**Find Element and FindElements by XPath in Selenium WebDriver:**

Why do you need Find Element/s command?

Interaction with a web page requires a user to locate the web element. Find Element command is used to uniquely identify a (one) web element within the web page. Whereas, Find Elements command is used to uniquely identify the list of web elements within the web page. There are multiple ways to uniquely identify a web element within the web page such as ID, Name, Class Name, Link Text, Partial Link Text, Tag Name and XPATH.

FindElement command syntax:

Selenium Find Element command takes in the By object as the parameter and returns an object of type WebElement in Selenium. By object in turn can be used with various locator strategies such as find element by ID Selenium, Name, Class Name, XPATH etc. Below is the syntax of FindElement command in Selenium web driver.

WebElement elementName = driver.findElement(By.LocatorStrategy("LocatorValue"));

Locator Strategy can be any of the following values.

ID

Selenium find element by Name

Class Name

Tag Name

Link Text

Partial Link Text

XPATH

Locator Value is the unique value using which a web element can be identified. It is the responsibility of developers and testers to make sure that web elements are uniquely identifiable using certain properties such as ID or name.

Example:

WebElement loginLink = driver.findElement(By.linkText("Login"));

**FindElements command syntax:**

FindElements in Selenium command takes in By object as the parameter and returns a list of web elements. It returns an empty list if there are no elements found using the given locator strategy and locator value. Below is the syntax of find elements command.

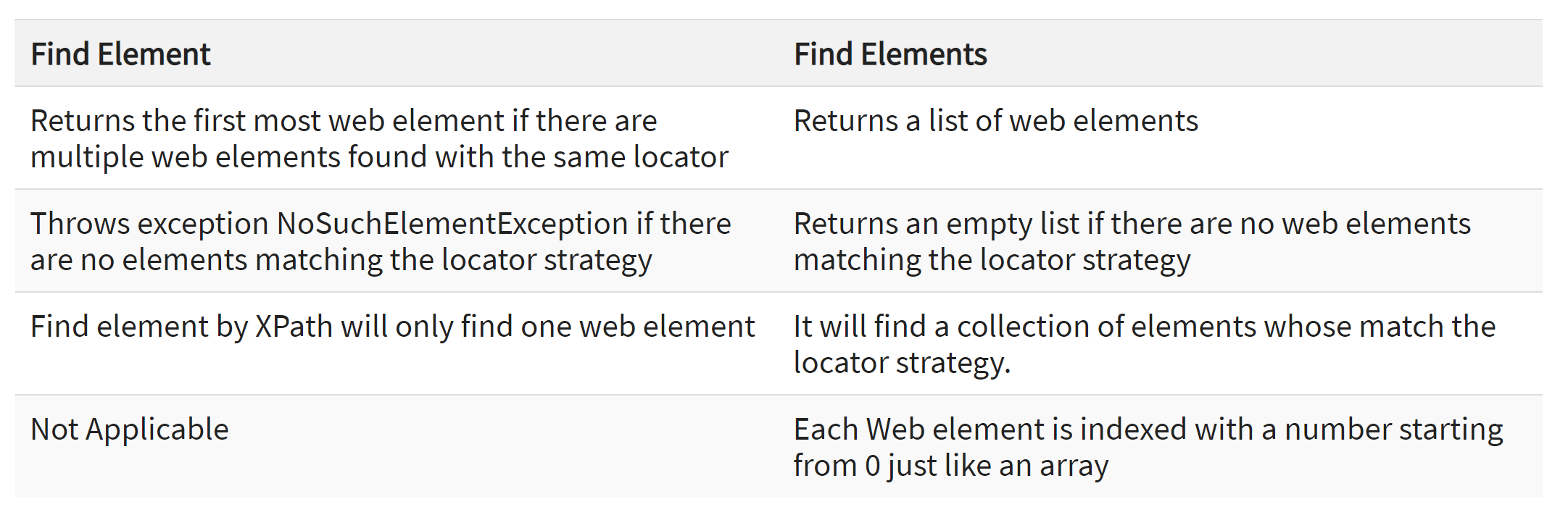
List<WebElement> elementName = driver.findElements(By.LocatorStrategy("LocatorValue"));

Example:

List<WebElement> listOfElements = driver.findElements(By.xpath("//div"));

Find element Vs Find elements

Below are the major differences between find element and find elements commands.



