Assisted Practice: 1.5 WebDriver Installation and Integration in Eclipse

This section will guide you to:

- Integrate WebDriver in Eclipse.

Development Environment

- Eclipse IDE for Enterprise Java Developers Version Oxygen.3a Release (4.7.3a)
- Java Development Kit Version 8

This lab has mainly three subsections, namely:

1.1.1 Downloading Selenium Standalone Server jar

1.1.2 Launching eclipse and creating a Java project

1.1.3 Configuring WebDriver with Eclipse

Step 1.1.1: Downloading Selenium Standalone Server jar

- Selenium is already installed in your practice lab. (Refer FSD: Lab Guide - Phase 5)

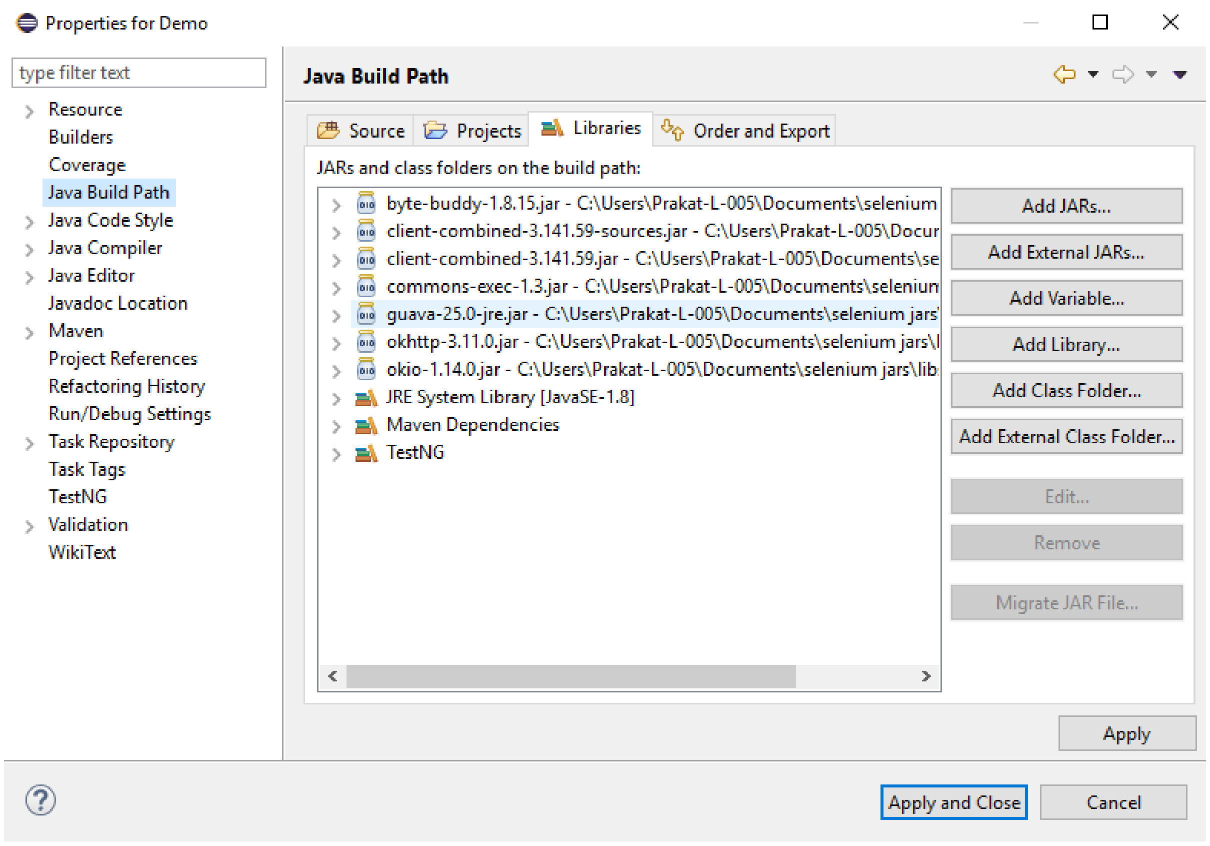
Step 1.1.2: Launching eclipse and creating a Java project

- Open Eclipse and create a Workspace.
- Create a Project.
- Click on File -> New -> Java Project

