## Selenium: Creating Driver\\

| Slno | Action | Selenium code | Description |
|---|---|---|---|
| 1 | Firefox driver | System.setProperty("webdriver.gecko.driver",PATH); WebDriver driver = new FirefoxDriver(); | PATH = path of Gecko exe file; |
| 2 | IE driver | System.*setProperty*("webdriver.ie.driver",PATH); WebDriver driver = **new** InternetExplorerDriver(); | PATH = path of IEDriver exe file; |
| 3 | chrome Driver | System.*setProperty*("webdriver.ie.driver",PATH); WebDriver driver = **new** ChromeDriver(); | PATH = path of chrome exe file; |

## Selenium: opening an application

| slno | Action | code |
|---|---|---|
| 1 | Opening an application | driver.get("url");<br><br>driver.navigate.to(url); |

## Selenium: Indentify Elements

| Slno | Action | Code | Description |
|---|---|---|---|
| 1 | Find single Element | driver.findElement(locator) | Locator is a location of element |
| 2 | Find multiple Elements | driver.findElements(locator) | Locator is a location of element |

## Selenium: Locating UI Elements

| Slno | Action | Code | Description |
|---|---|---|---|
| 1 | By ID | driver.findElement(By.id(str)); | Str is id of element |
| 2 | By Name | driver.findElement(By.name(str)); | Str is name of element |
| 3 | By class name | driver.findElement(By.className(str)); | Str is class value of element |
| 4 | By css selector | driver.findElement(By.cssSelector(str)); | Str is cssSelector of element |
| 5 | By link text | driver.findElement(By.linkText(str)); | Str is link text of element |
| 6 | By partial link text | driver.findElement(By.partialLinkText(str)); | Str is partial text of element |
| 7 | By tag name | driver.findElement(By.tagName(str)); | Str is tag name of element |
| 8 | By XPath | driver.findElement(By.xpath(xpath)); | Str is xpath of element |

## Selenium: User Actions

| Slno | Action | Code | Description |
|---|---|---|---|
| 1 | Write in text fields | driver.findElement(locator).sendKeys(text); | Text: what u want to write<br>locator is a location element |
| 2 | Click button or click radio button or check box | driver.findElement(locator).click(); | locator is a location element |
| 3 | Clear text in text field | driver.findElement(locator).clear(); | locator is a location element |
| 4 | Navigate back and forward in browser | driver.navigate().back();<br>driver.navigate().forward(); | |
| 5 | Navigate to frame | driver.switchTo().frame(frame); | frame can be integer value |

| | | | |
|---|---|---|---|
| | | | represents position of frame or string represents id of frame or WebElement represents frame of frame. |
| 6 | Navigate to next window or pop up window | driver.switchTo().window(hashCode); | hashCode is hash code of window |
| 7 | Get inner text of element or inner text of table | driver.findElement(locator).getText(); | locator is a location element |
| 8 | Working on auto complete/suggestions Or Calendar pop up | driver.findElement(locator).click(); | Get the locator of hidden division or element and perform required operation |

## Selenium: Drag, Drop and Mouse Over, Mouse Events

We use Actions Class for drag and drop

1st Create an object to action class

```
Actions act = new Actions(driver);
```

There are 2 ways to do drag and drop

| slno | Action | code | description |
|---|---|---|---|
| 1 | Drag and Drop using source and destination | act.dragAndDrop(src, des).build().perform(); | Src and dest is the webElement object of source and destination of drag and drop element. |
| 2 | Drag and Drop to specific position | act.dragAndDropBy(src, x,y).build().perform(); | x and y are integer values for specific position. |
| 3 | Mouse over on specific element. | act.moveToElement(element).build().perform(); | Element is an object of WebElement which points to required element. |
| 4 | Mouse right click | act.contextClick(element).build().perform(); | Element is an object of WebElement which points to required element. |
| 5 | Mouse movement after right click | act.sendKeys(Keys.<keyboardstrokes>).build().perform(); | Keys is a class contains all key strokes such as right left, enter, back button etc |

## Selenium: File Download

To automate file download in fire fox we have to do following steps:

1. **Set the path where to download.**
2. **Set browser preferences not to ask confirmation.**
1. **Set the path where to download**

For this we have set *browser.download.folderList* preference to integer value. The values are
*0 ------ to save in desktop*
*1 ------ Save in downloads folder in your computer*
*2 ----- User defined path*

If the value is 2 then we have to set *browser.download.dir* preferences to path of specific folder.

