**Download Selenium WebDriver Java client**

[***Click here***](http://docs.seleniumhq.org/download) to go on WebDriver Java client driver download page for WebDriver download file.

## Add External Jars to Java build path

Now you need to add Selenium WebDriver’s Jar files in to Java build path.

1) Right click on Project ‘***OnlineStore***‘ > ***Select Properties*** > **Java build path.**Then navigate to ***Libraries*** tab and click ***Add External JARs***.

### Scenario to Automate

* Launch the Firefox browser
* Open website “www.techvisionit.com”
* Print a Message to display that the website is opened successfully
* Wait for 5 Seconds
* Close the Browser

So far we have not done any learning, I would suggest you to copy the below code in the class which was created in the previous chapter and run the test.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | package automationFramework;    public class FirstTestCase {    public static void main(String[] args) {    // Create a new instance of the Firefox driver  WebDriver driver = new FirefoxDriver();            //Launch the Online Store Website  driver.get("http://www.techvisionit.com");            // Print a Log In message to the screen          System.out.println("Successfully opened the website www.Store.Techvisionit.com");    //Wait for 5 Sec  Thread.sleep(5000);            // Close the driver          driver.quit();      }  } |

## 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);***

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | driver.get("http://www.google.com");    //Or can be written as    String URL = "http://www.google.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();***

|  |  |
| --- | --- |
| 1  2  3  4  5 | 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();***

|  |  |
| --- | --- |
| 1  2  3  4  5 | driver.getCurrentUrl();    //Or can be written 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();***

|  |  |
| --- | --- |
| 1  2  3  4 | 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();***

|  |  |
| --- | --- |
| 1 | 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();***

|  |  |
| --- | --- |
| 1 | driver.quit(); |

### Practice Exercise – 1

1. Launch a new Firefox browser.
2. Open Store.Techvisionit.com
3. Get Page Title name and Title length
4. Print Page Title and Title length on the Eclipse Console.
5. Get Page URL and verify if the it is a correct page opened
6. Get Page Source (HTML Source code) and Page Source length
7. Print Page Length on Eclipse Console.
8. Close the Browser.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | 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.store.techvisionit.com";    //Launch the Techvisionit WebSite  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();  }  } |

## 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.

|  |  |
| --- | --- |
| 1 | driver.navigate().to("http://www.Techvisionit.com"); |

## 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.

|  |  |
| --- | --- |
| 1 | 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 youback by one page on the browser’s history.

|  |  |
| --- | --- |
| 1 | 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.

|  |  |
| --- | --- |
| 1 | driver.navigate().refresh(); |

### Practice Exercise

1. Launch new Browser
2. Open Techvisionit.com website
3. Click on Registration link using “driver.findElement(By.*xpath*(“.//\*[@id=’menu-item-374′]/a”)).click();“
4. Come back to Home page (Use ‘Back’ command)
5. Again go back to Registration page (This time use ‘Forward’ command)
6. Again come back to Home page (This time use ‘To’ command)
7. Refresh the Browser (Use ‘Refresh’ command)
8. Close the Browser

## 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();***

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 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”);***

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | WebElement element = driver.findElement(By.id("UserName"));  element.sendKeys("Techvisionit");    //Or can be written as    driver.findElement(By.id("UserName")).sendKeys("Techvisionit"); |

## Click Command

***click( ) : void*** – This simulates the clicking of any element. Accepts nothing as a parameter and returns nothing.

**Command**– ***element.click();***

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | WebElement element = driver.findElement(By.linkText("Techvisionit"));  element.click();    //Or can be written as    driver.findElement(By.linkText("Techvisionit")).click(); |

## 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();***

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | WebElement element = driver.findElement(By.id("UserName"));  boolean status = element.isDisplayed();    //Or can be written as    boolean staus = driver.findElement(By.id("UserName")).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();***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | 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("Techvisionit");  } |

## 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();***

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 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();***

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 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();***

|  |  |
| --- | --- |
| 1  2 | 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();***