**Example: How to use Find Element command**

package com.sample.stepdefinitions;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class NameDemo {

public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver", "D:\\3rdparty\\chrome\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

**driver.manage().window().maximize();**

driver.get("http://demo.guru99.com/test/ajax.html");

// Find the radio button for “No” using its ID and click on it

driver.findElement(By.id("no")).click();

driver.findElement(By.id("buttoncheck")).click();} }

**Example: How to use Find Elements command**

package com.sample.stepdefinitions;

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class NameDemo {

public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver", "X://chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("http://demo.guru99.com/test/ajax.html");

List<WebElement> elements = driver.findElements(By.name("name"));

System.out.println("Number of elements:" +elements.size());

for (int i=0; i<elements.size();i++){

System.out.println("Radio button text:" + elements.get(i).getAttribute("value"));

}

}

}

Summary:

Find Element command returns the web element that matches the first most element within the web page.

Find Elements command returns a list of web elements that match the criteria.

Find Element by XPath in Selenium command throws NoSuchElementException if it does not find the element matching the criteria.

Find Elements command returns an empty list in Selenium if there are no elements matching the criteria

**Selenium Form WebElement**: TextBox, Button, sendkeys(), click()

Forms are the fundamental web elements to receive information from the website visitors. Web forms have different GUI elements like Text boxes, Password fields, Checkboxes, Radio buttons, dropdowns, file inputs, etc.

We will see how to access these different form elements using Selenium Web Driver with Java. Selenium encapsulates every form element as an object of WebElement. It provides API to find the elements and take action on them like entering text into text boxes, clicking the buttons, etc. We will see the methods that are available to access each form element.

**Input Box**

Input boxes refer to either of these two types:

**Text Fields**- Selenium input text boxes that accept typed values and show them as they are.

**Password Fields**- text boxes that accept typed values but mask them as a series of special characters (commonly dots and asterisks) to avoid sensitive values to be displayed.

**Locators**

The method findElement() takes one parameter which is a locator to the element. Different locators like By.id(), By.name(), By.xpath(), By.CSSSelector() etc. locate the elements in the page using their properties like id, name or path, etc.

You can use plugins like Fire path to get help with getting the id, xpath, etc. of the elements.

**sendkeys in Selenium**

sendkeys() in Selenium is a method used to enter editable content in the text and password fields during test execution. These fields are identified using locators like name, class, id, etc. It is a method available on the web element. Unlike the type method, sendkeys() method does not replace existing text in any text box.

Entering Values in Input Boxes

To enter text into the Text Fields and Password Fields, sendKeys() is the method available on the WebElement in Selenium.

**Deleting Values in Input Boxes**

The **clear()** method is used to delete the text in an input box. This method does not need a parameter. The code snippet below will clear out the text from the Email or Password fields

**Buttons**

The Selenium click button can be accessed using the click() method.

**Submit Buttons**

Submit buttons are used to submit the entire form to the server. We can either use the click () method on the web element like a normal button as we have done above or use the submit () method on any web element in the form or on the submit button itself.

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.\*;

public class Form {

public static void main(String[] args) {

// declaration and instantiation of objects/variables

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

String baseUrl = "http://demo.guru99.com/test/login.html";

driver.get(baseUrl);

// Get the WebElement corresponding to the Email Address(TextField)

WebElement email = driver.findElement(By.id("email"));

// Get the WebElement corresponding to the Password Field

WebElement password = driver.findElement(By.name("passwd"));

email.sendKeys("abcd@gmail.com");

password.sendKeys("abcdefghlkjl");

System.out.println("Text Field Set");

// Deleting values in the text box

email.clear();

password.clear();

