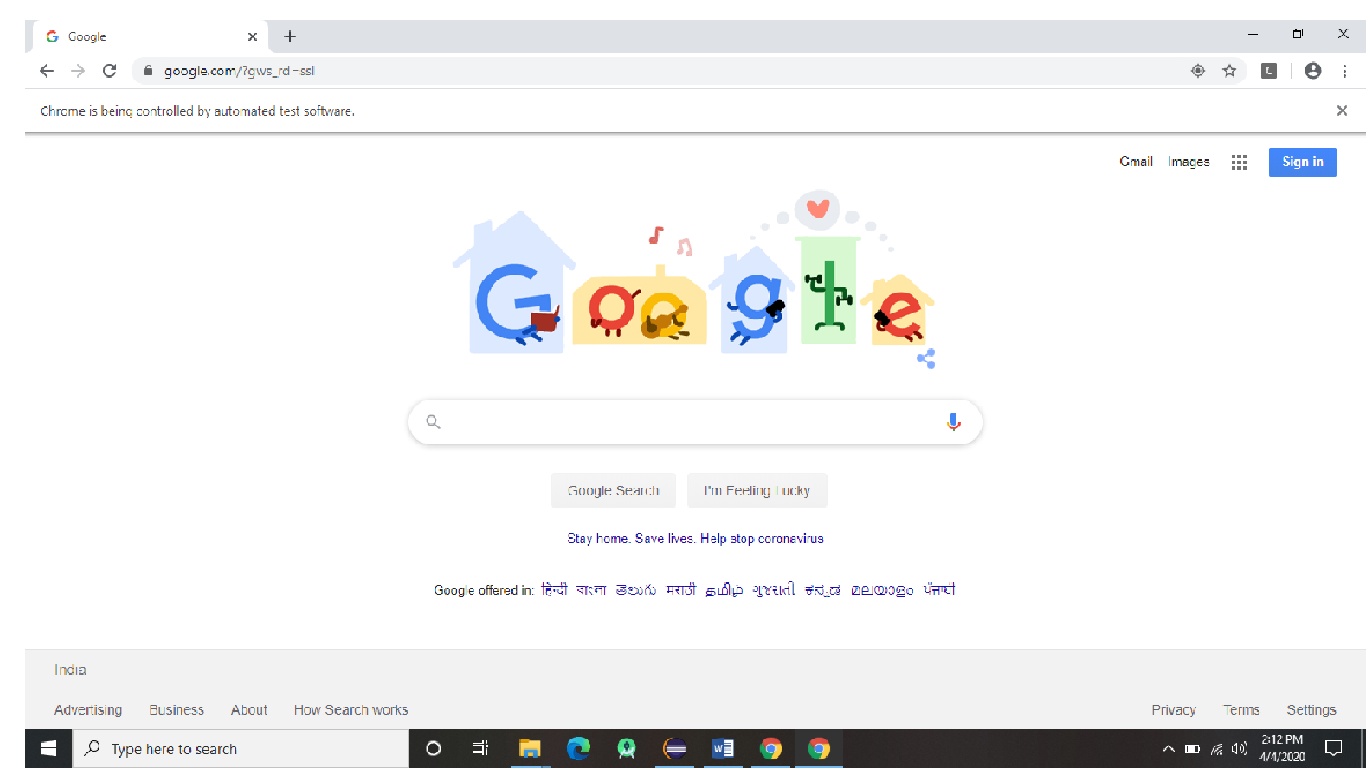
**Handling Multiple Elements:**

* we can handle multiple elements with the help of findElements method “findElements()”.
* Return type of findElement() is List<WebElement>.
* If specified locator of element is not present we will get empty list “[]”.

****

In the above picture we are checking how many links are there is nothing but “anchor tags”.

we are handling multiple Elements in that page.

