**Interview questions page**

https://www.facebook.com/permalink.php?story\_fbid=863296403759120&id=180086875413413

[How to type in textbox using Selenium WebDriver (Selenium 2) with Java?](https://stackoverflow.com/questions/16481418/how-to-type-in-textbox-using-selenium-webdriver-selenium-2-with-java)

package facebook;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.interactions.Actions;

public class Facebook {

public static void main(String args[]){

WebDriver driver = new FirefoxDriver();

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

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

Actions builder = new Actions(driver);

Actions seriesOfActions = builder.moveToElement(email).click().sendKeys(email, "gati.naveen@gmail.com");

seriesOfActions.perform();

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

WebElement login =driver.findElement(By.id("u\_0\_b"));

Actions seriesOfAction = builder.moveToElement(pass).click().sendKeys(pass, "naveench").click(login);

seriesOfAction.perform();

driver.

}

}

In Selenium .properties files are mainly used to store GUI locators / elements, and also Global fields like database configuration details

**'.properties' files are mainly used in Java programs to maintain project configuration data, database config or project settings etc**. Each parameter in properties file are stored as a pair of strings, in key and value format, where each key is on one line. You can easily read properties from some file using object of type Properties.

Below is a example program which demonstrate to read the data from .properties file using Java.

File file = new File("D:/Dev/ReadData/src/datafile.properties");

fileInput = new FileInputStream(file);

Properties prop = new Properties();

prop.load(fileInput);

WebDriver driver = new FirefoxDriver();

driver.get(prop.getProperty("URL")); driver.findElement(By.id("Email")).sendKeys(prop.getProperty("username"));

Enumeration KeyValues = prop.keys();

while (KeyValues.hasMoreElements())

{String key = (String) KeyValues.nextElement();

String value = prop.getProperty(key);

System.out.println(key + ":- " + value);

}

**objectmap.properties file**

Username\_field =id:login\_login\_username  
Password\_field =id:login\_login\_password  
Login\_button =id:login\_submit  
online\_user=cssSelector:#sb-onlineusers > h3

String locator = properties.getProperty(ElementName);  
String locatorType = locator.split(":")[0];  
String locatorValue = locator.split(":")[1];  
**if**(locatorType.toLowerCase().equals("id"))  
                 **return**By.id(locatorValue);

WebElement username = driver.findElement(objmap.getLocator("Username\_field"));

# [Implementing Assert and Verify logic in Selenium WebDriver](http://www.testerlogic.com/assert-verify-logic-selenium-webdriver/)

Selenium IDE has built in VerifyTextPresent, AssertElementPresent commands. In Selenium WebDriver there are definitely many ways to do it in with support of testing frameworks like JUnit or TestNG etc.

**1.1 Verify Text Present:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **if(driver.getPageSource().contains("**Text - Testing with Arif"))  {  System.out.println("Text is Present");  }  else  {  System.out.println("Text is not Present");  } |

OR

|  |  |
| --- | --- |
| 1  2  3  4  5 | try {  **assertTrue(**driver.findElement(By.cssSelector("BODY")).getText().matches("^[\\s\\S]\*verify text is present[\\s\\S]\*$"));  } catch (Error e) {  verificationErrors.append(e.toString());  } |

OR

|  |  |
| --- | --- |
| 1  2 | **driver.findElement(By.xpath("//**span[contains(.,'Transaction was added successfully')]"));  System.out.println("Transaction successful"); |

OR

|  |  |
| --- | --- |
| 1  2  3  4  5 | try {  **assertEquals("**VerifyText in Element", driver.findElement(By.cssSelector("div.bbMargin")).getText());  } catch (Error e) {  verificationErrors.append(e.toString());  } |

### **1.2 Assert Text Present**

we want our scripts to stop if certain assertion fails, we can use assert methods from Junit, TestNG (and other alternative frameworks) to implement the assertion strategy but without Exception handling

(consider all above examples without try-catch)

### **2.1 Verify Element Present**

|  |  |
| --- | --- |
| 1 | **!driver.findElements(By.id("xyz")).isEmpty(); ---not confirmed maybe IDE** |

OR

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | if**(isElementPresent(**By.linkText("Submit"))) **----not confirmed maybe IDE**  {  System.out.println("SUBMIT Link/Button found");  }  else  {  System.out.println("SUBMIT Link/Button not found");  } |

OR

|  |  |
| --- | --- |
| 1  2  3  4  5 | try {  **assertTrue(isElementPresent(**By.cssSelector("div.bbMargin")));  } catch (Error e) {  verificationErrors.append(e.toString());  } |

### **2.2 Assert Element Present**

1. use both assert and isElementPresent together

**assertTrue(isElementPresent(**By.cssSelector("div.bbMargin")));

**Assert.assertEquals(**0, wd.findElements(By.locator("locator")).size());

[Extracting a number from a text with selenium webdriver/java](https://stackoverflow.com/questions/32602322/extracting-a-number-from-a-text-with-selenium-webdriver-java)

locate element according to your convenience and replace it with first line of code

String OSNAMES= Driver.findElement(By.xpath("YOUR XPATH")).getAttribute("value");

String[] parts = OSNAMES.split(" ");

String OS = parts[3];

int a=Integer.parseInt(OS);

System.out.println(a);

# [Reading text from textarea in webdriver](https://stackoverflow.com/questions/15513918/reading-text-from-textarea-in-webdriver)

WebElement text = wd.findElement(By.id("edit-pi-analytics-tms-id"));

String textagain = text.getAttribute("value");

# [Extracting URLs from a text document using Java + Regular Expressions](https://stackoverflow.com/questions/1806017/extracting-urls-from-a-text-document-using-java-regular-expressions)

import java.util.\*;

import java.util.regex.\*;

class FindUrls

{

public static List<String> extractUrls(String input) {

List<String> result = new ArrayList<String>();

Pattern pattern = Pattern.compile(

"\\b(((ht|f)tp(s?)\\:\\/\\/|~\\/|\\/)|www.)" +

"(\\w+:\\w+@)?(([-\\w]+\\.)+(com|org|net|gov" +

"|mil|biz|info|mobi|name|aero|jobs|museum" +

"|travel|[a-z]{2}))(:[\\d]{1,5})?" +

"(((\\/([-\\w~!$+|.,=]|%[a-f\\d]{2})+)+|\\/)+|\\?|#)?" +

"((\\?([-\\w~!$+|.,\*:]|%[a-f\\d{2}])+=?" +

"([-\\w~!$+|.,\*:=]|%[a-f\\d]{2})\*)" +

"(&(?:[-\\w~!$+|.,\*:]|%[a-f\\d{2}])+=?" +

"([-\\w~!$+|.,\*:=]|%[a-f\\d]{2})\*)\*)\*" +

"(#([-\\w~!$+|.,\*:=]|%[a-f\\d]{2})\*)?\\b");

Matcher matcher = pattern.matcher(input);

while (matcher.find()) {

result.add(matcher.group());

}

return result;

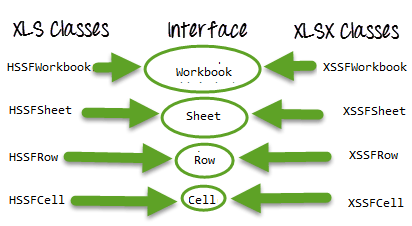
}

}

# Read & Write Data from Excel File in Selenium Webdriver: POI & JXL

To read or write an Excel,Apache provides a very famous library POI. This library is capable enough to read and write both**XLS** and**XLSX** file format of Excel.