2. **Set browser preferences not to ask confirmation**
   The *browser.helperApps.neverAsk.saveToDisk* preference to be set to the MIME Type of file, we need to download.

1st we need to create fire fox profile for browser by creating object to FirefoxProfile:

```
FirefoxProfile prof = new FirefoxProfile();
```

| Slno | Action | Code | Description |
|---|---|---|---|
| 1 | Set browser.download.folderList | prof.setPreference("browser.download.folderList",integerVal); | Use object created for fire fox profile and integerVal is 0 or 1 or 2 based location where you want to save. |
| 2 | Set browser.download.dir | prof.setPreference("browser.download.dir", "d:\\"); | If integerVal in above code is 2 then set the path where you need to download the file. |
| 3 | Set browser.helperApps.neverAsk.saveToDisk | prof.setPreference("browser.helperApps.neverAsk.saveToDisk", MimeType); | MimeType is MIME type of file |
| 4 | Use profile created to create driver | WebDriver driver = new FirefoxDriver(prof); | |

## Selenium: Handling Java Script Alerts

To handle alert first we need to switch to alert.

```
Alert al = driver.switchTo().alert();
```

The Actions list.

| slno | Action | code |
|---|---|---|
| 1 | Click on ok in alert | al.accept(); |
| 2 | Click on cancel. | al.dismiss() |
| 3 | Type in alert box. | al.sendKeys(“hi”); |
| 4 | Get text from alert box. | al.getText(); |

## Selenium: Handling Un Trusted Certificate Exception

The below code used to handle un trusted certificate exception.

```
FirefoxProfile prof = new FirefoxProfile(); //create firefox profile
prof.setAcceptUntrustedCertificates(true);
prof.setAssumeUntrustedCertificateIssuer(true);
WebDriver driver = new FirefoxDriver(prof);
```

## Selenium: Capture Screen Shot of Browser

| Slno | Action | Code | Description |
|---|---|---|---|
| 1 | Capture screen | File scrFile1 = ((TakesScreenshot)*driver*).getScreenshotAs(OutputType.*FILE*); | It captures screen shot of particular page and stores it in variable |
| 2 | Save to disk | FileUtils.*copyFile*(scrFile1, **new** File("c:\\tmp\\k2.png")); | Save screen shot as k2.png |

## Selenium: Work on drop down list (select drop down list)

Using Select class we can work on select drop down. Create select object for specific select drop down.

```
WebElement usrs = driver.findElement(By.name("users"));  //create WebElement for select drop down
  Select usr = new Select(usrs);
```

We can select options of drop down in 3 different ways as explained below

| slno | Action | code | description |
|---|---|---|---|
| 1 | Select by using id of option tag | usr.selectById(id); | Id is string, value of id attribute of option. |
| 2 | Select by using index of option tag | usr.selectByIndex(i); | I is the position of option |
| 3 | Select by using visible text in option tag | usr.selectByVisibleText(str) | str is the text inside option tag. |

