**Ways to select the options of Dropdown in Selenium Webdriver:**

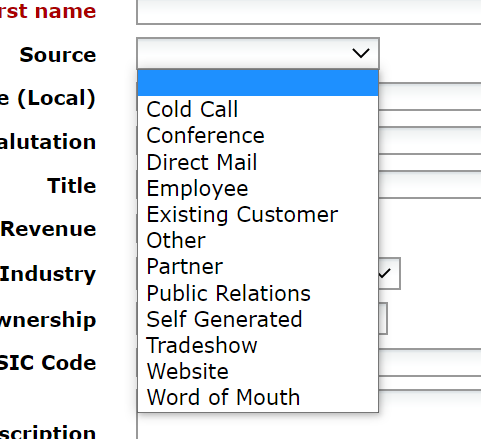
 Dropdown box are most widely used in all applications to pick a particular value from multiple options under a category. The selection can be of Single select or Multiple selection

Fig:1

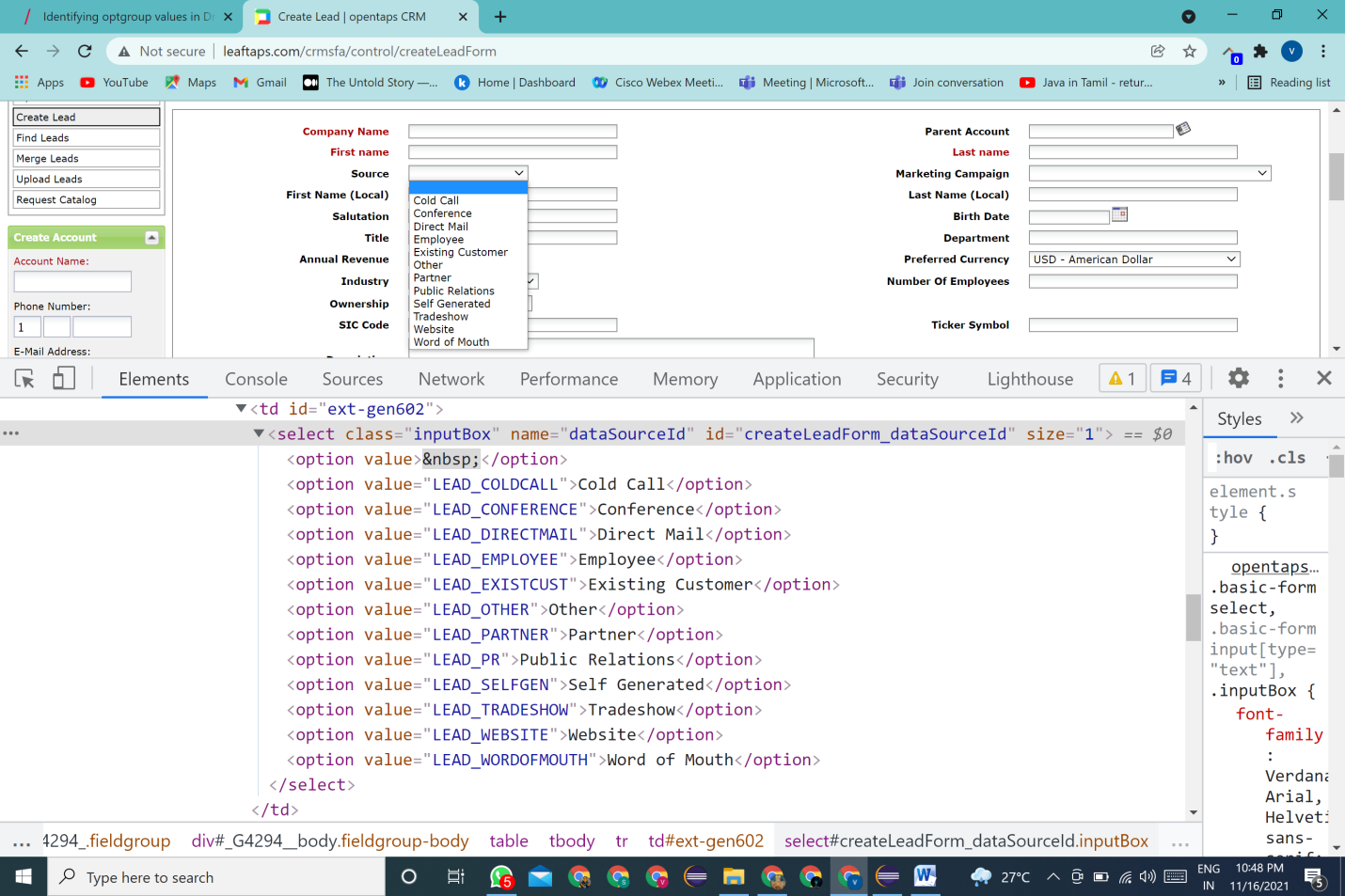
 Let’s have a look into the DOM Structure of dropdown of the above given Fig.1

Fig:2

Fig:2 Explains the DOM Structure of the dropdown element.

Dropdowns are usually identified in the DOM with the html Tag name <Select> and the options are identified with tag name <Options>.In some web applications, we may also find the dropdowns with tag names<li> and <ul> in the Dom. Let’s look into, how selenium helps us to handle these kinds of element in the Web Application.

Selenium Webdriver provides us the class **Select** which helps us to custom our selection with different helper methods to select and deselect the options provided with the web element.

If we want to use the select html tag in our automation selenium script, we have to first instantiate the Select Class of Selenium with **new** Keyword. The syntax is

**Select** options =new **Select** (WebElement);

**import** org.openqa.selenium.support.ui.Select;

(Select class should be imported from selenium.support.ui.Select)

**Different Helper Methods:**  To Select and deselect the option, Select class provides the following methods.

**To Select Single value in dropdown:**

***Method: 1 -> selectbyIndex (int index);***

Usage of this method is to select the required option using index value. Index of Select class starts from 0.It does not return any value.

