Navigation Commands

**Navigate To Command**

to(String arg0) : void – This method Loads a new web page in the current browser window. It accepts a String parameter and returns nothing.

Command – driver.navigate().to(appUrl);

It does exactly the same thing as the driver.get(appUrl) method. Where appUrl is the website address to load. It is best to use a fully qualified URL.

**Forward Command**

forward() : void – This method does the same operation as clicking on the Forward Button of any browser. It neither accepts nor returns anything.

Command – driver.navigate().forward();

Takes you forward by one page on the browser's history.

**Back Command**

back() : void – This method does the same operation as clicking on the Back Button of any browser. It neither accepts nor returns anything.

Command – driver.navigate().back();

Takes youback by one page on the browser's history.

**Refresh Command**

refresh() : void – This method Refresh the current page. It neither accepts nor returns anything.

Command – driver.navigate().refresh();

Perform the same function as pressing F5 in the browser.

**Browser Commands**

**Get Command**

get(String arg0) : void – This method Load a new web page in the current browser window. Accepts String as a parameter and returns nothing.

Command – driver.get(appUrl);

Where appUrl is the website address to load. It is best to use a fully qualified URL.

driver.get("http://www.google.com");

**Get Title Command**

getTitle() : String – This method fetches the Title of the current page. Accepts nothing as a parameter and returns a String value.

Command – driver.getTitle();

As the return type is String value, the output must be stored in String object/variable.

driver.getTitle();

//Or can be used as

String Title = driver.getTitle();

driver.getTitle();

//Or can be used as

String Title = driver.getTitle();

**Get Current URL Command**

getCurrentUrl() : String – This method fetches the string representing the Current URL which is opened in the browser. Accepts nothing as a parameter and returns a String value.

Command – driver.getCurrentTitle();

As the return type is String value, the output must be stored in String object/variable.

**Get Page Source Command**

getPageSource() : String – This method returns the Source Code of the page. Accepts nothing as a parameter and returns a String value.

Command – driver.getPageSource();

As the return type is String value, the output must be stored in String object/variable.

driver.getPageSource();

//Or can be written as

String PageSource = driver.getPageSource();

**Close Command**

close() : void – This method Close only the current window the WebDriver is currently controlling. Accepts nothing as a parameter and returns nothing.

Command – driver.close();

Quit the browser if it's the last window currently open.

**Quit Command**

quit() : void – This method Closes all windows opened by the WebDriver. Accepts nothing as a parameter and returns nothing.

Command – driver.quit();

Close every associated window.

**Practice Program**

```
package automationFramework;

	import org.openqa.selenium.WebDriver;

	import org.openqa.selenium.firefox.FirefoxDriver;

public class WebDriverCommands {


	public static void main(String[] args) {

		// Create a new instance of the FireFox driver

		WebDriver driver = new FirefoxDriver();
```

```
// Storing the Application Url in the String variable
String url = "http://www.amazon.in";

//Launch the amazon site
driver.get(url);

// Storing Title name in the String variable
String title = driver.getTitle();

// Storing Title length in the Int variable
int titleLength = driver.getTitle().length();

// Printing Title & Title length in the Console window
System.out.println("Title of the page is : " + title);
System.out.println("Length of the title is : "+ titleLength);

// Storing URL in String variable
String actualUrl = driver.getCurrentUrl();

if (actualUrl.equals(url)){
        System.out.println("Verification Successful - The correct Url is opened.");
}else{
        System.out.println("Verification Failed - An incorrect Url is opened.");
        //In case of Fail, you like to print the actual and expected URL for the record
purpose
        System.out.println("Actual URL is : " + actualUrl);
        System.out.println("Expected URL is : " + url);
}
```

```
        // Storing Page Source in String variable
        String pageSource = driver.getPageSource();

        // Storing Page Source length in Int variable
        int pageSourceLength = pageSource.length();

        // Printing length of the Page Source on console
        System.out.println("Total length of the Pgae Source is : " + pageSourceLength);

        //Closing browser
        driver.close();
    }
```

**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>
```

**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.

**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();

**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();

**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();

**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();

**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();

**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);
```

FindElement And FindElements Commands

FindElement And FindElements Commands

In the previous chapter of WebElement Commands, we learned different types to actions which can be performed on a WebElement object. Thus the next thing to do is to interact with a web page to use WebElement Commands/Actions. First thing to locate an element on the web page before interacting with it and locating elements can be done on the WebDriver Instance(driver) itself or on a WebElement. WebDriver gives us Find Element and Find Elements methods to locate element on the web page.

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 method of WebDriver returns WebElement.

WebElementCommands_02

But the findElement() method accepts something as a Parameter/Argument and which is By Object. By is the mechanism used to locate elements within a document with the help of locator value. A normal syntax of By looks like this:

FindElements_02

Locating Element using By Strategy

Locating elements in WebDriver is done by using the findElement(By.locator()) method. The findElement methods take a locator or query object called 'By'. In the eclipse code window type driver.findElement(By dot), Eclipse intellicense will populate the list of different locators. 'By' strategies are listed below.

