**UNIT – IV**

**Working On Popups**

**Syllabus:** Alerts, Prompts, Confirmation, Working on Frames and Windows, Introduction to Test NG Designs, Annotations in TestNG. Apache POI Jar Files 3.17 for Reading, Writing Excel Files. Page Object Model-Property List.

#### Popups:

* There are 3 type in javascript pop-up. They are,
  + Alert
  + Confirmation
  + Prompt

**Characteristics of javascript pop-up**

* We can not move the pop up.
* We can not inspect the pop up.
* It is black and white in color
* If it contains only ok button then it is alert pop-up.
* If it contains only ok and cancel button then it is confirmation pop-up.
* If it contains only Text box, ok and cancel button then it is prompt pop-up.
* we can handle any javascript pop up by using the statement Alert a = driver.switchTo().alert();
* In Alert interface we have different methods. They are
  + accept() click on Ok
  + dismiss() click on cancel
  + getText() get the text
  + sendKeys() enter the text
* If the javascript pop-up is not present and still if we try to switch into it, then it will throw NoAlertPresentException

### Simple:

* + In this method, we just click single ok button in the alert.

### Example program:

**public class** Dummy2 {

**public static void** main(String[] args) **throws** InterruptedException { System.*setProperty*("webdriver.gecko.driver",

"C:/Users/siva/workspace/Selenium/driver/geckodriver.exe"); WebDriver driver = **new** FirefoxDriver(); driver.get("https://mail.rediff.com/cgi-bin/login.cgi"); driver.findElement(By.*xpath*("//input[@type='submit']")).click(); Alert a = driver.switchTo().alert(); System.***out***.println(a.getText());

a.accept(); driver.switchTo().defaultContent();

driver.findElement(By.*id*("login1")).sendKeys("venkat123"); driver.quit();

