**SELENIUM USER GUIDE**

1. If there is a existing text in a text box, will selenium append the text or overwrite ?

Selenium will **append** the text, if you want to overwrite the text, we have to use **clear()** method and then enter the text using **Sendkeys**

1. What is point class in Selenium ? It is used to find the coordinates(x,y) of a button element in a web page.
2. Generics 🡪 To store only specific web elements for dropdown. To ensure no other list like numbers or strings are stored. Say Instance List<web elements> <name>
3. Can we use send keys to select drop down values 🡪 Yes we can use sendkeys to select a value from drop-down
4. Different types of x-path 🡪 There are 2 types one is Absolute and another one is Relative. **Absolute** start with root code **single slash(/),** say eg (html/body/div/..), where as **relative** xpath start in middle of the block, with **double slash(//)** say eg(“//[@id=’contentblock..]/selection/div..)

**Methods:**

**getLocation()** 🡪 To get the X and Y coordinated from a web element

getX() 🡪 Once get the coordinates of a button, to get the value of X coordinate.

getY() 🡪 Once get the coordinates of a button, to get the value of Y coordinate.

**get CssValue 🡪** To get the background colour of an element

**getSize( ) 🡪** To get the size of an element

**getHeight( ) 🡪** Once get the size of a button, to get the Height value of web element.

**getWidth( ) 🡪** Once get the size of a button, to get the Width value of web element.

**getOptions()** 🡪 To get the number of options available in drop down

**size()** 🡪 To get the size of options captured in getOptions() method.

**When Xpath not recognised 2ndtime:** when xpath is not recognised by script second time, we can remove (“) with (‘) and delete (\) like shown below

**Before**

driver.findElement(By.*xpath*("//\*[@id=\”contentblock\”]/section/div[1]/div[1]/input"));

**After:**

driver.findElement(By.xpath("//\*[@id='contentblock']/section/div[1]/div[1]/input"));

**Class 🡪 ButtonsExample**

// Find button Get positions and getting the X,y coordinates

WebElement findposition = driver.findElement(By.*id*("position"));

Point xyvalue = findposition.getLocation();

**int** xvalue = xyvalue.getX();

**int** yvalue = xyvalue.getY();

System.***out***.println("Value of X is :"+xvalue+" and "+"Value of Y is :"+yvalue);

//2.Find button Color

WebElement findcolor = driver.findElement(By.*id*("color"));

String colour = findcolor.getCssValue("background-color");

System.***out***.println("Colour of Button is : "+colour);

//3. Find Height and Width of button

WebElement findheight = driver.findElement(By.*id*("size"));

**int** hig = findheight.getSize().getHeight();

**int** lgt = findheight.getSize().getWidth();

System.***out***.println

("Height of the buttion is : "+hig+ " and "+"Width of the button is : "+lgt);

**Class 🡪 DropDownExample**

Drop down and alert are slightly different elements in selenium. In order to get the values from dropdown we are using **select** class(built in).

We can select the values from dropdown using these methods() SelectByIndex(), SelectByValue() and selectByVisibleText() of select class

// Selecting DropDown values using select ByIndex, ByValue and ByVisible text methods

Select values = **new** Select(dropdown); // Creating new object for select class

values.selectByIndex(1);

values.selectByValue("2");

values.selectByVisibleText("Loadrunner");

// Getting number of dropdown options.

WebElement dropdown2 = driver.findElement(By.*className*("dropdown"));

Select values2 = **new** Select(dropdown2);

List<WebElement> listoptions = values2.getOptions();

**int** listsize = listoptions.size();

System.***out***.println("Number of options in dropdown is : "+listsize);

**Class 🡪 AlertBoxExample**

**Alerts 🡪,** like dropdown element, Alert is also unique element. In order to access the alert box, we are using a built-in interface called **Alert** in selenium.

WebElement alertbox = driver.findElement(By.*xpath*

("//\*[@id=\'contentblock\']/section/div[1]/div/div/button"));

alertbox.click();

Alert alertboxclick = driver.switchTo().alert(); //Since alert works differently in webpage, we are using inbuild interface called Alert and switching our driver(browser) control to alert box using driver.switchTo().alert();

alertboxclick.accept();

// 3. Same as above example, here clicking on Cancel on confirm box, instead of OK button.

WebElement confirmbox = driver.findElement(By.*xpath*("//\*[@id=\'contentblock\']/section/div[2]/div/div/button"));

confirmbox.click();

Alert confirmboxclick = driver.switchTo().alert();

Thread.*sleep*(500);

confirmboxclick.dismiss();

// 4. Clicking on Promptbox and entering the words using send keys and clicking OK button.

WebElement promptbox = driver.findElement(By.*xpath*("//\*[@id=\'contentblock\']/section/div[3]/div/div/button"));

promptbox.click();

Alert promptboxclick = driver.switchTo().alert();

promptboxclick.sendKeys("Hello World");

Thread.*sleep*(500);

promptboxclick.accept();

**Class 🡪 CheckBoxExample**

**Deselect the checkbox**, which is already selected. Here we have 2 checkboxes one is selected another one is not selected

we can directly find the element of selected checkbox which is selected by default and make it deselect using **click(),**

but here we are using if condition, to identify which one is selected and deselecting the same

WebElement checkbox4 = driver.findElement(By.*xpath*("//\*[@id=\"contentblock\"]/section/div[3]/div[1]/input"));

**if**(checkbox4.isSelected()) {

checkbox4.click();

}

WebElement checkbox5 = driver.findElement(By.*xpath*("//\*[@id=\"contentblock\"]/section/div[3]/div[2]/input"));

**if**(checkbox5.isSelected()) {

checkbox5.click();

}

**Class 🡪 WindowsExample**

**getWindowHandle()**

It’s a method in selenium to find the current window(parent/old window).

String oldwindow = driver.getWindowHandle();

**getWindowHandles()**

Almost same as above, but it is used to find the old and newly opened windows(both parent and child windows)

Set<String>allwindow = driver.getWindowHandles(); // all opened windows are stored in allwindow

//To change to newpop window from old window we are using for each loop

**for** (String popupwindow : allwindow) {

driver.switchTo().window(popupwindow);

}

//Once jumped to popup window, we are clicking the editbox button and closing the popup windows

WebElement editbox = driver.findElement(By.*xpath*("//\*[@id=\"post-153\"]/div[2]/div/ul/li[1]/a/img"));

driver.close();

**Difference between driver.Close() and driver.quit()**

Driver.close() 🡪 It will close only the current browser opened by driver

Driver.quit() 🡪 It will close all opened browser by driver.

Set<String> allwindow3 = driver.getWindowHandles(); // Method to find all windows new and old windows(both parent and child)

**for** (String popup2 : allwindow3) {

**if**(!popup2.equals(oldwindow)) {

driver.switchTo().window(popup2); //Switching control to new window(child)

driver.close();

}

**Class 🡪 FramesExample**

**What is Frames in a webpage:** Frames is also a HTML page, it will be added inside a main HTML with the tag iframe. Eg(Ads in a web page is a frame).

Also while inspecting an element in a web page, we get a new option “**View Frame Source**” , which refers to a Frame

We may think we can use usual find element method to identity the frame and perform any action.