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 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();***

|  |  |
| --- | --- |
| 1  2 | 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();***

|  |  |
| --- | --- |
| 1  2  3 | 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();***

|  |  |
| --- | --- |
| 1  2  3 | WebElement element = driver.findElement(By.id("SubmitButton"));  Point point = element.getLocation();  System.out.println("X cordinate : " + point.x + "Y cordinate: " + point.y); |

## 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.



**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:

|  |  |
| --- | --- |
| 1  2  3 | WebElement element = driver.findElement(By.id("submit"));  // Action can be performed on Input Button element  element.submit(); |

## 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 attributeand 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.

|  |  |
| --- | --- |
| 1  2  3 | WebElement element = driver.findElement(By.name("firstname"));  // Action can be performed on Input Text element  element.sendKeys("Techvisionit"); |

## By ClassName

***className(String className) : By*** – This finds elements based on the value of the CLASSattribute. 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:

|  |  |
| --- | --- |
| 1  2  3 | WebElement parentElement = driver.findElement(By.className("button"));  WebElement childElement = parentElement.findElement(By.id("submit"));  childElement.submit(); |

## 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:

|  |  |
| --- | --- |
| 1  2  3 | 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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 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**](http://toolsqa.com/selenium-webdriver/choosing-effective-xpath/) which we will come across during our learning journey on Techvisionit 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.

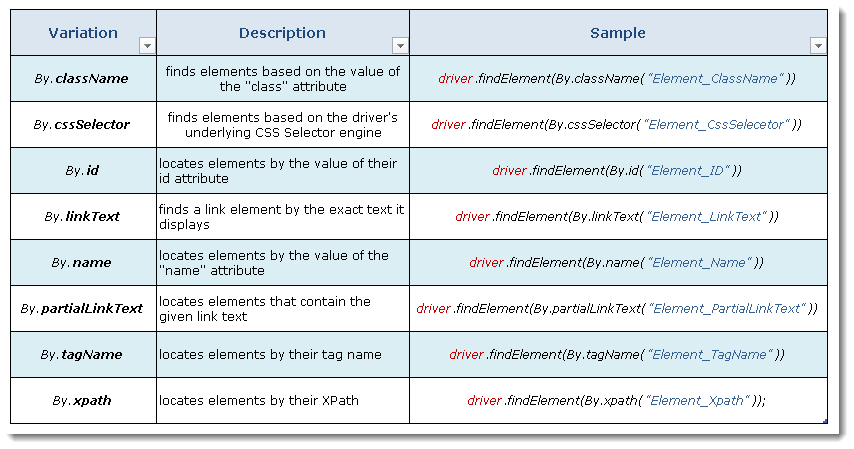
***findElement()***

* 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

## Summary



### Practice Exercise 1

1. Launch new Browser
2. Open URL http://techvisionit.com/automation-practice-form/
3. Type Name & Last Name (Use Name locator)
4. Click on Submit button (Use ID locator)

### Practice Exercise 2

1. Launch new Browser
2. Open URL http://techvisionit.com/automation-practice-form/
3. Click on the Link “Partial Link Test” (Use ‘patialLinkTest’ locator and search by ‘Partial’ word)
4. Identify Submit button with ‘tagName’, convert it in to string and print it on the console
5. Click on the Link “Link Test” (Use ‘linkTest’ locator)

## DropDown & Multiple Select Operations

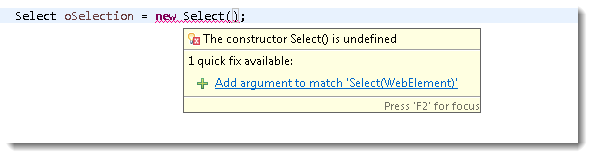
It is just an ordinary operation like selecting any other type of element on a webpage. You can choose it by ID, Name, Css & Xpath etc. But to perform any action on this element it is required to import ‘***import org.openqa.selenium.support.ui.Select***' package and to use it we need to create a new ***Select Object*** of class ***Select***.