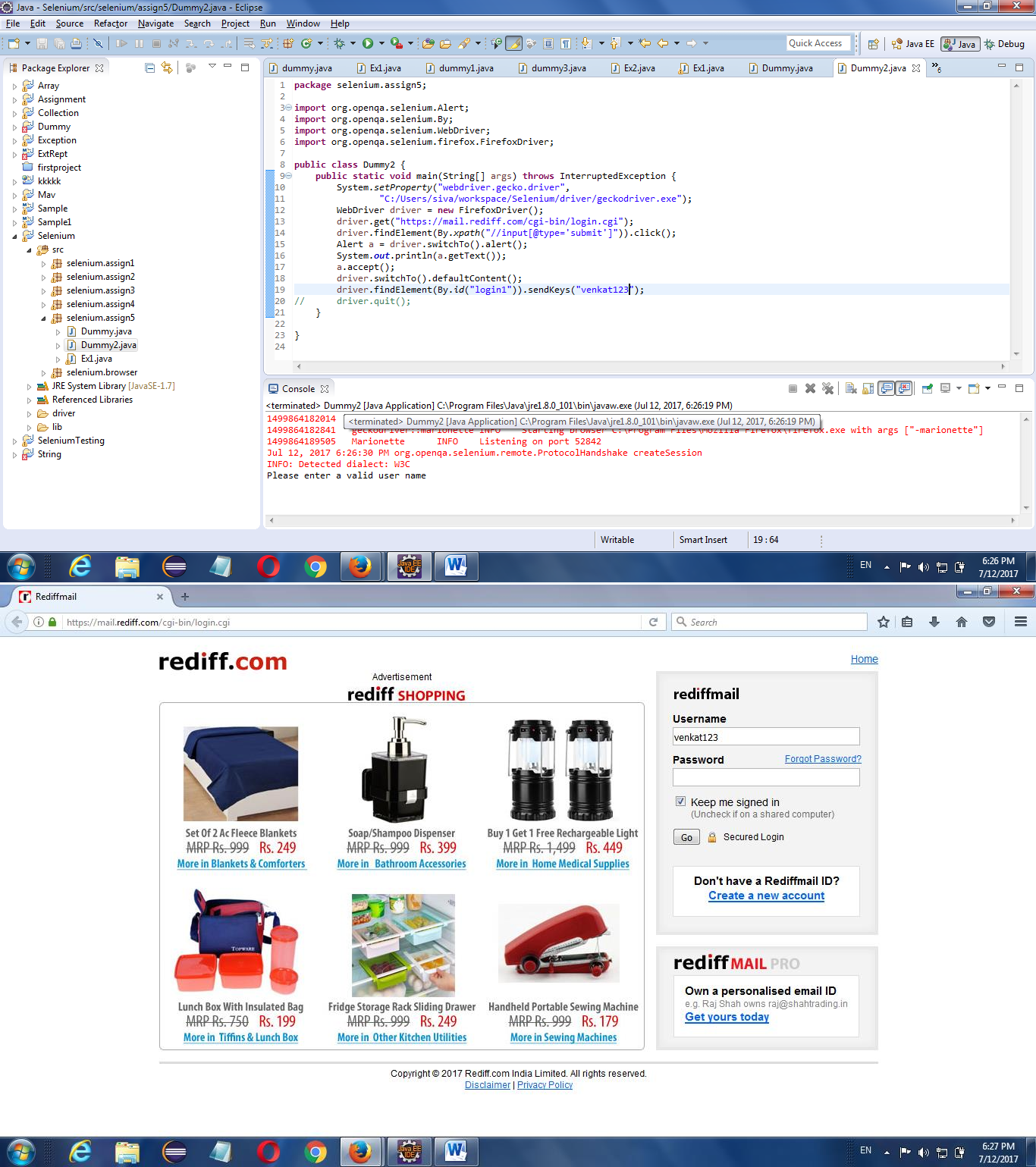
}

}

Here,

alert  class

### Output:



1. **Confirm:**
   * In this method, we have to click OK or Cancel, if we click OK/Cancel, then it will confirm

### Example program:

**public class** Dummy3 {

**public static void** main(String[] args) **throws** InterruptedException { System.*setProperty*("webdriver.gecko.driver",

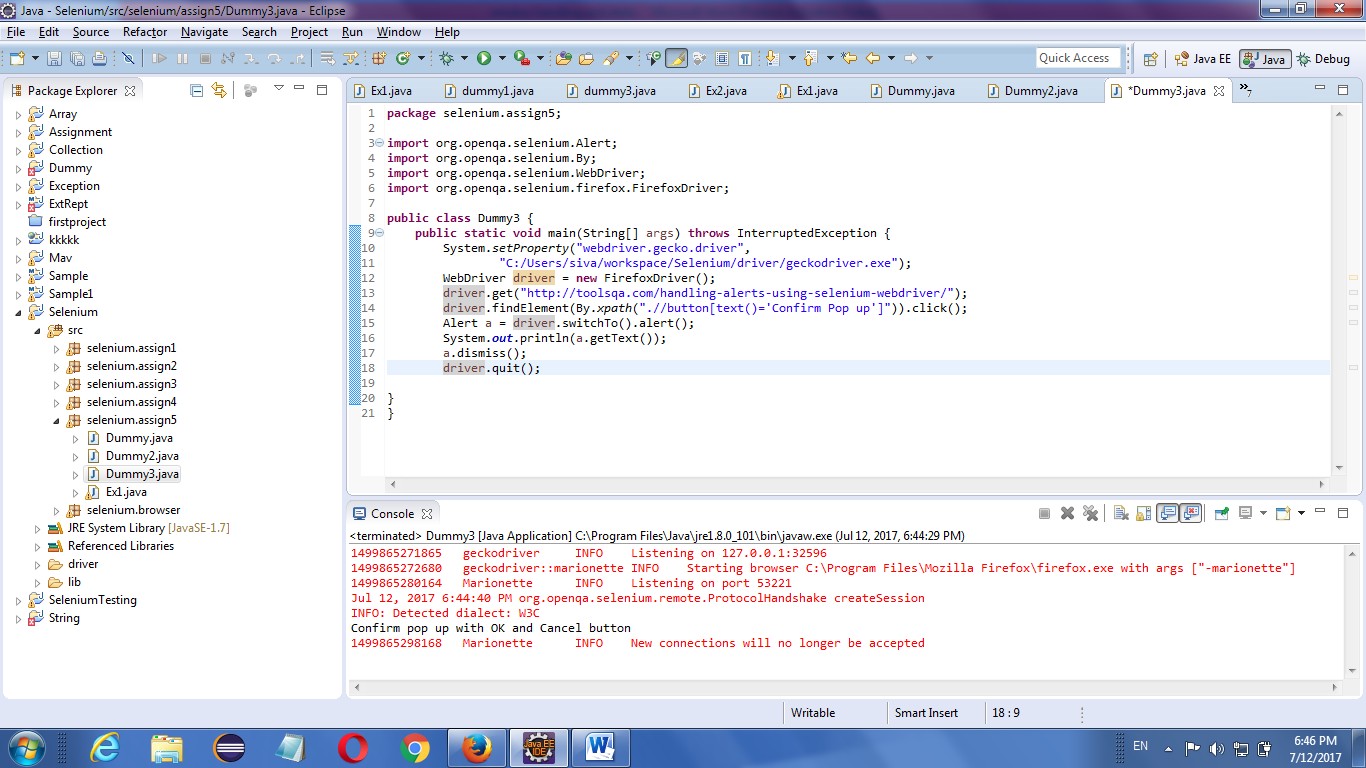
"C:/Users/siva/workspace/Selenium/driver/geckodriver.exe"); WebDriver driver = **new** FirefoxDriver();