But script **WebDriver driver = new ChromDriver();** will be able to identify only the base/core HTML content, whichever code written under nested HTML say instance iFrame. It cannot be accessed through driver script



Hence we are using new function called **frame()**

Instruct the script to find an element which is inside the frame.

driver.switchTo().frame(0); //Instructing script to switch from native browser to frames, since there is no attributes like(id, name) we are using giving zero(0). Zero to 1st frame.

WebElement button1 = driver.findElement(By.*id*("Click"));

button1.click();

If there is no id to identify the frame, we can use the number (0,1…9,10)

0 refers to 1st frame in the webpage

2 refers to 2nd frame in the webpage

Like that it goes on

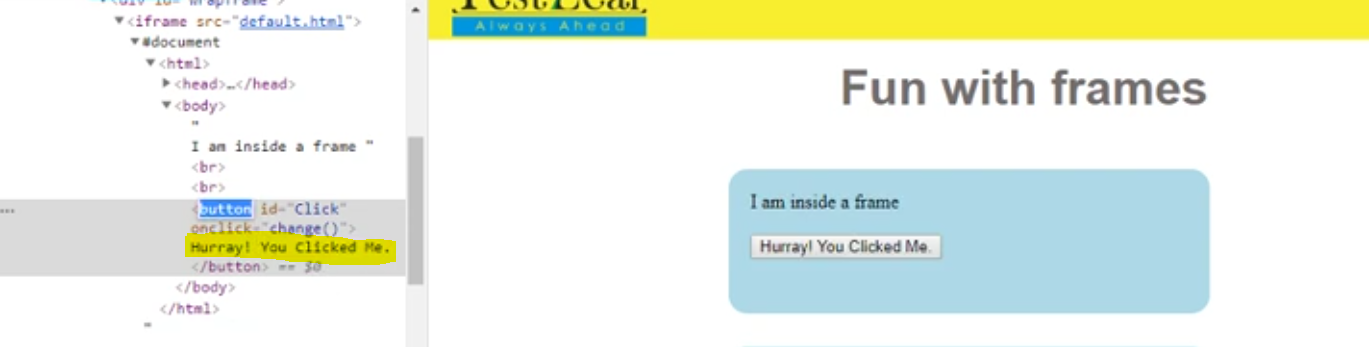
**Difference between getText() and getAttribute()**

To get the value between tags use 🡪**getText(),**

String gettext = driver.findElement(By.*id*("Click")).getText(); //getText is ised to get the text of a button

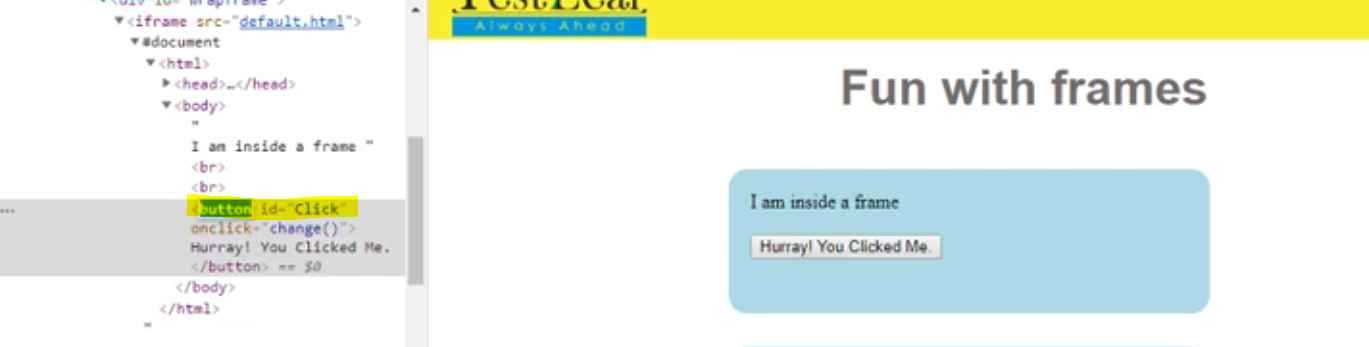
System.***out***.println("Text inside click me button is : "+gettext);

**Result 🡪** Text inside click me button is : Hurray! You Clicked Me.

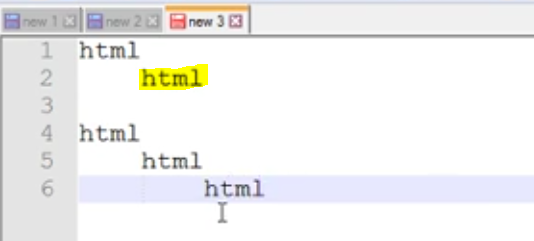


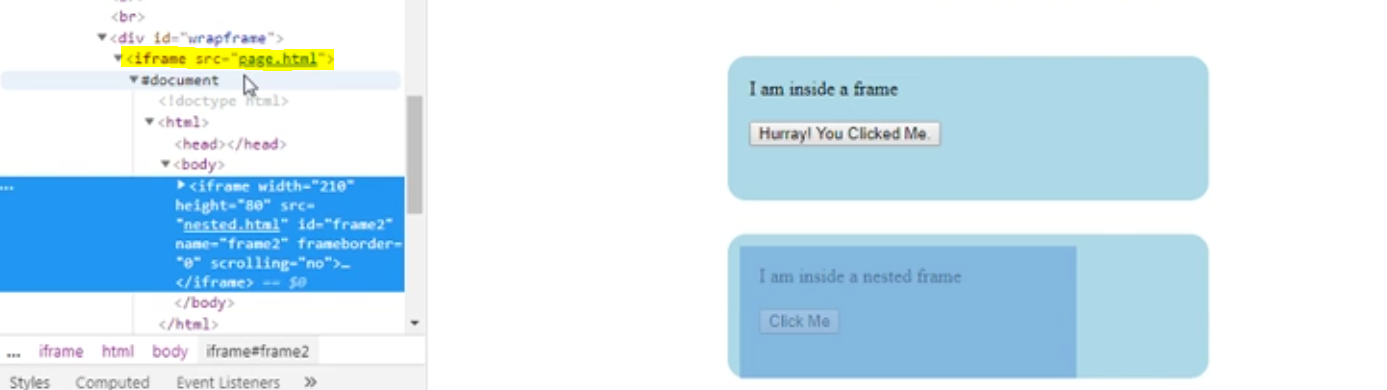
To take the value of an attribute inside tags use 🡪 **getAttribute()**

We have getAttribute() example in ToolTipExample class down the line.



