email ends with 'mail'

```
css=input[id$='mail']
```

It will find input tag which contains 'id' attribute containing 'mai' text. Email contains 'mai'

```
css=input[id*='mai']
```