driver.get("<http://toolsqa.com/handling-alerts-using-selenium-webdriver/>"); driver.findElement(By.*xpath*(".//button[text()='Confirm Pop up']")).click(); Alert a = driver.switchTo().alert();

System.***out***.println(a.getText()); a.dismiss();

driver.quit();}

### Output:





1. **Prompt:**
   * In this method, first we have insert Yes/No and then we will click OK/Cancel

### Example program:

**public class** Dummy4 {

**public static void** main(String[] args) **throws** InterruptedException { System.*setProperty*("webdriver.gecko.driver",

"C:/Users/siva/workspace/Selenium/driver/geckodriver.exe"); WebDriver driver = **new** FirefoxDriver();

driver.get("<http://toolsqa.com/handling-alerts-using-selenium-webdriver/>"); driver.findElement(By.*xpath*(".//button[text()='Prompt Pop up']")).click(); Alert a = driver.switchTo().alert();

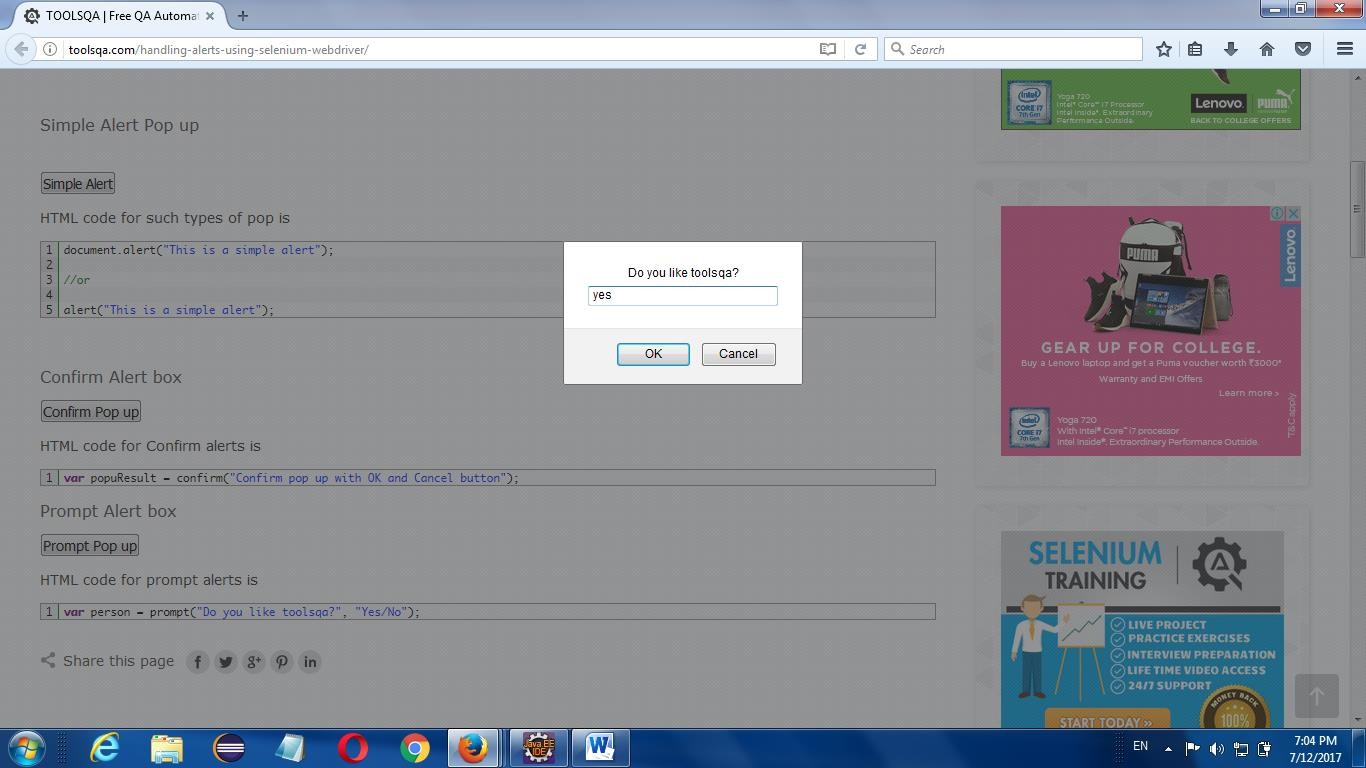
System.***out***.println(a.getText()); a.sendKeys("yes"); Thread.*sleep*(3000); a.dismiss();

driver.quit();

