Chapter 4

Locating the elements in Web Pages

So far in our Selenium Learning journey we have done ***WebDriver Commands*** and ***Navigation Commands***. Soon we will be identifying the different ***WebElement*** on webpages and performing various actions on it. This chapter is all about ***Selenium WebDriver WebElement Commands***. But before moving on to finding different WebElements, it better to cover that what all operations we can perform on a *WebElement*. In this chapter we will learn ***What is WebElement*** and the ***List of Actions*** can be performed on various *WebElements*.

What is WebElement?

WebElement represents an HTML element. HTML documents are made up by HTML elements. HTML elements are written with a start tag, with an end tag, with the content in between: <tagname> content </tagname>

The HTML element is everything from the start tag to the end tag: My first HTML paragraph.

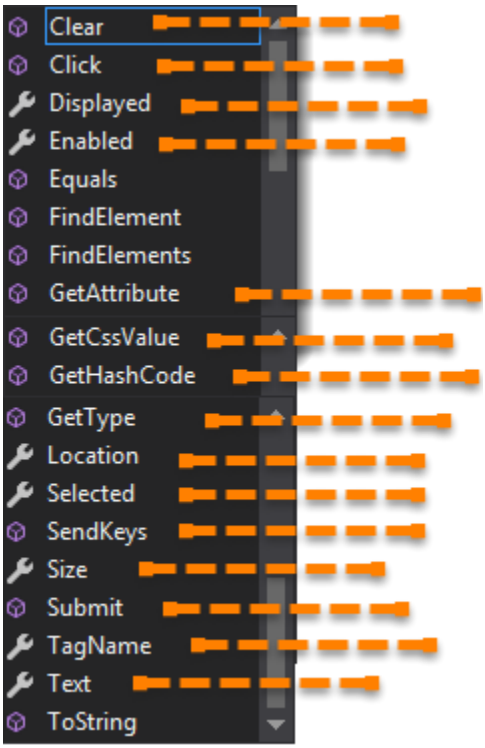
HTML elements can be nested (elements can contain elements). All HTML documents consist of nested HTML elements.

XHTML

```
1<html>
2 <body>
3 <h1> My First Heading </h1>
4 <p> My first paragraph. </p>
5 </body>
6</html>
```

List of WebElement Commands/Actions

All interesting operations to do with interacting with a page will be performed through this ***IWebElement Interface.***



Before going through each and every action of *IWebElement*, let's just understand that how we get a *IWebElement* object/element. As in the previous chapters we learned that every method of the ***IWebDriver*** either returns something or return void(means return nothing). The same

way ***FindElement*** command of ***IWebDriver*** returns ***IWebElement***.

```
IWebElement ISearchContext.FindElement(By by)
Finds the first IWebElement using the given method.
```

So, to get the *IWebElement* object write the below statement:

IWebElement element = driver.FindElement(By.Id("UserName"));

And now if you type ***element dot***, Eclipse's intellisence will populate the complete list of actions just like the above image.

One more thing to notice that *IWebElement* can be of any type, like it can be a ***Text, Link, Radio Button, Drop Down, WebTable*** or any *HTML element*. But all the actions will always populate against the any element irrespective of whether the action is valid on the *IWebElement* or not. For e.g. ***Clear() command***, even if you have a link element still you get the option to choose *Clear() command* on it, which if you choose may result in some error or may not does anything.

Clear Command

void IWebElement.Clear() – If this element is a text entry element, this will clear the value. This method accepts nothing as a parameter and returns nothing.

Command – ***element.Clear();***

This method has no effect on other elements. Text entry elements are ***INPUT*** and ***TEXTAREA*** elements.

```
IWebElement element = driver.findElement(By.Id("UserName"));
element.Clear();

//Or can be written as

driver.FindElement(By.Id("UserName")).Clear();
```

SendKeys Command

void IWebElement.SendKeys(string text) – This simulate typing into an element, which may set its value. This method accepts *string* as a parameter and returns nothing.

Command – element.SendKeys("text");

This method works fine with text entry elements like ***INPUT*** and ***TEXTAREA*** elements.

```
IWebElement element = driver.FindElement(By.Id("UserName"));
element.SendKeys("ToolsQA");

//Or can be written as

driver.FindElement(By.Id("UserName")).SendKeys("ToolsQA");
```

Click Command

void IWebElement.Click()– This simulates the clicking of any element. Accepts nothing as a parameter and returns nothing.

Command – element.Click();

Clicking is perhaps the most common way of interacting with web elements like text elements, links, radio boxes and many more**.**

```
IWebElement element = driver.FindElement(By.LinkText("ToolsQA"));
element.Click();

//Or can be written as

driver.FindElement(By.LinkText("ToolsQA")).Click();
```

***Note**: Most of the time we click on the links and it causes a new page to load, this method will attempt to wait until the page has loaded properly before handing over the execution to next statement. But If Click() causes a new page to be loaded via an event or is done by sending a native event for example through javascript, then the method will not wait for it to be loaded.*

*There are some preconditions for an element to be clicked. The element must be **Visible** and it must have a **Height and Width** greater than 0.*

Displayed Command

bool IWebElement.Displayed{ get; } – This method determines if an element is currently being displayed or not. This accepts nothing as a parameter but returns boolean value(true/false).

