**What is the meaning of ‘StaleElementReference’ exception? When do we get this exception?**

**🡪** ‘StaleElementReference’ exception means address (reference) of the element is old (stale). We get this exception

* After finding element
* If page is refreshed
* Then if we try to perform the action on that element

**What are the difference between ‘findElement()’ & ‘findElements()’?**

|  |  |
| --- | --- |
| **findElement()** | **findElements()** |
| 1. Return type 🡪 **WebElement** 2. Returns **first matching** element 3. If locator is not matching then it throws ‘**NoSuchElement**’ exception. | 1. Return type 🡪 **List<WebElement>** 2. Returns **all matching** elements. 3. If locator is not matching it returns **empty List**.It won’t throw any exception |

**Handling Different types of Pop-Ups:**

**Q) List the different types of Pop-ups in Web-Application?**

1. Java Script/Alert/Confirmation Pop-up.
2. Hidden Division Pop-up.
3. File Upload Pop-up.
4. File Download Pop-up.
5. Child Browser Pop-up.
6. Windows Pop-up

Note: In order to identify the type of the pop up properly, always use the browser opened by the WebDriver

**Java Script Pop-up/Alert pop-up/Confirmation pop-up**

**Characteristics:**

* We **cannot** **inspect** this pop-up.
* We **can** **move** this pop-up.
* It will not have any Minimize, Maximize & Close buttons.
* Once pop-up appears background will be blocked.
* It will have ‘OK’ button or ‘OK’ & ‘Cancel’ or ‘Yes’ & ‘No’ buttons.

**Solution:**

* To handle ‘Java Script’ pop-up (Alert pop-up) we use “**switchTo().alert()**” statement which will transfer the control to the Java Script pop-up.
* After transferring the control, we use ‘**getText()**’ method to get the message present on the pop-up.
* We use ‘**accept()**’ method to click on ‘**OK’** button & ‘**dismiss()**’ method to click on ‘**Cancel’** button which are present on the pop-up.
* While using any of these above methods if the pop-up is not present we get “**selenium.NoAlertPresentException**”

**Hidden Division Pop-up:**

**Characteristics:**

* We **can** **inspect** this pop-up.
* We **cannot** **move** this pop-up.
* Pop-up will be in different colors.
* It may or may not have buttons.

**Solution:**

* We can handle this pop-up using ‘**findElement()**’ method itself. (Example for Hidden Division pop-up is ‘Calendar’)

|  |
| --- |
| driver.get("http://vrlbus.in/");  *//Click to open Calendar*  driver.**findElement*(By.id("TRIPSTARTDATE"))*.click();**  *//Select date*  driver.**findElement*(By.linkText("17")).*click();**  driver.close(); |

Code to select different Dates:

|  |
| --- |
| ***LocalDate*** d1=**LocalDate.now();** 🡪 To select Today’s date  ***LocalDate*** d2=**LocalDate.now().plusDays(5);** 🡪 To add days to current date  ***LocalDate*** d3=**LocalDate.now().plusMonths(2);** 🡪 To add months to current date  ***LocalDate*** d4=**LocalDate.now().plusYears(3);** 🡪 To add years to current date |

**File Upload Pop-Up:**

**Characteristics:**

* We **cannot** **inspect** this pop-up.
* We **can move** this pop-up.
* Title of the pop-up will be ‘**File Upload**’ & it will have ‘**Open’** & ‘**Cancel’** buttons.
* We can’t inspect “Browse” button
* If we see the source code of the browse button, it should contain the following code.
* <input type=”file”>

**Note:-** This pop-up is used to browse & select the file which needs to be uploaded. When we click on ‘Open’ button it will send complete path of the file to the browser (Web application). Some of the application will upload the file automatically & some of the application will upload only after clicking on some other button.

**Solution:**

* To handle this pop-up we specify complete path of the file for ‘**sendKeys()**’ method for ‘Upload or Browse’ button.

|  |
| --- |
| driver.get("https://www.naukri.com/");  ***String*** xp="//input[@value='Post your CV']";  *//Find browse button & send file path*  driver.**findElement*(By.xpath(xp))*.sendKeys*("D:\\BesantTechnology\\AutomationTesting\\PopUps.docx"*);**  driver.close(); |

**Note:** While **specifying path of the file we can use ‘\\’ (double backward slashes) or ‘/’ (single forward slash). Path is not case sensitive but it will not** allow Relative path(i.e. “//”).

