Day-33

WebElement in Selenium

WebElement Commands

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>***

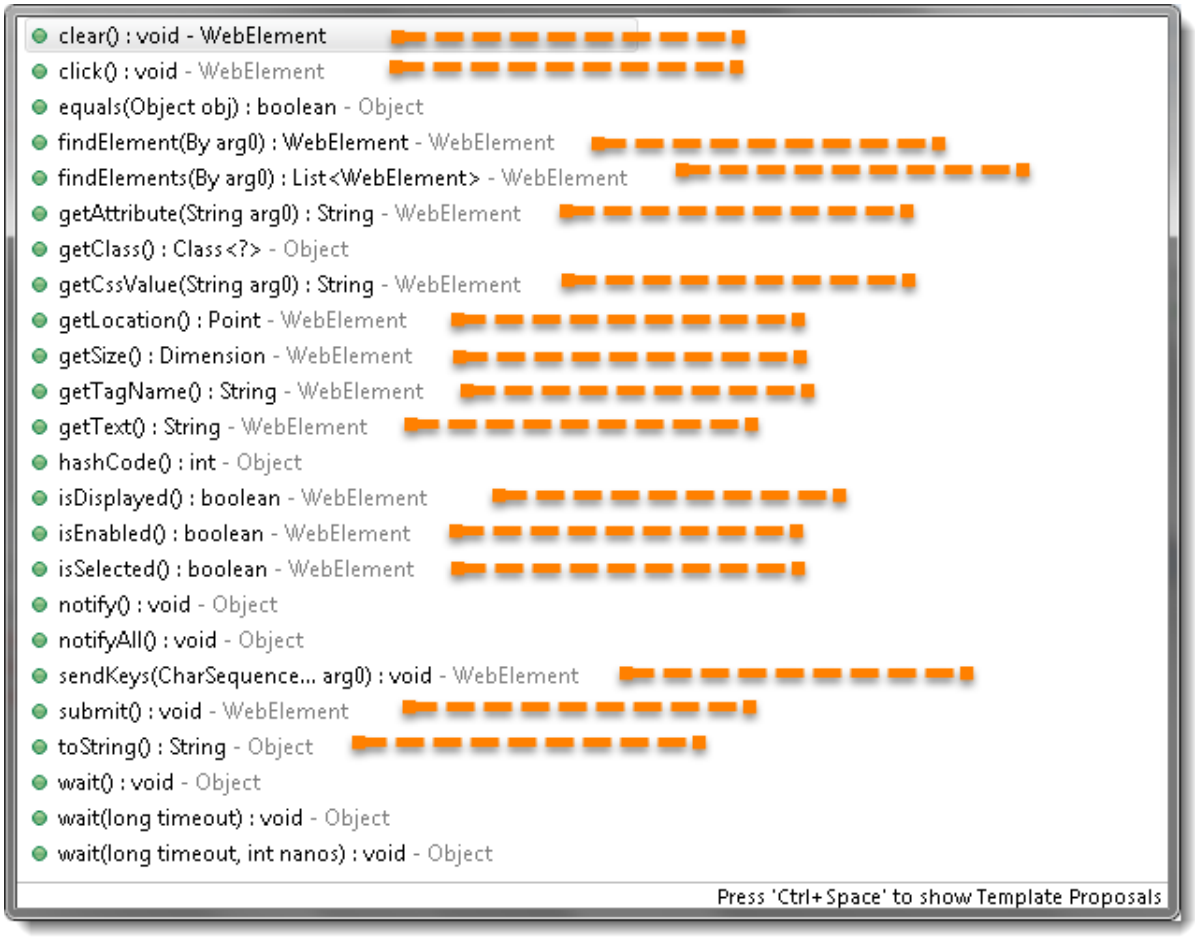
The HTML element is everything from the start tag to the end tag: ***My first HTML paragraph***.

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 intellisence 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 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 simulates 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("Cognizant ");

//Or can be written as

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

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("Cognizant"));
element.click();

//Or can be written as

driver.findElement(By.linkText("Cognizant")).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 a boolean value(true/false).

Command - element.isDisplayed();

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

//Or can be written as

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

***Note**: Do not confuse this method with elements 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 to 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("Cognizant");
}
```

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();
```

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*) innerText of the element. This accepts nothing as a parameter but returns a String value.