Select options=new Select(sourceId)  
 options .selectbyindex (1);-> select the second option(ColdCall) in the source tab(ref Fig:1)

***Method: 2 -> selectbyValue (String Value);***

This method is to select the required option using the attribute and its value. It takes String as argument.

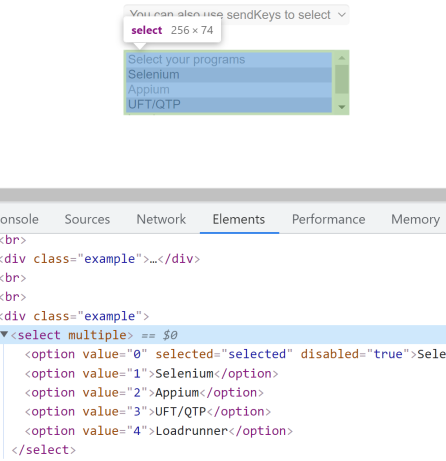
Select options=new Select(sourceId)  
 options .selectbyValue(“ColdCall”);

***Method:3->selectByVisibleText(String text);***

SelectbyVisisbleText provides the way to pick the argument with the given text provided from the user which is present in the DOM

Select options=new Select(sourceId)  
 options.selectByVisibleText (“ColdCall”);->Pick the text from the options which matches the text in the DOM

**To Select Multiple Values in dropdown:**

****

**Fig:3**

Fig:3 explains the multi-select option in a dropdown. The availability to select multiple option can be identified with the <select> tag with multiple attribute. To validate the dropdown allow the user to multi-select the webElements, the method **isMultiple()** can be used.

WebElement ele=driver.findElement(By.xpath(“//div[@class='example'][6]/select”)

**ele.isMultiple();** ->returns the Boolean value. After validating the element is available for multiselect, the user can select the values with required methods.To opt for multi-select the options,

Select options=new Select(ele)  
 options .selectbyValue(“3”);

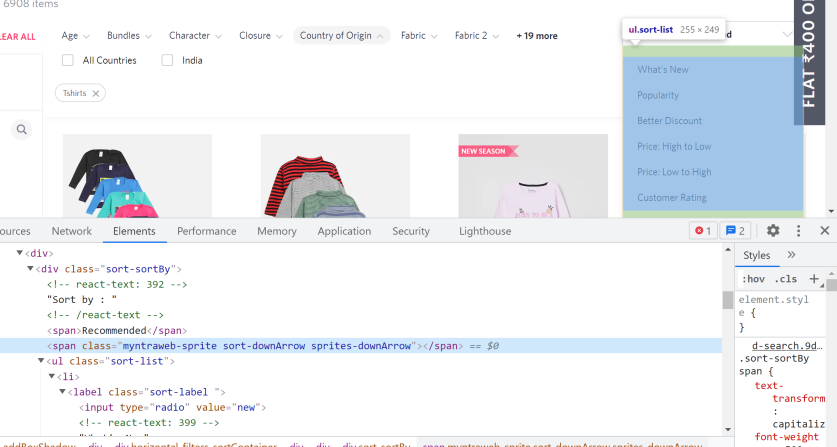
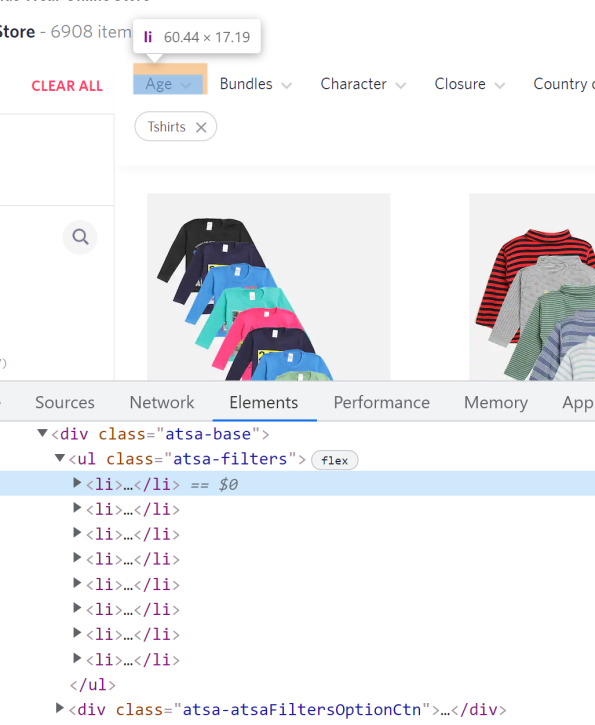
options.selectbyindex(2);

options.selectByVisibleText (“Appium”);

Select class also provides methods to retrieve the data of selected options and also deselect the options from the dropdown. Below methods allow to the user for Multiple Selections

* options.getAllSelectedOptions();-> It returns all the selected options as List
* options.getFirstSelectedOptions(); It returns the first selected or current option
* options.getOptions();-> It returns all the available options in the dropdown as List
* options.deselectAll();-> It clears all the selected options in the dropdown

Other than the dropdown with select tag, the web page will also contains <ul> and <li> tags to pick the value under a list of elements. In these types, the user can directly use xpath to select the values.



In above examples, the element can be selected by using the xpath

List<WebElement> sort=driver.findElements(By.className("sort-sortBy"));

From the list values, the required value will be selected by

for (WebElement eachBrand : sort) {

String brand = eachBrand.getText();

System.out.println("BrandName:" + brand);

}