* In order to upload the file using its **relative path** we use ‘**getAbsolutePath()**’ method of ‘**File’** class.

|  |
| --- |
| ***File*** f=**new File*("./Photos/cv.docx");***  ***String*** path = f.**getAbsolutePath();**  WebDriver driver=new ChromeDriver();  driver.get("https://www.naukri.com/");  String xp="//input[@value='Post your CV']";  driver.**findElement*(By.xpath(xp))*.sendKeys(path);**  driver.quit(); |

**File Download Pop-Up:**

**Characteristics:**

* We **cannot inspect** this pop-up.
* We **can move** this pop-up.
* It will have 2-radio buttons. (‘Open with’ & ‘Save File’)
* It will have “Ok” and “Cancel” buttons
* It will have a check box “Do this automatically”

**Solution:**

* To handle this pop-up we use ‘**Robot’** class.

Code for ‘Firefox’ browser:

|  |
| --- |
| driver.get("https://www.seleniumhq.org/download/");  ***String*** xp="//td[.='Java']/../td[4]/a";  driver.**findElement*(By.xpath(xp)).click();***  *//Press Alt+S to select ‘Save’ radio button & Press ‘Enter’*  ***Robot*** r=**new Robot();**  r.**keyPress*(KeyEvent.VK\_ALT);***  r.**keyPress*(KeyEvent.VK\_S);***  r.**keyRelease*(KeyEvent.VK\_ALT);***  r.**keyPress*(KeyEvent.VK\_ENTER);***  driver.close(); |

* In ‘**Chrome’** browser when we click on ‘Download’ button link it will automatically downloads the file without displaying ‘File Download’ pop-up. In such cases we **don’t use** ‘**Robot’** class.

|  |
| --- |
| WebDriver driver=new ChromeDriver();  driver.get("https://www.seleniumhq.org/download/");  ***String*** xp="//td[.='Java']/../td[4]/a";  driver.**findElement*(By.xpath(xp))*.click();**  driver.close(); |

**Child Browser Pop-Up:**

**Characteristics**:

* We **can inspect** this pop-up.
* We **can move** this pop-up.
* It will have Minimize & Maximize buttons.

**Solution**:

* To handle this pop-up we use ‘**driver.switchTo().window(*String*)**’. In this statement ‘*String’* stands for window handle of the browser in the String format which is a unique alpha numeric string generated by the OS for every browser window.
* In order to get this string, we use **getWindowHandle()** method

**Q) What is the difference between ‘getWindowHandle()’ & ‘getWindowHandles()?**

**🡪** ‘**getWindowHandle()**’ returns window handle of current browser (**String**) whereas ‘**getWindowHandles()**’ returns window handle of all the browsers (**Set <String>**). (Parent & all the child browsers).

**Q) What is the difference between ‘close()’ & ‘quit()’?**

🡪 ‘**close()**’ method closes the **current browser** whereas ‘**quit()**’ method closes the **all the bowsers**.

**Q) Write script to close all the browsers without using ‘quit()’ method?**

|  |
| --- |
| driver.get("https://www.naukri.com/");  ***Set<String>*** allWindows = driver**.getWindowHandles();**  **for(*String* wh:allWindows)**  **{**  driver.**switchTo().window(*wh*).close();**  **}** |

**Q) Write script to print title of all the browsers?**

|  |
| --- |
| driver.get("https://www.naukri.com/");  ***Set<String>*** allBrowsers = driver.**getWindowHandles();**  **for(*String* wh:allBrowsers)**  **{**  ***WebDriver*** browser = driver.**switchTo().window(*wh*);**  String title=browser.**getTitle();**  System.out.println(title);  browser.close();  **}** |

**Windows Pop-Up:**

If the pop-up displayed on the application does not belong to any of the above categories then it is called as Window Pop-up.

**Characteristics**:

* We **cannot inspect** this pop-up.
* Some of the window pop-ups can be moved & some are not.

**Solution**:

* We can handle this pop-up using ‘**Robot’** class & ‘**AutoIT’**.

**Q) Write code to open ‘Print’ pop-up & click on ‘Cancel’ button?**

|  |
| --- |
| driver.get("http://localhost:8080/login.do");  ***Robot* r=new Robot();**  *//Press 'Ctrl+P' to open Print pop-up*  r.**keyPress*(KeyEvent.VK\_CONTROL);***  r.**keyPress*(KeyEvent.VK\_P);***  r.**keyRelease*(KeyEvent.VK\_CONTROL);***  *//Click on 'Cancel' button*  r.**keyPress*(KeyEvent.VK\_ENTER)***;  driver.close(); |

**AutoIT:**

AutoIT is an automation tool which is used to automate the window based applications.

* AutoIT uses the coding language ‘**SciTE Script’** which is similar to VB-Script.

**Download Link**: - <https://www.autoitscript.com/site/autoit/downloads>

Copy the downloaded file in the required location and install it.