### **Classes and Interfaces in POI:**

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc5.png)

Following is a list of different Java Interfaces and classes in**POI**for reading**XLS**and**XLSX**file-

* **Workbook**: XSSFWorkbook and HSSFWorkbook classes implement this interface.
* **XSSFWorkbook**: Is a class representation of XLSX file.
* **HSSFWorkbook**: Is a class representation of XLS file.
* **Sheet**: XSSFSheet and HSSFSheet classes implement this interface.
* **XSSFSheet**: Is a class representing a sheet in an XLSX file.
* **HSSFSheet**: Is a class representing a sheet in an XLS file.
* **Row**: XSSFRow and HSSFRow classes implement this interface.
* **XSSFRow**: Is a class representing a row in the sheet of XLSX file.
* **HSSFRow**: Is a class representing a row in the sheet of XLS file.
* **Cell**: XSSFCell and HSSFCell classes implement this interface.
* **XSSFCell**: Is a class representing a cell in a row of XLSX file.
* **HSSFCell:** Is a class representing a cell in a row of XLS file.

File file = new File(filePath+"\\"+fileName);

FileInputStream inputStream = new FileInputStream(file);

Workbook guru99Workbook = null;

String fileExtensionName = fileName.substring(fileName.indexOf("."));

if(fileExtensionName.equals(".xlsx")){

guru99Workbook = new XSSFWorkbook(inputStream);

}

else if(fileExtensionName.equals(".xls")){

guru99Workbook = new HSSFWorkbook(inputStream);

}

Sheet guru99sheet = guru99Workbook.getSheet(sheetName);

int rowCount = guru99sheet.getLastRowNum()-guru99sheet.getFirstRowNum();

**Read purpose:**

for (int i = 0; i < rowCount+1; i++) {

Row row = guru99Sheet.getRow(i);

for (int j = 0; j < row.getLastCellNum(); j++) {

System.out.print(row.getCell(j).getStringCellValue()+"|| ");

}

System.out.println();

}

**Write Purpose:**

**//Create a new row and append it at last of sheet**

Row row = sheet.getRow(0);

Row newRow = sheet.createRow(rowCount+1);

for(int j = 0; j < row.getLastCellNum(); j++){

Cell cell = newRow.createCell(j);

cell.setCellValue(dataToWrite[j]);

}

inputStream.close();

**//Create an object of FileOutputStream class to create write data in excel file**

FileOutputStream outputStream = new FileOutputStream(file);

guru99Workbook.write(outputStream);

outputStream.close();

}

# Handling Web Tables, Frames, and Dynamic Elements in Selenium

**html tables:**

1.’table’ tag defines html table.  
2.’tbody’ tag defines container for rows and columns.  
3.’tr’ defines rows in an html table.  
4.’td’/’th’ define column of an html table.

**Dynamic example:**

|  |  |  |
| --- | --- | --- |
| 1 | WebElement htmltable=**driver.findElement(By.xpath**("//\*[@id='main']/table[1]/tbody")); | |
| 2 | List<WebElement> rows=**htmltable.findElements(By.tagName**("tr")); |

|  |  |
| --- | --- |
| 3 |  |
| 4 | for(int rnum=0;rnum<rows.size();rnum++) | |

|  |  |
| --- | --- |
| 5 | { |
| 6 | List<WebElement> columns=rows.get(rnum).findElements(By.tagName("th")); | |

|  |  |  |
| --- | --- | --- |
| 7 | System.out.println("Number of columns:"+columns.size()); | |
| 8 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | for(int cnum=0;cnum<columns.size();cnum++) | | |
| 10 | | { |

|  |  |  |
| --- | --- | --- |
| 11 | System.out.println(columns.get(cnum).getText()); | |
| 12 | } |

|  |  |
| --- | --- |
| 13 | } |

### **Dynamic elements:**

**Problem Type 1:**If part of the attribute value changes**.**

driver.findElement(By.xpath(“//\*[contains(@id,’username’)]”)).sendKeys(“username”);  
driver***.***findElement(By.xpath(“//\*[starts-with(@id,’user’)]”)).sendKeys(“username”);

‘contains’ is a java method which checks if id contains the substring username.  
starts-with() checks if any attribute starts with “user”.

**Problem Type 2:**If entire value of the attribute changes dynamically.

Use of sendKeys. (tab key, enter keys, F5 etc)

1.driver.findElement(By.id(“password”)).sendKeys(“password”))

2.driver.findElement(By.id(“password”)).sendKeys(Keys.TAB));

**How to switch back to the Main Frame**

driver.switchTo().parentFrame(); //parent frame

driver.switchTo().defaultContent(); //most parent or main frame

**total number of iframes present inside the page**

int size = driver.findElements(By.tagName("iframe")).size();

**total number of iframes present inside the frame**

int sizeO = driver.findElements(By.tagName("iframe")).size();

driver.switchTo().frame(0);

sizeI = driver.findElements(By.tagName("iframe")).size();

# Flash Testing with Selenium

Flash Testing is testing type used to check the flash based video, games, movies, etc. are working as expected.

* Main difference between flash and other element is that Flash is embedded in SWF files, while other elements are embedded in HTML files

**Tools-**Below are the testing tools which are useful in flash testing.

1. Selenium
2. Soap UI
3. TestComplete
4. Test Studio etc.

[download flashwebdriver jar files](https://drive.google.com/file/d/0B2SE8u2xJC83Y3VzVTRkRmRPek0/view):

import Flash.FlashObjectWebDriver;

FlashObjectWebDriver flashApp **= new FlashObjectWebDriver(driver, "myFlashMovie");**

driver.get("http://demo.guru99.com/flash-testing.html");

Thread.sleep(5000);

flashApp.**callFlashObject("Play");**

Thread.sleep(5000);

flashApp.callFlashObject("StopPlay");

Thread.sleep(5000);

flashApp.callFlashObject("SetVariable","/:message","Flash testing using selenium Webdriver");

Java: To read lines from a text file, you can use this (uses try-with-resources):

String line;

try (

InputStream fis = new FileInputStream("the\_file\_name");

InputStreamReader isr = new InputStreamReader(fis, Charset.forName("UTF-8"));

BufferedReader br = new BufferedReader(isr);

) {

while ((line = br.readLine()) != null) {

// Do your thing with line

}

}

# Verify Image Presence in Web Page using Selenium WebDriver

@Test

public void CheckImage() throws Exception {

driver.get(baseUrl);

