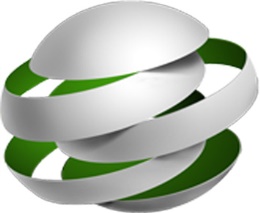
**

**Softech Solutions Inc.**

[www.softechsolution.tech](http://www.softechsolution.tech)

Email: saney.alam@softechsolution.tech

*This part is a step-by-step introduction to setup and configure WebDriver with Eclipse.*

WebDriver Commands Part 1

***Lecture Notes***

WebDriver Commands Part 1

|  |  |
| --- | --- |
| ***Table of Contents Page*** | |
| *WebDriver Browser Commands* |  |
| *Get Command* | ***02*** |
| *Get Title Command* | ***04*** |
| *Get Current URL Command* | ***04*** |
| *Get Page Source Command* | ***04*** |
| *Close Command* | ***04*** |
| *Quit Command* | ***05*** |
| *Browser Navigation Commands* |  |
| *Navigate to command* | ***06*** |
| *Froward Command* | ***06*** |
| *Back Command* | ***06*** |
| *Refresh Command* | ***07*** |
| *FindElement & FindElements Command* |  |
| *Locating Elements using By Strategy* |  |
| *By Id* |  |
| *By Name* |  |
| *By Class Name* |  |
| *By Tag Name* |  |
| *By LinkText & PartialLinkText* |  |
| *By XPath* |  |
| *Difference between FindElement and FindElements Commands* |  |
| *Web Element Commands* |  |
| *What is Web Element?* |  |
| *Lists of Web Element Commands* |  |
| *Clear Command* |  |
| *Click Command* |  |
| *IsDisplayed Command* |  |
| *IsEnabled Command* |  |
| *IsSelected Command* |  |
| *Submit Command* |  |
| *GetText Command* |  |
| *getTagName Command* |  |
| *getCssValue Command* |  |
| *getAttribute Command* |  |
| *getSize Command* |  |
| *getLocation Command* |  |
|  |  |

**Browser Commands**

The very first question which comes to my mind and has been asked in many interviews is **what is Selenium WebDriver**? Is it an Automation Tool? Is it a Class? Is it an Interface or what actually it is? To answer this question we need to understand the Advance Java OOPs concepts first and then we would be able to visualize the WebDriver Implementation. For the sake of simplicity, we will avoid this WebDriver Implementation topic for now and will cover this in later chapters. As of now we start with all the methods we get from WebDriver.

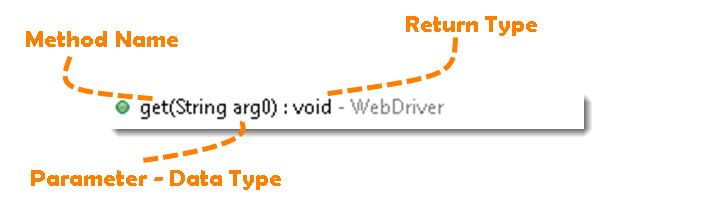
Now the next question is, How to access the methods of WebDriver? To check what all we have in WebDriver, create a driver object from WebDriver and press dot key. This will list down all the methods of WebDriver.



**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.*

* *The suggestions marked in* Blue Color *are Nested Classes under WebDriver and will be covered in detail separately in the following chapters.*
* *The suggestions marked in* Green Color *are also Interfaces like WebDriver and will be covered in detail separately in the following chapters.*
* *The suggestions marked in* Violet Color *are similar methods like* Orange *but will be covered in detail separately in the following chapters*.

Let’s just start discussing the **Orange** **colored** methods of **Selenium WebDriver** but before that try to understand the syntax of the suggestions display by Eclipse for WebDriver.



**Method:** A Java method is a collection of statements that are grouped together to perform an operation.

* **Method Name:** *To access any method of any class, we need to create an object of class and then all the public methods will appear for the object.*
* **Parameter**: *It is an argument which is passed to a method as a parameter to perform some operation. Every argument must pass with the same data type. For e.g.* **get(String arg0) : void.** *This is asking for a* String type *argument.*
* **Return Type:** *Method can return a value or returning nothing (void). If the* void *is mentioned after the method, it means the method is returning no value. And if it is returning any value, then it must display the type of the value for e.g.***getTitle() : String*.***

Now it would be very easy to understand the WebDriver commands in the below chapter. The very first thing you like to do with Selenium is to Opening a new browser, Perform few tasks and Closing the browser. Below are the numbers of commands you can apply on the Selenium opened browser.

## ***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");*

*//Or can be written as*

*String URL = "http://www.DemoQA.com";*

*driver.get(URL);*

## ***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();*

## ***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.getCurrentUrl();**

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

*driver. getCurrentUrl ();*

*//Or can be used as*

*String CurrentUrl= driver.getCurrentUrl();*

## ***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 used 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.

*driver.close();*

## ***Quit Command***

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

**Command** **– driver.quit();**

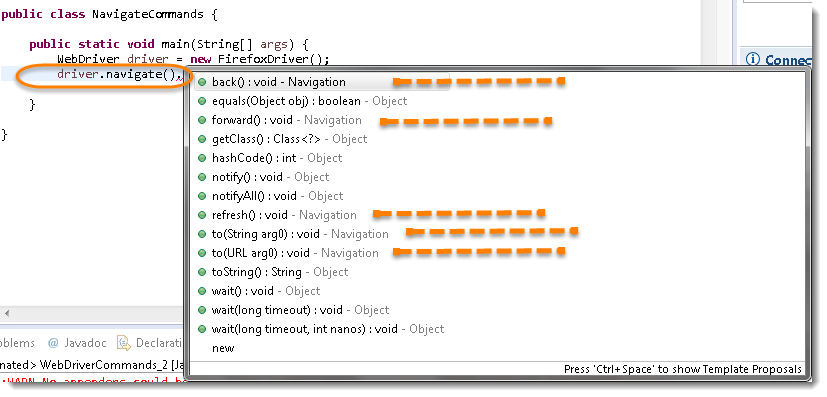
Close every associated window.