FindElements_01

Browser tools for Element Inspector

Firefox: Firebug add on. Right click on any element and select Inspect Element or F12

Chrome: Build in Page analyzing feature (right click –> Inspect Element / F12)

IE: Developers Tool (Tools –> Developers Tools/ F12)

I would suggest to take a look at the small chapter of Finding Elements using Browser Inspector before moving on to this chapter.

**By ID**

id(String id) : By – This is the most efficient and preferred way to locate an element, as most of the times IDs are unique. It takes a parameter of String which is a Value of ID attribute and it returns a BY object to findElement() method.

Command – driver.findElement(By.id("Element ID"));

With this strategy, If no element has a matching id attribute, a NoSuchElementException will be raised.

Example: If an element is given like this:

FindElements_04

Actual Command

```
WebElement element = driver.findElement(By.id("submit"));
// Action can be performed on Input Button element
element.submit();
```

```
WebElement element = driver.findElement(By.id("submit"));
// Action can be performed on Input Button element
element.submit();
```

Note: Common pitfalls that UI developers make is having non-unique id's on a page or auto-generating the id, both should be avoided.

**By Name**

name(String name) : By – This is also an efficient way to locate an element but again the problem is same as with ID that UI developer make it having non-unique names on a page or auto-generating the names. It takes a parameter of String which is a Value of NAME attribute and it returns a BY object to findElement() method.

Command – driver.findElement(By.name("Element NAME"));

With this strategy, the first element with the name attribute value matching the location will be returned. If no element has a matching name attribute, a NoSuchElementException will be raised.

Example: If an element is given like this:

FindElements_05

Actual Command

**By ClassName**

className(String className) : By – This finds elements based on the value of the CLASS attribute. It takes a parameter of String which is a Value of CLASS attribute and it returns a BY object to findElement() method.

Command – driver.findElement(By.className(“Element CLASSNAME”));

If an element has many classes then this will match against each of them.

Example: If an element is given like this:

FindElements_04

Actual Command

```
WebElement parentElement = driver.findElement(By.className("button"));
WebElement childElement = parentElement.findElement(By.id("submit"));
childElement.submit();
```

Note: This method is a life saver. As said, class can contain many elements, many times when you end up with duplicate IDs and Names, just go for the ClassName first and try to locate the element with ID. That will work fine, as the selenium will look for the ID which is in the mentioned class.

**By TagName**

tagName(String name) : By – With this you can find elements by their TAGNAMES. It takes a parameter of String which is a Value of TAG attribute and it returns a BY object to findElement() method.

Command – driver.findElement(By.tagName(“Element TAGNAME”));

Locating Element By Tag Name is not too much popular because in most of cases, we will have other alternatives of element locators. But yes if there is not any alternative then you can use element's DOM Tag Name to locate that element in WebDriver.

Example: If an element is given like this:

FindElements_04

Actual Command

```
WebElement element = driver.findElement(By.tagName("button"));
// Action can be performed on Input Button element
element.submit();
```

**By LinkText & PartialLinkText**

linkText(String linkText) : By – With this you can find elements of "a" tags(Link) with the link names. Use this when you know link text used within an anchor tag. It takes a parameter of String which is a Value of LINKTEXT attribute and it returns a BY object to findElement() method.

partialLinkText(String linkText) : By – With this you can find elements of "a" tags(Link) with the partial link names.

Command – driver.findElement(By.linkText("Element LINKTEXT"));

Command – driver.findElement(By.partialLinkText("Element LINKTEXT"));

If your targeted element is link text then you can use by link text element locator to locate that element. Partial Link Text is also same as Link text, but in this we can locate element by partial link text too. In that case we need to use By.partialLinkText at place of By.linkText.

Example: If an element is given like this:

FindElements_06

Actual Command

```
WebElement element = driver.findElement(By.linkText("Partial Link Test"));
element.clear();
//Or can be identified as
WebElement element = driver.findElement(By.partialLinkText("Partial");
element.clear();
```

**By XPath**

xpath(String xpathexpression) : By – It is most popular and majorly used locating element technique or the easiest way to locate element in WebDriver. It takes a parameter of String which is a XPATHEXPRESSION and it returns a BY object to findElement() method.

Command – driver.findElement(By.xpath("Element XPATHEXPRESSION"));

The best thing in xpath is that it provides many different technique to locate elements. It gives you feature to locate single element in many ways.

We have a complete chapter on XPath techniques which we will come across during our learning journey on ToolsQA latter.

**Difference between FindElement & FindElements Commands**

The difference between findElement() and findElements() method is the first returns a WebElement object otherwise it throws an exception and the latter returns a List of WebElements, it can return an empty list if no DOM elements match the query.

findElement()

On Zero Match : throws NoSuchElementException

On One Match : returns WebElement

On One+ Match : returns the first appearance in DOM

findElements()

On Zero Match : return an empty list

On One Match : returns list of one WebElement only

On One+ Match : returns list with all matching instance