

**Softech Solutions Inc.**

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

Email: saney.alam@softechsolution.tech

*This part is a step-by-step explanation of different Selenium webdriver command*

WebDriver Commands Part 2

***Lecture Notes***

WebDriver Commands Part 2

|  |  |
| --- | --- |
| ***Table of Contents Page*** | |
| *CheckBox & Radio Button Operations* |  |
| *By Id* | ***02*** |
| *With IsSelected* | ***02*** |
| *With Value* | ***03*** |
| *By CssSelector* | ***04*** |
| *Drop Down & Multiple Select Operations* |  |
| *Drop Down & Multiple Select Operations* | ***05*** |
| *Different Select Commands* | ***06*** |
| *selectByVisibleText Command* | ***07*** |
| *selectByIndex Command* | ***07*** |
| *selectByValue* | ***08*** |
| *getOptions* | ***08*** |
| *Print all the Options* | ***09*** |
| *DeSelect Methods* | ***09*** |
| *isMultiple* | ***10*** |
| *Multi Select Methods* | ***11*** |
| *Handle Dynamic WebTables in Selenium WebDriver* |  |
| *What is table in HTML?* | ***11*** |
| *Handle Dynamic WebTables* | ***12*** |
|  |  |

**Checkbox & Radio button Operations**

**Checkbox & Radio Button Operations** are easy to perform and most of the times the simple **ID attributes** work fine for both of these. But ***selection*** and ***d-selection*** is not the only thing we want with ***Check Boxes*** and ***Radio Buttons***. We might like to check that if the ***Check Box*** is already checked or if the ***Radio Button*** is selected by default or anything. ***Check Boxes*** and ***Radio Button*** deals exactly the same way and you can perform below mentioned operations on either of them.

## ***Different Selection Method***

## ***By ID***

If **ID** is given for the ***Radio Button/Checkbox*** and you just want to click on it irrespective of its value, then the command will be like this:

*WebElement radioBtn = driver.findElement(By.id("someId"));*

*radioBtn.click();*

## ***With IsSelected***

If your choice is based on the ***pre-selection*** of the ***Radio Button/Check Box*** and you just need to select the deselected ***Radio Button/Check Box***. Assume there are two ***Radio Buttons/Check Boxes***, one is selected by default and you want to select the other one for your test. With **IsSelected** **statement, you can get to know that the element is selected or not.**

*// Store all the elements of same category in the list of WebLements*

*List oRadioButton = driver.findElements(By.name("toolsqa"));*

*// Create a boolean variable which will hold the value (True/False)*

*boolean bValue = false;*

*// This statement will return True, in case of first Radio button is selected*

*bValue = oRadioButton.get(0).isSelected();*

*// This will check that if the bValue is True means if the first radio button is selected*

*if(bValue = true){*

*// This will select Second radio button, if the first radio button is selected by default*

*oRadioButton.get(1).click();*

*}else{*

*// If the first radio button is not selected by default, the first will be selected*

*oRadioButton.get(0).click();*

*}*

**Note:** *Name is always same for the same group of Radio Buttons/Check Boxes but their Values are different. So if you find the element with the name attributes then it means that it may contain more than one element, hence we need to use findElements method and store the list of WebElements.*

## ***With Value***

You can even select ***Radio Buttons/Check Boxes*** with their Values.

*// Find the checkbox or radio button element by Name*

*List oCheckBox = driver.findElements(By.name("tool"));*

*// This will tell you the number of checkboxes are present*

*int iSize = oCheckBox.size();*

*// Start the loop from first checkbox to last checkboxe*

*for(int i=0; i < iSize ; i++ ){*

*// Store the checkbox name to the string variable, using 'Value' attribute*

*String sValue = oCheckBox.get(i).getAttribute("value");*

*// Select the checkbox it the value of the checkbox is same what you are looking for*

*if (sValue.equalsIgnoreCase("toolsqa")){*

*oCheckBox.get(i).click();*

*// This will take the execution out of for loop*

*break;*

*}*

*}*

## ***By CssSelector***

A simple way of selecting a check-box or radio button is by using its value:

*WebElement oCheckBox = driver.findElement(By.cssSelector("input[value='Tools QA']"));*

*oCheckBox.click();*

**Drop Down & Multiple Select Operations**

Just like ***Checkbox & Radio Buttons***, **Dropdown** & **Multiple Select** Operations also works together and almost the same way. To perform any action, the first task is to identify the element group. I am saying it a group, as ***Dropdown*** /***Multiple Select*** is not a single element. They always have a single name but and they contains one or more than one elements in them. I should rather say more than one option in ***Dropdown*** and ***Multiple Select***. The only difference between these two is deselecting statement & multiple selections are not allowed on ***DropDown***. Let’s look at the different operations:

## ***DropDown & Multiple Select Operations***

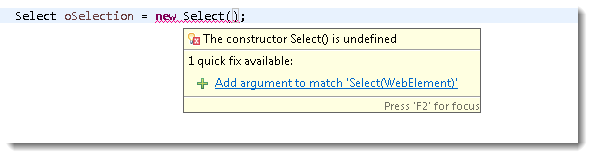
It is just an ordinary operation like selecting any other type of element on a webpage. You can choose it by ***ID, Name, and Css & Xpath*** etc. But to perform any action on this element it is required to import ‘***import org.openqa.selenium.support.ui.Select***' package and to use it we need to create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



It clearly says that Select is asking for an element type object for its constructor. The code will be:

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

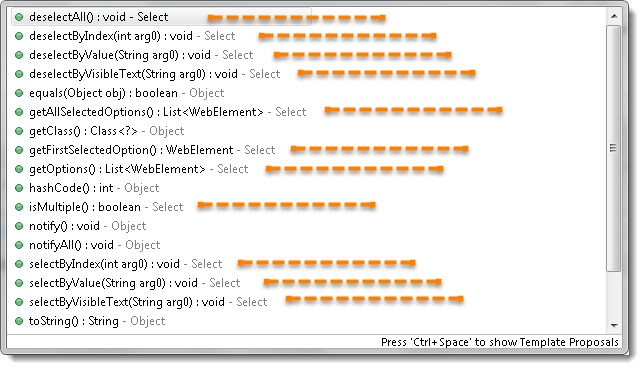
*Select oSelect = new Select(element);*

*//Or it can be also written as*

*Select oSelect = new Select(driver.findElement(By.id("Country")));*

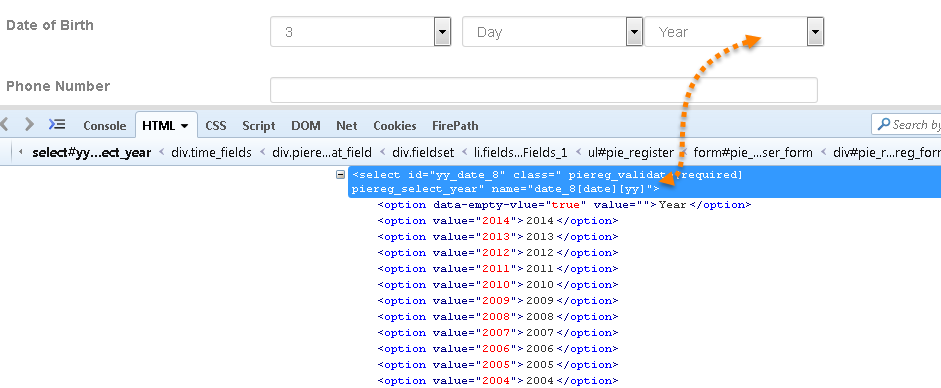
**Note:***Select class only works for elements with <select> tags.*

Now, once you got the oSelect object which is a ***SELECT object***, you can access all the methods resides in side the ***SELEC T***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.

*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\****

**Code –** To select the value 2010 using index.

*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 one of the value of ***Select element***and it returns nothing.

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

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

**Code –** To select the value 2014.

*Select oSelect = new Select(driver.findElement(By.id("yy\_date\_8")));*

*oSelect.selectByValue("2014");*

**Note:***The value of an option and the text of the option may not be always same and there can be a possibility that the value is not assigned to Select WebElement. If the value is given in the Select tag then only you can use the selectByValue method.*

## ***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\****

**Code –** To get the **Count** of the total elements inside ***SELECT***.

*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.***

*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);*

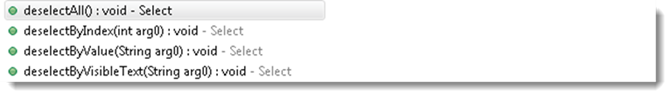
*}*

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

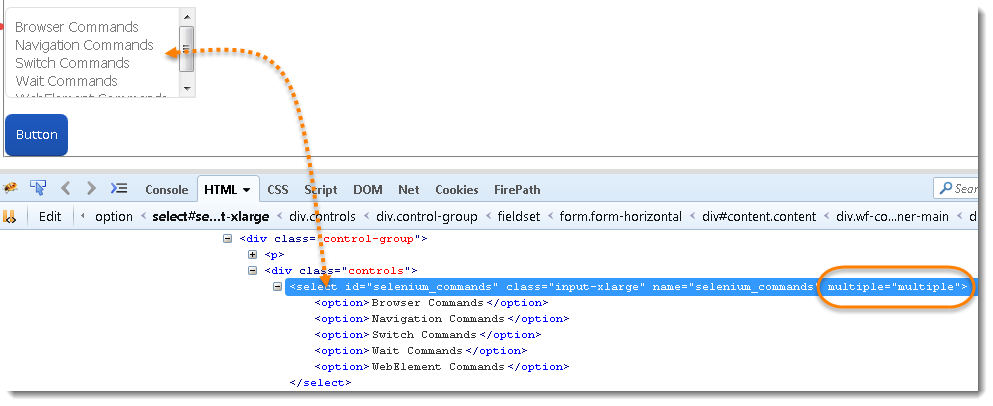
## ***DeSelect Methods***

The way we select different values of Dropdown & Multi Select, the same way we can also deselect the values. But the only challenges 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 looks 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 by 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.