Normal Frame 🡪 HTML tag inside a HTML



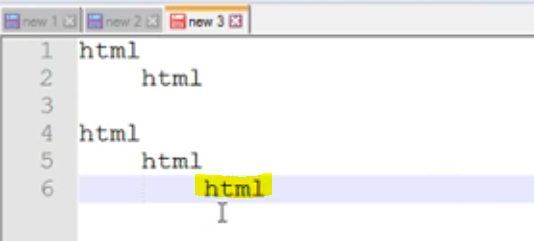


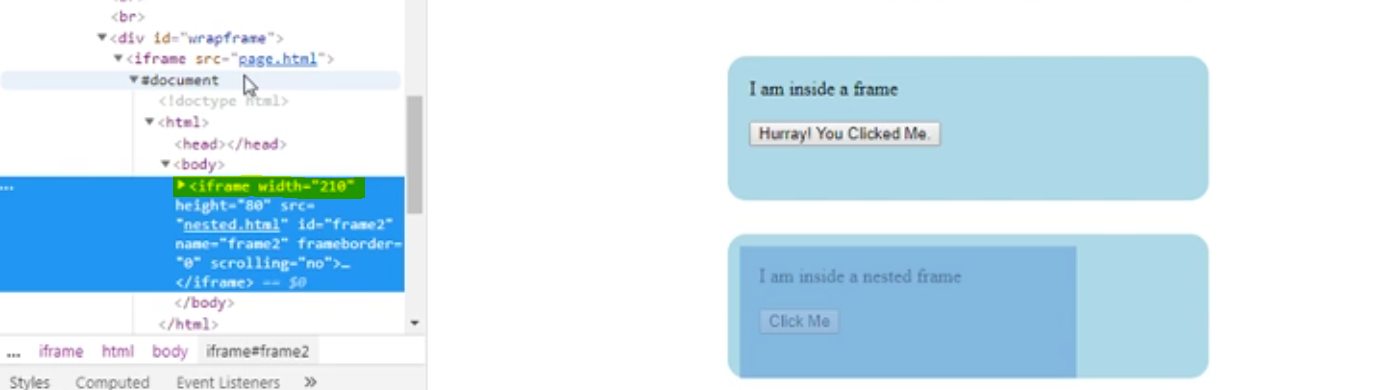
driver.switchTo().frame(0); //Instructing script to switch from native browser to frames

WebElement button1 = driver.findElement(By.*id*("Click"));

button1.click();

Nested Frame 🡪 HTML inside a 2nd HTML





driver.switchTo().frame(1); //now instructing script to switch to the 1st frame

driver.switchTo().frame("frame2"); //since it's a nested frame, we are navigating to frame(nested frame) inside a frame

WebElement button2 = driver.findElement(By.*id*("Click1"));

button2.click();

**To find the number of Frames in a page: findElements()**

We have to use findElements(),

In order to find the number of times a tag is presented in a webpage. **By.tagName()**

Here in this scenario we are trying to identify the count of frames, so we using theBy.*tagName*("iframe"));

List<WebElement> noofframes = driver.findElements(By.*tagName*("iframe"));

**int** framescount = noofframes.size();

System.***out***.println("Total no of Frames in this page is : "+framescount);

**Result 🡪** Total no of Frames in this page is : 3

**defaultContent()**

To move the control back to native browser from frames

driver.switchTo().defaultContent(); // Switching back to native browser from frames

**Class 🡪 DragDropExample**

To perform actions using mouse,(drag and drop) or keyboard(CTRL, SHIFT..etc) we have to use Actions class.

// 3.Creating object for Actions

Actions **dragdrop** = **new** Actions(driver); // now control moved to drag and drop

There are 2 methods we can perform this drag and drop action

1. dragdrop.dragAndDrop(drag, drop).build().perform();

After creating the actions class **dragdrop**. Using **dragAndDrop(source, target)** function, by passing source(**drag**) and target(**drop**) we can execute the same

1. dragdrop.clickAndHold(drag).moveToElement(drop).release(drop).build().perform();

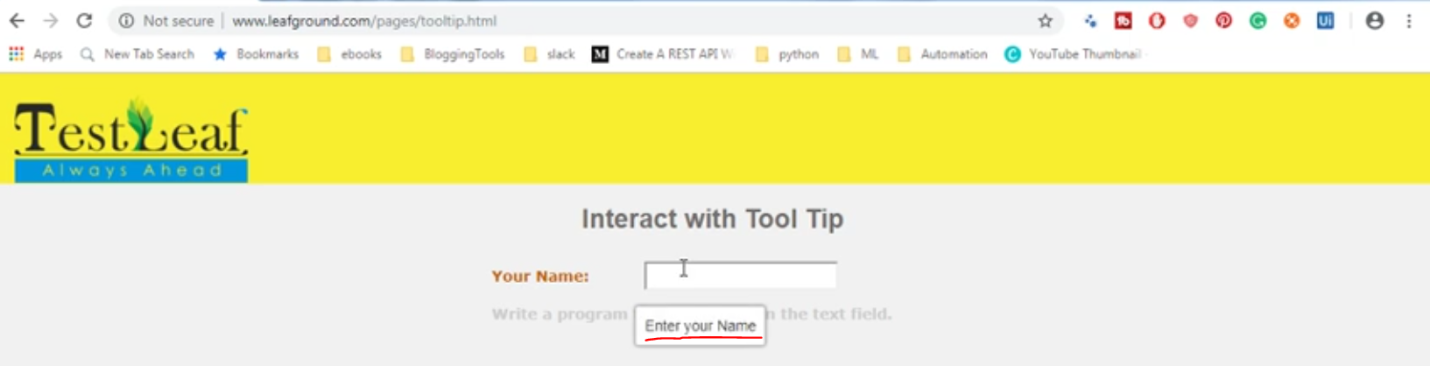
After creation the action class **dragdrop.**

Using **clicnAndHold(source),** **moveToElement(target),** **release(target)** we can execute the same.

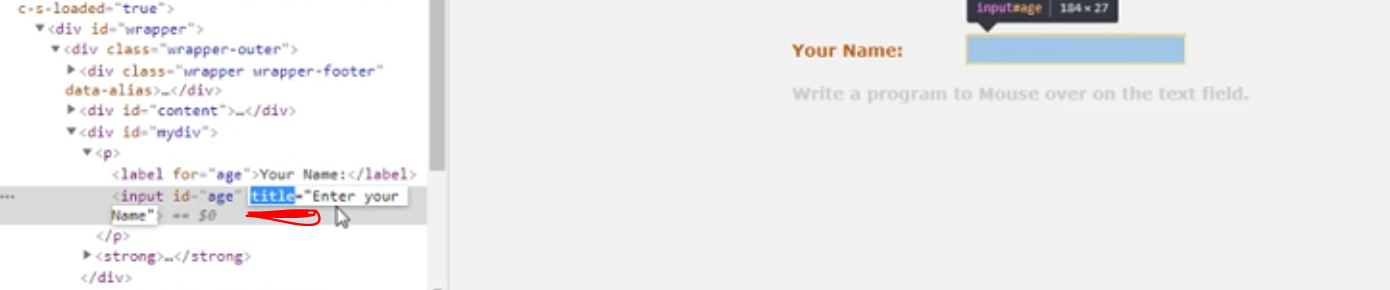
**Build().perform()** 🡪 What is the purpose of this function. Whenever we use Actions class we have to use this build().perform(). In order to complete Drag and Drop (or) CTRL action, using keyboard or Mouse

**Class 🡪 ToolTipExample**

What is tool tip text, when we hover(move) the mouse to a web element(say eg, textbox) it will display what needs to be filled in that particular element.



To perform this action, we can simply find the element using(id,name, class…etc) and get the attribute value of the tool tip text. In our case the attribute where tool tip is stored **title = “Enter your Name”**



Using getAttribute(), we can get the tool tip of a web element.