WebElement ImageFile = driver.findElement(By.xpath("//img[contains(@id,'Test Image')]"));

Boolean ImagePresent = (Boolean) ((JavascriptExecutor)driver).executeScript("return arguments[0].complete && typeof arguments[0].naturalWidth != \"undefined\" && arguments[0].naturalWidth > 0", ImageFile);

if (!ImagePresent)

{

System.out.println("Image not displayed.");

}

else

{

System.out.println("Image displayed.");

}

}

[Extracting a number from a text with selenium webdriver/java](https://stackoverflow.com/questions/32602322/extracting-a-number-from-a-text-with-selenium-webdriver-java)

locate element according to your convenience and replace it with first line of code

String OSNAMES= Driver.findElement(By.xpath("YOUR XPATH")).getAttribute("value");

String[] parts = OSNAMES.split(" ");

String OS = parts[3];

int a=Integer.parseInt(OS);

System.out.println(a);

# [Reading text from textarea in webdriver](https://stackoverflow.com/questions/15513918/reading-text-from-textarea-in-webdriver)

WebElement text = wd.findElement(By.id("edit-pi-analytics-tms-id"));

String textagain = text.getAttribute("value");

# [Extracting URLs from a text document using Java + Regular Expressions](https://stackoverflow.com/questions/1806017/extracting-urls-from-a-text-document-using-java-regular-expressions)

import java.util.\*;

import java.util.regex.\*;

class FindUrls

{

public static List<String> extractUrls(String input) {

List<String> result = new ArrayList<String>();

Pattern pattern = Pattern.compile(

"\\b(((ht|f)tp(s?)\\:\\/\\/|~\\/|\\/)|www.)" +

"(\\w+:\\w+@)?(([-\\w]+\\.)+(com|org|net|gov" +

"|mil|biz|info|mobi|name|aero|jobs|museum" +

"|travel|[a-z]{2}))(:[\\d]{1,5})?" +

"(((\\/([-\\w~!$+|.,=]|%[a-f\\d]{2})+)+|\\/)+|\\?|#)?" +

"((\\?([-\\w~!$+|.,\*:]|%[a-f\\d{2}])+=?" +

"([-\\w~!$+|.,\*:=]|%[a-f\\d]{2})\*)" +

"(&(?:[-\\w~!$+|.,\*:]|%[a-f\\d{2}])+=?" +

"([-\\w~!$+|.,\*:=]|%[a-f\\d]{2})\*)\*)\*" +

"(#([-\\w~!$+|.,\*:=]|%[a-f\\d]{2})\*)?\\b");

Matcher matcher = pattern.matcher(input);

while (matcher.find()) {

result.add(matcher.group());

}

return result;

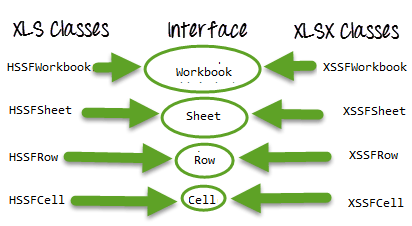
}

}

# Read & Write Data from Excel File in Selenium Webdriver: POI & JXL

To read or write an Excel,Apache provides a very famous library POI. This library is capable enough to read and write both**XLS** and**XLSX** file format of Excel.

### **Classes and Interfaces in POI:**

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc5.png)

Following is a list of different Java Interfaces and classes in**POI**for reading**XLS**and**XLSX**file-

* **Workbook**: XSSFWorkbook and HSSFWorkbook classes implement this interface.
* **XSSFWorkbook**: Is a class representation of XLSX file.
* **HSSFWorkbook**: Is a class representation of XLS file.
* **Sheet**: XSSFSheet and HSSFSheet classes implement this interface.
* **XSSFSheet**: Is a class representing a sheet in an XLSX file.
* **HSSFSheet**: Is a class representing a sheet in an XLS file.
* **Row**: XSSFRow and HSSFRow classes implement this interface.
* **XSSFRow**: Is a class representing a row in the sheet of XLSX file.
* **HSSFRow**: Is a class representing a row in the sheet of XLS file.
* **Cell**: XSSFCell and HSSFCell classes implement this interface.
* **XSSFCell**: Is a class representing a cell in a row of XLSX file.
* **HSSFCell:** Is a class representing a cell in a row of XLS file.

File file = new File(filePath+"\\"+fileName);

FileInputStream inputStream = new FileInputStream(file);

Workbook guru99Workbook = null;

String fileExtensionName = fileName.substring(fileName.indexOf("."));

if(fileExtensionName.equals(".xlsx")){

guru99Workbook = new XSSFWorkbook(inputStream);

}

else if(fileExtensionName.equals(".xls")){

guru99Workbook = new HSSFWorkbook(inputStream);

}

Sheet guru99sheet = guru99Workbook.getSheet(sheetName);

int rowCount = guru99sheet.getLastRowNum()-guru99sheet.getFirstRowNum();

**Read purpose:**

for (int i = 0; i < rowCount+1; i++) {

Row row = guru99Sheet.getRow(i);

for (int j = 0; j < row.getLastCellNum(); j++) {

System.out.print(row.getCell(j).getStringCellValue()+"|| ");

}

System.out.println();

}

**Write Purpose:**

**//Create a new row and append it at last of sheet**

Row row = sheet.getRow(0);

Row newRow = sheet.createRow(rowCount+1);

for(int j = 0; j < row.getLastCellNum(); j++){

Cell cell = newRow.createCell(j);

cell.setCellValue(dataToWrite[j]);

}

inputStream.close();

**//Create an object of FileOutputStream class to create write data in excel file**

FileOutputStream outputStream = new FileOutputStream(file);

guru99Workbook.write(outputStream);

outputStream.close();

}

**//it seems jxl more easy to use thn this**

**Handling Web Tables, Frames, and Dynamic Elements in Selenium**

**html tables:**

1.’table’ tag defines html table.  
2.’tbody’ tag defines container for rows and columns.  
3.’tr’ defines rows in an html table.  
4.’td’/’th’ define column of an html table.

**Dynamic example:**

|  |  |  |
| --- | --- | --- |
| 1 | WebElement htmltable=**driver.findElement(By.xpath**("//\*[@id='main']/table[1]/tbody")); | |
| 2 | List<WebElement> **rows**=**htmltable.findElements(By.tagName**("tr")); |

|  |  |
| --- | --- |
| 3 |  |
| 4 | for(int rnum=0;rnum<rows.size();rnum++) | |

|  |  |
| --- | --- |
| 5 | { |
| 6 | List<WebElement> columns=**rows.get(rnum).findElements(By.tagName("th"));** | |

|  |  |  |
| --- | --- | --- |
| 7 | System.out.println("Number of columns:"+columns.size()); | |
| 8 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | for(int cnum=0;cnum<columns.size();cnum++) | | |
| 10 | | { |

|  |  |  |
| --- | --- | --- |
| 11 | System.out.println(columns.get(cnum).getText()); | |
| 12 | } |

|  |  |
| --- | --- |
| 13 | } |

### **Dynamic elements:**

**Problem Type 1:**If part of the attribute value changes**.**

driver.findElement(By.xpath(“//\*[**contains(@**id,’username’)]”)).sendKeys(“username”);  
driver***.***findElement(By.xpath(“//\*[**starts-with(**@id,’user’)]”)).sendKeys(“username”);

‘contains’ is a java method which checks if id contains the substring username.  
starts-with() checks if any attribute starts with “user”.