Example Program:

**package** basic\_selenium;

**import** java.util.List;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**public** **class** Handling\_multiple\_elements

{

**static**

{

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

}

**public** **static** **void** main(String[] args)

{

WebDriver driver=**new** ChromeDriver();

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

driver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***);

driver.get("http://www.google.com");

List<WebElement> links = driver.findElements(By.*xpath*("//a"));

**int** count = links.size();

System.***out***.println(count);

**for** (WebElement link : links)

{

String text = link.getText();

System.***out***.println(text);

}

System.***out***.println("we are getting index 1 webpage link text :"+links.get(1).getText());

}

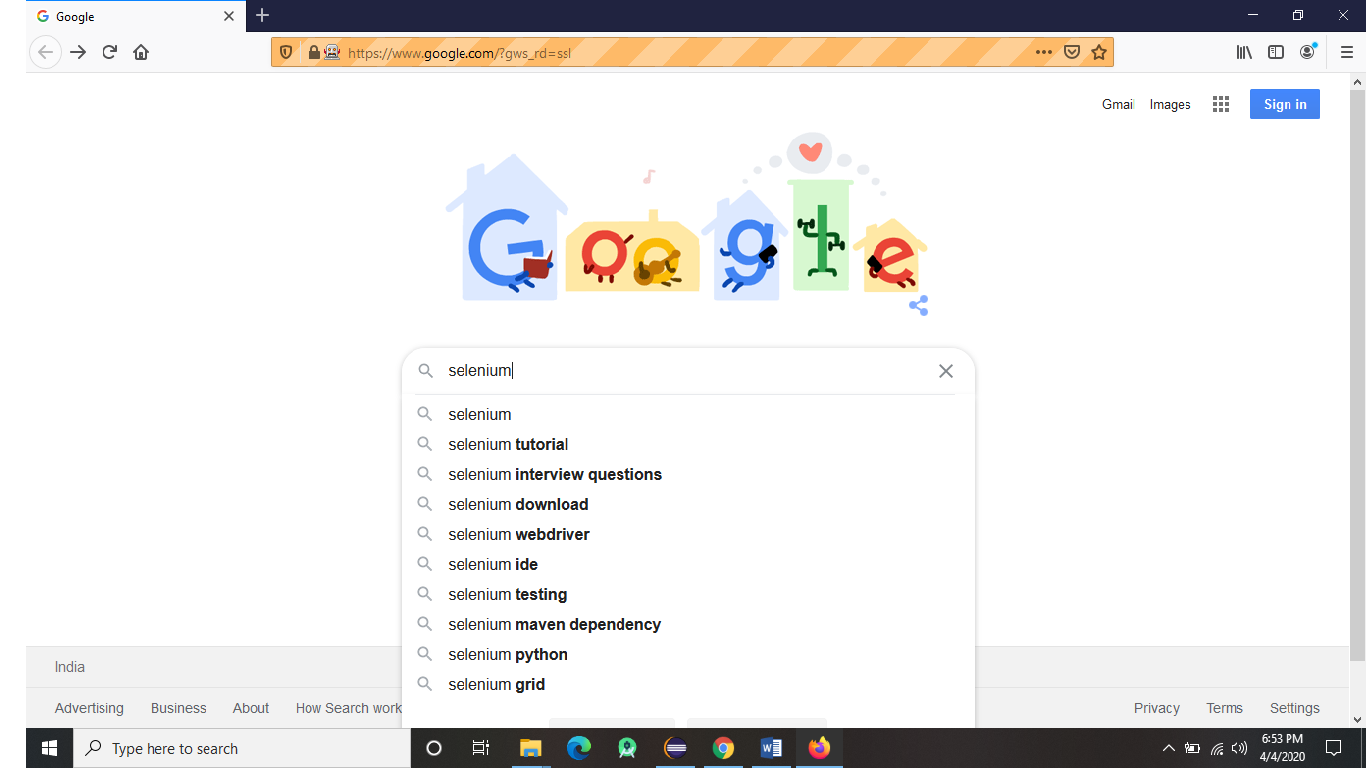
}

Difference between findElement() and findElements()

|  |  |
| --- | --- |
| findElement() | findElements() |
| Return type of FindElement is WebElement. | Return type of FindElements is  List <WebElement>. |
| If specific locator of Element is not matching it will  Throw NoSuchElementException. | If specific locator of Element is not matching it will give Empty List ([]). |
| If specific locator of Element is matching with multiple Element it will give first Matching element. | If specific locator of Element is matching with multiple Element it will give All the Matching elements. |

**Handling Auto Suggestion:**

* To handle Auto Suggestion we should use findElements().
* Return type of findElements() is List<WebElement>.



In the above picture when we search on google Text box like selenium it will automatically give suggestion. with the help of findElements() we can handle Auto suggestion. And we can select any suggestion.