Step 1.1.3: Configuring WebDriver with Eclipse

- Add selenium standalone server jars
- Right click on Project -> select Properties -> Select Java Build Path
- Navigate to Libraries tab and click on Add External Jars button

- Add selenium standalone server Jar files.
- Click on Apply and Close button.
- In eclipse it looks like:



JARs and class folders on the build path:

```
byte-buddy-1.8.15.jar - C:\Users\Prakat-L-005\Documents\selenium
client-combined-3.141.59-sources.jar - C:\Users\Prakat-L-005\Docur
client-combined-3.141.59.jar - C:\Users\Prakat-L-005\Documents\se
commons-exec-1.3.jar - C:\Users\Prakat-L-005\Documents\seleniun
guava-25.0-jre.jar - C:\Users\Prakat-L-005\Documents\selenium jars
okhttp-3.11.0.jar - C:\Users\Prakat-L-005\Documents\selenium jars\l
okio-1.14.0.jar - C:\Users\Prakat-L-005\Documents\selenium jars\lib
JRE System Library [JavaSE-1.8]
Maven Dependencies
TestNG
```

Assisted Practice: 1.7 Locating Web Page Elements

This section will guide you to:

- How to locate elements in Multiple ways using selenium web driver

This lab has mainly eight subsections, namely :

1.2.1 Using ID as a Locator

1.2.2 Using class name as a Locator

1.2.3 Using name as a Locator

1.2.4 Using Link Text as a Locator

1.2.5 Using Xpath as a Locator

1.2.6 Using CSS Selector as a Locator

1.2.7 Using XPath handling complex and dynamic elements

1.2.8 Pushing the code to GitHub repositories

Step 1.2.1: Using ID as a Locator

- Open Eclipse
- Finding Web element using Locator **ID**
  a. Syntax : id = id of the element
  b. Example : driver.findElement(By.id("Email"));

Step 1.2.2 Using class name as a Locator

- Finding Web element using Locator **ClassName**
  a. Syntax : class = Class Name of the element
  b. Example : driver.findElement(By.class("classname"));

Step 1.2.3 Using Name as a Locator

- Finding Web element using Locator **Name**
    a. Syntax : name = Name of the element
    b. Example : driver.findElement(By.name(“name”));

Step 1.2.4 Using LinkText as a Locator

- Finding Web element using Locator **Link Text**
    a. Syntax : link = partialLink of the element
    b. Example : driver.findElement(By.partialLinkText(“plink”));

Step 1.2.5 Using Xpath as a Locator

- Finding Web element using Locator **Xpath**
- Xpath can be created in two ways
    a. **Relative Xpath**
        - Syntax : relativeXpath : //*[@class=’relativexapath’]
        - Example : driver.findElement(By.xpath(“//*[@class=’relativexapath’]”));

    b. **Absolute Xpath**
        - Syntax : absoluteXpath : html/body/div[1]/div[1]/div/h4[1]/b
        - Example : driver.findElement(By.xpath(“html/body/div[1]/div[1]/div/h4[1]/b”));

Step 1.2.6 Using Xpath as a **CSS Selector**

- CSS Selector have many formats, namely
    a. **Tag and ID**
        - Syntax :”css = tag#id”
        - Example : driver.findElement(By.cssSelector(“input#email”));

    b. **Tag and Class**
        - Syntax : “css = tag.class”
        - Example : driver.findElement(By.cssSelector(”input.inputtext”));

    c. **Tag and Attribute**
        - Syntax : “css = tag[attribute=value]”

- Example : driver.findElement(By.cssSelector(“input[name=lastName]”));

d. **Tag, Class and Attribute**
    - Syntax : “tag.class[attribute=value]”
    - Example : driver.findElement(By.cssSelector(“input.inputtext[tabindex=1]”));

e. **Inner text**
    - Syntax : ”css = tag.contains(“innertext”)”
    - Example : driver.findElement(By.cssSelector(font:contains(“Boston”)));