**Problem Type 2:**If entire value of the attribute changes dynamically.

Use of sendKeys. (tab key, enter keys, F5 etc)

1.driver.findElement(By.id(“password”)).sendKeys(“password”))

2.driver.findElement(By.id(“password”)).sendKeys(Keys.TAB));

**How to switch back to the Main Frame**

driver.switchTo().parentFrame(); //parent frame

driver.switchTo().defaultContent(); //most parent or main frame

**total number of iframes present inside the page**

int size = driver.findElements(By.tagName("iframe")).size();

**total number of iframes present inside the frame**

int sizeO = driver.findElements(By.tagName("iframe")).size();

driver.switchTo().frame(0);

sizeI = driver.findElements(By.tagName("iframe")).size();

# Flash Testing with Selenium

C Testing is testing type used to check the flash based video, games, movies, etc. are working as expected.

* Main difference between flash and other element is that Flash is embedded in SWF files, while other elements are embedded in HTML files

**Tools-**Below are the testing tools which are useful in flash testing.

1. Selenium
2. Soap UI
3. TestComplete
4. Test Studio etc.

[download flashwebdriver jar files](https://drive.google.com/file/d/0B2SE8u2xJC83Y3VzVTRkRmRPek0/view):

import Flash.FlashObjectWebDriver;

FlashObjectWebDriver flashApp **= new FlashObjectWebDriver(driver, "myFlashMovie");**

driver.get("http://demo.guru99.com/flash-testing.html");

Thread.sleep(5000);

flashApp.**callFlashObject("Play");**

Thread.sleep(5000);

flashApp.callFlashObject("StopPlay");

Thread.sleep(5000);

flashApp.callFlashObject("SetVariable","/:message","Flash testing using selenium Webdriver");

Java: To read lines from a text file, you can use this (uses try-with-resources):

String line;

try (

InputStream fis = new FileInputStream("the\_file\_name");

InputStreamReader isr = new InputStreamReader(fis, Charset.forName("UTF-8"));

BufferedReader br = new BufferedReader(isr);

) {

while ((line = br.readLine()) != null) {

// Do your thing with line

}

}

# Verify Image Presence in Web Page using Selenium WebDriver

@Test

public void CheckImage() throws Exception {

driver.get(baseUrl);

WebElement ImageFile = driver.findElement(By.xpath("//img[contains(@id,'Test Image')]"));

Boolean ImagePresent = (Boolean) ((JavascriptExecutor)driver).executeScript("return arguments[0].complete && typeof arguments[0].naturalWidth != \"undefined\" && arguments[0].naturalWidth > 0", ImageFile);

if (!ImagePresent)

{

System.out.println("Image not displayed.");

}

else

{

System.out.println("Image displayed.");

}

}

**Connect To Database From Selenium**

|  |
| --- |
| **Connection con** = DriverManager.getConnection("jdbc: odbc: makeConnection", "userID", "password"); |

We can **handle multiple windows using Windows Handlers** in selenium webdriver.

**Step 1:** After opening the website, we need to get the main window handle by using driver.getWindowHandle(); (Returns String)  
The window handle will be in a form of lengthy alpha numeric   
**Step 2:** We now need to get all the window handles by using driver.getWindowHandles(); (Returns Set)  
**Step 3:** We will compare all the window handles with the main Window handles and perform the operation the window which we need.

*//To close all the other windows except the main window.*

String openWindowHandle = driver.getWindowHandle();;

public static boolean closeAllOtherWindows(String openWindowHandle)

{

Set<String> allWindowHandles = driver.getWindowHandles();

for (String currentWindowHandle : allWindowHandles)

{

if (!currentWindowHandle.equals(openWindowHandle))

{

driver.switchTo().window(currentWindowHandle);

driver.close();

}

}

driver.switchTo().window(openWindowHandle);

if (driver.getWindowHandles().size() == 1)

return true;

else return false;

}

**Connection object for the following things:**- creates the Statement, PreparedStatement and CallableStatement objects for executing the SQL  
statements.  
-to Commit or roll back a JDBC transaction.  
-to know about the database or data source (using DatabaseMetaData)

-to close the data source. The Connection.isClosed() method returns true only if the  
Connection.close() has been called. This method is used to close all the connection.

# Generate logs in Selenium using TestNG reporter

# Using log4j.jar:

# private static Logger Log = Logger.getLogger(Log.class.getName());//

Log.info("Click action performed on Log out link");

Using reporter:

Reporter.log("Browser Opened");

# [switch tabs using Selenium WebDriver with Java](https://stackoverflow.com/questions/12729265/switch-tabs-using-selenium-webdriver-with-java)

ArrayList<String> tabs2 = new ArrayList<String> (driver.getWindowHandles());

driver.switchTo().window(tabs2.get(1));

driver.close();

driver.switchTo().window(tabs2.get(0));

driver.findElement(By.cssSelector("body")).sendKeys(Keys.CONTROL +"\t"); // not works always

String parentHandle = driver.getWindowHandle(); // get the current window handle

System.out.println(parentHandle); //Prints the parent window handle

String anchorURL = anchor.getAttribute("href"); //Assuming u are clicking on a link which opens a new browser window

anchor.click(); //Clicking on this window

for (String winHandle : driver.getWindowHandles()) { //Gets the new window handle

System.out.println(winHandle);

driver.switchTo().window(winHandle); // switch focus of WebDriver to the next found window handle (that's your newly opened window)

}

//Now your driver works on the current new handle

//Do some work here.....

//Time to go back to parent window

driver.close(); // close newly opened window when done with it

driver.switchTo().window(parentHandle); // switch back to the original window

# How to Download files in Selenium Webdriver

[**http://learn-automation.com/how-to-download-files-using-selenium-webdriver/**](http://learn-automation.com/how-to-download-files-using-selenium-webdriver/)

FirefoxProfile profile=new FirefoxProfile();

// Set preferences for file type

profile.setPreference("browser.helperApps.neverAsk.openFile", "application/octet-stream");

// Open browser with profile

WebDriver driver=new FirefoxDriver(profile);

# How to Capture Tooltip in Selenium Webdriver

[**http://learn-automation.com/how-to-capture-tooltip-in-selenium-webdriver/**](http://learn-automation.com/how-to-capture-tooltip-in-selenium-webdriver/)

// find the tooltip xpath  **(use firepath to get the xpath)**

WebElement username\_tooltip=driver.findElement(By.xpath("html/body/div[2]/div[1]"));

 builder.moveToElement(username\_tooltip).perform();

 String tooltip\_msg=username\_tooltip.getText();

## **Execute Failed test cases using Selenium**

Steps

1-If your test cases are failing then once all test suite completed then you have to refresh your project . Right click on project > Click on refresh or Select project and press f5.

