PROFENAA TECHNOLOGIES

SOFTWARE TESTING COURSES

Syllabus for software testing courses

Advanced Selenium

- Basics of Selenium
- Installation steps
- Browser Launching
- Locators
- Debug
- Actions
- Navigations
- Alert
- Screenshot
- JAVA script executor
- Scroll up and Scroll Down
- Drop Downs
- Frames
- Windows Handling
- Web table

Frameworks

- Basics of Framework
- Maven Tool
- Base class Creation
- Page object model

TDD Frameworks

- J Unit
- Test NG

BDD Frameworks

- Cucumber
    1. Cucumber Introduction
    2. Cucumber Annotations
    3. Test case Preparations
    4. Cucumber Report
    5. Cucumber Architecture

Manual Testing

- Basic concepts of Software Testing
- SDLC
- STLC
- Manual Test case Preparation
- Levels of Testing
- Testing Techniques
- SDLC Methodologies-Agile
- Defect life cycle
- Defect Management-JIRA
- Priority and Severity

Basics of Domain Knowledge
Interview preparation classes
CV preparation guidelines
Mock Test preparation

ADVANCED SELENIUM

Basics of Selenium.

What is selenium?

1. Selenium is the automation testing tool
2. It is popular open source testing tool used for web application testing
3. It is used to automate the web based applications
4. It is used to test the Functionality of Web applications

Advantages of Selenium

1. It is the open source tool
2. It supports lot of programming languages including java, python, ruby, JavaScript etc.
3. It supports various operating systems include Windows, Linux and Mac...
4. It supports various browsers includes chrome, Mozilla Fire Fox, Internet Explorer, opera.

Disadvantages of Selenium

1. It supports Web based applications only not for mobile based and mobile web based applications
2. We can’t automate images, barcodes, captcha, puzzles, OTP etc.

Selenium Components

1. **Selenium IDE**
    - IDE- Integration Development Environment
    - It is an open source test automation tool that can record and playback your actions on web

    Drawbacks

    - It supports only on Firefox browser

2. **Selenium RC**
    - RC- Remote Control
    - It is an intermediate server to communicate with the browsers
    - It acts as mediator between the selenium commands and the browsers

    Drawbacks

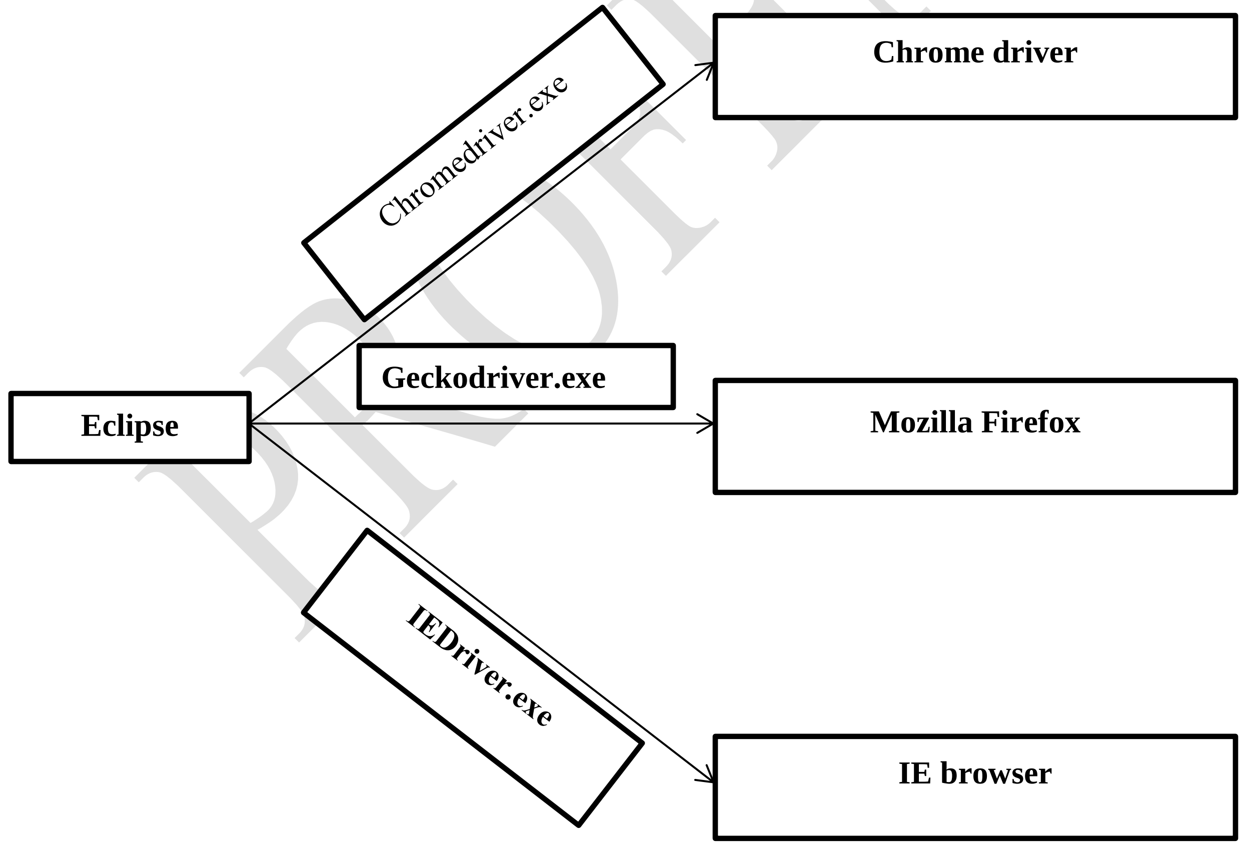
    - Speed of Execution will be slow compared to the selenium WebDriver