### Select Class in Selenium

Models a ***SELECT*** tag, providing helper methods to select and deselect options. Select is a class which is provided by Selenium to perform multiple operations on DropDown object and Multiple Select object. This class can be found under the ***Selenium’s Support.UI.Select***package. As Select is also an ordinary class, so it’s object is also created by a New keyword with regular class creation syntax.

**Select oSelect = new Select());**

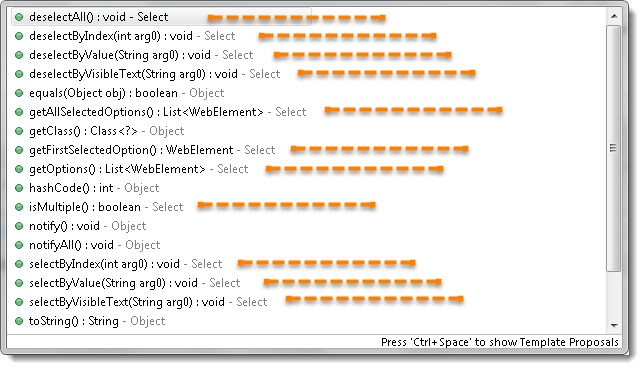
The above code will generate compile time error in Eclipse, as Select() is asking for constructor. Bring the cursor over Select(), Eclipse will populate a suggestion.



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | WebElement element = driver.findElement(By.id("Country"));  Select oSelect = new Select(element);    //Or it can be also written as    Select oSelect = new Select(driver.findElement(By.id("Country"))); |

***Note:***Select class only works for elements with <select> tags.

Now, once you got the oSelect object which is a SELECT object, you can access all the methods resides in side the SELECT class by typing ***oSelect + dot***.



## Different Select Commands

In this chapter we will learn how to deal with DropDown and Multi Select elements. There will be many interesting operations are available on these elements. But you may be wondering that how a DropDown looks like in the HTML code. Will use the same example for the reference of different select commands.

## selectByVisibleText

***selectByVisibleText(String arg0) : void***– It is very easy to choose or select an option given under any dropdowns and multiple selection boxes with selectByVisibleText method. It takes a parameter of String which is one of the Value of Select element and it returns nothing.

**Command**– ***oSelect.selectByVisibleText(“text”);***

***Example –*** Refer the above Screen shot of YEAR Drop Down\*

***Code –***To select the value 2010.

|  |  |
| --- | --- |
| 1  2  3 | Select oSelect = new Select(driver.findElement(By.id("yy\_date\_8")));    oSelect.selectByVisibleText("2010"); |

## selectByIndex

***selectByIndex(int arg0) : void***– It is almost the same as selectByVisibleText but the only difference here is that we provide the index number of the option here rather the option text.It takes a parameter of int which is the index value of Select element and it returns nothing.

**Command**– ***oSelect.selectByIndex(int);***

***Example –*** Refer the above Screen shot of YEAR Drop Down\*

|  |  |
| --- | --- |
| 1  2  3 | Select oSelect = new Select(driver.findElement(By.id("yy\_date\_8")));    oSelect.selectByIndex(4); |

***Note***: Index starts from Zero, so the fifth position value will be at index 4.

## selectByValue

***selectByValue(String arg0) : void***–It is again the same what we have discussed earlier, the only difference in this is that it ask for the value of the option rather the option text or index. It takes a parameter of String which is on of the value of Select element and it returns nothing.

**Command**– ***oSelect.selectByValue(“text”);***

***Example –*** Refer the above Screen shot of YEAR Drop Down\*

|  |  |
| --- | --- |
| 1  2  3 | Select oSelect = new Select(driver.findElement(By.id("yy\_date\_8")));    oSelect.selectByValue("2014"); |

## getOptions

***getOptions( ) : List<WebElement>***–This gets the all options belonging to the Select tag. It takes no parameter and returns List<WebElements>.

**Command**– ***oSelect.getOptions();***

Sometimes you may like to count the element in the dropdown and multiple select box, so that you can use the loop on Select element.