Step 1.2.7 Using Xpath Handling complex and Dynamic elements

- Dynamic Xpath has many formats, Namely

a. **Contains();**
    - Syntax : “xpath = //*[contains(text(),’text’)]
    - Example : driver.findElement(By.xpath(”//*[contains(text(),’sub’]”));

b. **Using OR & AND**
    - Syntax : xpath=//*[@type=’submit’ or @name=’btnReset’]
    - Example :

driver.findElement (By.xpath(”=//*[@type=’submit’ or @name=’btnReset’]”));

c. **Start-with function**
    - Syntax : xpath= //label[starts-with(@id,’message’)]
    - Example :

driver.findElement (By.xpath(”//label[starts-with(@id,’message’)]”));

d. **Text();**
    - Syntax : xpath=//td[text()=’UserID’]
    - Example : : driver.findElement (By.xpath(”=//td[text()=’UserID’]”));

e. **Following**
    - Syntax : xpath=//*[@type=’text’]//following::input

- Example : driver.findElement(By.xpath(”=//*[@type=’text’]//following::input”));

f. **Preceding**
   - Syntax : xpath=//*[@type=’text’]//preceding::input
   - Example : driver.findElement(By.xpath(”//*[@type=’text’]//preceding::input”));

g. **Following - sibling**
   - Syntax : xpath=//*[@type=’submit’]//preceding::input
   - Example :

   driver.findElement (By.xpath (”//*[@type=’text’]//following-sibling::input”));

Step 1.2.8: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m “Changes have been committed.”

- Push the files to the folder you initially created using the following command:

git push -u origin master

Assisted Practice: 1.9 Locating Elements through CSS and XPath

This section will guide you to:

- How to locate elements in the Web page.

This lab has mainly three subsections, namely:

1.3.1 To find the element present on the page by using CSS Selector.

1.3.2 To find the element present on the page by using XPath.

1.3.3 Pushing the code to your GitHub repositories

Step 1.3.1: To find the element present on the page using CSS Selector.

- Using CSS Selectors in Selenium. As we all know, CSS stands for Cascading Style Sheets. By using CSS selectors, we can find or select HTML elements on the basis of their id, class or other attributes. CSS is faster and simpler than XPath particularly in case of IE Browser where Path works very slowly.
- Open Eclipse
- Use Path as a CSS Selector
- CSS Selector have many formats, namely
  a. **Tag and ID**
     - Syntax: “css = tag#id”
     - Example: driver.findElement(By.cssSelector(“input#email”));

  b. **Tag and Class**
     - Syntax: “css = tag.class”
     - Example: driver.findElement(By.cssSelector(”input.inputtext”));

  C. **Tag and Attribute**
     - Syntax: “css = tag[attribute=value]”

- Example:
  driver.findElement(By.cssSelector(“input[name=lastName]”));

c. **Tag, Class and Attribute**
   - Syntax: “tag.class[attribute=value]”
   - Example:
     driver.
   findElement(By.cssSelector(“input.inputtext[tabindex=1]”));

d. **Inner text**
   - Syntax: “css = tag.contains(“innertext”)”
   - Example:
     driver.findElement(By.cssSelector(font:contains(“Boston”)));

Step 1.3.2: To find the element present on the page using Path.

- In Selenium automation, if the elements are not found by the general locators like id, class, name, etc. then XPath is used to find an element on the web page.
- XPath contains the path of the element situated at the web page. Standard syntax for creating XPath is:
  XPath=//tagname[@attribute='value']

  - **//:** Select current node.
  - **Tagname:** Tagname of the particular node.
  - **@:** Select attribute.
  - **Attribute:** Attribute name of the node.
  - **Value:** Value of the attribute.
- Types of XPath:
  There are two types of XPath:
  a. **Absolute XPath**
     - It is direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.
     - The key characteristic of XPath is that it begins with the single forward slash (/), which means you can select the element from the root node.

- Syntax for absolute Path: html/body/div[1]/div[1]/div/h4[1]/b
- Example: driver.findElement(By.xpath(“html/body/div[1]/div[1]/div/h4[1]/b”));
- Writing absolute XPath on the elements which are present in the webpage will be very lengthy. To reduce the length, we use relative XPath.

b. Relative XPath

- For relative XPath the path starts from the middle of the HTML DOM structure. It starts with the double forward slash (//), which means it can search the element anywhere at the webpage.
- You can start from the middle of the HTML DOM structure and no need to write long XPath.
- Syntax for relativeXPath: //*[@class=’relativexapath’]
- Example: driver.findElement(By.xpath(“//*[@class=’relativexapath’]”))

Step 1.3.3: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m “Changes have been committed.”