Example Program:

**package** basic\_selenium;

**import** java.util.List;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** Handling\_Auto\_Suggestion

{

**static**

{

System.*setProperty*("webdriver.gecko.driver","./driver/geckodriver.exe");

}

**public** **static** **void** main(String[] args)

{

WebDriver driver=**new** FirefoxDriver();

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

driver.manage().timeouts().implicitlyWait(20,TimeUnit.***SECONDS***);

driver.get("http://www.google.com");

WebElement googleSearch = driver.findElement(By.*name*("q"));

googleSearch.sendKeys("selenium");

List<WebElement> sel = driver.findElements(By.*xpath*("//span[contains(text(),'selenium')]"));

**int** count = sel.size();

System.***out***.println(count);

**for**(WebElement s:sel)

{

String text = s.getText();

System.***out***.println(text);

}

sel.get(6).click();

}

}

**Handling List Box:**

* To handle list box we use select class.
* Select class present in import “org.openqa.selenium.support.ui.select;”
* Select class has a parameterized constructor it will take arguments as WebElement.
* We can handle single select list box and multiple select list box with the help of select class.

In select class we have following methods:

* isMultiple();
* selectByIndex();
* selectByValue();
* selectByVisibleText();
* deselectAll();
* deselectByIndex();
* deselectByValue();
* deselectByVisibleText();
* getOptions();
* getWrappedElement();
* getAllSelectedOptions();
* getFirstSelectedOption();

Example html code for single select list box:

QSP :<select id="single\_select">

<option value="Manual Testing">Manual Testing</option>

<option value="selenium Testing">Selenium Testing</option>

<option value="java">java</option>

<option value="sql">sql</option>

</select>

Example html code for Multiple select list box:

JSP :<select id="multiple\_select" multiple>

<option value="HTML">HTML</option>

<option value="CSS">CSS</option>

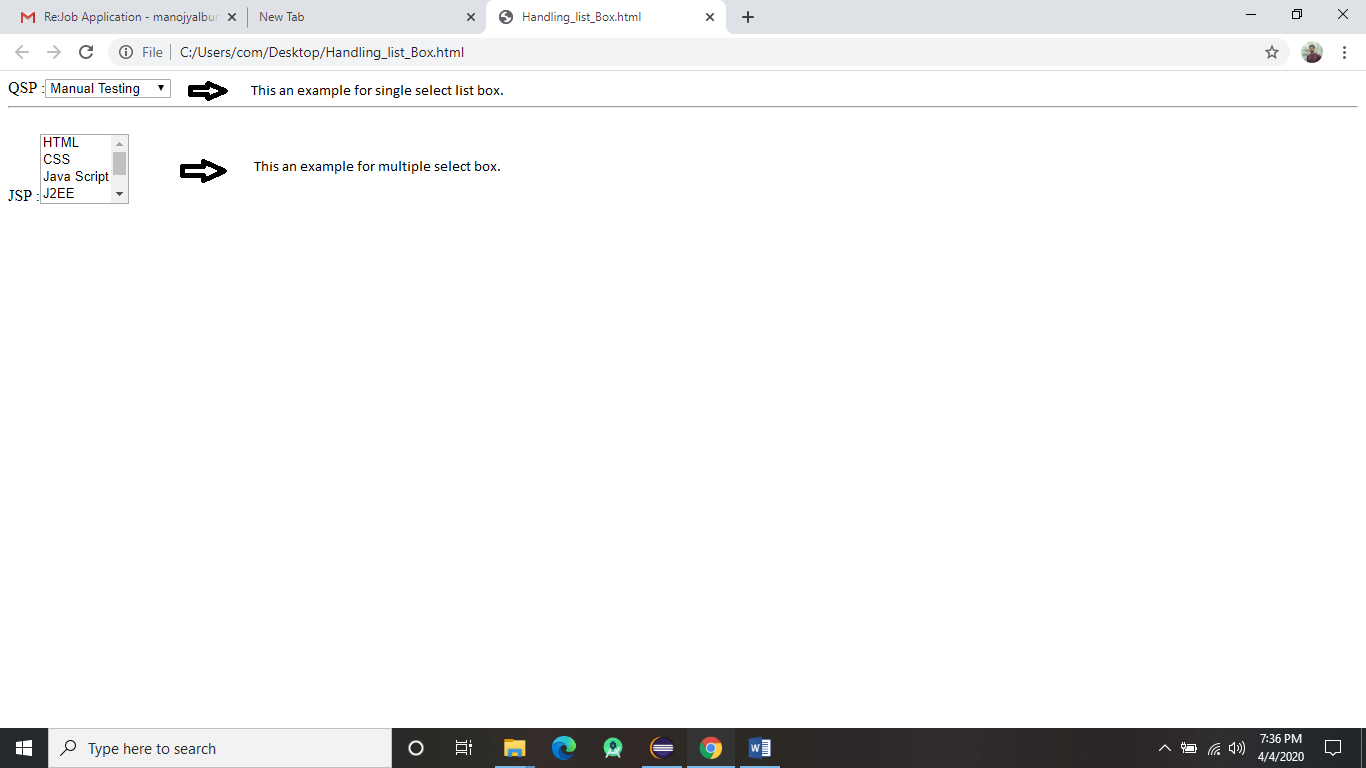
<option value="Java Script">Java Script</option>