3. **Selenium WebDriver**
    - It is the mostly used component and currently used one
    - It supports all browsers
    - It does not need any intermediate server it interact directly with the browsers
    - 4.8.1 is the latest stable version of selenium
    - Selenium 3.141.59 is the version used.

4. **Grid**
    - It is the advanced one of selenium
    - Selenium grid can be used to perform the cross browser testing at a scale by running the test on different browser –device combination

Driver

- It can acts as bridge between the eclipse and the browser



Installation Steps

1. **Download Selenium jar file:**
   - Go to google-→ search selenium download→click on first link→latest stable version 4.1.3

2. **Download Chrome driver:**
   - Go to google→search selenium download→click on first link→scroll down till browser→click on chrome documentation→click on version→ chromedriver_win32.zip→extract.

3. **Download gecko driver:**
   - Go to google→search selenium download→click on first link→scroll down till browser→click on Firefox documentation→gecko driver releases→click on version→geckodriver_win64.zip→extract

4. **Download IE Driver server**
   - Go to google→search selenium download→click on first link→64Bit windows IE

Browser Launching

1. **Create new Project**
    - Click on file→select new→java project→give the project name→click ok

2. **Create folder for Selenium jar file**
    - Right click on newly created project→select new→Folder→give the folder name→click ok
    - Copy and paste the downloaded selenium jar file in the newly created folder

3. **Configure jar file**
    - Right click on the jar file in the newly created folder→select Buildpath→add to Buildpath

4. **Create folder for drivers**
    - Right click on the project→select new→Folder→give the folder name→click finish
    - Copy and paste all the extracted chrome, Firefox and IE driver files in the newly created folder.

5. After creating the new project we have two folders one is JRE system library and another one is src

6. Right click on the src folder→select new→select packages→ give the package name→click Finish→the package is created under src folder.

7. Right click on the newly created package→select new→select class→give the class name→click Finish→the new class is created under the package name.