- Push the files to the folder you initially created using the following command:

  ```
  git push -u origin master
  ```

Assisted Practice: 1.11 Handling Various Web Elements

This section will guide you to:

- Handling Various Web Elements present on the page.

This lab has divided into different types, namely:

1.4.1 Edit box

1.4.2 Link

1.4.3 Button

1.4.4 Image, image link, an image button

1.4.5 Text area

1.4.6 Checkbox

1.4.7 Radio button

1.4.8 Dropdown list

1.4.9 Web table /HTML table

1.4.10 Frame

1.4.11 Switching between tabs in same browser window

1.4.12 Pushing the code to GitHub repositories

Step 1.4.1: Edit box

- Open Eclipse

- It is a basic text control that enables a user to type a small amount of text.
- Operations on Edit box
    - Enter a Value,
    - Clear the Value,
    - Check enabled status,
    - Check edit box existence,
    - Get the value

Step 1.4.2: Link

- link is more appropriately referred to as a hyperlink and connects one web page to another. It allows the user to click their way from page to page.
- Operations on Link
    - Click Link,
    - Check the link existence,
    - Check the link enabled status,
    - Return the Link Name

Step 1.4.3: Button

- This represents a clickable button, which can be used in forms and places in the document that needs a simple, standard button functionality.
- Operations on Button
    - Click
    - Check Enabled status

    - Display status

Step 1.4.4: Image, image link, an image button

- It helps in performing actions on images like clicking on the image link or the image button, etc.
- Operations Image
    - Three types of Image elements in Web Environment
    - General Image (No functionality)
    - Image Button (Submits)
    - Image Link (Redirects to another page/location)

Step 1.4.5: Text area

- It is an inline element used to designate a plain-text editing control containing multiple lines.
- Return / Capture Text Area or Error message from a web page

Step 1.4.6: Checkbox

- This is a selection box or a tick box which is a small interactive box that can be toggled by the user to indicate an affirmative or a negative choice.
- Operations on Check box
    - Check if the check box is displayed or not?
    - Check if the check box is enabled or not?
    - Check if the check box is Selected or not?
    - Select the Check box

- Unselect the Check box

Step 1.4.7: Radio button

- It is an option button which is a graphical control element that allows the user to choose only one predefined set of mutually exclusive options.
- Operations on Radio Button
    - Select Radio Button
    - Verify if the Radio Button is Displayed or not?
    - Verify if the Radio Button is enabled or not?
    - Verify if the Radio Button is Selected or not?
- Example:

```
oRadioButton.get(1).click();
```

Step 1.4.8: Dropdown list

- It is a graphical control element, similar to the list box, which allows the user to choose one value from the list. When this drop-down list is inactive, it displays only a single value.
- Operations on Drop down list
    - Check the Dropdown box existence
    - Check if the Drop down is enabled or not?
    - Select an item
    - Items Count
- Example:

```
Select fruits = new Select(driver.findElement(By.id("fruits")));
```

fruits.selectByVisibleText("Banana");

fruits.selectByIndex(1);

Step 1.4.9: Web table /HTML table

- Operations on Web table /HTML Table
    - Get cell value
    - Rows Count
    - Cells Count

Step 1.4.10: Frame

- Operations on Frame
    - Switch from Top window to a frame
    - Switch from a frame to Top window
- Example
    - driver.switchTo().frame("iframe1");
    - driver.switchTo().frame("id of the element");

Step 1.4.11: Switching between tabs in same browser window

- Operations on Switching between tabs in same browser window
- Open a new tab using Ctrl + t
- Driver control automatically switches to the newly opened tab
- Perform the required operations here.
- Next switch back to the old tab using Ctrl + Tab. You need to keep pressing this unless you reach the desired tab.

- Once the desired tab is reached, then perform the operations in that tab.
- Example:

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

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

Step 1.4.12: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m "Changes have been committed."

- Push the files to the folder you initially created using the following command:

  git push -u origin master

Assisted Practice: 1.13 Working with External Elements

This section will guide you to:

- How to handle External elements using Selenium.