<option value="J2EE">J2EE</option>

<option value="java">java</option>

<option value="sql">sql</option>

</select>



The above picture is an example Single select list box and multiple select list box.

Example Program:

**package** basic\_selenium;

**import** java.util.List;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.ie.InternetExplorerDriver;

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

**public** **class** Handling\_list\_Box

{

**static**

{

System.*setProperty*("webdriver.ie.driver","./driver/IEDriverServer.exe");

}

**public** **static** **void** main(String[] args) **throws** InterruptedException

{

WebDriver driver=**new** InternetExplorerDriver();

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

driver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***);

driver.get("file:///C:/Users/com/Desktop/Handling\_list\_Box.html");

System.***out***.println("To Handle Single List Box");

WebElement single = driver.findElement(By.*id*("single\_select"));

Select ss = **new** Select(single);

System.***out***.println("we are checking select box is single select are multi select:"+ss.isMultiple());

ss.selectByIndex(1);

ss.selectByValue("sql");

ss.selectByVisibleText("Manual Testing");

WebElement swe = ss.getWrappedElement();

String text = swe.getText();

System.***out***.println(text);

List<WebElement> sop = ss.getOptions();

**for** (WebElement op : sop)

{

String tex = op.getText();

System.***out***.println(tex);

}

System.***out***.println("To Handle Multi select list box");

WebElement mul = driver.findElement(By.*id*("multiple\_select"));

Select ms = **new** Select(mul);

System.***out***.println(ms.isMultiple());

ms.selectByIndex(3);

ms.selectByValue("java");

ms.selectByVisibleText("CSS");

ms.selectByIndex(0);

ms.selectByValue("Java Script");

ms.selectByVisibleText("sql");

List<WebElement> alls = ms.getAllSelectedOptions();

**for** (WebElement a: alls)

{

String texx = a.getText();

System.***out***.println(texx);

}

Thread.*sleep*(2000);

ms.deselectByIndex(0);

Thread.*sleep*(2000);

ms.deselectByValue("java");

Thread.*sleep*(2000);

ms.deselectByVisibleText("CSS");

WebElement fs = ms.getFirstSelectedOption();

System.***out***.println(fs.getText());

ms.deselectAll();

}

}

Robot Class:

Robot class in selenium is used to handle KeyBoard and Mouse actions.

Robot class present in AWT Package.

Example Program:

**package** basic\_selenium;

**import** java.awt.AWTException;

**import** java.awt.Robot;

**import** java.awt.event.KeyEvent;

**import** java.io.IOException;

**public** **class** Robott {

**public** **static** **void** main(String[] args) **throws** IOException, AWTException {

Runtime.*getRuntime*().exec("notepad");

Robot r = **new** Robot();

r.keyPress(KeyEvent.***VK\_SHIFT***);

r.keyPress(KeyEvent.***VK\_M***);

r.keyRelease(KeyEvent.***VK\_SHIFT***);

r.keyPress(KeyEvent.***VK\_O***);

r.keyPress(KeyEvent.***VK\_M***);

}

}

When we run the code it will open Notepad and it will print Mom.