WebElement textbox = driver.findElement(By.*id*("age"));

String gettooltip = textbox.getAttribute("title");

System.***out***.println("Tool tip of textbox is : "+gettooltip);

**Result 🡪** Tool tip of textbox is : Enter your Name

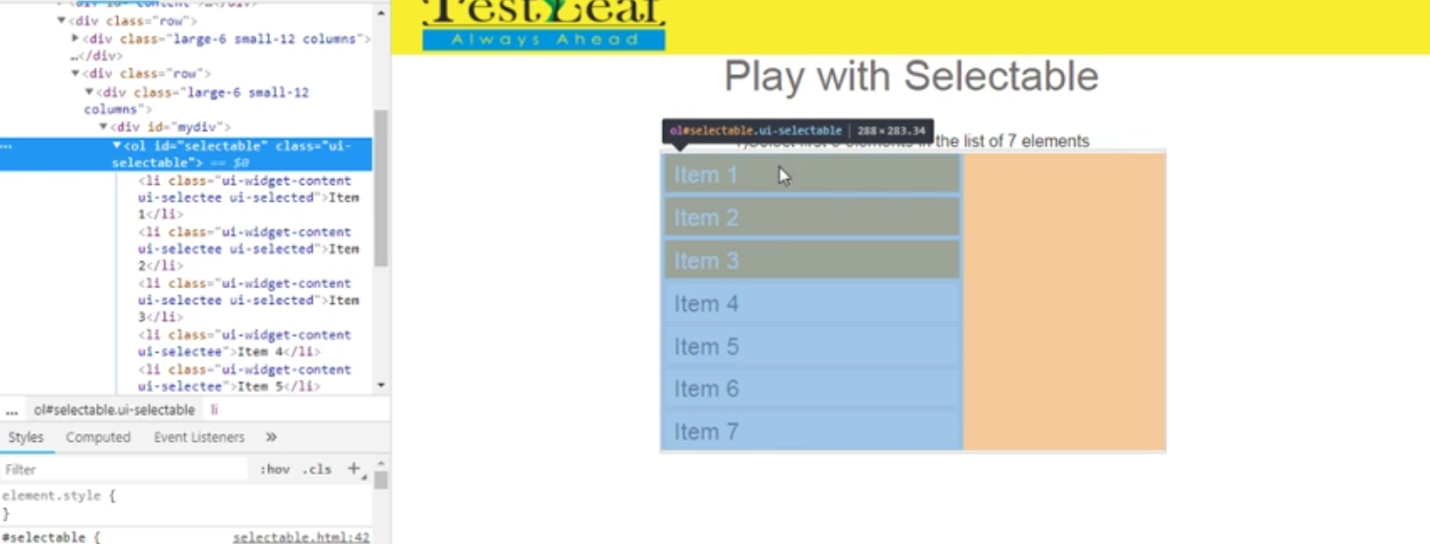
**Class 🡪 SelectableExample**

Selectable elements are nothing but list of elements where we can select using CTRL key and using mouse in a webpage.

These selectable web elements will be available under OL 🡪 Ordered List, (*FYI.. There are 2 types of list in HTML* ***OL*** *🡪 Ordered List and* ***UL*** *🡪 Unordered List*)

In this case even though we have locators like id, class, etc. We are using xpath to find the element reason because.

Locators lik <ol id = “selectable”> or <class = “ui-selectable”> refers to single web element, but we have to select multiple web elements like(Item 1, Item 2... etc)



Hence we are using Xpath locator also adding **/li** in the end of X-path, to ensure it refers to list of elements in ordered list.

Hence it will consider all the items in the selectable.



Using Actions class we can achieve this Multi select /Selectable using Keyboard or Mouse.

1. **Using KeyBoard**

// 1. Creating an Action class, in order to interact in webpage using Keyboard or Mouse

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

//2. Using Keyboard

selectitems.keyDown(Keys.***CONTROL***).click(selectables.get(0)).click(selectables.get(2)).click(selectables.get(4)).build().perform();

1. **Using Mouse**

//3. Using Mouse

selectitems.clickAndHold(selectables.get(1));

selectitems.clickAndHold(selectables.get(3));

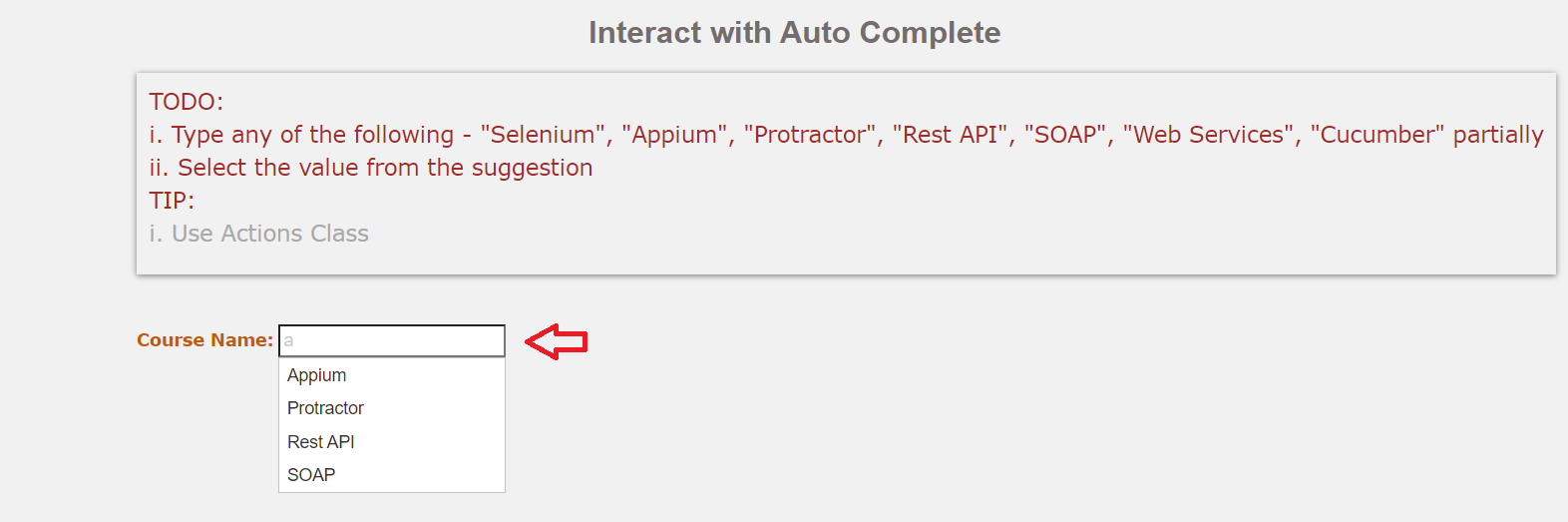
selectitems.build().perform();

**Class 🡪 AutoCompletionExample**

**What is Auto Completion 🡪** When we enter some text in textbox to search something, based on the letter we key in, it will auto populate few words for better search.

Say Instance(Google Search)

In this example we have to identify a textbox which supports auto completion and select an item form the list by key in only partial letters.



// Identifying Web Elements

WebElement textbox = driver.findElement(By.*id*("tags"));

textbox.sendKeys("a");

Thread.*sleep*(3000);

*// Since it’s an UL(Unordered List), we are finding this elements using find****Elements****(By.xpath) also adding “****/li****” at the end of xpath. and assigning it to return type* ***List<WebElement>*** List<WebElement>listelements = driver.findElements(By.*xpath*("//\*[@id=\"ui-id-1\"]/li"));

// Using println, to know how many items in the list are identified by List <WebElement>,in our case it is 4

System.***out***.println("Number of items in the list : "+listelements.size());

// **Result 🡪** Number of items in the list : 4

// Using for each loop to select our item from the auto completion list.