***Example –*** Refer the above Screen shot of YEAR Drop Down\*

|  |  |
| --- | --- |
| 1  2  3 | Select oSelect = new Select(driver.findElement(By.id("yy\_date\_8")));  List <WebElement> elementCount = oSelect.getOptions();  System.out.println(elementCount.size()); |

## Print all the Options

***Code –***To get the ***Count*** of the total elements inside SELECT and to ***Print*** the text value of every element present in the SELECT.

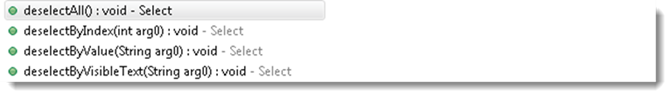
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | Select oSelect = new Select(driver.findElement(By.id("yy\_date\_8")));  List <WebElement> elementCount = oSelect.getOptions();  int iSize = elementCount.size();    for(int i =0; i<iSize ; i++){  String sValue = elementCount.get(i).getText();  System.out.println(sValue);  } |

All of the above methods work on both Dropdown and Multiple select box.

## DeSelect Methods

The way we select different values of DropDown & Multi Select, the same way we can also ***deselect*** the values. But the only challenge in these methods are they do not work for DropDown and only work for Multi Select elements.

In case you want to deselect any pre-selected option, that can be done with either deselectAll(), deselectByIndex, deselectByValue and deselectByVisibletext.



Multi Select element look like this:

***deselectAll( ) : void***– Clear all selected entries. This is only valid when the SELECT supports multiple selections.

**Command**– ***oSelect.deselectAll;***

***deselectByIndex(int arg0) : void***–Deselect the option at the given index.

**Command**– ***oSelect.deselectByIndex;***

***deselectByValue(String arg0) : void***–Deselect all options that have a value matching the argument.

**Command**– ***oSelect.deselectByValue;***

***deselectByVisibleText(String arg0) : void***– Deselect all options that display text matching the argument.

**Command**– ***oSelect.deselectByVisibleText***

## isMultiple

***isMultiple( ) : boolean***– This tells whether the SELECT element support multiple selecting options at the same time or not. This accepts nothing but returns boolean value(true/false).

**Command**– ***oSelect.isMultiple();***

This is done by checking the value of the “multiple” attribute.

***Example –*** Refer the above Screen shot of MULTI SELECT for **multiple** attribute\*

## Multi Select Methods

This one also just works on Multiple selection boxes and not on regular List boxes or dropdowns. There is no additional logic behind selecting multiple options of Select element. All you need to do is to fire select commands on multiple elements one by one that’s it.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | Select oSelect = new Select(driver.findElement(By.id(Element\_ID)));  oSelect.selectByIndex(index)  oSelect.selectByIndex(index)    // Or can be used as    oSelect.selectByVisibleText(text)  oSelect.selectByVisibleText(text)    // Or can be used as    oSelect.selectByValue(value)  oSelect.selectByValue(value) |

### Practice Exercise -1 (Drop Down Box/List)

1. Launch new Browser
2. Open “http://techvisionit.com/automation-practice-form/”
3. Select ‘Continents’ Drop down ( Use Id to identify the element )
4. Select option ‘Europe’ (Use selectByIndex)
5. Select option ‘Africa’ now (Use selectByVisibleText)
6. Print all the options for the selected drop down and select one option of your choice
7. Close the browser

### Practice Exercise -2 (Multiple Selection Box/List)

1. Launch new Browser
2. Open “http://techvisionit.com/automation-practice-form/”
3. Select ‘Selenium Commands’ Multiple selection box ( Use Name locator to identify the element )
4. Select option ‘Browser Commands’  and then deselect it (Use selectByIndex and deselectByIndex)
5. Select option ‘Navigation Commands’  and then deselect it (Use selectByVisibleText and deselectByVisibleText)
6. Print and select all the options for the selected Multiple selection list.
7. Deselect all options
8. Close the browser