## **Methods to implement the Robot class**

To implement the Robot class, we require a few methods that help in easy execution of test scripts. We have:

1. KeyPress()
2. KeyRelease()
3. MouseMove()
4. MousePress()
5. MouseRelease()

* **KeyPress()**: This method is called when you want to press any key

Ex: robot.keyPress(keyEvent*.VK\_UP*);

* **KeyRelease()**: This method is used to release the pressed key on the keyboard

Ex: robot.keyRelease(keyEvent*.VK\_CAPS\_LOCK*);

This will release the pressed capslock key in the keyboard

* **MouseMove()**: This method is called when you want to move the mouse pointer in the X and Y co-ordinates

Ex: robot.mouseMove(coordinates.get.X(),coordinates.get.Y());

This method is called when you want to move the mouse over X and Y coordinates

* **MousePress()**: This is used to press the left button of the mouse

Ex: robot.mousePress(InputEvent.BUTTON1\_MASK);

Helps in pressing the mouse button

* **MouseRelease()**: This method helps in releasing the pressed button of the mouse

Ex: robot.mouseRelease(InputEvent.BUTTON3\_DOWN\_MASK)

This method is used to release the right click of the mouse

**Actions Class:**

In Selenium Action is an interface where as Actions is class.

Actions class present in

import org.openqa.Selenium.interactions.Actions;

Actions class has a parameterized constructor it will take type of arguments such as Webdriver driver,keyboard,keybord and mouse.

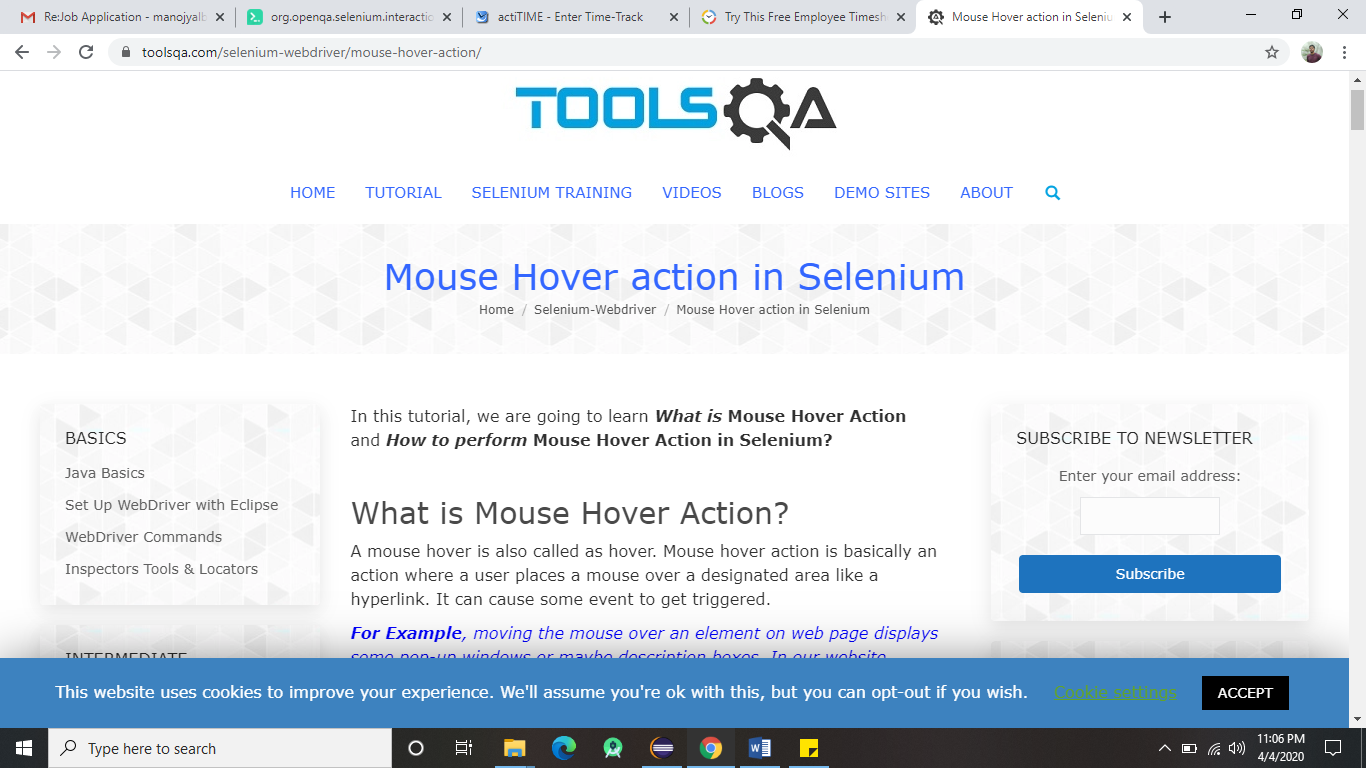
In Actions class when we call any of actions class methods at the end we should call perform() method.