**for** (WebElement webElement : listelements) {

**if** (webElement.getText().equals("Protractor")) {

webElement.click();

// Using println, to know which item has been clicked by script

System.***out***.println("selected element is : "+webElement.getText());

**break**;

**Break 🡪**If my condition matches in the 1st iteration itself, then I no need to wait for the next iteration to complete the loop, it will cause unnecessary delay.

To avoid this situation and terminate the loop, we are using break statement

**Class 🡪 DownloadFileExample**

In this scenario we are going to download a file from webpage and check whether it is downloaded in the correct location.

WebElement downloadhome = driver.findElement(By.*xpath*("//\*[@id=\"post-153\"]/div[2]/div/ul/li[18]/a/img"));

downloadhome.click();

Thread.*sleep*(2000);

WebElement downloadexcel = driver.findElement(By.*linkText*("Download Excel"));

downloadexcel.click();

Thread.*sleep*(3000);

We cannot check file downloaded to respective location using Selenium script. Since Selenium is used only for Webpage.

Hence we are going to use Java script, to execute the same. We have a class called **File**

//Creating new object with file download location with variable name filelocation

File filelocation = **new** File("C:\\Users\\Senthil\\Downloads");

//we have to get the list of files available/downloaded in the above path, using listFiles(), method

File[] getallfiles = filelocation.listFiles(); //storing all available files in **getallfiles** variable from the above location

//Using for each loop, to identify our downloaded file

**for** (File file : getallfiles) {

**if** (file.getName().equals("testleaf.xlsx")){

System.***out***.println("File available in downloads");

**break**; // using break statement to exit the loop

**Class 🡪 UploadFileExample**

File upload can be done in 3 different ways in selenium.

1) Auto IT tool,

2) Robot Class

3) Using SendKeys.

In this scenario we are going to upload a file to webpage using **Robot** Class. To interact with windows pop-up and file upload we are going to use **Toolkit, StringSelection,** along with **Robot** Class**.**

**Actions() 🡪**Creating an Action class in order to click the "choose File" button using keyboard. Reason mouse action click() is not working on the web page due to change in code.

Actions buttonclick = **new** Actions(driver); //Creating new Actions class

WebElement button1 = driver.findElement(By.*name*("filename"));

//Instructing code to click the "Choose File" button using keyboard.

buttonclick.moveToElement(button1).click().build().perform();

**StringSelection() 🡪** This class will convert/transfer the Java string to plain text. Whenever we use upload files, we have to use this string selection class.

//Using string selection class we are transferring the actual url to plain text and storing in a variable **urlpath**

StringSelection urlpath = **new** StringSelection

("C:\\Users\\Senthil\\Downloads\\testleaf.xlsx");

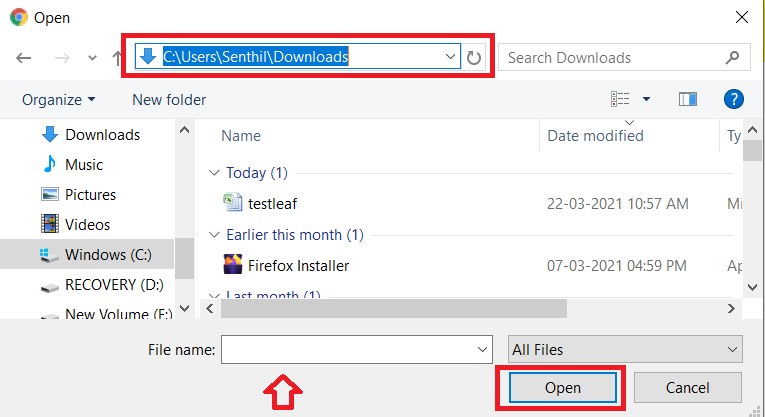
**Toolkit() 🡪** It’s used to access the windows tools, say instance(systemSelection, systemClipboard.. etc). Since selenium cannot do any action on windows based app/tools we have to use **Toolkit**

//Using ToolKit class we are setting content to clipboard, **urlpath** refers to transferred text

Toolkit.*getDefaultToolkit*().getSystemClipboard().setContents(urlpath, **null**);

**Robot()** 🡪 To stimulate keyboard actions(like **Actions** class used for web elements).Using Robot class we are going to perform keyboard operation in windows based app/tools.

Here we going to past the file location(C:\\Users\\Senthil\\Downloads\\testleaf.xlsx) to filename text Box and click Enter/open to upload the file to web page.



Robot copypasteactions = **new** Robot(); // Creating new object for Robot

// using this robot class doing keyboard action to press **CTRL + V**

copypasteactions.keyPress(KeyEvent.***VK\_CONTROL***);

copypasteactions.keyPress(KeyEvent.***VK\_V***);

// using this robot class doing keyboard action to Release **CTRL + V** in order to past the content

copypasteactions.keyRelease(KeyEvent.***VK\_V***);

copypasteactions.keyRelease(KeyEvent.***VK\_CONTROL***);

// using this robot class doing keyboard action to Press and Release **ENTER** key in order to upload the file.

copypasteactions.keyPress(KeyEvent.***VK\_ENTER***);

copypasteactions.keyRelease(KeyEvent.***VK\_ENTER***);

**Class 🡪 ImageExample**

In this scenarios we are going to work with images, using click() operation. It’s simple as other elements like button. Identify the element and do a click operation.

We have one more scenario, to identify whether the image is broken or not. To do the same we are using a CSS attribute “**naturalWidth**”. If it set to zero(0), it means image is broken.

// Identifying web element,

WebElement imagebroken = driver.findElement(By.*xpath*("//\*[@id=\"contentblock\"]/section/div[2]/div/div/img"));

//Using CSS attribute "naturalWidth"in **if** condition to identify whether image is broken

**if**(imagebroken.getAttribute("naturalWidth").equals("0"))

{

System.***out***.println("Image is Broken");

}

**else** {

System.***out***.println("Image is not broken");

}

**Class 🡪 HyperlinkExample**

In this scenario we are going to work with **links** complex task and going to explore 2 new methods **getTitle()** and driver.**navigate()** in this

// Clicking home page link

WebElement homepage = driver.findElement(By.*linkText*("Go to Home Page"));

homepage.click();

Thread.*sleep*(2000);

// going back to home page using **driver.navigate()**

driver.navigate().back();

// Finding where link is supposed to go without clicking it

WebElement findlink = driver.findElement(By.*partialLinkText*("Find where"));

String findele = findlink.getAttribute("href");

System.***out***.println("Link goes to : "+findele);

// Verifying whether the link is broken using **getTitle()**

WebElement brokenlink = driver.findElement(By.*partialLinkText*("Verify"));

brokenlink.click();

String pagetitle = driver.getTitle();

**if**(pagetitle.contains("404")) {

System.***out***.println

("This page you have trying to access is broken and getting :"+pagetitle);

}

// going back to home page using **driver.navigate()**

driver.navigate().back();

Thread.*sleep*(2000);