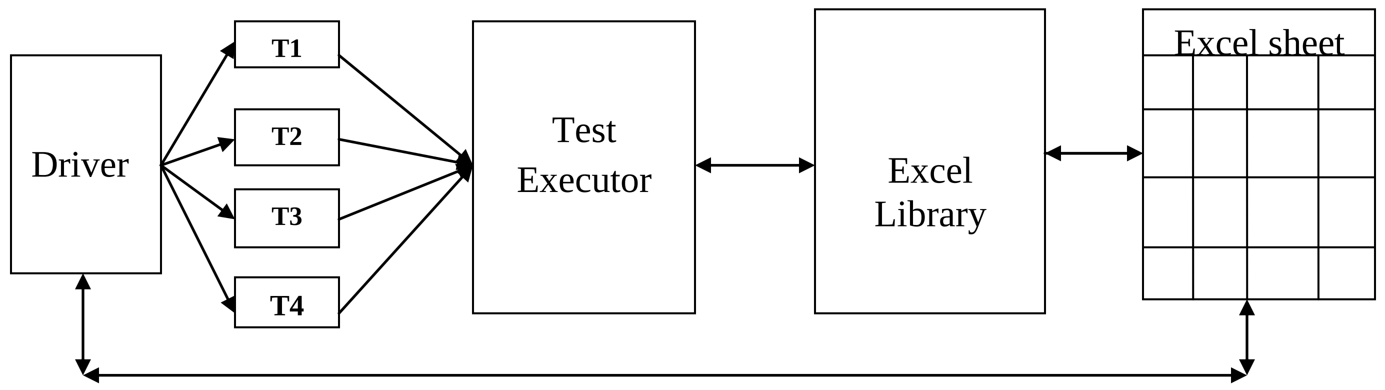
## Selenium: Working on excel sheet

Before working on excel first we need to read excel in input stream using file io stream.

```
FileInputStream fis = new FileInputStream("D:\\jul_weekend\\hybridframework\\tests\\testscenarios.xlsx");
```

| slno | Action | Code | Description |
|---|---|---|---|
| 1 | Convert file io into workbook | `Workbook wb = WorkbookFactory.create(fis);` | Create function creates work book. |
| 2 | Get into specified sheet | `Sheet s = wb.getSheet(sheetName);`<br>Or<br>`Sheet s = wb.getSheetAt(sheetNum);` | sheetName is name of the sheet<br>sheetNum is index of sheet |
| 3 | Get into specified row | `Row r = s.getRow(rowNum);` | |
| 4 | Get into specified column | `Cell c = r.getCell(colNum);` | |
| 5 | Get cell value | `String retVal = c.getStringCellValue();`<br>Or<br>`boolean b = c.getBooleanCellValue();`<br>or<br>`Date d = c.getDateCellValue();`<br>Or<br>`int I = c.getNumericCellValue();` | Get cell value based on value in excel cell |
| 6 | Get row count | int  I = s.getLastRowNum(); | |
| 7 | Get Column count | int  j = r. getLastCellNum (); | |
| 8 | Write back to excel | c.setCellValue("PASS1");<br>FileOutputStream fos = **new** FileOutputStream("C:\\Documents and Settings\\mahesh\\Desktop\\Book1.xlsx");<br>wb.write(fos);<br>fos.close(); | |

## Selenium: Hybrid Frame Work Explanation



**Frame Explain**

1. Our frame work is keyword hybrid frame work which runs from excel.
2. Driver is the starting point of execution. Driver launches the application, adds the required tests to the test suite, and runs the test suite. The test required to run will be present in excel file called "TestScenarios". The sheet called test suit in excel file contains list of all the test cases to be run with execution status against each script. If execution status is yes the test is executed else it is skipped.
3. Driver has the code to read this excel sheet and based on the execution status, script will be added to the test suite.
4. We use TestN(JUnit) to execute the tests.
5. When driver runs the tests, it creates an object of test executor class and calls the execute test method.
6. Test executor connects to excel through the particular sheet of the scenario which is running, loops through all the test steps, and executes the steps.
7. To connect with excel, apache POI API are used. The inbuilt methods are used get data from excel sheet.
8. When each step is executed, script checks the element is present and performs the action. For every action performed log message is generated which is written back to the excel sheet.