* **Build**: build() method is used to generate a composite action containing all actions.  
  In this case, it’s just a single move mouse action. If you observe, we have not invoked it in our above command. This is because the build is executed in the perform method internally.
* **Perform**: perform() method performs the actions we have specified. But before that, it internally invokes build() method first. After the build, the action is performed.

In actions class we have following methods:

* Mouse Hovering: Mouse Hovering is nothing but we can preform mouse actions on element.
* contextClick: contextClick is nothing but Right click.
* compositClick: compositClick is nothing but we can perform keyboard and mouse action at a time.
* dragAndDrop: dragAndDrop is nothing but we are moving one element to another.
* doubleClick: doubleClick is nothing but we are clicking double time on element .

Mouse Hovering: Mouse Hovering is nothing but we can preform mouse actions on element.



Above picture is before performing mouse hovering.

Example code:

**package** basic\_selenium;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.interactions.Actions;

**public** **class** Actionss

{

**static**

{

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

}

**public** **static** **void** main(String[] args)

{

WebDriver driver=**new** ChromeDriver();

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

driver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***);

driver.get("https://www.toolsqa.com/selenium-webdriver/mouse-hover-action/");

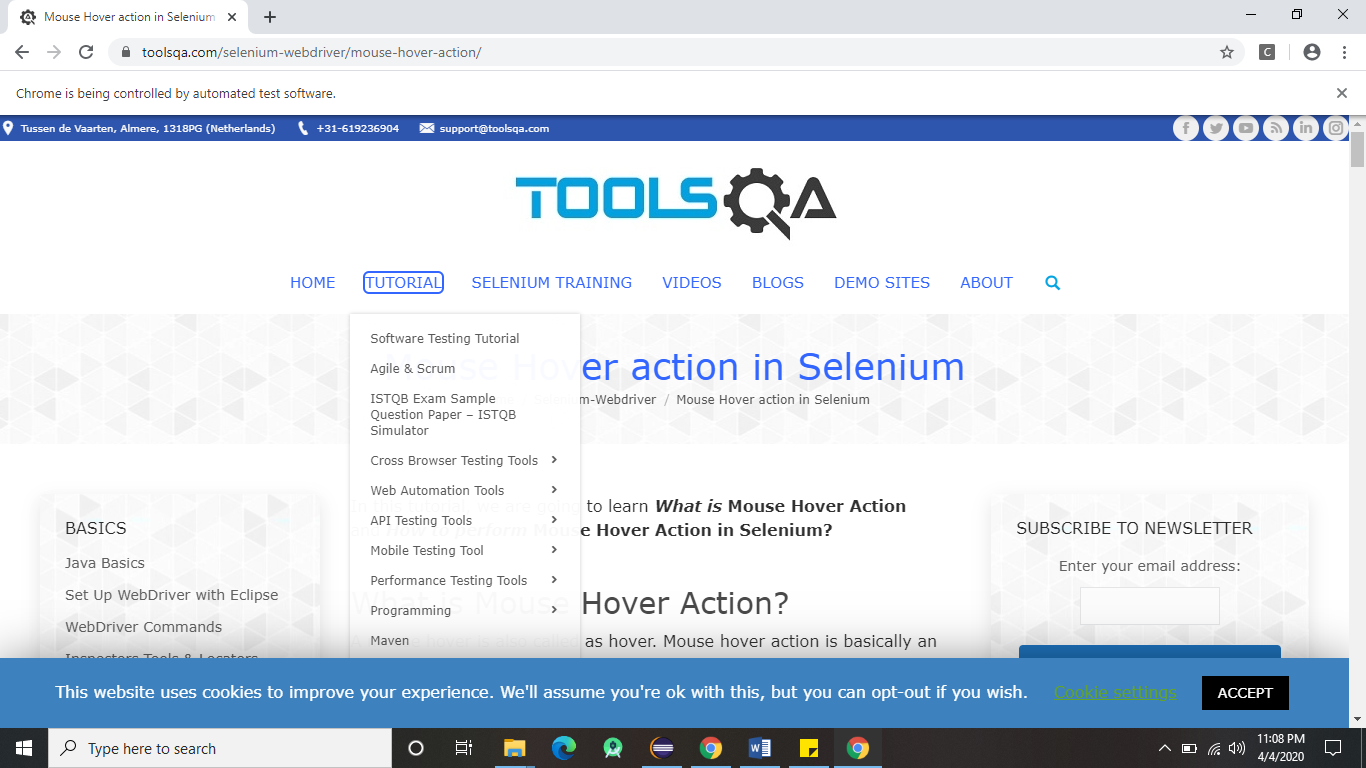
WebElement ele = driver.findElement(By.*xpath*("//span[text()='Tutorial']"));