System.out.println("Text Field Cleared");

// Find the submit button

WebElement login = driver.findElement(By.id("SubmitLogin"));

// Using click method to submit form

email.sendKeys("abcd@gmail.com");

password.sendKeys("abcdefghlkjl");

login.click();

System.out.println("Login Done with Click");

//using submit method to submit the form. Submit used on password field

driver.get(baseUrl);

driver.findElement(By.id("email")).sendKeys("abcd@gmail.com");

driver.findElement(By.name("passwd")).sendKeys("abcdefghlkjl");

driver.findElement(By.id("SubmitLogin")).submit();

System.out.println("Login Done with Submit");

//driver.close();

}

}

**How to Select CheckBox and Radio Button in Selenium WebDriver:**

**Radio Button**

Radio Buttons too can be toggled on by using the click() method.

Using http://demo.guru99.com/test/radio.html for practise, see that radio1.click() toggles on the "Option1" radio button. radio2.click() toggles on the "Option2" radio button leaving the "Option1" unselected.

## What is a Radio Button?

A ***Radio Button*** is an [***HTML***](https://en.wikipedia.org/wiki/HTML) element, which allows the user to select only one of the given options. We generally organize Radio buttons rally organized in a group that contains mutually exclusive options. That is to say, we can select only one option out of the give options.

A radio button in ***HTML*** is defined using ***<input>*** tag and an attribute “***type”,***which will have the value as “***radio”.*** So any [***locator strategy***](https://www.toolsqa.com/selenium-webdriver/selenium-locators/) that uses [***DOM***](https://en.wikipedia.org/wiki/Document_Object_Model) for identifying and locating the elements will use the ***<input>*** tag for recognizing the radio buttons.

Let’s take an example on the page: [***http://www.demoqa.com/radio-button***](https://www.demoqa.com/radio-button)

driver.findElement(By.id("yesRadio")).click();

***Selenium WebDriver*** provides certain methods that can *pre and post validate*the states of *Radion Buttons,* checkbox, and so on.

. Few of these methods are:

* ***isSelected():****Checks whether a radio button is selected or not.*

The isSelected() method checks that if an element is selected on the web page or not. It returns a boolean value (true) if selected, else false for deselected. It can be executed only on a  radio button, checkbox, and so on.

The syntax for isSelected() method in Selenium is as follows:

**Syntax:**

boolean isSelected()

or    driver.findElement(By.locatorType(“path”)).isSelected();

* ***isDisplayed():****Checks whether a radion button is displayed on the web page or not.*

The isDisplayed() method is used to check whether an element is displayed on a web page or not. It returns a boolean value (true) if the target element is displayed otherwise returns false.

The syntax for isDisplayed() method in Selenium is as follows:

**Syntax:**

boolean isDisplayed()

The syntax can be implemented in coding like this:

driver.findElement(By.locatorType(“path”)).isDisplayed();

* ***isEnabled():****Checks whether a radion button is enabled or not*

isEnabled() method is used to check if the web element is enabled or disabled within the web page. This method returns “true” value if the specified web element is enabled on the web page otherwise returns “false” value if the web element is disabled on the web page.

The syntax for isEnabled() method in Selenium WebDriver is as follows:

**Syntax:**

boolean isEnabled()

The above syntax can be implemented in coding like this:

driver.findElement(By.locatorType(“path”)).isEnabled();

**Check Box**

**What is a CheckBox?**

The checkbox is a GUI element that allows the user to make certain choices for the given options. Users may get a list of choices, and the checkbox records the choices made by the user. The checkbox allows users to select either single or multiple choices out of the given list.

We can define a checkbox in ***HTML*** using ***<input type=“checkbox”> tag***.

To understand more about CheckBoxes, let’s consider the example of checkboxes (as highlighted below) given on the page “[***http://www.demoqa.com/automation-practice-form***“.](https://www.demoqa.com/automation-practice-form)

**We have to perform the click() operation on that CheckBoxes**

***Selenium WebDriver***provides certain methods that we can use for a *pre and post validation* of the states of a *radio buttons,* checkbox, and so on.