2-Check test-output folder, at last, you will get testng-failed.xml

3- Now simply run testng-failed.xml.

# How to Handle Proxy in Selenium Webdriver

Proxy p=new Proxy();

p.setHttpProxy("localhost:7777");

DesiredCapabilities cap=new DesiredCapabilities();

cap.setCapability(CapabilityType.PROXY, p);

WebDriver driver=new FirefoxDriver(cap);

# Taking ScreenShot ONLY for Failed Tests

# http://www.seleniumeasy.com/testng-tutorials/how-to-take-screenshot-for-only-failed-tests-using-webdriver

**Method 1:** we need to create a **class and then implement**[***TestNG 'ITestListener'***](http://testng.org/javadoc/org/testng/ITestListener.html). We will have a **method called *'onTestFailure'***. We need to add the code to take the screen shot in this method. Instead of just taking the screen shot, we will get the Test method name and take a screen shot with test name and place it is destination folder.

**public class TestListener implements ITestListener** {

WebDriver driver=null;

String filePath = "D:\\SCREENSHOTS";

@Override

public **void onTestFailure(ITestResult result) {**

System.out.println("\*\*\*\*\* Error "+result.getName()+" test has failed \*\*\*\*\*");

String methodName=result.getName().toString().trim();

takeScreenShot(methodName);

}

public void takeScreenShot(String methodName) {

//get the driver

driver=TestBase.getDriver();

File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

//The below method will save the screen shot in d drive with test method name

try {

FileUtils.copyFile(scrFile, new File(filePath+methodName+".png"));

System.out.println("\*\*\*Placed screen shot in "+filePath+" \*\*\*");

} catch (IOException e) {

e.printStackTrace();

}

}

we need to add *TestListener* class in testng.xml file as below:

<**listeners**>

       <**listener** class-name="com.pack.listeners.TestListener"/>

  </**listeners**>

**Method 2: @Aftermethod**

* 1. **Just create a @Aftermethod in testNG as per below -**

# @AfterMethod public void takeScreenShot(ITestResult result) throws IOException { if(result.getStatus()==2){ String testName = result.getMethod().getMethodName(); Utility.screenshot(driver, result.getname()); } }

* 1. **Just create a @Aftermethod in testNG as per below -**

@AfterMethod (alwaysRun=true)

public void tearDown(**ITestResult result**) ) throws IOException {

// Here will compare if test is failing then only it will enter into if condition

if(**ITestResult.FAILURE==result.getStatus())**

{

try

{

TakesScreenshot ts=(TakesScreenshot)driver;

File source=ts.getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(source, new File("./Screenshots/"+result.getName()+".png"));

}

catch (Exception e)

{

System.out.println("Exception while taking screenshot "+e.getMessage());

}

**//** We can create Utility class which can have one method which will capture the screenshot. It will increase the clarity.

# Take Screenshot and place it in a folder with Test Class name

public class TestListener extends TestListenerAdapter {

WebDriver driver;

private static String fileSeperator = System.getProperty("file.separator");

@Override

public void onTestFailure(ITestResult result) {

System.out.println("\*\*\*\*\* Error " + result.getName() + " test has failed \*\*\*\*\*");

driver = TestBase.getDriver();

String testClassName = getTestClassName(result.getInstanceName()).trim();

String testMethodName = result.getName().toString().trim();

String screenShotName = testMethodName + ".png";

if (driver != null) {

String imagePath = ".." + fileSeperator + "Screenshots"

+ fileSeperator + "Results" + fileSeperator + testClassName

+ fileSeperator

+ takeScreenShot(driver, screenShotName, testClassName);

System.out.println("Screenshot can be found : " + imagePath);

}

}

public static String takeScreenShot(WebDriver driver,

String screenShotName, String testName) {

try {

File file = new File("Screenshots" + fileSeperator + "Results");

if (!file.exists()) {

System.out.println("File created " + file);

file.mkdir();

}

File screenshotFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

File targetFile = new File("Screenshots" + fileSeperator + "Results" + fileSeperator + testName, screenShotName);

FileUtils.copyFile(screenshotFile, targetFile);

return screenShotName;

} catch (Exception e) {

System.out.println("An exception occured while taking screenshot " + e.getCause());

return null;

}

}

public String getTestClassName(String testName) {

String[] reqTestClassname = testName.split("\\.");

int i = reqTestClassname.length - 1;

System.out.println("Required Test Name : " + reqTestClassname[i]);

return reqTestClassname[i];

}

}

# Handle Unexpected Alerts In Selenium WebDriver

Some times when we browsing software web application, Display some **unexpected alerts** due to some error or some other reasons. This kind of **alerts**not display every time but they are displaying only some time.

@Test

public void Text() throws InterruptedException {

**//To handle unexpected alert on page load.**

**try{**

**driver.switchTo().alert().dismiss();**

**}catch(Exception e){**

**System.out.println("unexpected alert not present");**

**}**

driver.findElement(By.xpath("//input[@name='fname']")).sendKeys("fname");

}

# [How to click an element in Selenium WebDriver using JavaScript](https://stackoverflow.com/questions/11947832/how-to-click-an-element-in-selenium-webdriver-using-javascript) (**org.openqa.selenium.JavascriptExecutor**)

# In case, when locators do not work you can use JavaScriptExecutor. You can use JavaScriptExecutor to perform an desired operation on a web element.

1. Use the WebDriver methods for locating the elements.

WebElement element = driver.findElement(By.id("gbqfd"));

JavascriptExecutor executor = (JavascriptExecutor)driver;

executor.executeScript**("arguments[0].click();", element**);

2.  Use the WebDriver methods for locating the elements.

JavascriptExecutor jse = (JavascriptExecutor)driver;

jse.executeScript("**document.getElementById('gbqfb').click();**");

//Vertical scroll down by 600 pixels

js.executeScript("window.scrollBy(0,600)");

//Call executeAsyncScript() method to wait for 5 seconds

js.executeAsyncScript("window.setTimeout(arguments[arguments.length - 1], 5000);");

//Perform Click on LOGIN button using JavascriptExecutor

js.executeScript("arguments[0].click();", button);

Js.executeScript("alert('hello world');");

//refresh browser

driver.executeScript("history.go(0)");

String DomainName = js.executeScript("return document.domain;").toString();

String url = js.executeScript("return document.URL;").toString();

String TitleName = js.executeScript("return document.title;").toString();

js.executeScript("window.location = 'http://demo.guru99.com/'");

# [click command in selenium webdriver does not work](https://stackoverflow.com/questions/11676790/click-command-in-selenium-webdriver-does-not-work)

driver.findElement(By.name("submit")).click(); //doesn’t work

**then simulate the click - if**.Click()**isn't working**

driver.findElement(By.name("submit")).sendKeys(Keys.Return);

or

driver.findElement(By.name("submit")).sendKeys(Keys.Enter);

# [How can I work with file uploads during a Webdriver test?](https://sqa.stackexchange.com/questions/12851/how-can-i-work-with-file-uploads-during-a-webdriver-test)

Problem Description: As when I want to upload files. However, this opens a native file picker, which Webdriver doesn't seem to support interacting with.