*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)*

**Handle Dynamic Web Tables in Selenium WebDriver**

## ***What is Table in HTML?***

**Table** is a kind of **HTML** data which is displayed with the help of**<table> tag** in conjunction with the ***<tr>*** and ***<td>*** tags. Although there are other tags for creating tables, these are the basics for creating a table in HTML. Here tag **<tr>** defines the **row** and tag**<td>** defines the **column** of the table.

Excel sheet is a simple example of table structures. Whenever we put some data in to excel we give them some heading as well. In HTML we use **<th> tag for headings** which defines heading of the table.

Each cell in the Excel sheet can be represented as ***<td>*** in the HTML table. The ***<td>*** elements are the data containers and these can contain all sorts of HTML elements like text, images, lists, other tables, etc.

Look at the below simple example of HTML table where first row is defined as header and later two rows are containing data.

*<table>*

*<tbody>*

*<tr>*

*<th>Automation Tool</th>*

*<th>Licensing</th>*

*<th>Market response</th>*

*</tr>*

*<tr>*

*<td>Selenium</td>*

*<td>Free</td>*

*<td>In</td>*

*</tr>*

*<tr>*

*<td>QTP</td>*

*<td>Paid</td>*

*<td>Out</td>*

*</tr>*

*</tbody>*

*</table>*

|  |  |  |
| --- | --- | --- |
| **Automation Tool** | **Licensing** | **Market response** |
| Selenium | Free | In |
| QTP | Paid | Out |

## ***Handle Dynamic Web Tables in Selenium Webdriver***

There is no rocket science in handling of tables. All you need to is to inspect the table cell and get the HTML location of it. In most cases tables contain text data and you might simply like to extract the data given in the each row or column of the table. But sometimes tables have link or images as well, and you can easily able to perform any action on those elements if you can find the HTML location of the containing cell.

***Example 1:***

Let’s take an example of above table and choose Column 1 of Row 2 which is ‘Selenium’ in above case:

*/html/body/div[1]/div[2]/div/div[2]/article/div/table/tbody/tr[2]/td[1]*

If we divide this xpath in to three different parts it will be like this

* **Part 1** ***–*** *Location of the table in the webpage* ***</html/body/div[1]/div[2]/div/div[2]/article/div/>***
* **Part 2** ***–*** *Table body (data) starts from here* ***<table/tbody/>***
* **Part 3** ***–*** *It says table row 2 and table column 1* ***<tr[2]/td[1]>***

If you use this xpath you would be able to get the Selenium cell of the table. Now what? How to get the text ‘Selenium’ from the table cell?

You need to use the ‘getText()’ method of the WebDriver element.

***driver.findElement(By.xpath(“/html/body/div[1]/div[2]/div/div[2]/article/div/table/tbody/tr[2]/td[1]”)).getText();***

### ***Example 2:***

Table operations are not always so simple like above because tables can contain a large amount of data and may be you need to pass rows and columns dynamically in your test case.

In that case you need to build your xpath with using variables and you will pass rows and columns to your xpath in the form of variables.

*String sRow = "2";*

*String sCol = "1";*

*driver.findElement(By.xpath("/html/body/div[1]/div[2]/div/div[2]/article/div/table/tbody/tr["+sRow+"]/td["+sCol+"]")).getText();*

### ***Example 3:***

The above example is still easy as at-least you know the row number and the column number to be fetched from the table and you are able to provide it in the xpath from external excel sheet or any source of data sheet. But what would you do when the row and columns are itself dynamic and all you know is the Text value of any cell only and you like to take out the correspondence values of that particular cell.

For example all you know is the Text ‘Licensing’ in the above example and you like to record the value for that particular column only such as Free & Paid.

*String sColValue = "Licensing";*

*//First loop will find the 'ClOCK TWER HOTEL' in the first column*

*for (int i=1;i<=3;i++){*

*String sValue = null;*

*sValue = driver.findElement(By.xpath(".//\*[@id='post-2924']/div/table/tbody/tr[1]/th["+i+"]")).getText();*

*if(sValue.equalsIgnoreCase(sColValue)){*

*// If the sValue match with the description, it will initiate one more inner loop for all the columns of 'i' row*

*for (int j=1;j<=2;j++){*

*String sRowValue= driver.findElement(By.xpath(".//\*[@id='post-2924']/div/table/tbody/tr["+j+"]/td["+i+"]")).getText();*

*System.out.println(sRowValue);*

*}*

*break;*

*}*

*}*