**Steps to inspect controls using AutoIT:**

**Note:** In AutoIT, elements are called as controls

1. Open ‘**AutoIT Window Info**’ tool.
2. Drag & drop the ‘**Finder Tool**’ on the window pop-up which needs to be handled. It will display properties value of the control such as Class, Name, ID & Text…etc..

Steps to write AutoIT script & execute:

1. Open ‘**SciTE Script Editor’** tool.
2. Write AutoIT code.
3. Save As the file (File extension will be **.au3** , Ex:- **test.au3**)
4. Go to ‘**Tools**’, select ‘**Compile** (Ctrl+F7). It will generate ‘**.exe**’ file (Ex:- **test.exe**) in the same location where .au3 file was saved.
5. Double click on ‘.exe’ file to execute or press ‘F5’ in AutoIT.

Sample code for AutoIT:

Code to enter value into notepad & close notepad.

|  |
| --- |
| *;open notepad and run this code*  WinWaitActive("Untitled - Notepad")  Sleep(1000)  Send("This is my AutoIT code")  Sleep(1000)  WinClose("Untitled - Notepad")  Send("{ENTER}")  Sleep(1000)  Send("MyNotepad.txt")  Sleep(1000)  Send("{ENTER}") |

**How do you execute ‘AutoIT’ script from Selenium? Write code for this?**

**🡪** Using ‘**Runtime’** class. i.e ‘**Runtime.getRuntime().exec(**"*path of .exe file*"**);**

|  |
| --- |
| public class AutoIT\_Example  {  public static void main(String[] args) throws IOException  {  *//Open Notepad*  **Runtime.getRuntime().exec*("notepad.exe"*);**  *//Run AutoIT script*  **Runtime.getRuntime().exec("E://Folder1//test1.exe");**  }  } |

**Handling Frames:**

Frames are used to create embedded webpages (Web page inside another webpage).

If any element is present inside the frame then before performing the operation on that element, we should transfer the control into that frame which can be done using any one of the following statements.

1.driver.switchTo().frame(int)🡪 Based on index

2.driver.switchTo().frame(string)🡪 Name of the frame

3.driver.switchTo().frame(WebElement)🡪 Frame Element

In-order to transfer the control back to the main page from the frame, we should use the following statement.

Driver.switchTo().defaultContent();

Note: If the element is present inside the frame then if you right click on that element,

1.Firefox browser will display “This Frame” in the context menu

2.Chrome browser will display “view frame source” in the context menu

**Data Driven Testing:**

Testing the application with multiple set of data is called as data driven testing. We always take data from external source such as text file, csv, xml, database or excel files.

1. **POI Dependencies:**

|  |
| --- |
| <URL:-> <http://mvnrepository.com/artifact/org.apache.poi>  **Group ID:-** org.apache.poi  **Artifact ID:-** poi-ooxml  **Version:-** 3.16 |

|  |
| --- |
| **Group ID:-** org.apache.poi  **Artifact ID:-** poi  Version:- 3.16 |

|  |
| --- |
| **Group ID:-** org.apache.commons  **Artifact ID:-** commons-io  **Version:-** 1.3.2 |

Excel files are most frequently used. In-order to handle excel file we need to use the jar files provided by apache with the name POI(poor obfuscation implementation)

POI jar files can be downloaded from [**http://poi.apache.org/download.html**](http://poi.apache.org/download.html)

1.Download <https://www.apache.org/dyn/closer.lua/poi/release/bin/poi-bin-4.0.0-20180907.zip> --- for windows available under “Binary Distribution”.

2.Download commons-compress-1.18-bin.zip from <http://mirrors.fibergrid.in/apache//commons/compress/binaries/commons-compress-1.18-bin.zip>

After downloading, place the zip files in the required location and unzip it.

Right click on the **java project** , Click on **Properties**, Click on **Java Build Path**, Click on **Libraries** tab, Click on **Add External Jars** and choose all the jar files to associate with the project.

**How to ,**

1.Read Data From Excel File

2.Write Data Into Excel File

**Note:**

1.path of the excel and name of the sheet is not case sensitive

2.Row index and cell index starts from 0

3.We get the following **checked exceptions** while handling excel files

1.IOException from FileInputStream

2. EncryptedDocumentException by create() method of WorkbookFactory

4. We get “NullPointerException” (**unchecked exception**) if name of the sheet, index of the row or colomn is invalid.

5.If any cell or row is blank in b/w, we get NullPointerexception so we should use nested try catch block to handle(one try catch for row and one for cell)

6. There is no method to check whether the row or cell is present or not

7.If excel has String value then use **getStringCellValue()** . If it has numeric value, use **getNumericCellValue()**

8. **How do you check whether the cell has string/numeric value?**

Ans: Using “getCellType()” method. If it has string, it returns “1” . If it has numeric value, it returns “0”.