**Using Webdriver** (when < input type=”file”>)

driver.findElement(By.id("myUploadElement")).sendKeys("<absolutePathToMyFile>");

# Using RemoteWebDriver

driver.setFileDetector(new LocalFileDetector());

WebElement upload = driver.findElement(By.id("myfile"));

upload.sendKeys("/Users/sso/the/local/path/to/darkbulb.jpg");

# [Selenium](https://stackoverflow.com/questions/19155417/difference-between-selenium-server-and-selenium-server-standalone-jars) Webdriver vs Selenium Server

There are some reasons though to **use the Selenium-Server with Selenium-WebDriver**.

1. You are using Selenium-Grid to distribute your tests over multiple machines or virtual machines (VMs).
2. You want to connect to a remote machine that has a particular browser version that is not on your current machine.
3. You are not using the Java bindings (i.e. Python, C#, or Ruby) and would like to use HtmlUnit Driver.

🡪 Selenium RC requires starting the server to kick of the execution of **Selenium Automated RC** test suites. The Selenium server is the intermediator between browser and Selenium RC because Selenium RC won’t make direct call to browser. So we have to start Selenium server prior to start running Selenium RC test cases.  
In case of **Selenium WebDriver**, it does not required to start Selenium Server for executing test scripts. Selenium WebDriver makes the calls between browser & automation script. Selenium WebDriver has native supports for each browser to supports Test Automation; on same machine (**both Selenium WebDriver Automation tests & browsers are on same machine.**)

# How do you get all the values from Drop Down list?

1. **Use select**

**Select dropdown=new Select(d.findElement(By.id("month")));**  
List<WebElement> list=dropdown.getOptions();  
for(WebElement we:list) {  
System.out.println(we.getText());  
}

1. **Without Select**

**WebElement sel =** myD.findElement(By.name("dropdown\_name"));  
List<WebElement> lists = **sel.findElements(By.tagName("option"));**

# Skip Test Case from execution

There are two ways to skip a test case :  
  
1. **By using testNg annotation** , ***enabled=false***  
  
@Test(enabled=false)  
Public void testCase(){  
// code  
}  
  
2. **By using "Skip Exception"**  
  
@Test  
public void testCase1(){  
throw Skip Exception(" skipping the test ");  
}

#### How Will You Fix The Import Errors For The Missing Packages?

It’s very easy to resolve such compile time issues.

* Make sure you’ve added the related library into your projects like the TestNG library or the Selenium Webdriver standalone Jar.
* You don’t need to remember the names of the required packages. Just press the CTRL+Shift+O keyboard shortcut. And it will automatically add the required imports to your code.

**What Is Selenium And What Are Its Popular Versions?**

-tool for web-based UI automation. It exposes a set of APIs to support multiple platforms

-also covers Android platform where Appium is the tool which implements Selenium Webdriver interface for mobile automation

Selenium had three major releases

**Selenium 1.0 Or Selenium RC.** - provided an API set which used a server to exchange commands and responses with the browsers.

**Selenium 2.0 Or Selenium Webdriver** - These new APIs completely replaced the server component and interacted natively with the target browsers

**Selenium 3.0** - The main change it brought up is the W3C specifications of Webdriver APIs for browser automation. It is based on the philosophy that each major browser would provide its own implementation of Webdriver APIs

**Selenium Grid :** run multiple tests in parallel, on multiple machines, in a heterogeneous environment.

**What Is Selenium Server And How Does It Differ From Selenium Hub?**  
-Selenium server is a standalone application for using a single server as a test node. Selenium hub acts as a proxy in front of one or more Selenium node instances. A hub + node(s) is called a Selenium grid. Running Selenium server is similar to creating a Selenium grid from a hub and a single node on the same host

# Log4j with Selenium Tutorial

Log4j is a fast, flexible and reliable logging framework It is a tool used for small to large scale Selenium Automation projects.

# [How Selnium WebDriver Overcomes Same Origin Policy](http://howcanfix.com/31451/how-selnium-webdriver-overcomes-same-origin-policy)

**Same Origin policy** prohibits JavaScript code from accessing elements from a domain that is different from where it was launched. Selenium WebDriver overcome same origin policy. Same origin policy problem is in Selenium RC

Selenium RC, testers needed to install local copies of both Selenium Core (a JavaScript program) and the web server containing the web application being tested so they would belong to the same domain.

**How it is avoided???**   
To avoid **“Same Origin Policy”** proxy injection method is used, in proxy injection mode the **Selenium Server** acts as a client configured HTTP proxy , which sits between the browser and application under test and then masks the **AUT**under a fictional URL

**Close browser**

1. webDriver.Close() - Close the browser window that the driver has focus of
2. webDriver.Quit() - Calls Dispose()
3. webDriver.Dispose() Closes all browser windows and safely ends the session

# Parallel Execution of test methods in TestNG

TestNG provides multiple ways to execute tests in separate threads. In testng.xml, if we set**'parallel'**attribute on the tag to 'tests', testNG will run all the ‘@Test’ methods in tag in the same thread, but each tag will be in a separate thread.

If we want to run methods/classes in separate threads, we need to set 'parallel' attribute on the tag to 'methods' / 'classes'

This helps us to run test methods / classes / tests in parallel. By using parallel execution, we can reduce the 'execution time' as tests are executed simultaneously in different threads.

In testNG we can achieve parallel execution by two ways. One with testng.xml file and we can configure an independent test method to run in multiple threads.

**Selenium webdriver uses following** **locators** to find the elements on web page. Locators can be categorized into two categories:

* Structure-based locators
* Attributes-based locators

**Structure-based locators**: locators that rely on the structure of the page to find elements. 

1. XPath (Relative and absolute)

XPath is the standard navigation tool for XML. XPath is used everywhere where there is XML. It is a very powerful language to express which element to target. If we use it correctly then it can produce very reliable locators.

1. DOM

DOM stands for Document Object Model. DOM works by locating elements that matches the Javascript expression referring to an element in the DOM of the page. DOM is convention for representing objects in HTML documents.

e.g. *dom=document.div[‘abc’].button[0]*

*dom=function foo() {return document.getElementById(“abc”);};*

1. CSS

**CSS**: CSS locator can be used to identify a large number of variations. CSS might be not used so general like XPath but it executes faster than XPath. CSS locator uses CSS selectors to find the elements in the webpage.