}

}

Output:



### WINDOW HANDLING:

* It is used to move one window to another window

### Example :

**To Handle 2 windows:**

**public class** Dummy9 {

**public static void** main(String[] args) **throws** InterruptedException { System.*setProperty*("webdriver.gecko.driver",

"C:/Users/siva/workspace/Selenium/driver/geckodriver.exe"); WebDriver driver = **new** FirefoxDriver(); driver.get("https:/[/www](http://www.hdfcbank.com/).[hdfcbank.com/](http://www.hdfcbank.com/)");

driver.findElement(By.*xpath*(".//\*[@id='cee\_closeBtn']/img")).click();

String parentWindowId = driver.getWindowHandle(); System.***out***.println("Parent Window ID:" + parentWindowId); driver.findElement(By.*id*("loginsubmit")).click(); Set<String> allWindowId = driver.getWindowHandles();

**for** (String x : allWindowId) {

**if** (!parentWindowId.equals(x)) { System.***out***.println("Child Window ID:" + x); driver.switchTo().window(x); Thread.*sleep*(3000);

driver.findElement(By.*xpath*("html/body/div[4]/div[2]/div[1]/a"))

.click(); driver.manage().window().maximize(); Thread.*sleep*(2000); driver.switchTo().defaultContent();

}

}

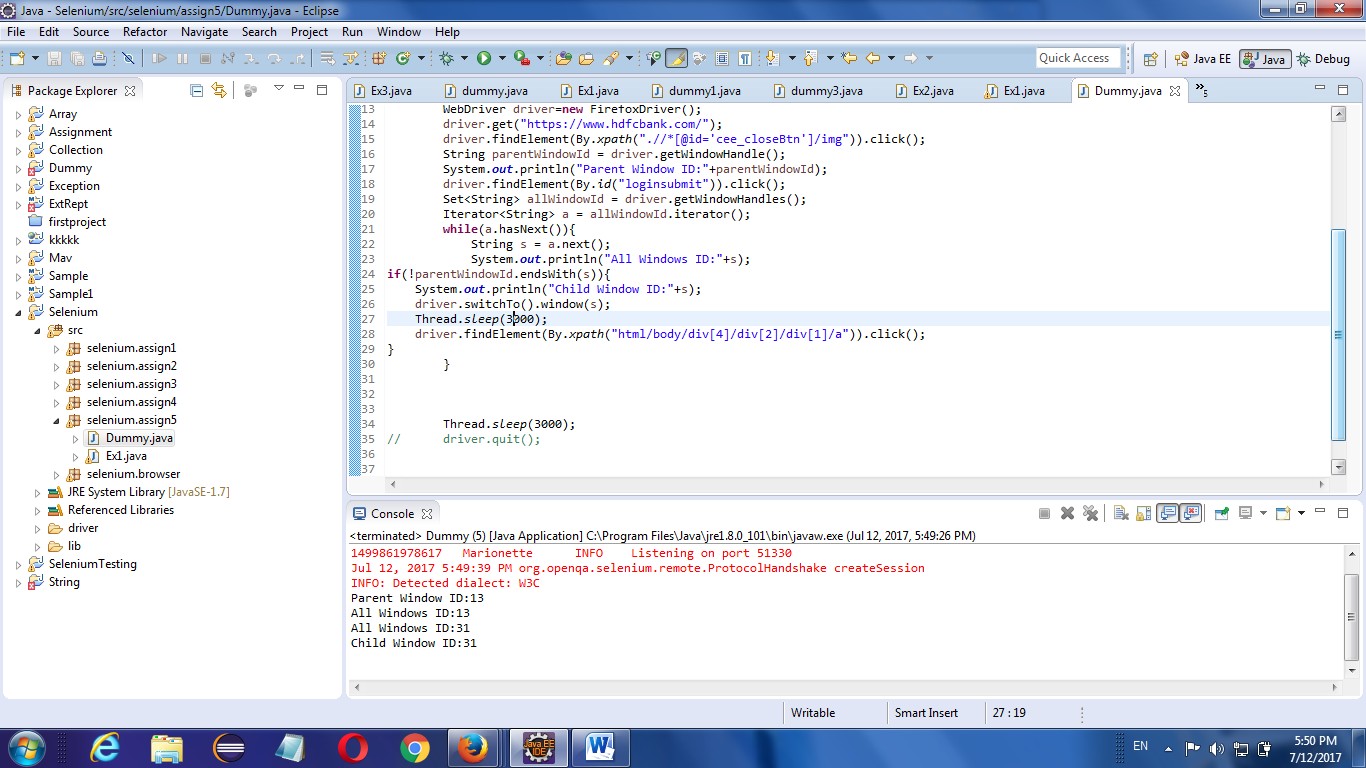
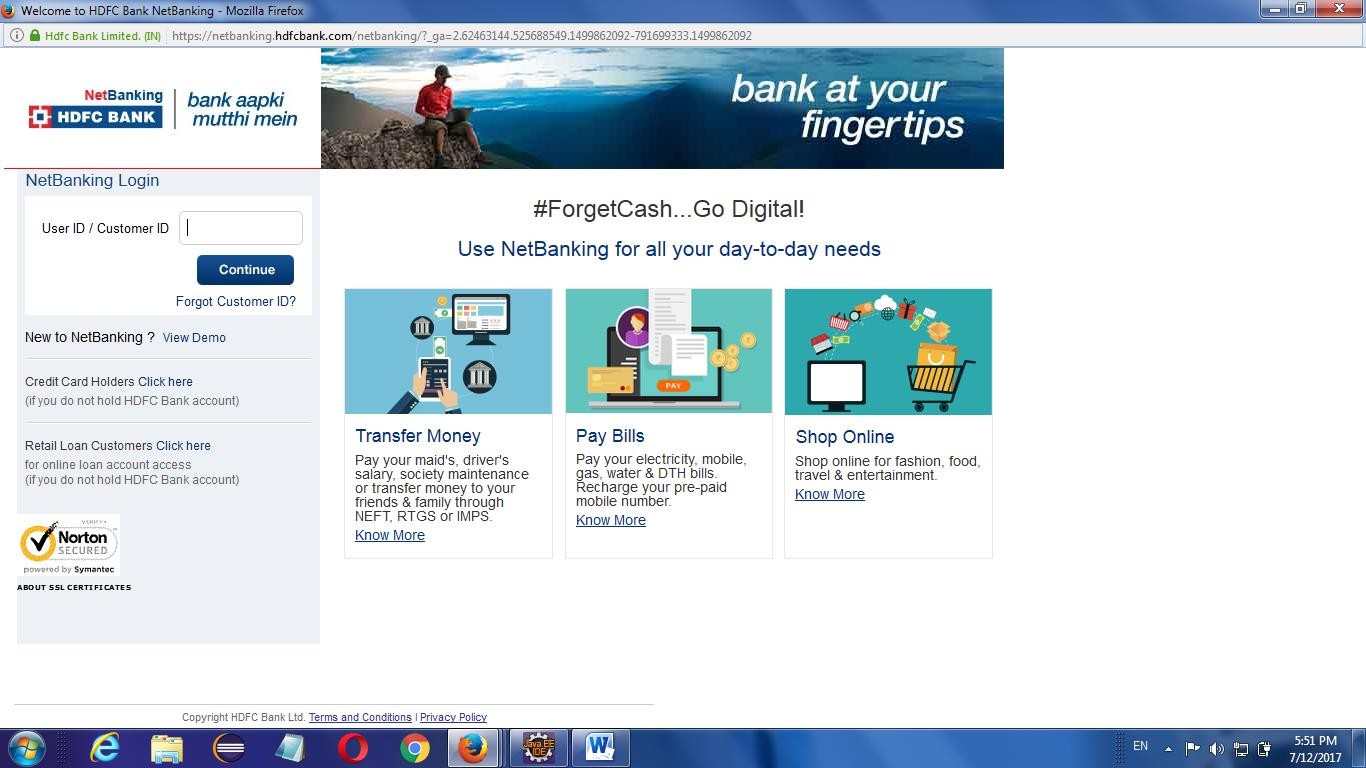
}

Here,

Thread.*sleep*(3000); driver.quit(); }

* Home page(window)  parent window
* Next window  Child window
* getWindowHandle()  It is a method used to get parent window id
* getWindowHandles()  It is a method used to get all windows id
* So, we have A(parent window id) and C(A,B) but we don’t have B.
* C  all windows id
* We have to find B using Iterator or enhancement for
* defaultContent()  it is a method used to move previous window

