SELENIUM WEBELEMENTS COMMANDS – PART 2

Lecture Notes

A COMPLETE SELENIUM GUIDE

Softech Solutions Inc.

www.softechsolutionsgroup.com

saney.alam@softechsolutionsgroup.com

Selenium WebElements Commands – Part 2

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.

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("Male"));
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("Male"));

 // 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 attribute 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("Female")){
                              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=Male]"));

 oCheckBox.click();
```

DropDown & 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.*

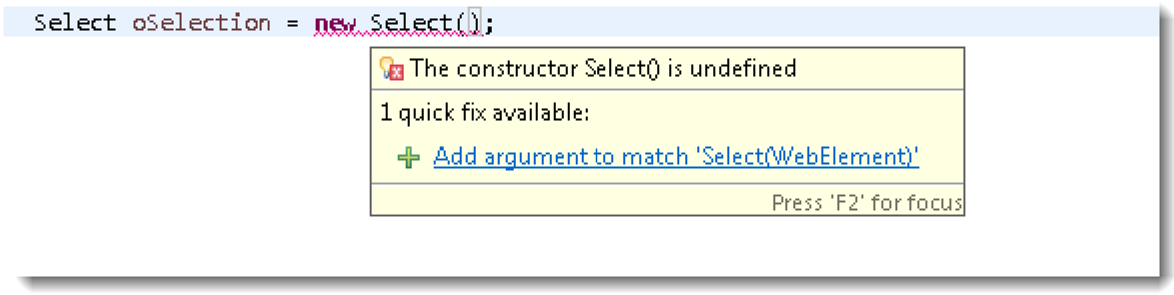
It is just an ordinary operation like selecting any other type of element on a webpage. You can choose it by **ID, Name, Css & Xpath** *etc. But to perform any action on this element it is required to import '*`import org.openqa.selenium.support.ui.Select'` *package and to use it we need to create a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



```
Select oSelection = new Select();
```

The constructor Select() is undefined

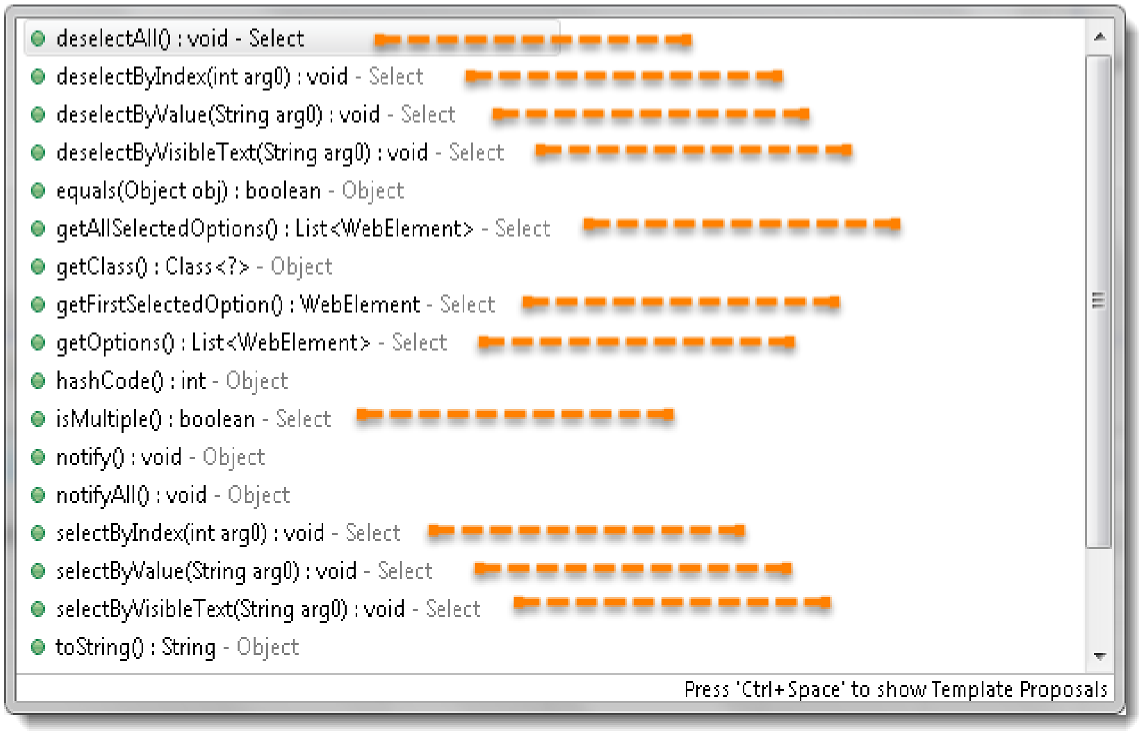
1 quick fix available:

It clearly says that Select is asking for a 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")));
```