// Clicking on the same element 2nd time to go to homepage. Hence finding element again

// Else we get stale element exception exception in selenium

WebElement homepage2= driver.findElement(By.*linkText*("Go to Home Page"));

homepage2.click();

// going back to home page

driver.navigate().back();

// Finding how many links are available in this web page using **tagName()**

List<WebElement> nooflinks = driver.findElements(By.*tagName*("a"));

System.***out***.println("No of link in this web page : "+nooflinks.size());

Thread.*sleep*(2000);

//Closing browser

driver.close();

**Difference between driver.get() and driver.navigate()**

**driver.get()** 🡪 Using this method, we cannot refresh, Go back and forth on a webpage. It will just take you the web page

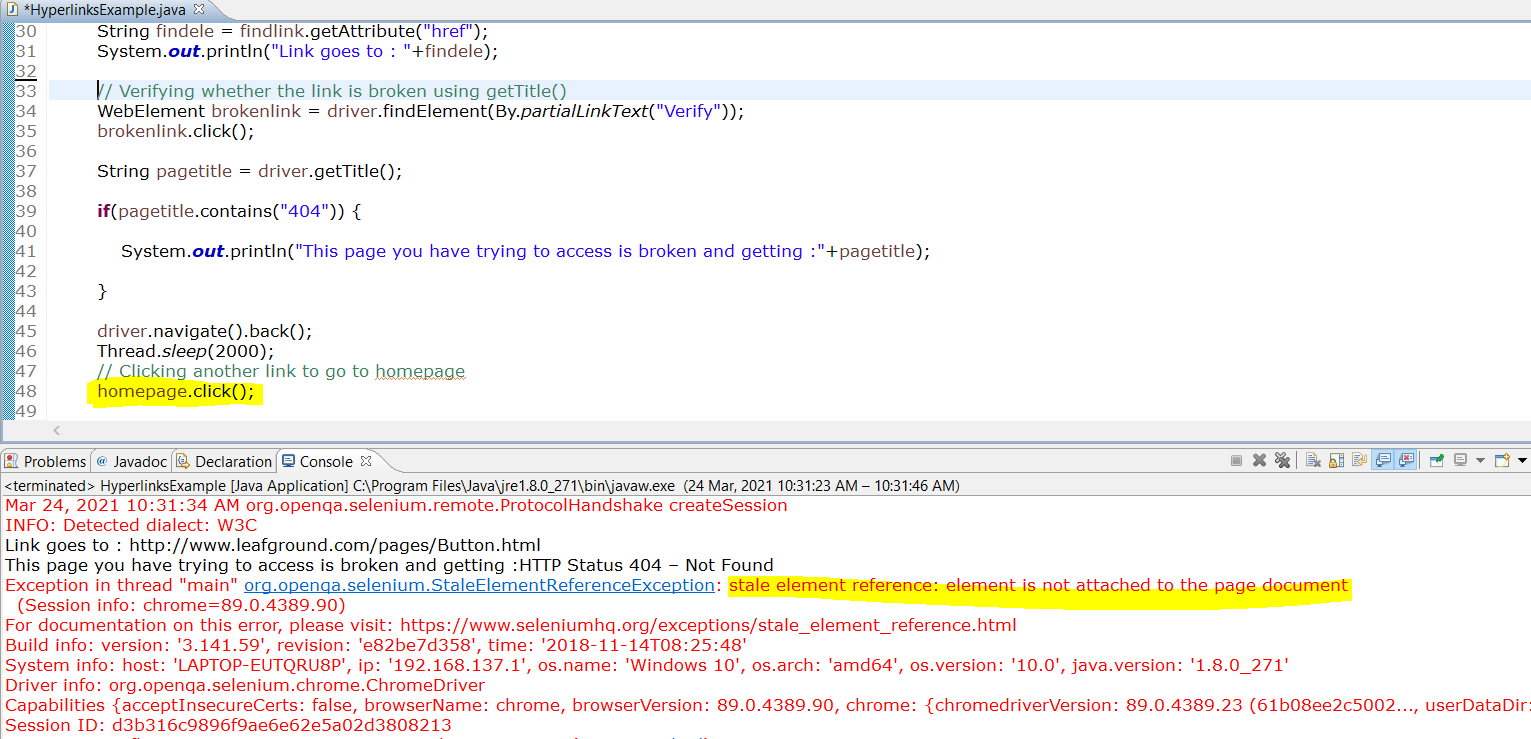
**driver.navigate() 🡪** Using this method, we can access the web page, also it will refresh, Go back and forth on a webpage.

**getTitle() 🡪** This method will get you the title of webpage.

**Stale Element Reference Exception 🡪**

When we identified an element and performed some action, and then, when you try to access the same web element once the webpage is refreshed/re-directed.

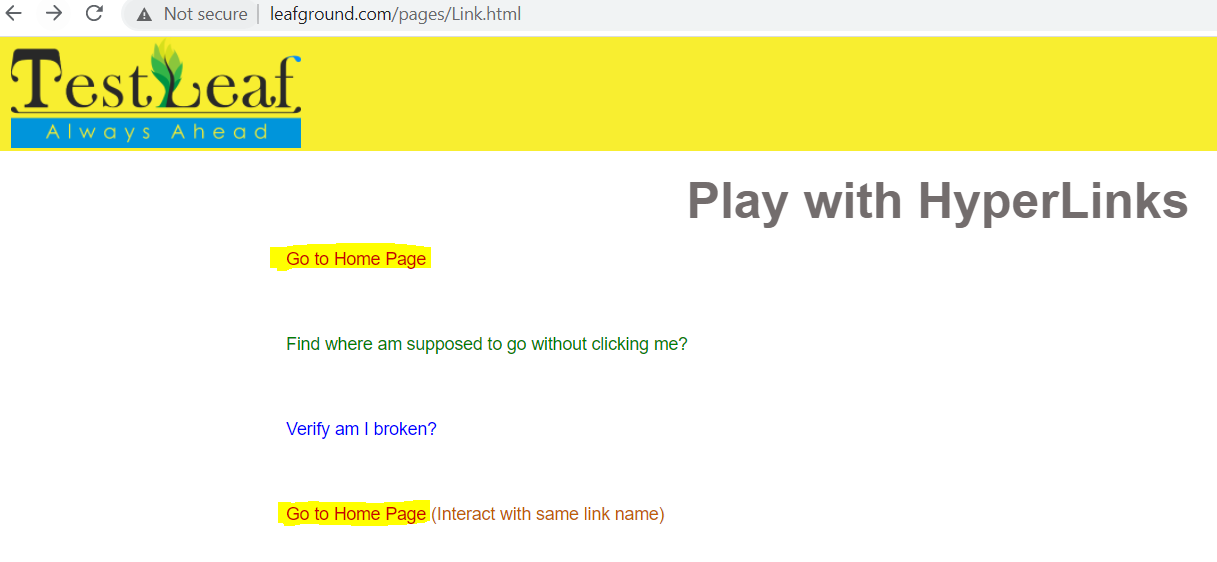
We cannot access the web element, reason after refresh/page re-direct, peoperties of elements also get refreshed. In that case we have to identify the element again in order to perform any action.



Say Instance, in the below image, we have clicked on the **1st home page** link, after performing some actions in the page, we are again trying to click the **4th home page** link.