Few of these methods are:

* *isSelected(): Checks whether a checkbox is selected or not.*
* *isDisplayed(): Checks whether a checkbox displays on the web page or not.*
* *isEnabled(): Checks whether a checkbox is enabled or not*

*examples:*

*isSelected():*

WebElement checkBoxElement = driver.findElement(By.cssSelector("label[for='hobbies-checkbox-1']"));

boolean isSelected = checkBoxElement.isSelected();

//performing click operation if element is not checked

if(isSelected == false) {

checkBoxElement.click();

isDisplayed():

WebElement checkBoxElement = driver.findElement(By.cssSelector("label[for='hobbies-checkbox-1']"));

boolean isDisplayed = checkBoxElement.isDisplayed();

// performing click operation if element is displayed

if (isDisplayed == true) {

checkBoxElement.click();

**isEnabled()**:

As the name suggests, this method validates if the given web element is enabled or not.

WebElement checkBoxElement = driver.findElement(By.cssSelector("label[for='hobbies-checkbox-1']"));

boolean isEnabled = chckBxEnable.isEnabled();

// performing click operation if element is enabled

if (isEnabled == true) {

checkBoxElement.click();

**example:**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.\*;

public class Form {

public static void main(String[] args) {

// declaration and instantiation of objects/variables

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("http://demo.guru99.com/test/radio.html");

WebElement radio1 = driver.findElement(By.id("vfb-7-1"));

WebElement radio2 = driver.findElement(By.id("vfb-7-2"));

//Radio Button1 is selected

radio1.click();

System.out.println("Radio Button Option 1 Selected");

//Radio Button1 is de-selected and Radio Button2 is selected

radio2.click();

System.out.println("Radio Button Option 2 Selected");

// Selecting CheckBox

WebElement option1 = driver.findElement(By.id("vfb-6-0"));

// This will Toggle the Check box

option1.click();

// Check whether the Check box is toggled on

if (option1.isSelected()) {

System.out.println("Checkbox is Toggled On");

} else {

System.out.println("Checkbox is Toggled Off");

}

//Selecting Checkbox and using isSelected Method

driver.get("http://demo.guru99.com/test/facebook.html");

WebElement chkFBPersist = driver.findElement(By.id("persist\_box"));

for (int i=0; i<2; i++) {

chkFBPersist.click ();

System.out.println("Facebook Persists Checkbox Status is - "+chkFBPersist.isSelected());

}

//driver.close();

}

}

How to Select Value from DropDown using Selenium Webdriver

**Select Class in Selenium**

The Select Class in Selenium is a method used to implement the HTML SELECT tag. The html select tag provides helper methods to select and deselect the elements. The Select class is an ordinary class so New keyword is used to create its object and it specifies the web element location.

Select Option from Drop-Down Box

Following is a step by step process on how to select value from dropdown in Selenium:

Before handling dropdown in Selenium and controlling drop-down boxes, we must do following two things:

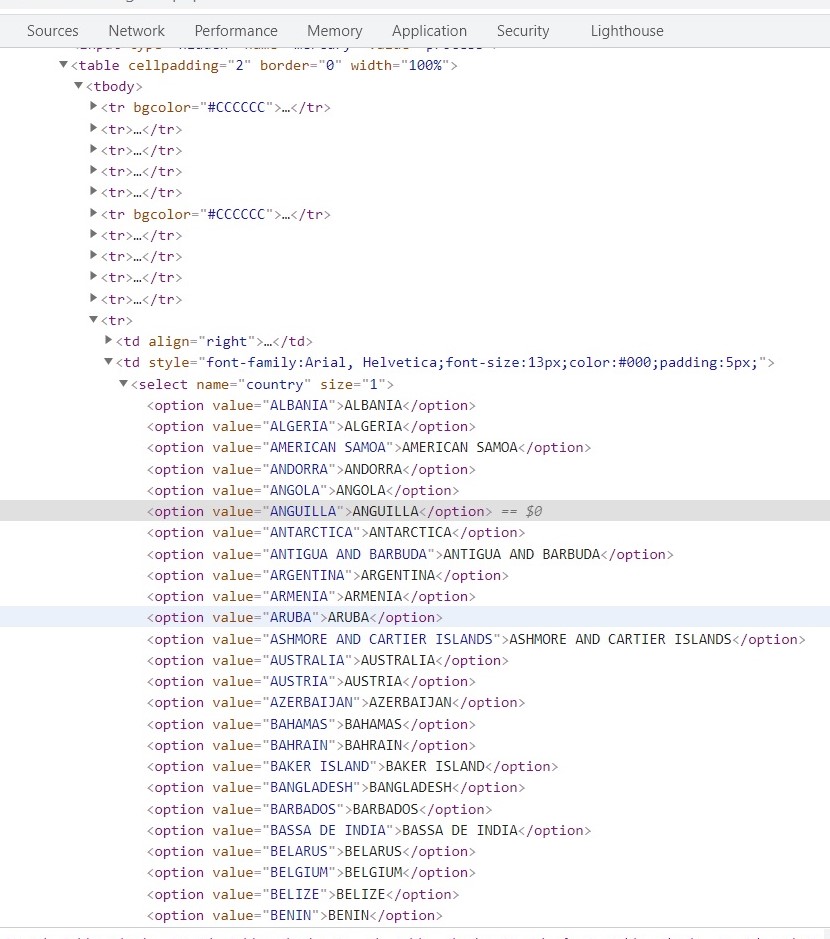
Import the package org.openqa.selenium.support.ui.Select

Instantiate the drop-down box as an object, Select in Selenium WebDriver

As an example, go to Mercury Tours' Registration page (http://demo.guru99.com/test/newtours/register.php) and notice the "Country" drop-down box there.

### How to select a value from a dropdown in Selenium?

As highlighted in the above figure, the Select class of Selenium WebDriver  provides the following methods to select an option/value from a drop-down:



* selectByIndex:

This method selects the dropdown option by its index number. We provide an integer value as the index number as an argument.

Select se = new Select(driver.findElement(By.xpath("//\*[@id='oldSelectMenu']")));

// Select the option by index

se.selectByIndex(3);

* selectByValue:

This method selects the dropdown option by its value. We provide a string value as the value as an argument.

Select se = new Select(driver.findElement(By.xpath("//\*[@id='oldSelectMenu']")));

// Select the option with value "ARUBA"

se.selectByValue("ARUBA");

* selectByVisibleText:

This method enables one to select one option from the dropdown or multi-select dropdown based on the dropdown text. You need to pass the String value of the <select> element as an argument.

Select se = new Select(driver.findElement(By.xpath("//\*[@id='oldSelectMenu']")));

// Select the option using the visible text

se.selectByVisibleText("AUSTRIA");

### **How to select multiple values from a dropdown in Selenium?**

If the <select > tag contains multiple attributes, it means that the dropdown allows selecting multiple values.

Use the web page link to select multiple values “[***https://demoqa.com/select-menu***](https://demoqa.com/select-menu)“:

#### **How to check whether dropdown is Multi-Select?**

As we discussed, the Select class provides the “**isMultiple()**” method, which determines whether the web element in say supports multiple selections. It returns a boolean value, i.e., True/False, without taking any argument. It checks the attribute ‘multiple’ in the HTML code for the web element. Consequently, it possesses the following syntax:

Select oSel = new Select(driver.findElement(By.xpath(//\*[@id='cars']);

if(oSel.isMultiple()){

//Selecting multiple values by index

oSel.selectByIndex(1);

oSel.selectByIndex(2);

oSel.selectByValue("volvo");

