Automating window Controls with Selenium:

· Handling Window Authentication Pop Up  
http://Username:Password@SiteURL

· Driver.get();

· Handling File Upload from Windows using AutoIT  
What is AutoIT  
Install AutoIT  
AutoIT Scripting  
Integrating AutoIT with Selenium

//Shift focus to the file upload windows

//set text/path into file name edit box

//click open to upload file

Au3info- record window component objects

Build Script -scite.exe

Save it- .au3 extenstion

Convert file into .exe by compiling .au3 file

Call .exe file with Runtime class in java into your selenium tests

ControlFocus("Open","","Edit1")

ControlSetText("Open","","Edit1","C:\Users\rahul\Documents\check\visit.pdf")

ControlClick("Open","","Button1")

239. Handling Window Authentication pop ups with Selenium

We know that Selenium works only on web-based applications.

So, if there is any window-based controls, like window popups, or if you want to upload any file

into your web application from your system, so file could be in your Documents folder

or a Test app folder, if you want to upload it, then you have to go to specific folder and select it,

All those cannot be done by Selenium, because Selenium automates only web browser.

Selecting a file from your local drive does not belong to any web, so Selenium do not have any knowledge about it.

all this, you cannot do only with Selenium, You need some external software to support in achieving all this window controls.

So, for Window Authentication Pop Up,we need not depend on any other software, Selenium guys came up with one trick to handle that.

Basically, it'll ask credentials to navigate to that URL. That's where we generally get this.

This time, you have to tie up with this Username and Password.

You have to send these values as well along with your URL from Selenium.

When you hit like this in driver.get, in this fashion, so Selenium will capture this Username and Password and keep that in memory of the browser.

So, whenever any popup comes like this, after it hits the URL, automatically, Selenium sent this Username and Password into these fields, and it'll click on Sign in.

Selenium will take care of extracting this Username and Password from this URL,

and it'll put this into these edit fields and that will click on Sign in.

240. Examples on handling pop ups with modified webdriver url

**package** sec30\_windowpop\_uploadfile;

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

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

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

**public** **class** windowPopUp {

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

// **TODO** Auto-generated method stub

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

//http://Username:Password@SiteURL

//driver.get("https://the-internet.herokuapp.com/");

driver.get("http://admin:admin@the-internet.herokuapp.com/");

driver.findElement(By.*linkText*("Basic Auth")).click();

}

}

241. What is AutoIT? Installation details

So after you click on Choose File... So this Choose File can be clicked using Selenium.

Because this is web part, so you can simply inspect and get some locator out of it, where you can make sure this window is opening. After you open this window, you need to actually go to that specific folder, and you need to select this visit PDF. So how do you do that?

And you need to click on Open so that file will be uploaded. All this work should be done through automation. And that's possible with AutoIt, not with Selenium, bcze that's a desktop related thing.

Google – autoit download

Open autoit editor after download in this path - C:\Program Files (x86)\AutoIt3\SciTE

AutoIt can scan any of your desktop items, But here you need to specifically tell AutoIt

to shift focus to this specific window so that you can operate on that.

1. Shift focus to the file upload windows
2. Set text/path into file name editbox
3. Click open to upload file

242. Inspecting the window objects and converting into AutoIT code

 before you start with scripting, you have to open the object recording tool as well.

if you want to actually inspect elements in the browser, you right click and do inspect

so that you will get all the HTML attributes of that specific element, Similarly, there are some window attributes present for each and every component here.

( this is Au3Info\_x64, ) - Basically this is called a spy tool. This will spy your window objects

and display the properties of that specific control.

So first of all, we need to, step one is to shift our focus to that specific edit box.

So there is a simple method in AutoIt called control focus, so basically this will tell us to shift the focus to that specific element.

243. End to End example on uploading File with AutoIT Selenium

Create one new class( FileUpload)

Selenium does not have any knowledge of calling .exe files which are located in your Windows system, but Java does for us.

So in Java there is a class called Runtime and we do the .getRuntime. So basically we are executing something on Runtime. And exec, this is a method in that Runtime static class which will help us to execute.