Actions a = **new** Actions(driver);

a.moveToElement(ele).perform();

}

}



Above picture after performing mouse hovering on element Tutorial.

ContextClick:

contextClick is nothing but Right click on element.

Example code:

**package** basic\_selenium;

**import** java.awt.AWTException;

**import** java.awt.Robot;

**import** java.awt.event.KeyEvent;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.interactions.Actions;

**public** **class** Actions\_contextClick

{

**static**

{

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

}

**public** **static** **void** main(String[] args) **throws** AWTException

{

WebDriver driver=**new** ChromeDriver();

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

driver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***);

driver.get("http://localhost/login.do");

WebElement ele = driver.findElement(By.*xpath*("//a[text()='Actimind Inc.']"));

Actions a = **new** Actions(driver);

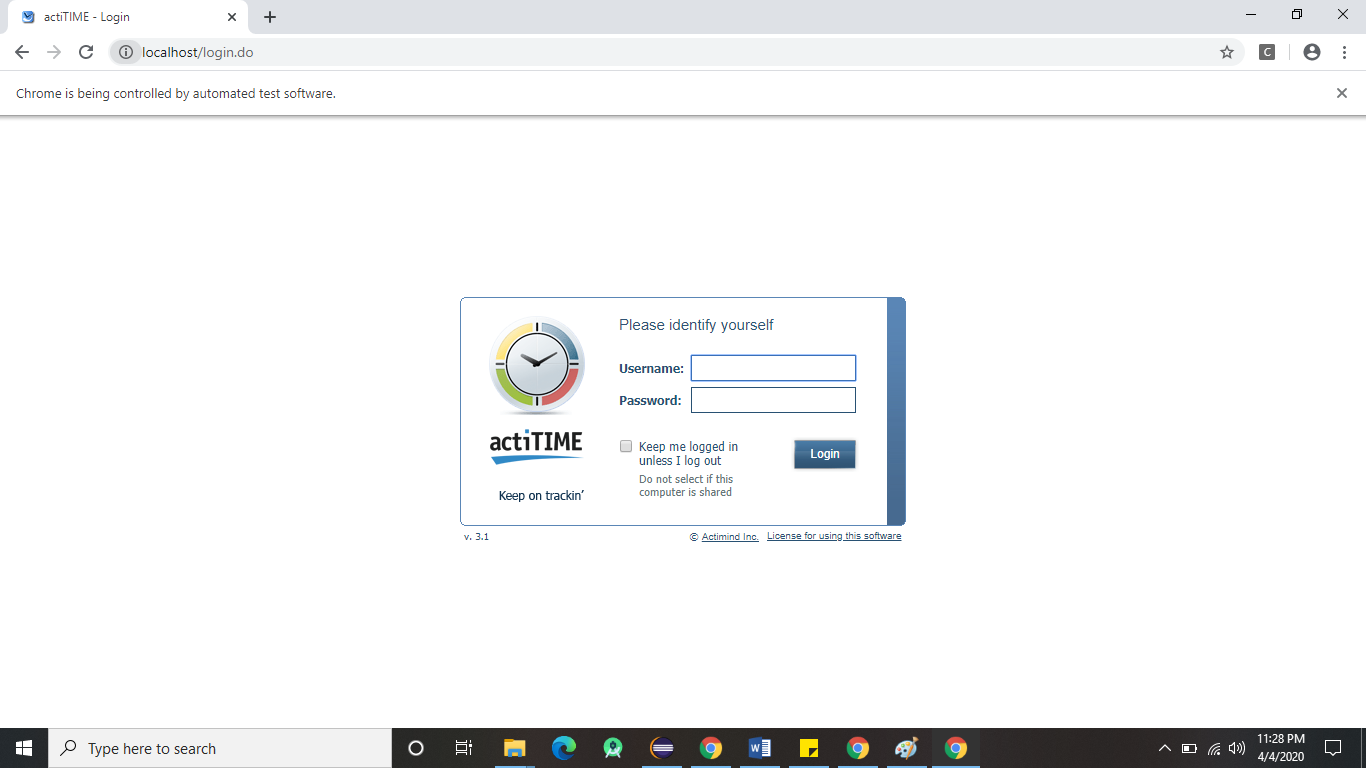
a.contextClick(ele).perform();