### Output:



**To Handling Multiple windows:**

**public class** Login {

**public static void** main(String[] args) **throws** InterruptedException, Throwable { System.*setProperty*("webdriver.chrome.driver",

"C:\\Users\\10657527\\Downloads\\chromedriver\_win32

1. \\chromedriver.exe");

WebDriver driver = **new** ChromeDriver(); driver.manage().window().maximize(); driver.get("https:/[/www](http://www.hdfcbank.com/).[hdfcbank.com/](http://www.hdfcbank.com/)"); driver.findElement(By.*xpath*(".//\*[@id='cee\_closeBtn']/img")).click(); String parentWindowId = driver.getWindowHandle(); System.***out***.println("Parent Window ID:" + parentWindowId); driver.findElement(By.*id*("loginsubmit")).click();

Set<String> allWindowId = driver.getWindowHandles(); List<String> l=**new** ArrayList<String>(allWindowId);

//By passing index we can switch the desired window driver.switchTo().window(l.get(1));

}

}

### IFRAME:

* IFrame is a web page which is embedded in another web page or an HTML document embedded inside another HTML document.
* iFrame is defined by an *<iframe></iframe>* tag in HTML. With this tag you can identify an iFrame while inspecting the HTML tree.
* Same HTML for IFrame,

<body>

<div class="box">

<iframe name="iframe1" id="IF1" height="600" width="400" src="[http://toolsqa.wpengine.com"](http://toolsqa.wpengine.com/)></iframe>

</div>

<div class="box">

<iframe name="iframe2" id="IF2" height="600" width="400" align="left" src="[http://demoqa.com](http://demoqa.com/)"></iframe>

</div>

</body>

</html>

### Identifying IFrame:

* We cannot detect the frames by just seeing the page or by inspecting Firebug.
* Observe the below image, Advertisement being displayed is an Iframe, we cannot locate or recognize that by just inspecting using Firebug. So the question is how can you identify the iframe?
* Right click on the element, If you find the option like 'This Frame' then it is an iframe.(Please refer the above diagram)
* Right click on the page and click 'View Page Source' and Search with the 'iframe', if you can find any tag name with the 'iframe' then it is meaning to say the page consisting an iframe.

### Ways to Switch IFrame:

* 1. Switch to frame by index
  2. Switch to frame by id or name
  3. Switch to frame by webelement
* We can find total number of IFrame by using below commands

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

### Switch to frame by index: Example:

**public class** Dummy9 {

@Test

**public void** upload() **throws** InterruptedException, IOException

{ System.*setProperty*("webdriver.gecko.driver",

"C:/Users/siva/workspace/Selenium/driver/geckodriver.exe"); WebDriver driver = **new** FirefoxDriver(); driver.get("<http://toolsqa.wpengine.com/iframe-practice-page/>");

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

System.***out***.println(size); driver.switchTo().frame(0);

driver.findElement(By.*name*("firstname")).sendKeys("vengat");

}

}

### Switch to frame by id or name: Using ID:

**public class** Dummy9 {

@Test

**public void** upload() **throws** InterruptedException, IOException { System.*setProperty*("webdriver.gecko.driver",

"C:/Users/siva/workspace/Selenium/driver/geckodriver.exe"); WebDriver driver = **new** FirefoxDriver(); driver.get("<http://toolsqa.wpengine.com/iframe-practice-> page/"); **int** size = driver.findElements(By.*tagName*("iframe")).size();

System.***out***.println(size); driver.switchTo().frame("IF1");

driver.findElement(By.*name*("firstname")).sendKeys("vengat");

}

}

### Using Name:

**public class** Dummy9 {

@Test

**public void** upload() **throws** InterruptedException, IOException { System.*setProperty*("webdriver.gecko.driver",

"C:/Users/siva/workspace/Selenium/driver/geckodriver.exe"); WebDriver driver = **new** FirefoxDriver(); driver.get("<http://toolsqa.wpengine.com/iframe-practice-page/>"); **int** size = driver.findElements(By.*tagName*("iframe")).size();

System.***out***.println(size); driver.switchTo().frame("iframe1");

driver.findElement(By.*name*("firstname")).sendKeys("vengat");