8. After creating the class under the package we have newly opened page with the class name.

```
package org.cts;

Public class SampleProgram {

	}
```

9. After the class is created create the main method by using main[ctrl+space]

```
Package org.cts;

public class SampleProgram {
public static void main(String[] args)  {

}}
```

10. After creating the main method we have to launch the browser by using system.setproperty
    - System.setproperty is the key and value pair combination.
    - key = “Webdriver.chrome.driver”
    - value=”location of the project”

system.setproperty (“Webdriver.chrome.driver”,”location”);

11. **To initialize webdriver**
    WebDriver driver = new Chromedriver ();
    - WebDriver is an interface so we can’t able to create the object
    - In order to initialize the webdriver we have to use the above syntax

12. **URL Launching**
    - To launch the URL we have to use the method **get ().**
    - **Driver.get(“url”);**

13. **To get url of the webpage**
    - To get the current url of the webpage we have to use the method **getcurrenturl();**

14. **To get title of the Webpage**
    - To get the title of the webpage we have to use the method **gettitle();**

15. **To quit**
    - To quit the entire program we have to use the method as **driver.quit();**

Sample Program

```
package org.cts;

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

public class SampleProgram {


             Public static void main (String [] args) {

      //launch browser
      System.setProperty ("Webdriver.chrome.driver", location");

      //initialize webdriver
      WebDriver driver = new ChromeDriver ();

      //to maximize the window
      driver. Manage().window().maximize();

      //to load the url
      driver.get("String url");

      //to get the URL of webpage
             String url = driver.getCurrentUrl();
             System.out.println(url);

      //to get title of webpage
             String title = driver.getTitle();
             System.out.println(title);

      //to quit the browser
             driver.quit ();


      }

}
```

OUTPUT

- The automated Facebook webpage is opened
- In the console area we can get the current url and title of the webpage
- At finally the webpage is closed.

Locators

- Locators are basically the HTML attributes of a web element. They help identify unique web elements on a page and command testing frameworks, such as Selenium WebDriver, to perform the action on those elements.
- Locators are also known as selectors.

Types of Locators

- Id
- Name
- Classname
- Xpath
- Tagname
- Linktext
- Partially linktext
- CSS selector

Webdriver-I

- Webdriver is an interface under the webdriver we have the method named as **find element ();**
- **Find element()** is the method used to find the locators
- **By ()-** abstract class which is used to locate the locators

WebElement-I

- Anything that is present on the web page is a WebElement such as text box, button, etc.
- Under the WebElement we have the method as **sendkeys()** to send or pass the value to the text boxes
- The sendkeys accept only the **String Data.**

DOM

- It is named as Document Object Model
- It contains only HTML tax
- **Tag name----purple color**
- **Attribute name---orange color**
- **Attribute value---Blue color**

Sample DOM structure

<input type="text" class="input text _55r1 input text _1kbt input text _1kbt" name="email" id="email" tab index="0" placeholder="Email address or phone number" value="" autofocus="1" autocomplete="username" aria-label="Email address or phone number">

XPATH:

- If the element does not contain id ,name and class we can go for Xpath
- The Xpath are in two types
    - ➢ Absolute Xpath
    - ➢ Relative Xpath

Absolute Xpath

- Element is taken from root tag to the desired Tagname
- It is difficult to use

Relative Xpath

- We can take element directly from the desired Tagname

1. **Syntax for Xpath**
    - //Tagname[@attributename = 'attributevalue']

Sample program for Xpath

```
WebElement btnclick = driver.findElement(By.xpath("//button[@type='submit']"));
             btnclick.click();
```

2. **Xpath with index**
    - (//Tagname[@attributename = 'attributevalue'])[index]

3. **Text Xpath**
    - If the DOM structure does not contains attribute name and attribute value we can go for text Xpath
    - The text in the DOM structure available in the **black color**

Syntax for Text Xpath

- //Tagname[text()='text value']
- **Example: >text<**

```
WebElement ref= driver.findElement(By.xpath("//div[text()='Log in to Facebook']"));
                    String text = ref.getText();
                    System.out.println(text);
```

4. **Contains text Xpath**
   - To inspect the paragraph means we can go for contains text Xpath
   - //Tagname[contains(text(),’partial text’)]

   - **Example**

```
WebElement r= driver.findElement(By.xpath("(//p[contains(text(),'PROFENAA ')])[1]"));

        String text = r.getText();
        System.out.println(text);
```