Command - element.getText();

This returns an innerText 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);
```

Introduction to WebElement, findElement(), findElements()

Selenium Web Driver encapsulates a simple form element as an object of **WebElement.**

There are various techniques by which the WebDriver identifies the form elements based on the different properties of the Web elements like ID, Name, Class, XPath, Tagname, CSS Selectors, link Text, etc.

Web Driver provides the following two WebElement methods to find the elements.

- **findElement()** – finds a single web element and returns as a WebElement Selenium object.
- **findElements()** – returns a list of WebElement objects matching the locator criteria.

FindElement command syntax:

Selenium Find Element command takes in the By object as the parameter and returns an object of type list WebElement in Selenium. By object in turn can be used with various locator strategies such as find element by ID Selenium, Name, Class Name, XPATH etc. Below is the syntax of FindElement command in Selenium web driver.

```
WebElement elementName =
driver.findElement(By.LocatorStrategy("LocatorValue"));
```

Locator Strategy can be any of the following values.

- ID
- Selenium find element by Name
- Class Name
- Tag Name
- Link Text
- Partial Link Text
- XPATH

Locator Value is the unique value using which a web element can be identified. It is the responsibility of developers and testers to make sure that web elements are uniquely identifiable using certain properties such as ID or name.
Example:

```
WebElement loginLink = driver.findElement(By.linkText("Login"));
```

FindElements command syntax:

FindElements in Selenium command takes in By object as the parameter and returns a list of web elements. It returns an empty list if there are no elements found using the given locator strategy and locator value. Below is the syntax of find elements command.

```
List<WebElement> elementName =
driver.findElements(By.LocatorStrategy("LocatorValue"));
```

Example:

```
List<WebElement> listOfElements = driver.findElements(By.xpath("//div"));
```

Example: How to use Find Element command

```
package com.sample.stepdefinitions;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class NameDemo {

public static void main(String[] args) {
// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver",
"D:\\3rdparty\\chrome\\chromedriver.exe");
WebDriver driver = new ChromeDriver();
driver.manage().window().maximize();

driver.get("http://demo.guru99.com/test/ajax.html");

// Find the radio button for "No" using its ID and click on it
driver.findElement(By.id("no")).click();

//Click on Check Button
driver.findElement(By.id("buttoncheck")).click();

}

}
```

Example: How to use Find Elements command

```
package com.sample.stepdefinitions;

import java.util.List;
```

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class NameDemo {

public static void main(String[] args) {

    System.setProperty("webdriver.chrome.driver", "X://chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.get("http://demo.guru99.com/test/ajax.html");

    List<WebElement> elements = driver.findElements(By.name("name"));
    System.out.println("Number of elements:" +elements.size());

    for (int i=0; i<elements.size();i++){
      System.out.println("Radio button text:" +
elements.get(i).getAttribute("value"));
    }
  }
}
```

Create Xpath for Webelement-

What is 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.

In this tutorial, we will learn about the Xpath and different XPath expression to find the complex or dynamic elements, whose attributes changes dynamically on refresh or any operations.

What is XPath in Selenium?

XPath in Selenium is an XML path used for navigation through the HTML structure of the page. It is a syntax or language for finding any element on a web page using XML path expression. XPath can be used for both HTML and XML documents to find the location of any element on a webpage using HTML DOM structure.
The basic format of XPath in selenium is explained below with screen shot.
Syntax for XPath selenium:

XPath contains the path of the element situated at the web page. Standard XPath syntax for creating XPath is.

```
Xpath=//tagname[@attribute='value']
```

- **// :** Select current node.
- **Tagname:** Tagname of the particular node.
- **@:** Select attribute.
- **Attribute:** Attribute name of the node.
- **Value:** Value of the attribute.

To find the element on web pages accurately there are different types of locators:

XPath Locators	Find different elements on web page
ID	To find the element by ID of the element
Classname	To find the element by Classname of the element
Name	To find the element by name of the element
Link text	To find the element by text of the link
XPath	XPath required for finding the dynamic element and traverse between various elements of the web page
CSS path	CSS path also locates elements having no name, class or ID.

Types of X-path

There are two types of XPath:

1) Absolute XPath

2) Relative XPath

Absolute XPath:

It is the direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.

The key characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the element from the root node.

Absolute XPath:

```
/html/body/div[2]/div[1]/div/h4[1]/b/html[1]/body[1]/div[2]/div[1]/div[1]/h4[1]/b[1]
```

Relative Xpath:

Relative Xpath starts from the middle of HTML DOM structure. It starts with double forward slash (//). It can search elements anywhere on the webpage, means no need to write a long xpath and you can start from the middle of HTML DOM structure. Relative Xpath is always preferred as it is not a complete path from the root element.

Relative Xpath :

```
Relative XPath: //div[@class='featured-box cloumnsize1']//h4[1]//b[1]
```