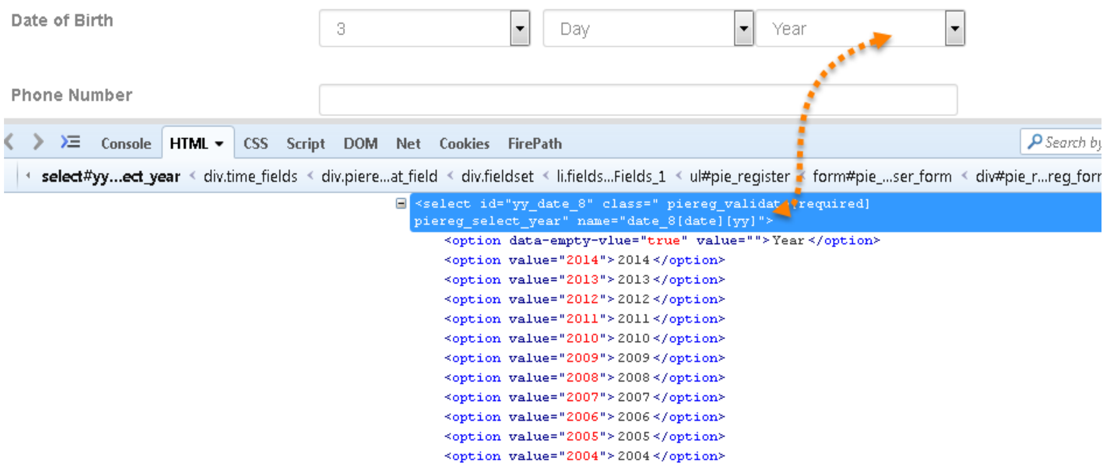
Note: Select class only works for elements with <select> tags.

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



```
deselectAll() : void - Select
deselectByIndex(int arg0) : void - Select
deselectByValue(String arg0) : void - Select
deselectByVisibleText(String arg0) : void - Select
equals(Object obj) : boolean - Object
getAllSelectedOptions() : List<WebElement> - Select
getClass() : Class<?> - Object
getFirstSelectedOption() : WebElement - Select
getOptions() : List<WebElement> - Select
hashCode() : int - Object
isMultiple() : boolean - Select
notify() : void - Object
notifyAll() : void - Object
selectByIndex(int arg0) : void - Select
selectByValue(String arg0) : void - Select
selectByVisibleText(String arg0) : void - Select
toString() : String - Object
Press 'Ctrl+Space' to show Template Proposals
```

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.



Date of Birth
3
Day
Year
Phone Number
Console
HTML
CSS
Script
DOM
Net
Cookies
FirePath
Search by
select#yy...ect_year < div.time_fields < div.piere...at_field < div.fieldset < li.fields...Fields_1 < ul#pie_register < form#pie_...ser_form < div#pie_r...reg_forr

```
<select id="yy_date_8" class=" piereg_validat [required]
piereg_select_year" name="date_8[date][yy]">
    <option data-empty-vlue="true" value="">Year</option>
    <option value="2014">2014</option>
    <option value="2013">2013</option>
    <option value="2012">2012</option>
    <option value="2011">2011</option>
    <option value="2010">2010</option>
    <option value="2009">2009</option>
    <option value="2008">2008</option>
    <option value="2007">2007</option>
    <option value="2006">2006</option>
    <option value="2005">2005</option>
    <option value="2004">2004</option>
```

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 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*
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.***

