SELENIUM WEBELEMENT COMMANDS

Lecture Notes

A COMPLETE SELENIUM GUIDE

Softech Solutions Inc.

www.softechsolutionsgroup.com

saney.alam@softechsolutionsgroup.com

Selenium WebElement Commands

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***.

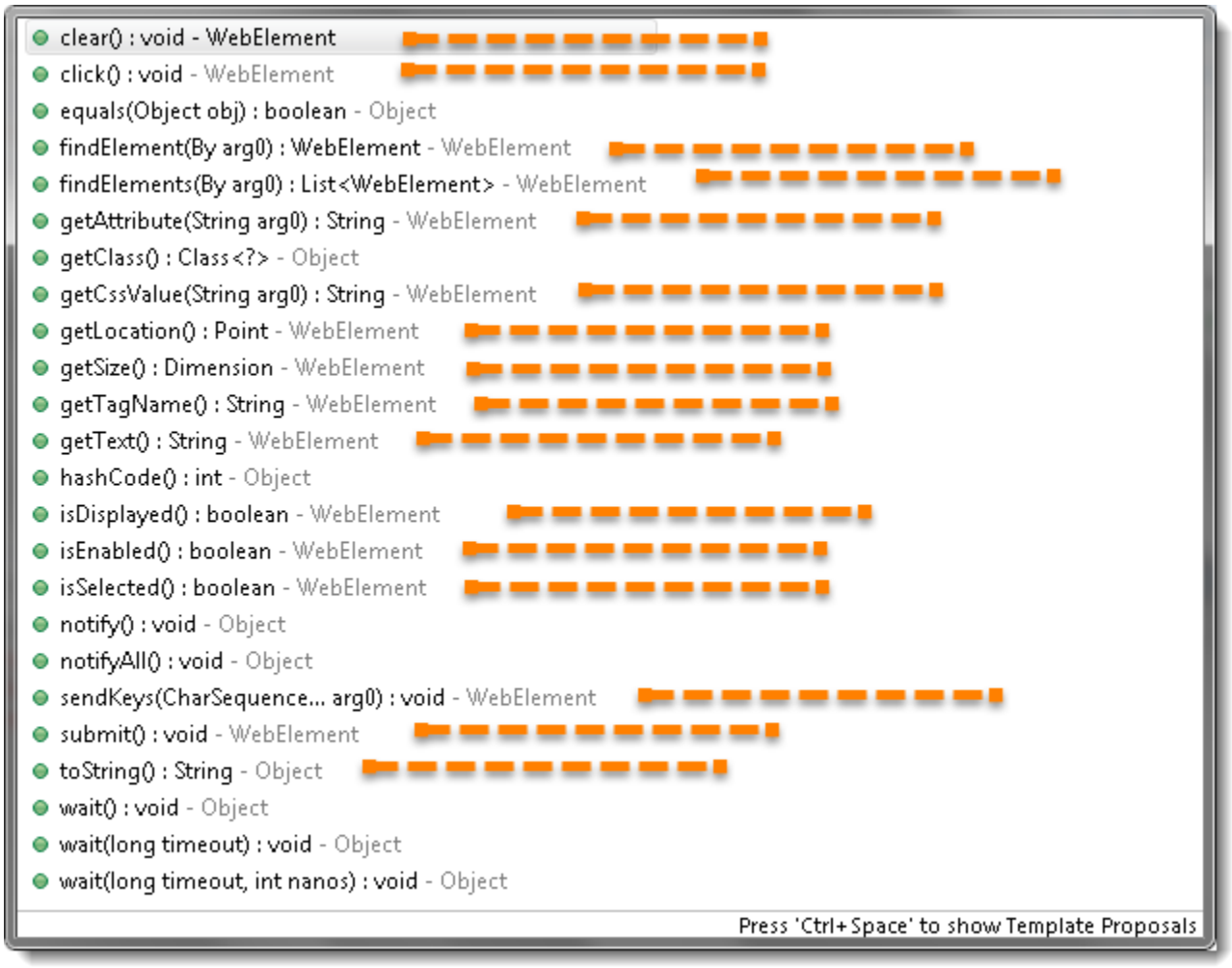
What is WebElement?