**package** sec30\_windowpop\_uploadfile;

**import** java.io.IOException;

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

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

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

**public** **class** fileupload {

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

// **TODO** Auto-generated method stub

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

// https://pdf-to-word-converter-online.com/

//C:\Users\Dell\Documents\Custom Office Templates

driver.get("https://pdf-to-word-converter-online.com/");

Thread.*sleep*(3000);

driver.findElement(By.*cssSelector*("label[type='button']")).click();

Thread.*sleep*(3000);

Runtime.*getRuntime*().exec("C:\\Users\\Dell\\Documents\\Custom Office Templates\\john.exe");

}

}

245. Steps to complete the flow to download file from Application with selenium

once we uploaded file using auto ID, our next step is to click on convert now and then download the file. so our selenium test should click on convert now and then it should download. So after it downloaded we have to verify if the file is downloaded in our system.

So here we will use Java file package to actually check if the file is successfully located

in that directory where it got downloaded.

246. Chrome driver options to configure download path of browser

System.getProperty. This will simply give the path of your project

whatever you are getting from this website, into this project, so if you can reroute that download file every time whenever it gets downloaded into this project then it's easy for us to build the dynamic path because we know that download path have the path to its project

now when you want to download that file, you don't want to get it into downloads folder

because if you download there, you cannot get generic path.

If you download inside this project , you will have a generic path.

But how to tell your Chrome browser specifically to stop downloading in that folder

and download only in this My Project folder ?

You can actually inform all these options, Chrome options before you invoke the browser,

so that Selenium remembers all your preferences, and based upon your preferences,

it will invoke Chrome browser, so that it remembers that whatever I download.

I have to download as per the preference mentioned by user in the script, so that knowledge,

we have to provide to our Chrome browser.

So that's how you need to come to approach in dynamically checking the path.

step 1 is to first of all, reroute file-checking to our own project.

Step 2 is to tell Chrome browser to download into our own project.

That's it and we can build a dynamic path.

Now, the only thing left for us is to put value here. For that, Chrome have given a documentation

to include these three lines of code ( Hashmap ) in to the class.

You can find these three lines in their official documentation saying how to set default directory path.

We have successfully redirected our directory to this project. And when you run this test now, the file, whatever you download through the script, will get downloaded into this exercise project.

And we are catching it here, and we are checking if it exists or not. And now, you see that there is no hard coding.

whenever you create something at the end, you should also have a step to delete.

How do you do that?

**package** sec30\_windowpop\_uploadfile;

**import** java.io.File;

**import** java.io.IOException;

**import** java.time.Duration;

**import** java.util.HashMap;

**import** org.testng.Assert;

**import** org.testng.annotations.Test;

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

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

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

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

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

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

**public** **class** filedownloadinproject {

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

// **TODO** Auto-generated method stub

String downloadPath=System.*getProperty*("user.dir");

//System.setProperty("webdriver.chrome.driver","C:\\work\\chromedriver.exe");

HashMap<String, Object> chromePrefs = **new** HashMap<String, Object>();

chromePrefs.put("profile.default\_content\_settings.popups", 0);

chromePrefs.put("download.default\_directory", downloadPath);

ChromeOptions options=**new** ChromeOptions();

options.setExperimentalOption("prefs", chromePrefs);

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

driver.get("https://altoconvertpdftojpg.com/");

//driver.findElement(By.cssSelector("[class\*='btn--choose']")).click();

driver.findElement(By.*id*("browse")).click();

Thread.*sleep*(3000);

Runtime.*getRuntime*().exec("C:\\\\Users\\\\Dell\\\\Documents\\\\Custom Office Templates\\\\john.exe");

WebDriverWait wait= **new** WebDriverWait(driver,Duration.*ofSeconds*(30));

wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*cssSelector*("button[class\*='medium']")));

driver.findElement(By.*cssSelector*("button[class\*='medium']")).click();

wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*linkText*("Download Now")));

driver.findElement(By.*linkText*("Download Now")).click();

Thread.*sleep*(5000);

File f=**new** File(downloadPath+"/converted.zip");

**if**(f.exists())

{

Assert.*assertTrue*(f.exists());

**if**(f.delete())

System.***out***.println("file deleted");

}

}

}