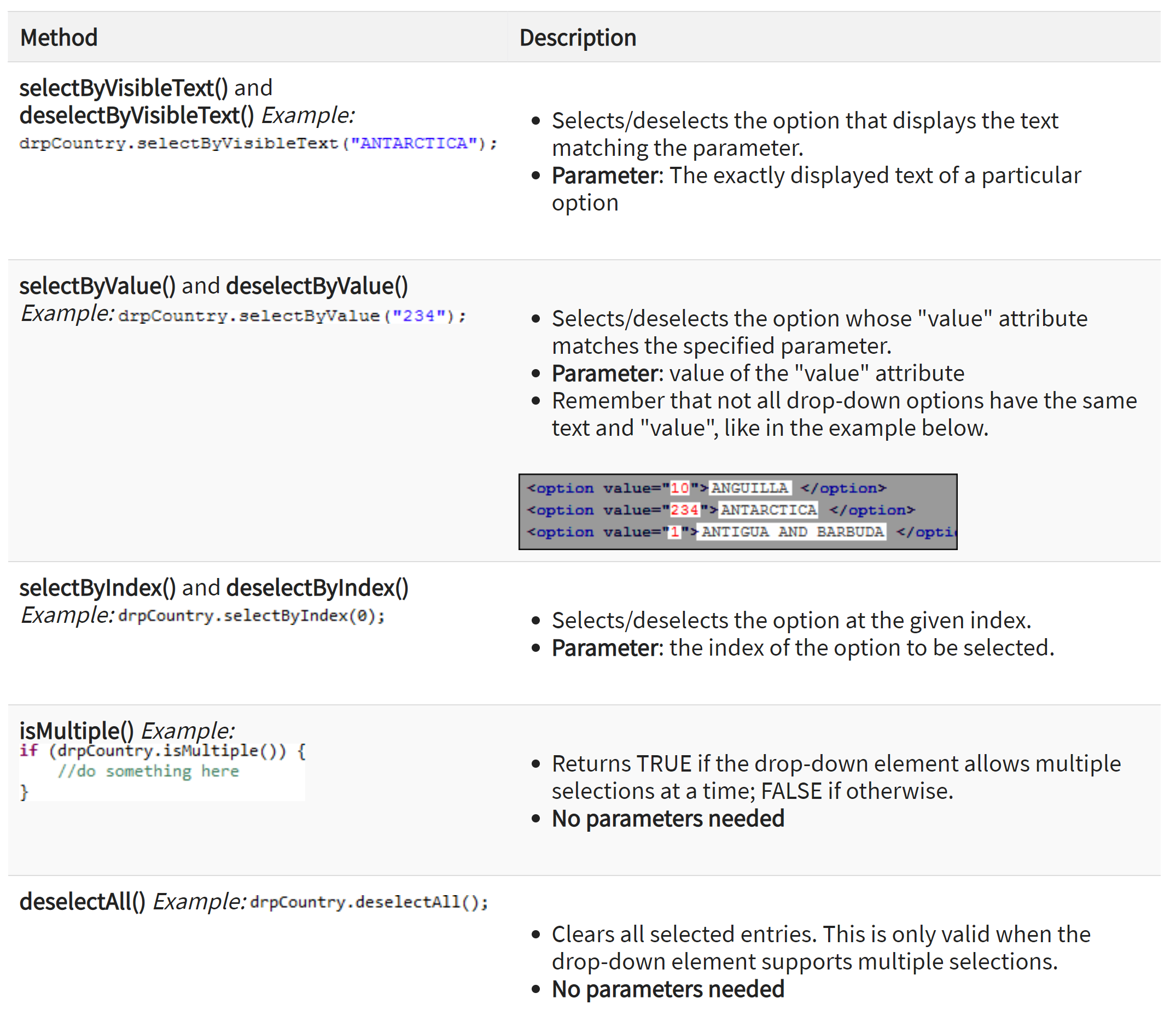
oSel.selectByValue("audi");

//Or selecting by visible text

oSel.selectByVisibleText("Volvo");

oSel.selectByVisibleText("Opel");

}



package newpackage;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

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

import org.openqa.selenium.By;

public class accessDropDown {

public static void main(String[] args) {

System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe");

String baseURL = "http://demo.guru99.com/test/newtours/register.php";