Command** – **element.Displayed;

```
IWebElement element = driver.FindElement(By.Id("UserName"));
bool status = element.Displayed;

//Or can be written as

bool staus = driver.FindElement(By.Id("UserName")).Displayed;
```

***Note**: Do not confuse this method with element present on the page or not. This will return **true** if the element is present on the page and throw a **NoSuchElementFound** exception if the element is not present on the page. This refers the property of the element, sometimes the*

*element is present on the page but the property of the element is set to **hidden**, in that case this will return **false**, as the element is present in the DOM but not visible to us.*

Enabled Command

bool IWebElement.Enabled{ get; } – This determines if the element currently is ***Enabled or not***? This accepts nothing as a parameter but returns boolean value(true/false).

Command – ***element.Enabled;***

This will generally return true for everything but I am sure you must have noticed many disabled input elements in the web pages.

```
IWebElement element = driver.FindElement(By.Id("UserName"));
bool status = element.Enabled;

//Or can be written as

bool staus = driver.FindElement(By.Id("UserName")).Enabled;

//Or can be used as
IWebElement element = driver.FindElement(By.Id("UserName"));
bool status = element.Enabled;

// Check that if the Text field is enabled, if yes enter value
if(status){
   element.SendKeys("ToolsQA");
}
```

Selected Command

bool IWebElement.Selected{ get; } – Determine whether or not this element is selected or not. This accepts nothing as a parameter but returns boolean value(true/false).

Command – ***element.Selected;***

This operation only applies to input elements such as ***Checkboxes***, ***Select Options*** and ***Radio Buttons***. This returns ***True*** if the element is currently *selected or checked*, ***false*** otherwise.

```
IWebElement element = driver.FindElement(By.Id("Sex-Male"));
bool status = element.Selected;

//Or can be written as

bool staus = driver.FindElement(By.Id("Sex-Male")).Selected;
```

Note*: In the later chapters of* ***Check Box & Radio Buttons*** and ***Drop Down & Multiple Selects,*** we have covered many examples around it.

Submit Command

void IWebElement.Submit() – This method works well/better than the *Click()* if the current element is a form, or an element within a form. This accepts nothing as a parameter and returns nothing.

Command – ***element.Submit();***

If this causes the current page to change, then this method will wait until the new page is loaded.

```
IWebElement element = driver.FindElement(By.Id("SubmitButton"));
element.Submit();

//Or can be written as

driver.FindElement(By.Id("SubmitButton")).Submit();
```

Text Command

string IWebElement.Text{ get; } – This method will fetch the visible (i.e. not hidden by CSS) innerText of the element. This accepts nothing as a parameter but returns a String value.

Command – ***element.Text;***

This returns an innerText of the element, including sub-elements, without any leading or trailing whitespace.

```
IWebElement element = driver.FindElement(By.Xpath("anyLink"));
String linkText = element.Text;
```

TagName Command

string IWebElement.TagName{ get; } – This method gets the tag name of this element. This accepts nothing as a parameter and returns a String value.

Command – ***element.TagName();***

This does not return the value of the name attribute but return the tag for e.g. "***input***" *for the element* ***<input name="foo"/>***.

```
IWebElement element = driver.FindElement(By.Id("SubmitButton"));
String tagName = element.TagName;

//Or can be written as

String tagName = driver.FindElement(By.Id("SubmitButton")).TagName;
```

GetCssValue Command

string IWebElement.GetCssValue(string propertyName) – This method Fetch CSS property value of the given element. This accepts string as a parameter which is property name.

Command – ***element.GetCssValue();***

Color values should be returned as rgba strings, so, for example if the “background-color” property is set as “green” in the HTML source, the returned value will be “rgba(0, 255, 0, 1)”.

GetAttribute Command

string IWebElement.GetAttribute(string attributeName) – This method gets the value of the given attribute of the element. This accepts the String as a parameter and returns a String value.

Command – ***element.GetAttribute();***

Attributes are Ids, Name, Class extra and using this method you can get the value of the attributes of any given element.

```
IWebElement element = driver.FindElement(By.Id("SubmitButton"));
String attValue = element.GetAttribute("id"); //This will return "SubmitButton"
```

Size Command

System.Drawing.Size IWebElement.Size{ get; } – This method fetch the width and height of the rendered element. This accepts nothing as a parameter but returns the Dimension object.

Command – ***element.Size();***

This returns the size of the element on the page.

```
1   IWebElement element = driver.FindElement(By.Id("SubmitButton"));
2   Dimension dimensions = element.Size();
3   Console.WriteLine("Height :" + dimensions.Height + "Width : "+ dimensions.Width);
```

Location Command

System.Drawing.Location IWebElement.Location{ get; } – This method locate the location of the element on the page. This accepts nothing as a parameter but returns the Point object.

Command – ***element.Location();***

This returns the ***Point object***, from which we can get X and Y coordinates of specific element.

```
1   IWebElement element = driver.FindElement(By.Id("SubmitButton"));
2   Point point = element.Location;
3   Console.WriteLine("X cordinate : " + point.X + "Y cordinate: " + point.Y);
```

Reference links :

http://learn-automation.com/webelements-commands-in-selenium-webdriver-with-c-sharp/

https://www.c-sharpcorner.com/article/overview-of-selenium-locators/