Development Environment

- Eclipse IDE for Enterprise Java Developers Version Oxygen.3a Release (4.7.3a)
- Java Development Kit Version 8
- Selenium standalone server Version 3.141.59

This lab has mainly three subsections, namely:

1.5.1 Handling External pop ups.

1.5.2 Handling new Tabs and new Windows.

1.5.3 Pushing the code to your GitHub Repository.

Step 1.5.1: Handling External pop ups.

- WebDriver does ability to interact with multiple windows,which includes alerts using the method switchTo. This method allows to switch the control to pop-up while keeping the browser open in the back ground.
- Open Eclipse
- Syntax for handling the various pop ups
- To click on 'OK' button in pop up

  Syntax: WebDrive driver = new chromeDriver();

driver.switchTo().alert().accept();

- To click on 'Cancel' button in pop up

  Syntax: WebDrive driver = new chromeDriver();

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

- To Capure the alert message

  Syntax: WebDrive driver = new chromeDriver()

  driver.switchTo().alert().getText();

- To enter the information

  Syntax: WebDrive driver = new chromeDriver()

  driver.switchTo().alert().sendKeys("text");

- To exit from the popup

  Syntax: WebDrive driver = new chromeDriver();

  driver.switchTo().alert().close();

Step 1.5.2: Handling new Tabs and new Window.

- Opening new tab

  Syntax: WebDrive driver = new chromeDriver();
  driver.findElement(By.id("xyz")).sendKeys(Keys.CONTROL + "t");

- Opening new Window

  Syntax: WebDriver driver = new chromeDriver();

  driver.findElements(By.id("xyz").sendKeys(Keys.CONTROL + "w");

Step 1.5.3: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m “Changes have been committed.”

- Push the files to the folder you initially created using the following command:

  git push -u origin master

1.15 Screenshots and Browser profiles

This section will guide you to:

- How to take screenshots using selenium web driver and how to set the browser profile.

Development Environment

- Eclipse IDE for Enterprise Java Developers Version Oxygen.3a Release (4.7.3a)
- Java Development Kit Version 8
- Selenium standalone server Version 3.141.59

This guide has mainly four subsections, namely:

1.6.1 Screenshots

1.6.2 Running the code

1.6.3 Pushing the code to your GitHub repositories

1.6.4 Browser profiles

1.6.5 Pushing the code to your GitHub repositories

Step 1.6.1: Screenshots

- Open Eclipse
- Convert web driver object to TakeScreenshot
- Call getScreenshotAs method to create image file
- Copy file to desire location

Step 1.6.1.1 Convert web driver object to TakeScreenshot

- Syntax : TakesScreenshot scrShot = (TakesScreenshot)driver;

Step 1.6.1.2 Call getScreenshotAs method to create image file

- Syntax: File srcFile = scrShot.getScreenshotAs(OutType.FILE);

Step 1.6.1.3 Copy file to desire location

- Syntax: FileUtils.copyFile(source, filePath);
- Open Eclipse and write the code giveln below

```
package screenshots.screenshot;
import java.io.File;
import java.io.IOException;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import com.sun.jna.platform.FileUtils;

public class Screenshots {
        public static void main(String[] args ) throws IOException
            {
                System.setProperty("webdriver.chrome.driver",
"C:\\Users\\Testing-L-064\\chromedriver_win32\\chromedriver.exe");
               WebDriver driver = new ChromeDriver();
               driver.get("https://www.flipkart.com/");
               WebElement upload =
driver.findElement(By.xpath("//*[@type='text']"));
               upload.click();
               TakesScreenshot ts = (TakesScreenshot)driver;
               File scr = ts.getScreenshotAs(OutputType.FILE);
               FileUtils.copyFile(scr, new File("/Screenshot/test.png"));


            }
}
```

Step 1.6.2: Running the code

- Run the code through eclipse.

Step 1.6.3: Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files

cc

Initialize your repository using the following command:

g

Add all the files to your git repository using the following command:

gi

Commit the changes using the following command:

gi[redacted]

Push the files to the folder you initially created using the following command:

gi[redacted]

Step 1.6.4: Browser profiles**:**

- First of all, close the Firefox if it is opened.
- Open Run (Windows+R) and type firefox.exe -p and click OK.
- A dialogue box will open with named ‘Firefox -choose user profile’.
- Select option ‘Create Profile’ from the window, and a Wizard will open, click on Next.
- Provide your profile name which you want to create and click on Finish button.