}

}

### Switch to frame by webelement: public class Dummy9 {

@Test

**public void** upload() **throws** InterruptedException, IOException { System.*setProperty*("webdriver.gecko.driver",

"C:/Users/siva/workspace/Selenium/driver/geckodriver.exe"); WebDriver driver = **new** FirefoxDriver(); driver.get("<http://toolsqa.wpengine.com/iframe-practice-> page/"); **int** size = driver.findElements(By.*tagName*("iframe")).size(); System.***out***.println(size);

WebElement web = driver.findElement(By.*id*("IF2")); driver.switchTo().frame(web);

driver.findElement(By.*name*("firstname")).sendKeys("venga t");

}

}

**Introduction to Test NG Designs**

* TestNG is a very important framework when you are actually developing the framework from scratch level.
* TestNG provides you full control over the test cases and the execution of the test cases. Due to this reason, TestNG is also known as a testing framework.
* Cedric Beust is the developer of a TestNG framework.
* If you want to run a test case A before that as a pre-request you need to run multiple test cases before you begin a test case A. You can set and map with the help of TestNG so that pre-request test cases run first and then only it will trigger a test case A. In such way, you can control the test cases.
* TestNG framework came after Junit, and TestNG framework adds more powerful functionality and easier to use.
* It is an open source automated TestNG framework. In TestNG, NG stands for "**Next Generation**".
* TestNG framework eliminates the limitations of the older framework by providing more powerful and flexible test cases with help of easy annotations, grouping, sequencing and parametrizing.

**Annotations in TestNG**

TestNG Annotation is a piece of code which is inserted inside a program or business logic used to control the flow of execution of test methods.

List of TestNG Annotations

|  |  |
| --- | --- |
| **TestNG Annotation** | **Description** |
| [@BeforeSuite](https://www.javatpoint.com/testng-beforesuite-annotation) | The @BeforeSuite annotated method will run before the execution of all the test methods in the suite. |
| [@AfterSuite](https://www.javatpoint.com/testng-aftersuite-annotation) | The @AfterSuite annotated method will run after the execution of all the test methods in the suite. |
| [@BeforeTest](https://www.javatpoint.com/testng-beforetest-annotation) | The @BeforeTest annotated method will be executed before the execution of all the test methods of available classes belonging to that folder. |
| [@AfterTest](https://www.javatpoint.com/testng-aftertest-annotation) | The @AfterTest annotated method will be executed after the execution of all the test methods of available classes belonging to that folder. |
| [@BeforeClass](https://www.javatpoint.com/testng-beforeclass-annotation) | The @BeforeClass annotated method will be executed before the first method of the current class is invoked. |
| [@AfterClass](https://www.javatpoint.com/testng-afterclass-annotation) | The @AfterClass annotated method will be invoked after the execution of all the test methods of the current class. |
| [@BeforeMethod](https://www.javatpoint.com/testng-beforemethod-annotation) | The @BeforeMethod annotated method will be executed before each test method will run. |
| [@AfterMethod](https://www.javatpoint.com/testng-aftermethod-annotation) | The @AfterMethod annotated method will run after the execution of each test method. |
| [@BeforeGroups](https://www.javatpoint.com/testng-beforegroups-annotation) | The @BeforeGroups annotated method run only once for a group before the execution of all test cases belonging to that group. |
| [@AfterGroups](https://www.javatpoint.com/testng-aftergroups-annotation) | The @AfterGroups annotated method run only once for a group after the execution of all test cases belonging to that group. |

**Apache POI**

Apache POI is a Java API to read and write Documents in **.xls** and **.xlsx** formats. It contains classes and interfaces. Apache POI is a powerful collection of java libraries that can do so much more than just “poor obfuscation implementation” – it allows Java applications to interact with Microsoft files like Excel, Powerpoint, and Word. An Apache POI library provides two implementations to read an excel file:

* Horrible Spreadsheet Format (HSSF) Implementation: It’s an API that works with Excel 2003 or earlier versions.
* XML Spreadsheet Format (XSSF) Implementation: It’s an API that works with Excel 2007 or later versions.

# ****Interfaces In Apache POI****

* **Workbook:** It represents an **Excel Workbook**. This is an interface implemented by **HSSFWorkbook** and **XSSFWorkbook**.
* **Sheet:** This is an interface that represents an **Excel worksheet**. A sheet is a structure of a workbook, which represents the grid of cells. A Sheet interface extends **java.lang.Iterable**.
* **Row:** It is an interface that represents the **row** of the spreadsheet. The Row interface extends **java.lang.Iterable**
* **Cell:** It is an interface. It is the representation of the **cell** in a row of a spreadsheet. **HSSFCell** and **XSSFCell** implement a Cell interface.