*driver.quit();*

**Browser Navigation Commands**

After successfully running our first test case on Firefox Browser now we are stepping towards grasping the essential **Browser Navigation Commands** in Selenium. Thus we are going to discuss about various navigation commands that we would be using in our day to day automation testing. The navigate interface exposes the ability to move backwards and forwards in the browser’s history.

To access the navigation’s method, just type **driver.navigate().**. The intellisence feature of eclipse will automatically display all the public methods of *Navigate Interface* shown in the below image.



**Note:***Only methods which are followed by Navigation keyword are belongs to navigate. Rest 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.*

***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.

*driver.navigate().forward();*

## ***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 you back by one page on the browser’s history.

*driver.navigate().back();*

## ***Refresh Command***

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

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

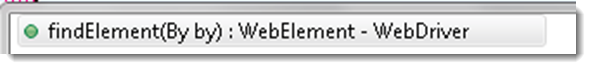
Perform the same function as pressing F5 in the browser.

*driver.navigate().back();*

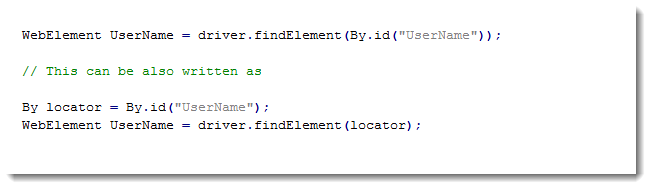
**FindElement & FindElements Commands**

In the previous chapter of [WebElement Commands](http://toolsqa.com/selenium-webdriver/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**.



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:



***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 intellisence will populate the list of different locators. ‘**By’** strategies are listed below.



*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**](http://toolsqa.com/selenium-webdriver/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:



**Actual Command**

*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****.***

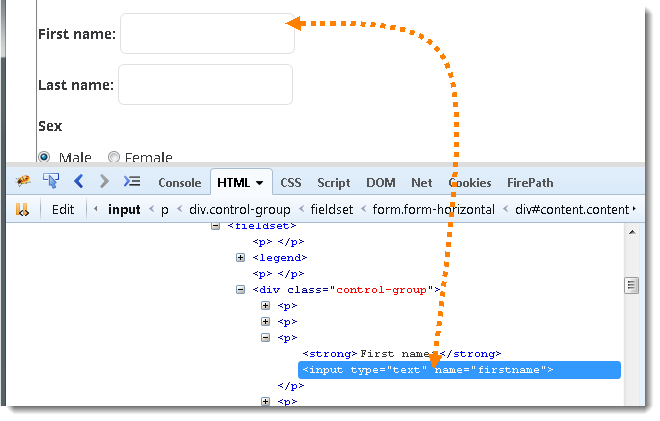
***By Name***

**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:



**Actual Command**

*WebElement element = driver.findElement(By.name("firstname"));*

*// Action can be performed on Input Text element*

*element.sendKeys("Saney");*

***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:



**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:



**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.

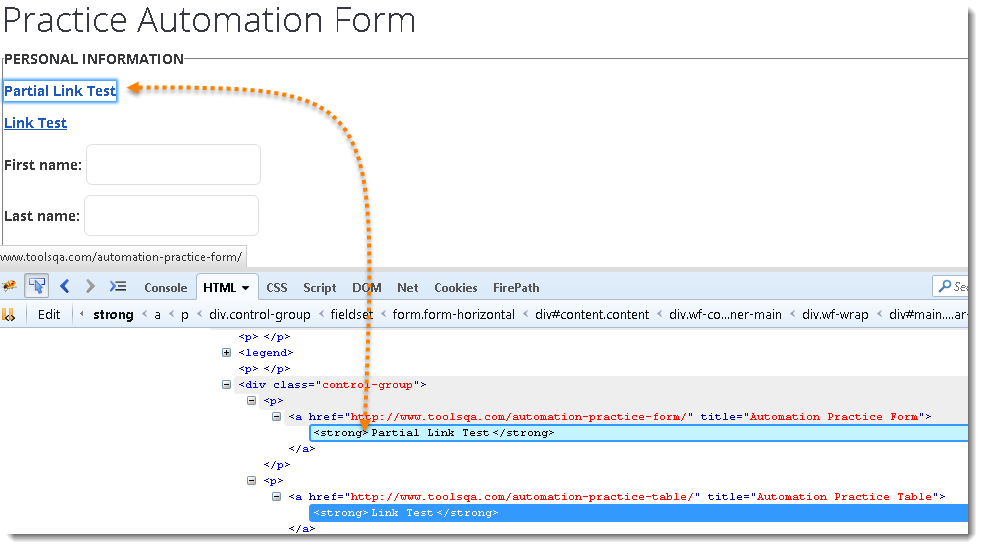
**partialLinkText(String linkText) : By** – With this you can find elements of ***“*a*” tags(*Link*)*** with the partial link names.

*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:



**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 techniques to locate elements. It gives you feature to locate single element in many ways.

## ***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*

## ***Summary***