Even though both the **home page** elements are similar, since page got refreshed and re-directed many times, we are not able to identify the web element.

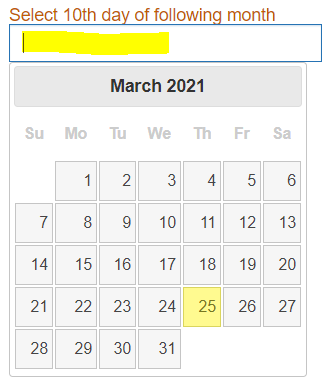
Hence we have to identify the same element again in order to go to home page.



**Class 🡪 CalendarExample**

There are different types of calendar, developers will build calendar based on the client requirement. So it’s vary webpage to webpage, most common calendar is **Jquery Calendar.**

**Sample Jquery Calendar**

****

**There are different ways to select date using calendar**

1. **Using Text Box**

If calendar comes with textbox, we can enter the dates using sendKeys.

WebElement txtbx = driver.findElement(By.*id*("datepicker"));

txtbx.sendKeys("04/10/2021"+Keys.***ENTER***);

1. **Using Calendar Icons**

// Identifying the element and clicking on it to open calendar

WebElement txtbx = driver.findElement(By.*id*("datepicker"));

txtbx.click();

// Identifying the element using Xpath and clicking on next icon on Calendar to select next month

WebElement nxtbtn = driver.findElement(By.*xpath*("//a[@title ='Next']"));

nxtbtn.click();

//Identifying the element using Xpath and Clicking on the date

WebElement selectdt = driver.findElement(By.*xpath*("//\*[@id=\"ui-datepicker-div\"]/table/tbody/tr[2]/td[7]/a"));

selectdt.click();

**Class 🡪 SortableExample**

In this scenario, we are going to sort the elementsfrom the list.

driver.get("http://www.leafground.com/pages/sortable.html");

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

// Creating Action class to perform actions using Keryboard/Mouse

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

// Identifying the items in the list and storing in a variable **items** using List<WebElement>

List<WebElement> items = driver.findElements(By.*xpath*("//\*[@id=\"sortable\"]/li"));

System.***out***.println("no of items in the list "+items.size());

There are 2 methods we can sort the items in the list

**Method 1**

// storing the element in a variable which we going to drag and drop

WebElement fromelement = items.get(6);

WebElement toelement = items.get(0);

moveitem.clickAndHold(fromelement);

moveitem.moveToElement(toelement);

moveitem.release(fromelement);

moveitem.build().perform();

**Method 2**

// Using variable(items) which declared to find list of items and declaring the items to be moved to index(0), (1).. etc inside **dragAndDrop ()**

moveitem.dragAndDrop(items.get(6), items.get(0)).build().perform();

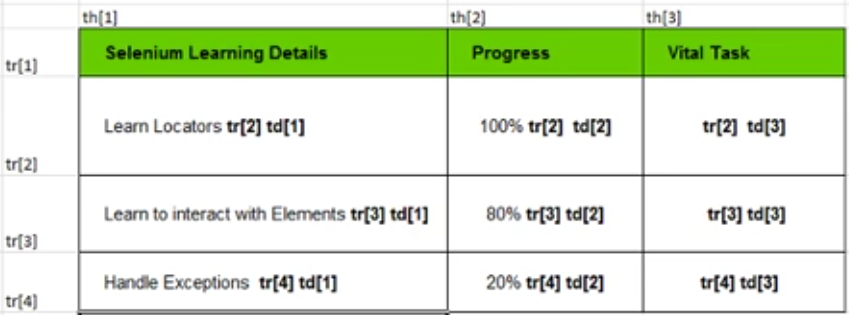
**Class 🡪 DynamicTableExample**

In this scenarios we are going to perform few task with Dynamic tables.

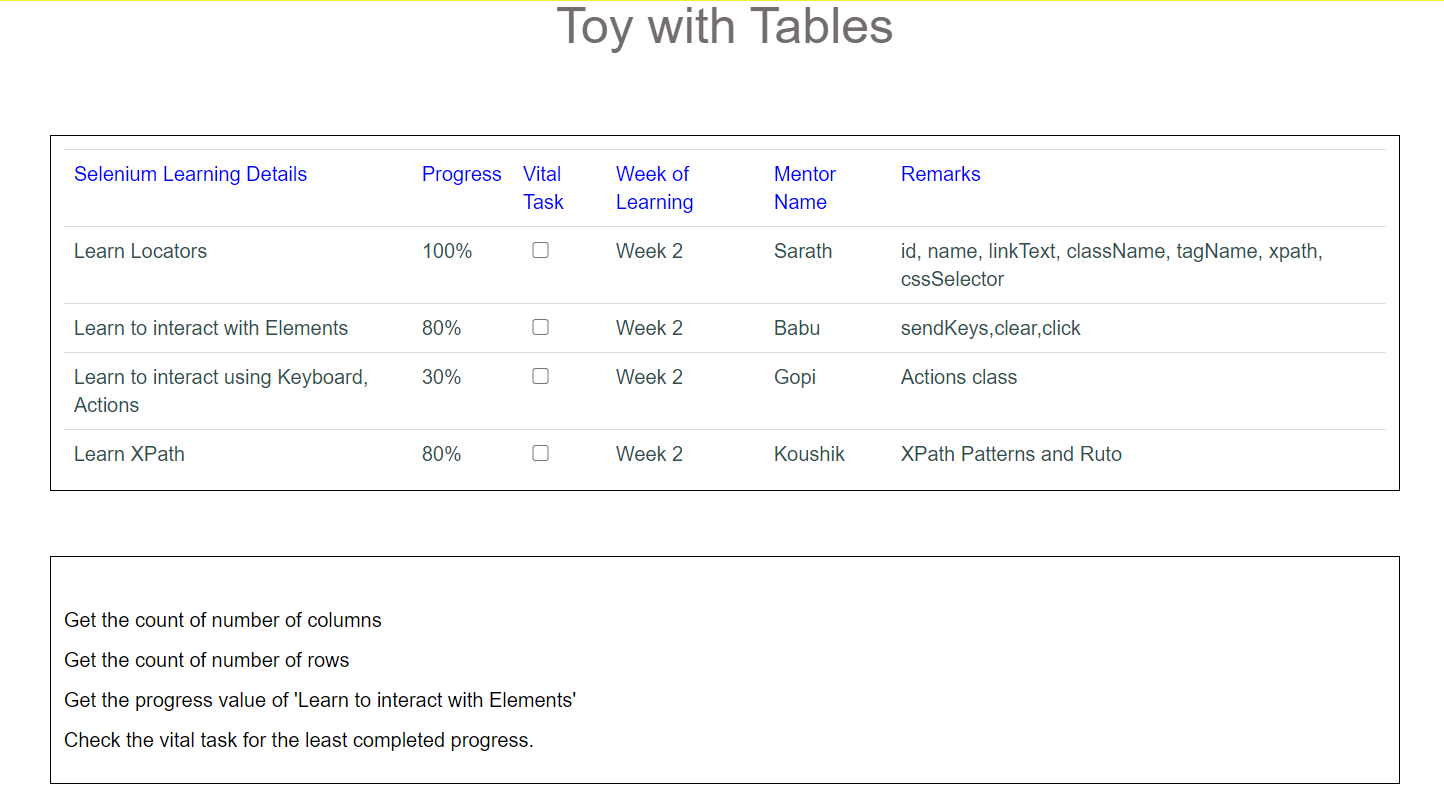
**Tasks to be performed**

1. Get the count of No. of Columns
2. Get the count of No. of Rows
3. Get the Progress value of “Learn to interact with Elements”
4. Check the Vital Task for the least completed progress.