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.

```
<html>
        <body>
                <h1> My First Heading </h1>
                <p> My first paragraph. </p>
        </body>
</html>
```

List of WebElement Commands/Actions

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



Note: ***Methods followed by*** **Object** ***keyword are the generic methods gets from Object Class in Java. You will find these methods for every object of java language.***

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

So, to get the WebElement object write the below statement:

WebElement element = driver.findElement(By.id("UserName"));

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

One more thing to notice that ***WebElement*** 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 WebElement 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

clear( ) : void – 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.

```
WebElement element = driver.findElement(By.id("UserName"));
element.clear();

//Or can be written as

driver.findElement(By.id("UserName")).clear();
```

SendKeys Command

sendKeys(CharSequence... keysToSend) : void – This simulate typing into an element, which may set its value. This method accepts CharSequence as a parameter and returns nothing.

Command – **element.sendKeys("text");**

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

```
WebElement element = driver.findElement(By.id("UserName"));
element.sendKeys("admin");

//Or can be written as

driver.findElement(By.id("UserName")).sendKeys("admin");
```

Click Command

click( ) : void – 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**.**

```
WebElement element = driver.findElement(By.linkText("Dress"));
element.click();

//Or can be written as

driver.findElement(By.linkText("Dress")).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.

IsDisplayed Command

isDisplayed( ) : boolean – 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.isDisplayed();**

```
WebElement element = driver.findElement(By.id("UserName"));
boolean status = element.isDisplayed();

//Or can be written as

boolean status = driver.findElement(By.id("UserName")).isDisplayed();
```

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.

IsEnabled Command

isEnabled( ) : boolean – This determines if the element currently is **Enabled or not**? This accepts nothing as a parameter but returns Boolean value (true/false).

Command – **element.isEnabled();**

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

```
WebElement element = driver.findElement(By.id("UserName"));
boolean status = element.isEnabled();

//Or can be written as

boolean staus = driver.findElement(By.id("UserName")).isEnabled();

//Or can be used as
WebElement element = driver.findElement(By.id("userName"));
boolean status = element.isEnabled();
// Check that if the Text field is enabled, if yes enter value
if(status){
   element.sendKeys("admin");
}
```

IsSelected Command

isSelected( ) : boolean – Determine whether or not this element is selected or not. This accepts nothing as a parameter but returns Boolean value (true/false).

Command – **element.isSelected();**

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.

```
WebElement element = driver.findElement(By.id("Sex-Male"));
boolean status = element.isSelected();

//Or can be written as

boolean staus = driver.findElement(By.id("Sex-Male")).isSelected();
```

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

Submit Command

submit( ) : void– 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.

```
WebElement element = driver.findElement(By.id("SubmitButton"));
element.submit();

//Or can be written as

driver.findElement(By.id("SubmitButton")).submit();
```

GetText Command

getText( ) : String– This method will fetch the visible (i.e. not hidden by CSS) inner Text of the element. This accepts nothing as a parameter but returns a String value.

Command – **element.getText();**

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

```
WebElement element = driver.findElement(By.xpath("anyLink"));
String linkText = element.getText();
```

getTagName Command

getTagName( ) : String– This method gets the tag name of this element. This accepts nothing as a parameter and returns a String value.

Command – **element.getTagName();**

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

```
WebElement element = driver.findElement(By.id("SubmitButton"));
String tagName = element.getTagName();

//Or can be written as

String tagName = driver.findElement(By.id("SubmitButton")).getTagName();
```

getCssValue Command

getCssvalue( ) : String– This method Fetch CSS property value of the give element. This accepts nothing as a parameter and returns a String value.

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

getAttribute(String Name) : String– 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.

```
WebElement element = driver.findElement(By.id("SubmitButton"));
String attValue = element.getAttribute("id");   //This will return "SubmitButton"
```

getSize Command

getSize( ) : Dimension – This method fetch the width and height of the rendered element. This accepts nothing as a parameter but returns the Dimension object.

Command – **element.getSize();**

This returns the size of the element on the page.

```
WebElement element = driver.findElement(By.id("SubmitButton"));
Dimension dimensions = element.getSize();
System.out.println("Height :" + dimensions.height + "Width : "+ dimensions.width);
```

getLocation Command

getLocation( ) : Point – This method locate the location of the element on the page. This accepts nothing as a parameter but returns the Point object.

Command – **element.getLocation();**

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

```
WebElement element = driver.findElement(By.id("SubmitButton"));
Point point = element.getLocation();
System.out.println("X cordinate : " + point.x + "Y cordinate: " + point.y);
```