## **Classes In Apache POI**

* Apache POI has divided into the below classes:

1. **HSSF**
2. **XSSF**

* **XLS Classes**
* **HSSFWorkbook:** It’s a class that represents the XLS file.
* **HSSFSheet:**  It’s a class that represents the sheet in an XLS file.
* **HSSFRow:**  It’s a class that represents a row in the sheet of the XLS file.
* **HSSFCell:** It’s a class that represents a cell in a row of XLS files.
* **XLSX Classes**
* **XSSFWorkbook:** It’s a class that represents the XLSX file.
* **XSSFSheet:** It’s a class that represents the sheet in an XLSX file.
* **XSSFRow:** It’s a class that represents a row in the sheet of the XLSX file.
* **XSSFCell:** It’s a class that represents a cell in a row of an XLSX file.

### ****Steps to Read Data From Excel file****

1. **Create a workbook instance from an excel sheet**
2. **Get to the desired sheet**
3. **Increment row number**
4. **Iterate over all cells in a row**
5. **Repeat steps 3 and 4 until all data is read**

Java program to read an excel file using Apache POI library.

ReadExcelDemo.java

FileInputStream file = **new** FileInputStream(**new** File("howtodoinjava\_demo.xlsx"));

*//Create Workbook instance holding reference to .xlsx file*

XSSFWorkbook workbook = **new** XSSFWorkbook(file);

*//Get first/desired sheet from the workbook*

XSSFSheet sheet = workbook.getSheetAt(0);

*//Iterate through each rows one by one*

Iterator<Row> rowIterator = sheet.iterator();

**while** (rowIterator.hasNext()) {

Row row = rowIterator.next();

*//For each row, iterate through all the columns*

Iterator<Cell> cellIterator = row.cellIterator();

**while** (cellIterator.hasNext()) {

Cell cell = cellIterator.next();

*//Check the cell type and format accordingly*

**switch** (cell.getCellType()) {

**case** Cell.CELL\_TYPE\_NUMERIC:

System.out.print(cell.getNumericCellValue() + "t");

**break**;

**case** Cell.CELL\_TYPE\_STRING:

System.out.print(cell.getStringCellValue() + "t");

**break**;

}

}

System.out.println("");

}

file.close();

Program Output:

Output

ID NAME LASTNAME

1.0 Amit Shukla

2.0 Lokesh Gupta

3.0 John Adwards

4.0 Brian Schultz

### ****Steps to Write Data From Excel file****

1. Create a workbook
2. Create a sheet in the workbook
3. Create a row in the sheet
4. Add cells to the sheet
5. Repeat steps 3 and 4 to write more data

Java program to write an excel file using Apache POI library.

1. WriteExcelDemo.java
2. *//Blank workbook*
3. XSSFWorkbook workbook = **new** XSSFWorkbook();
4. *//Create a blank sheet*
5. XSSFSheet sheet = workbook.createSheet("Employee Data");
6. Sheet.CreateRow(0);
7. Sheet.getRow(0).CreateCell(0).setcellvalue(“Hello”);
8. Sheet.getRow(0).CreateCell(1).setcellvalue(“Hi”);
9. Sheet.CreateRow(1);
10. Sheet.getRow(1).CreateCell(0).setcellvalue(“Hello”);
11. Sheet.getRow(1).CreateCell(1).setcellvalue(“Hi”);
13. *//Write the workbook in file system*
14. FileOutputStream out = **new** FileOutputStream(**new** File("howtodoinjava\_demo.xlsx"));
15. workbook.write(out);
16. out.close();
17. System.out.println("howtodoinjava\_demo.xlsx written successfully on disk.");
18. }

**Page Object Model-Property List.**

**We use the Page Object Model concept in Selenium Webdriver due to the following reasons:**

1. An object repository is created in this POM model. It is independent of test cases and can be reused for a different project.
2. The naming convention of methods is very easy, understandable and more realistic.
3. Under the Page object model, we create page classes that can be reused in another project.
4. The Page object model is easy for the developed framework due to its several advantages.
5. In this model, separate classes are created for different pages of a web application like login page, the home page, employee detail page, change password page, etc.
6. If there is any change in any element of a website then we only need to make changes in one class, and not in all classes.
7. The script designed is more reusable, readable and maintainable in the page object model approach.
8. Its project structure is quite easy and understandable.
9. Can use PageFactory in the page object model in order to initialize the web element and store elements in the cache.
10. TestNG can also be integrated into the Page Object Model approach.