**Sample Structure of HTML Table:**



**Table we are going to work with:**



1. **Get the count of No. of Columns**

// Finding no of columns using findElement by tagName "th"

List<WebElement> cols = driver.findElements(By.*tagName*("th"));

**int** countcols = cols.size();

System.***out***.println("No of Columns in this table is : "+countcols);

**Result 🡪** No of Columns in this table is : 6

1. **Get the count of No. of Rows**

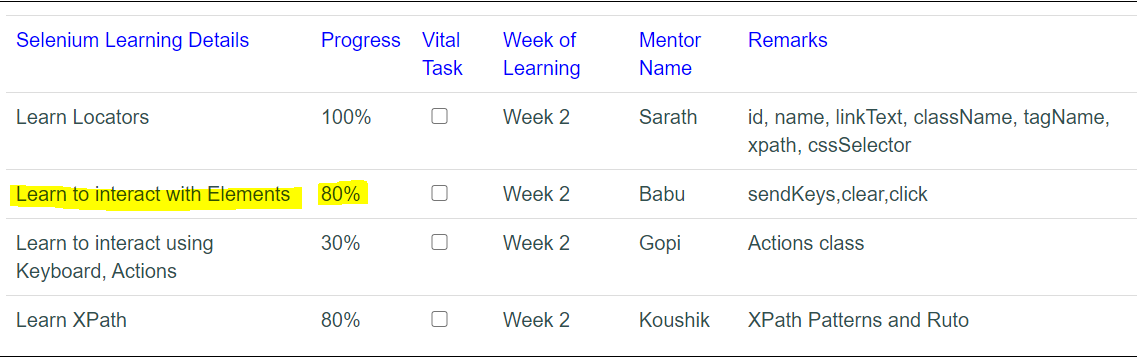
// Finding no of rowss using findElement by tagName "tr"

List<WebElement> rows = driver.findElements(By.*tagName*("tr"));

System.***out***.println("No of Rows in this table is : "+rows.size());

**Result 🡪** No of Rows in this table is : 5

1. **Get the Progress value of “Learn to interact with Elements”**

****

//Creating own X-path to get the progress value

/\* //td[normalize-space()=”Learn to interact with Elements”]//following::td[1] \*/

WebElement progvalue = driver.findElement(By.*xpath*("//td[normalize-space()=\"Learn to interact with Elements\"]//following::td[1]"));

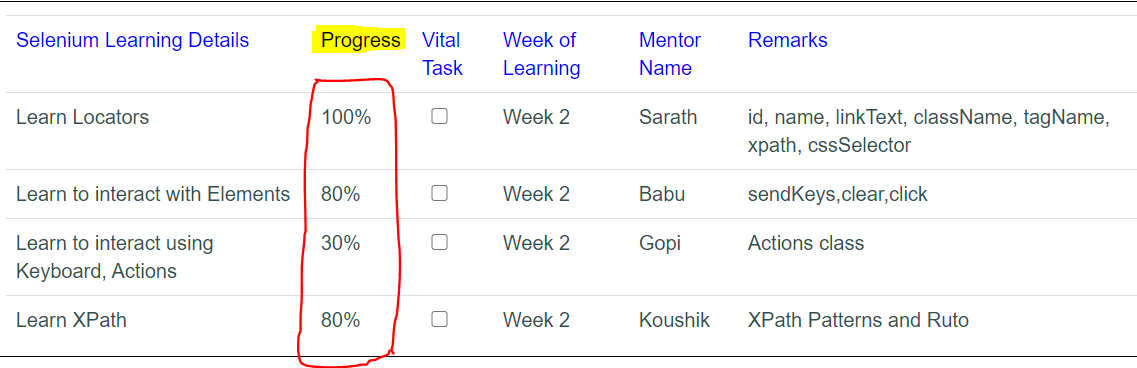
//Storing all progress value in a variable “**progvalue**”, getting the text of variable using getText() and printing the same.

String getval = progvalue.getText();

System.***out***.println("Progress Value of Learn to Interact with Elements is : "+getval);

**Result 🡪** Progress Value of Learn to Interact with Elements is : 80%

1. **Check the Vital Task for the least completed progress.**
2. In this scenario 1st we have to collate total no of values(String) under progress column
3. Once get the values, convert the String to Integer using Integer.ParseInt() inside for loop and store in a List<Integer>
4. Find out the least value in the list using collections.min(<variable name>)
5. Now again convert Integer to String Value using Integer.toString()
6. Create own x-path to identify the checkbox element next to least value(i.e 30%) and do a click operation.



//Creating own x-Path to get the list of values under progress tab, as highlighted above

List<WebElement> progresslist = driver.findElements(By.*xpath*("//td[2]"));

//Creating new object to store the values, converted to Integer from String

List<Integer> storevalues = **new** ArrayList<Integer>();

//Converting String to Integer Values and storing it in a variable "**storevalues**" using for loop

**Integer.*parseInt* () 🡪** To Convert string to Integer

**Replace(‘old’,’new;) 🡪** To replace the letters/special characters from a String

<**variable>.add() 🡪** To Store the converted Integers in a variable “storevalues”

**for** (WebElement webElement : progresslist) {

String values = webElement.getText().replace("%", "");

storevalues.add(Integer.*parseInt*(values));

}

System.***out***.println("List of Progress Value is : "+storevalues);

**Result 🡪** List of Progress Value is : [100, 80, 30, 80]

After getting the values form the list, we have to find the least number. So we using “**collections**” in Java and storing it **int** return type variable “**smallvalue**”

//Find least values

**int** smallvalue = java.util.Collections.*min*(storevalues);

System.***out***.println("Smallest num is : "+smallvalue);

**Result 🡪** Smallest num is : 30

//Now converting the Integer to String in order to get the x-path and click the next cell of least value

**Integer.toString ()🡪** Convert Integer to String, we adding “%” to that converted string

String convertedstring = Integer.*toString*(smallvalue)+"%";

//Expected X-path //td[normalize-space()="30%"]//following::td[1]//input derived from webpage, x-path to click on the checkbox next to value 30%

//Now creating our own X-path same as above, in order to check the correctness of x-path we are using print statement

**//td[normalize-space()="** 🡪 Upto this it’s a normal string

**+"\""+convertedstring+"\""+"] 🡪** Adding value **30%”]** which is stored in a variable **convertedstring**

**“+"//following::td[1]"+"//input"** 🡪 Adding rest of the values

String findxpath = "//td[normalize-space()="+"\""+convertedstring+"\""+"]"

+"//following::td[1]"+"//input";

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

**Result 🡪** //td[normalize-space()="30%"]//following::td[1]//input

//Once identified the correct x-path, clicking the checkbox

WebElement chkbox = driver.findElement(By.*xpath*(findxpath));

chkbox.click();

**Result 🡪**