Robot r = **new** Robot();

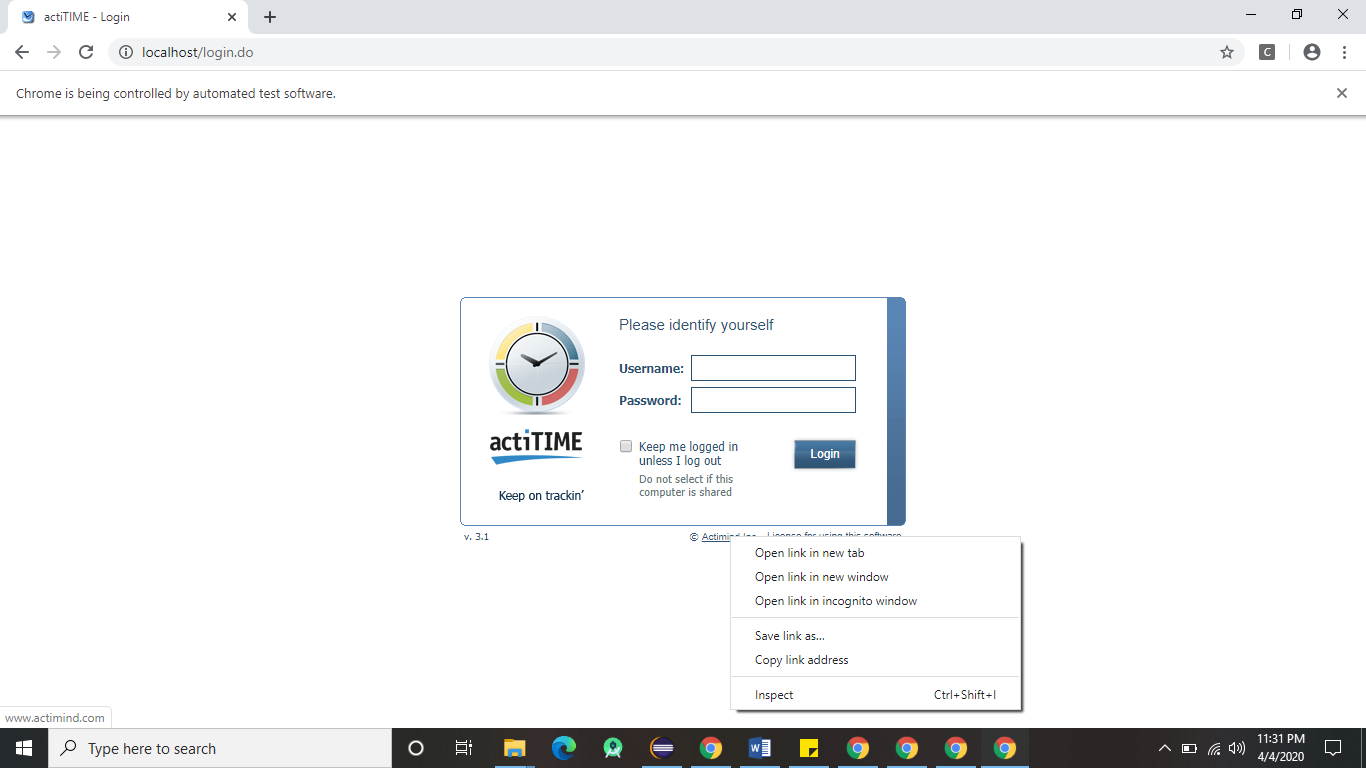
r.keyPress(KeyEvent.***VK\_T***);

}

}



Above picture before performing contextClick on element Actimind inc.



Above picture after performing contextClick on element Actimind inc.

After performing contextClick on element .we can use robot class to open new tab,new window etc..