Step 1.6.5: Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master

1.17 Handling File Uploads

This section will guide you to:

- How to upload the Desktop file in selenium WeDriver

Development Environment

- Eclipse IDE for Enterprise Java Developers Version Oxygen.3a Release (4.7.3a)
- Java Development Kit Version 8
- Selenium standalone server
- Autoit

This guide has mainly Two subsections, namely:

1.7.1 Handling File Upload by **SendKeys**

1.7.2 Handling File Upload by **AutoIT script**

1.7.3 Pushing the code to GitHub repositories

Step 1.7.1: Handling File upload by SendKeys

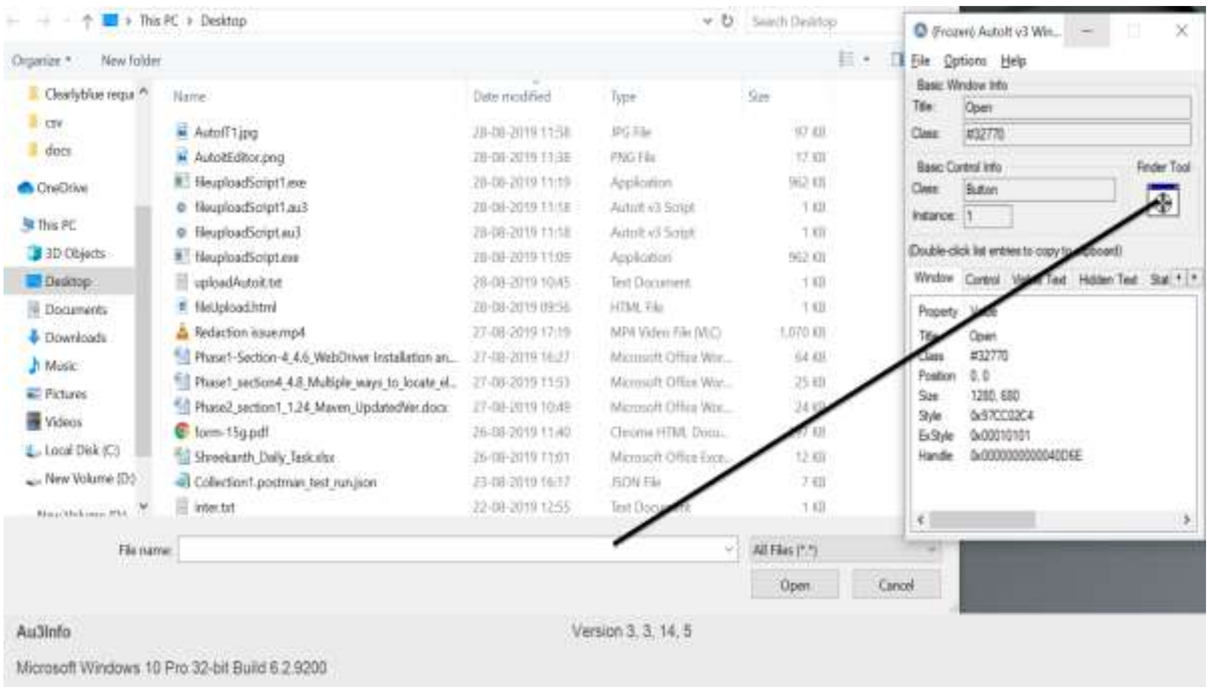
- Open Eclipse
- Create a project : Click on file->New->Java project
- Enter project name as UploadFile
- Click on Finish
- In the project explorer, Expand **UploadFile**
- Right click on **src** and choose **New->Class**
- In Package Name, enter **com.ecommerce** and in **Name** enter **Upload** and click on **Finish**
- Locate the browse button using chropath/firebug.
- Set the path using SendKeys. And the code looks like below

```
//Locating 'browse' button
 WebElement browse =driver.findElement(By.id("uploadfile"));
 //pass the path of the file to be uploaded using Sendkeys method
 browse.sendKeys("D:\\SoftwareTestingMaterial\\UploadFile.txt");
```