WebDriver driver = new FirefoxDriver();

driver.get(baseURL);

Select drpCountry = new Select(driver.findElement(By.name("country")));

drpCountry.selectByVisibleText("ANTARCTICA");

//Selecting Items in a Multiple SELECT elements

driver.get("http://jsbin.com/osebed/2");

Select fruits = new Select(driver.findElement(By.id("fruits")));

fruits.selectByVisibleText("Banana");

fruits.selectByIndex(1);

}

}

### **How to get options from a dropdown in Selenium?**

As highlighted by marker 2, in the image under the “**Select**” class section above, the Select class provides the following methods to get the options of a dropdown:

* getOptions()
* getFirstSelectedOption()
* getSelectedOptions()

#### **getOptions:**

There are times when you need to get all the options in a dropdown or multi-select box. This is where you can use the getOptions() method of the Select class. It possesses the following syntax:

getOptions(): List<WebElement>

As we can see, this method returns all the options of the dropdown as a list of ***WebElement***. The following code snippet shows how we can get all the options of the dropdown on the page “***https://demoqa.com/select-menu***“:

Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

// Get all the options of the dropdown

List<WebElement> options = select.getOptions();

For(WebElement list:options){

System.out.println(list.getText())

}

#### **getFirstSelectedOption():**

This method returns the first selected option of the dropdown. If it is a single-select dropdown, this method will return the selected value of the dropdown, and if it is a multi-select dropdown, this method will return the first selected value of the dropdown. It possesses the following syntax:

getFirstSelectedOption(): WebElement

ex:

Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

// Get the first selected option of the dropdown

WebElement firstSelectedOption = select.getFirstSelectedOption();

System.out.println(firstSelectedOption.getText());

#### **getAllSelectedOptions():**

This method returns all the selected options of the dropdown. If it is a single-select dropdown, this method will return the only selected value of the dropdown, and if it is a multi-select dropdown, this method will return all the selected values of the dropdown. It possesses the following syntax:

getAllSelectedOptions(): List<WebElement>

### **How to deselect a value from a dropdown in Selenium?**

Just like we select values in a DropDown & Multi-Select, we can deselect the values too. But the deselect method works only for Multi-Select. You can deselect pre-selected options from a Multi-select element using the different deselect methods discussed here.  As we would have observed in the screenshot showing methods of the “**Select**” class (shown by marker 3), the Select class provides the following methods to deselect values of a dropdown:

* deselectAll()

#### This method will clear all the selected entries of the dropdown.

deselectAll(): void

If there are few options already selected in a dropdown, you can deselect all the options using the method deselectAll(). The following code snippet shows a sample example, how we deselect all the values from the dropdown:

Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

//Deselect all the options

select.deselectAll();

* deselectByIndex()

Similar to the selectByIndex() method, the Select class also provides the method to deselect an option from the dropdown using the ***deselectByIndex()*** method. You can use the option’s index number to deselect it. It possesses the following syntax:

deselectByIndex(int arg0): void

So, if there are few options already selected in a dropdown, you can deselect one of the options using the method deselectByIndex(). The following code snippet shows a sample example, how we deselect one of the values from the dropdown by specifying its index:

select.deselectByIndex(1);

* deselectByValue()

Similar to the selectByValue() method, the Select class also provides the method to deselect an option from the dropdown using the ***deselectByValue()*** method. You can use the option’s value to deselect it

So, if there are few options already selected in a dropdown, you can deselect one of the options using deselectByValue(). The following code snippet shows a sample example, how we deselect one of the values from the dropdown by specifying its value:

select.deselectByValue("6");

* deselectByVisibleText()

Similar to the selectByVisibleText() method, the Select class also provides the method to deselect an option from the dropdown using the ***deselectByVisibleText()*** method. You can use the option’s text to deselect it. It possesses the following syntax:

So, if there are few options already selected in a dropdown, you can deselect one of the options using the method deselectByVisibleText(). The following code snippet shows a sample example, how we deselect one of the values from the dropdown by specifying its text:

select.deselectByVisibleText("White");