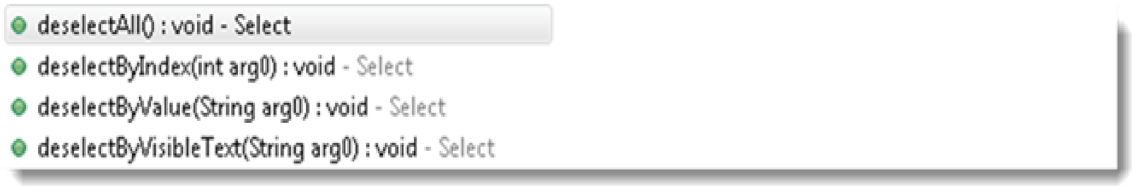
```
Select oSelect = new Select(driver.findElement(By.id("yy_date_8")));
List <WebElement> elementCount = oSelect.getOptions();
int iSize = elementCount.size();

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 boxes.

DeSelect Methods

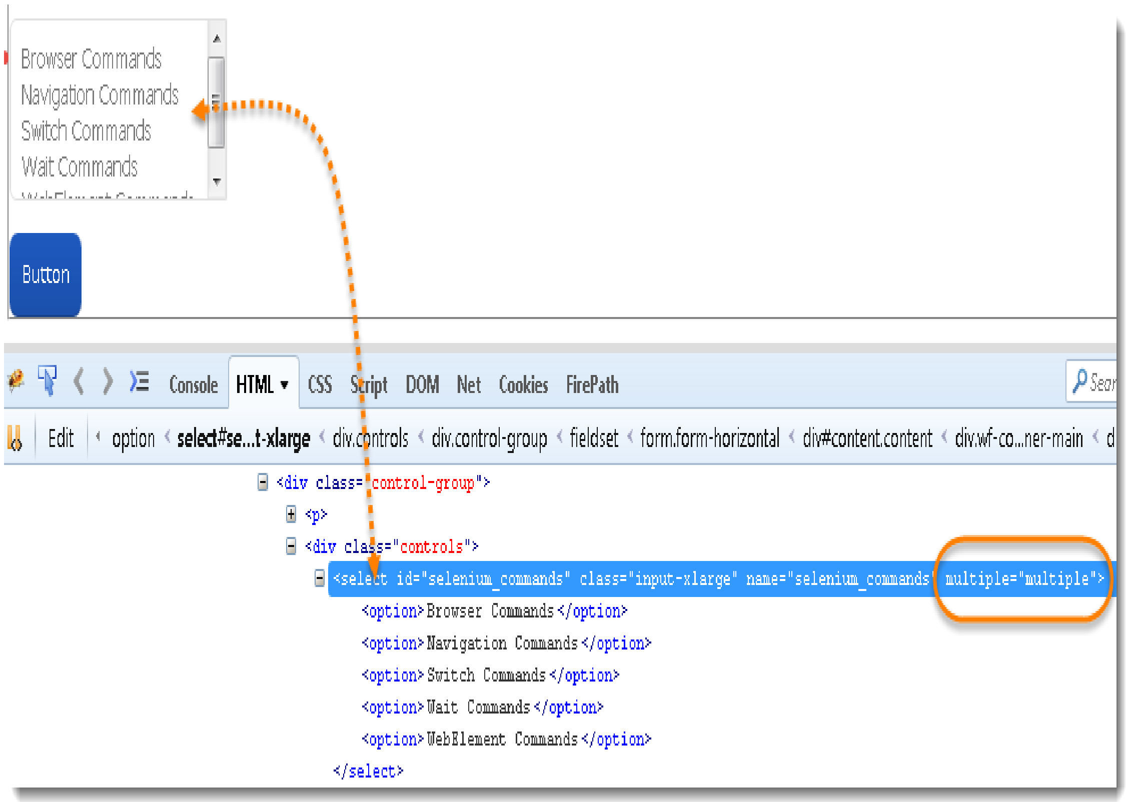
The way we select different values of ***DropDown*** & ***Multi Select***, the same way we can also **deselect** the values. But the only challenge in these methods is they do not work for ***DropDown*** and only work for ***Multi Select*** elements.

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



```
deselectAll() : void - Select
deselectByIndex(int arg0) : void - Select
deselectByValue(String arg0) : void - Select
deselectByVisibleText(String arg0) : void - Select
```

Multi Select element looks like this:



```
<div class="control-group">
  <p>
  <div class="controls">
    <select id="selenium_commands" class="input-xlarge" name="selenium_commands" multiple="multiple">
      <option>Browser Commands</option>
      <option>Navigation Commands</option>
      <option>Switch Commands</option>
      <option>Wait Commands</option>
      <option>WebElement Commands</option>
    </select>
```

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.

```
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)
```