[redacted]

Step 1.7.2: Handling File Upload by AutoIT script

- To open it go to **Start->Autoit v3->Autoit window info**
- Now drag the Finder tool box to the object in which you are interested



Name	Date modified	Type	Size
AutoIT1.jpg	28-08-2019 11:58	JPG File	97 KB
AutoitEditor.png	28-08-2019 11:38	PNG File	17 KB
fileuploadScript1.exe	28-08-2019 11:19	Application	962 KB
fileuploadScipt1.au3	28-08-2019 11:18	AutoIt v3 Script	1 KB
FileuploadScript.au3	28-08-2019 11:18	AutoIt v3 Script	1 KB
fileuploadScript.exe	28-08-2019 11:09	Application	962 KB
uploadAutoit.txt	28-08-2019 10:45	Text Document	1 KB
fileUpload.html	28-08-2019 09:56	HTML File	1 KB
Redaction issue.mp4	27-08-2019 17:19	MP4 Video File (VLC)	1,070 KB
Phase1-Section-4_4.6_WebDriver Installation an...	27-08-2019 16:27	Microsoft Office Wor...	64 KB
Phase1_section4_4.8_Multiple_ways_to_locate_el...	27-08-2019 11:53	Microsoft Office Wor...	25 KB
Phase2_section1_1.24_Maven_UpdatedVer.docx	27-08-2019 10:48	Microsoft Office Wor...	24 KB
form-15g.pdf	26-08-2019 11:40	Chrome HTML Docu...	97 KB
Shreekanth_Daily_Task.xlsx	26-08-2019 11:01	Microsoft Office Exce...	12 KB
Collection1.postman_test_run.json	23-08-2019 16:17	JSON File	7 KB
inter.txt	22-08-2019 12:55	Text Document	1 KB

Basic Window Info
Title: Open
Class: #32770

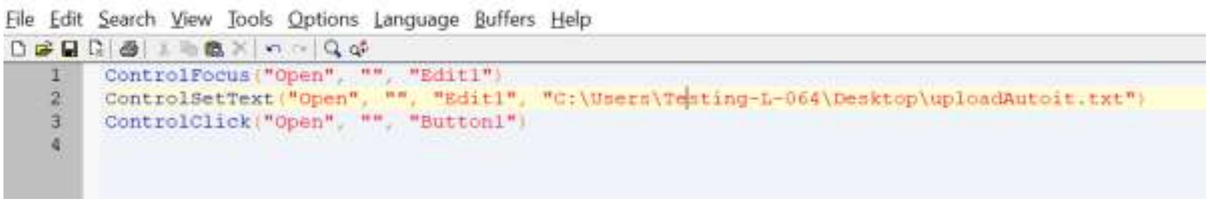
Basic Control Info
Class: Button
Instance: 1

(Double-click list entries to copy to clipboard)

Property	Value
Title	Open
Class	#32770
Position	0, 0
Size	1280, 680
Style	0x97CC02C4
ExStyle	0x00010101
Handle	0x000000000040D6E

Au3Info   Version 3, 3, 14, 5
Microsoft Windows 10 Pro 32-bit Build 6.2.9200

- Build an AutoIT script using **SciTE editor** and write the script using **ControlFocus, ControlsetText, and ControlClick** commands.
- And the script looks like below



```
ControlFocus("Open", "", "Edit1")
ControlSetText("Open", "", "Edit1", "C:\Users\Testing-L-064\Desktop\uploadAutoit.txt")
ControlClick("Open", "", "Button1")
```

- Save the Script with **.au3** extension.
- Compile the **.au3** script which converts it into **.exe** file.
- Pass the **.exe** path into selenium test script using method **Runtime.getRuntime().exec("C:\AutoIt\Autoitscript.exe")**
- Complete script looks like this

```
import java.io.IOException;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class AutoIt {

 private static WebDriver driver = null;

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

    driver = new FirefoxDriver();

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

    driver.get("http://toolsqa.com/automation-practice-form");

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

    Runtime.getRuntime().exec("D:\AutoIt\AutoItTest.exe");

    Thread.sleep(5000);

    driver.close();
```

Step 1.7.3: Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master