**Attributes-based locators**: locators that relies on the attributes of elements to locate them. 

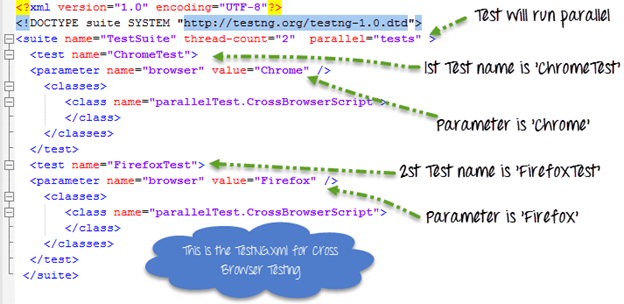
1. Identifier
2. Id
3. Name
4. Link Text
5. Partial Link Text
6. Tag Name
7. Class Name
8. CSS

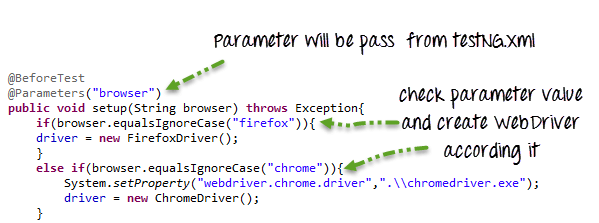
### Cross Browser Testing?

Web-based applications are totally different from Windows applications. A web application can be opened in any browser by the end user.

**Reason Cross Browser Issues**

1. Font size/style mismatch in different browsers.
2. JavaScript implementation can be different.
3. CSS,HTML validation difference can be there.
4. Some browser still not supporting HTML5.
5. Page alignment and div size.
6. Image orientation.
7. Browser incompatibility with OS. Etc.





# Parallel Execution of test methods in TestNG

<http://www.seleniumeasy.com/testng-tutorials/parallel-execution-of-test-methods-in-testng>

In testng.xml, if we set**'parallel'**attribute on the tag to 'tests', testNG will run all the ‘@Test’ methods in tag in the same thread, but each tag will be in a separate thread.

If we want to run methods/classes in separate threads, we need to set 'parallel' attribute on the tag to 'methods' / 'classes'

[<**suite**](http://testng.org/testng-1.0.dtd) name="Parallel test suite" **parallel="methods" thread-count="2">**

<**test** name="Regression 1">

<**classes**>    <**class** name="com.parallel.TestParallelOne"/>     </**classes**>

  </**test**> </**suite**>

To check the thread address-

@Test

public void **testCaseOne**()

{ System.out.println(Thread.currentThread().getId()); }

**HoW to estimate a script :**

[**http://www.articlesontesting.com/2014/01/how-to-estimate-automation-scripting.html**](http://www.articlesontesting.com/2014/01/how-to-estimate-automation-scripting.html)

we all understand is here :  
All test cases will not have same level of technical complexity involved while scripting.  
All test cases will not have same number of steps in it.  
So some sort of normalization will come in picture.

# Handling Pop ups in Selenium

Selenium doesn't give support on automating or handling Browser pop-ups and Native OS pop-ups. So we can make use of Alert API and Robot Class.

Web Applications generate 2 different types of PopUps

(**Alert box/ Pop up box/ confirmation Box/ Prompt/ Authentication Box)**

1. Windows based alert pop ups
2. Web based alert pop ups

### Web based alert pop up (Alert API)

# Alert is a small message box which displays on-screen notification to give the user some kind of information or ask for permission to perform certain kind of operation. It may be also used for warning purpose.

# **1**) Simple Alert****

# This simple alert displays some information or warning on the screen.

**2) Prompt Alert.**

This Prompt Alert asks some input from the user and selenium webdriver can enter the text using sendkeys(" input…. ").

**3) Confirmation Alert.**

This confirmation alert asks permission to do some type of operation.

Alert alrt = drv.switchTo().alert();

alrt.accept();

alrt.dismiss();

alrt.getText();

//prompt alert with single field

alrt.sendKeys("Hello Alert");

//Is used to Authenticate by passing the credentials alrt.authenticateUsing(**new** UserAndPassword("jagseer", "singh"));

//to wait for alert to appear

Alert alrt1 = wdw.until(ExpectedConditions.*alertIsPresent*());

### Window based alert pop up (Robot API or AutoIT)

To control keyboard or mouse **to interact with OS windows** like Download pop-up, Alerts, Print Pop-ups, etc. or native Operation System applications like Notepad, Skype, Calculator, etc. **Selenium Webdriver cannot handle these OS pop-ups/applications.**

In[Java](http://www.guru99.com/java-tutorial.html)version 1.3 **Robot Class** (in AWT Package) was introduced. Robot Class can handle OS popups/applications.

Robot rb = **new** Robot();

//keyboard

rb.keyPress(KeyEvent.***VK\_CAPS\_LOCK***);

rb.keyRelease(KeyEvent.***VK\_CAPS\_LOCK***);

//mouse

rb.mousePress(InputEvent.***BUTTON1\_DOWN\_MASK***);

rb.mousePress(InputEvent.***BUTTON2\_DOWN\_MASK***);

rb.mousePress(InputEvent.***BUTTON3\_DOWN\_MASK***);

rb.mouseRelease(InputEvent.***BUTTON3\_DOWN\_MASK***);

rb.mouseMove(12, 12);

rb.mouseWheel(3);

rb.getPixelColor((**int**)MouseInfo.*getPointerInfo*().getLocation().getX(),(**int**)MouseInfo.*getPointerInfo*().getLocation().getY());

rb.delay(1000);

### Disadvantages of Robot Class

1. Keyword/mouse event will only works on current instance of Window.
2. Most of the methods like mouseMove is screen resolution dependent so there might be a chance that code working on one machine might not work on other.

### Handling Keyboard & Mouse Events

**Handling special keyboard and mouse events** are done using the **Advanced User Interactions API**. It contains the **Actions** and the **Action** classes

**Step 1:**Import the **Actions** and **Action** classes.

**Step 2:**Instantiate a new Actions object.

**Step 3:**Instantiate an Action using the Actions object in step 2.

**Step 4:**Use the perform() method when executing the Action object we designed in Step 3.

**Action Class:**

If we are not using Action to store the actions e.g.

Actions act = **new** Actions(drv);

//no need of using Build() when performing single action only

act.moveToElement(upload).perform();

//use action to store the compiled actions as one

Action acti1 = act.moveToElement(upload).keyDown(Keys.***ALT***).build();

acti1.perform();

//use actions to store the mutilple without building as one

Actions acti2 = act.moveToElement(upload).keyDown(Keys.***ALT***);

acti2.build().perform();

//all functions for mouse

act.click();

act.clickAndHold();

act.contextClick();

act.doubleClick();

act.dragAndDrop(upload, upload);

act.release();

act.moveByOffset(2, 2);

act.moveToElement(upload);

//all functions for keyboard

act.sendKeys(Keys.***CLEAR***);

act.sendKeys(upload, Keys.***ENTER***);

act.keyDown(Keys.***ENTER***);

act.keyUp(Keys.***ENTER***);

act.wait(10);

Scenarios:

1. **Fetch the reference for Web Element Email (textbox).**
2. **Enter the text “hello” into the textbox after converting it into Capital letters.**
3. **Highlight the text in the textbox by clicking on it.**
4. **Lastly, right-click to open the context menu.**

// Handling Multiple Actions

      Actions act = new Actions(driver);

      Action seriesOfActions = act.moveToElement(userName).click().keyDown(userName, Keys.SHIFT)

            .sendKeys(userName, "hello").keyUp(userName, Keys.SHIFT).doubleClick(userName).contextClick().build();

      seriesOfActions.perform();

# Difference between action class and Robot Class

* 1. Action API handles special mouse events on web pages but Robot handles basic mouse events for OS pop ups only.
  2. Action object works on web driver object but Robot object works independently wihout web object.
  3. Action API and Robot both have keyboard basic handling only.
  4. Action API vs Robot API functions

|  |  |  |
| --- | --- | --- |
| **Event** | **Action** | **Robot** |
| Key press | keyUP (Keys.***ENTER***) | keyPress(KeyEvent.***VK\_CAPS\_LOCK***) |
| Key release | keyDown (Keys.***ENTER***) | keyRelease(KeyEvent.VK\_V) |
| Mouse press | click()  clickAndHold() | mousePress(InputEvent.***BUTTON3\_DOWN\_MASK***) |
| Mouse Release | release() | mouseRelease(InputEvent.***BUTTON3\_DOWN\_MASK***) |

# Handle Authentication Popup using Selenium Webdriver

**First approach: (Pass in URL itself)**

drv.get(“http://username:password@the-site.com”);

**Second Approach: (Using Alert)**

WebDriverWait wait = new WebDriverWait(driver, 10);

Alert alert = wait.until(ExpectedConditions.alertIsPresent());

alert.authenticateUsing(new UserAndPassword(username, password));

**Third Approach: (Using AutoIT)**

## **Attach File on a page**

There are two ways to attach file:  
**1.** **Using sendkeys command w**hen input type is “file” and pass the file path parameter..

WebElement fileInput = drv.findElement(By.*name*("uploadfile"));

fileInput.sendKeys("C:/path/to/file.jpg");

# 2. Upload file using Robot

* Click on the File Upload / Choose File button, so that the File Upload dialog is displayed.

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

* Copy your file's absolute path to the clipboard

StringSelection ss = new StringSelection("D:/Test/Test1.docx");

Toolkit.getDefaultToolkit().getSystemClipboard().setContents(ss, null);

* Paste the file's absolute path into the File name field of the File Upload dialog box

//native key strokes for CTRL, V and ENTER keys

Robot robot = new Robot();

robot.keyPress(KeyEvent.VK\_CONTROL);

robot.keyPress(KeyEvent.VK\_V);

robot.keyRelease(KeyEvent.VK\_V);

robot.keyRelease(KeyEvent.VK\_CONTROL);

robot.keyPress(KeyEvent.VK\_ENTER);

robot.keyRelease(KeyEvent.VK\_ENTER);

# 3. Using AutoIT (AutoIt v3 is a freeware BASIC-like scripting language designed for automating Windows GUI and general scripting. It uses a combination of simulated keystrokes, mouse movement and window/control manipulation in order to automate tasks.)

***Use robot class or Sikuli for chrome if autoIT is not working.***

**First Approach:**

# Need tool- Element Identifier and AutoIT editor,

Step 1: Identify the Windows control

Step 2: Build AutoIt script using identified windows control

Step 3: Compile the .au3 script and convert it in to .exe file using compile script

Step 4: Call the .exe file in to the Selenium test case

//Control Script --- fun (" title "," text ",controlID )

ControlFocus("Open",,"Edit1")

ControlSetText("Open",,"Edit1","E:\Resume\resume.doc")

ControlClick("Open",,"Button1")

//In selenium

Runtime.getRuntime().exec("E:\\AutoIT\\FileUpload.exe");

Control Script for Credentials

WinWaitActive("Windows Security")

Send("admin") Send("{TAB}") Send("admin") Send("{ENTER}"

**Second Approach:**

Using auto-it record feature–

* Open the Script Editor window, save the blank file with ‘.au3′ extension and then go to Tools > AU3Recorder or Alt + F6 on AutoIt Script Editor.
* Now type the username and password and do not use the mouse to click on any field, simply use keyboard’s tab button to do the thing.
* Then save the script if not saved and compile it and use in selenium as in above ex.

# [How to avoid Compound Class name error in Page Object?](https://stackoverflow.com/questions/17808521/how-to-avoid-compound-class-name-error-in-page-object)

|  |  |
| --- | --- |
|  | Use a CSS selector instead:­  .country.name  The important thing to note is that this example is wrong! If "country name" is meant as a name of a country, that is. Class names can't have spaces in them. In fact, the class attribute is a space-separated list of classes. That means that if you have a class country name, it's not one class, it's *two* different classes your element belongs to - the first is country, the second is name!  Therefore, fix your classes, if they're wrong. If they're not, use a CSS selector, it's the only reliable way to match multiple classes (apart from a very long and complicated XPath expression). Don't use trivial XPath expressions or CSS selectors with naive attribute comparison (//\*[@class='country name'] or \*[class='country name']), that's just plain wrong. |

# Waits

## **Implicit Wait**

 to wait for a certain amount of time before throwing an **exception** that it cannot find the element on the page.

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

## **Fluent Wait**

Each **FluentWait** instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as **NoSuchElementExceptions** when searching for an element on the page.

Wait wait = new FluentWait(driver)

    .withTimeout(30, SECONDS)

    .pollingEvery(5, SECONDS)

    .ignoring(NoSuchElementException.class);

WebElement foo = wait.until(new Function() {

    public WebElement apply(WebDriver driver) {

        return driver.findElement(By.id("foo"));

    }

});

## **Explicit Wait**

//explicit wait

Thread.*sleep*(100);

WebDriverWait wdw = **new** WebDriverWait(drv, 15); // time in seconds

wdw.until(ExpectedConditions.*presenceOfElementLocated*(By.*xpath*””)));

wait for any condition you might like.

Remember that there is a difference between several scenarios:

* An element not being present at all in the DOM.
* An element being present in the DOM but not visible.
* An element being present in the DOM but not enabled. (i.e. clickable)

public WebElement getWhenVisible(By locator, int timeout) {

WebElement element = null;

WebDriverWait wait = new WebDriverWait(driver, timeout);

element = wait.until(ExpectedConditions.visibilityOfElementLocated(locator));

return element;

}

public void clickWhenReady(By locator, int timeout) {

WebDriverWait wait = new WebDriverWait(driver, timeout);

WebElement element = wait.until(ExpectedConditions.elementToBeClickable(locator));

element.click();

}