5. **Contains attribute Xpath**
   - //Tagname[contains(@attribute name, ’attribute value’]

Debug

- Debug is the process of finding and fixing error in the test script
- It is the step by step execution

Steps to debug

- Set the breakpoint wherever we want
- Running the source code in debug mode
- Debug the code and fixing the error
- Stop the debug mode execution
- F6 is the key which is used to stepover action
- While debugging the code can appears in the green color

Actions

- In some areas list of options are appeared
- In that situations we can go for Action class
- The actions class are available in two ways
  - ➢ Mouse over actions
  - ➢ Keyboard related actions

- **Syntax for Actions**

  Actions ref.name = new Actions (WebDriver ref name);

1. **Mouse over Actions**
   The mouse over actions contains the method which is **move to element ()**

Syntax

Actions class ref name. Move To Element (WebElement ref name).perform ();

Sample program for move to elements:

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

System.setProperty("webdriver.chrome.driver", "location");

WebDriver driver = new ChromeDriver();
driver.manage().window().maximize();
driver.get("String url");
WebElement txtcourse = driver.findElement(By.xpath("//a[text()='COURSES']"));
Actions acc = new Actions(driver);
acc.moveToElement(txtcourse).perform();
WebElement opt = driver.findElement(By.xpath("//span[text()='Oracle Training']"));
acc.moveToElement(opt).perform();
WebElement opts = driver.findElement(By.xpath("//span[text()='SQL Certification']"));
opts.click();
```

2. **Drag and Drop**
   - It is also mouse related work
   - Create element for both drag and drop (source and target element)

Syntax

Actions class ref name.drag and drop (WebElement source refname, WebElement target refname).perform ();

3. **Right click**
   - Right click operation is also performed by using actions class
   - The method used for right click actions is **context click();**

Syntax

Actions class ref name .context click (WebElement refname) .perform ();

Sample program for Right click

```
public static void main(String[] args) {
             System.setProperty("webdriver.chrome.driver", "location");

             WebDriver driver = new ChromeDriver();
             driver.manage().window().maximize();
```

```
		driver.get("url");

	WebElement element = driver.findElement(By.xpath("//a[text()='Gmail']"));

		Actions acc = new Actions(driver);

		acc.contextClick(element).perform();
```

4. **Double click**
   - The double click operations are performed using Actions class
   - The method used for double click operations is **double click();**

Syntax

Actions class ref name. Double click (WebElement ref name) .perform ();

5. **Keyboard actions**
   - The keyboard actions are performed by the robot class

Syntax

Robot ref name = new Robot ();

It contains two methods

- Key press
- Key release

Sample program for Robot class

```
public static void main(String[] args) {
	System.setProperty("webdriver.chrome.driver""location");

		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();

		driver.get("String url");

	WebElement element =driver.findElement(By.xpath("//a[text()='Gmail']"));

		Actions acc = new Actions(driver);

		acc.contextClick(element).perform();

		Robot r = new Robot();

		r.keyPress(KeyEvent.VK_DOWN);
		r.keyRelease(KeyEvent.VK_DOWN);
		r.keyPress(KeyEvent.VK_DOWN);
		r.keyRelease(KeyEvent.VK_DOWN);
```

```
                r.keyPress(KeyEvent.VK_DOWN);
                r.keyRelease(KeyEvent.VK_DOWN);
                r.keyPress(KeyEvent.VK_DOWN);
                r.keyRelease(KeyEvent.VK_DOWN);
                r.keyPress(KeyEvent.VK_DOWN);
                r.keyRelease(KeyEvent.VK_DOWN);
                r.keyPress(KeyEvent.VK_DOWN);
                r.keyRelease(KeyEvent.VK_DOWN);
                r.keyPress(KeyEvent.VK_ENTER);
                r.keyRelease(KeyEvent.VK_ENTER);


          }
     }
```

Navigation commands

Difference between get and navigate

Get()	Navigate.to()
• It is used to go to the particular websites	• It is also used to go to the particular websites
• It cannot maintain any browser history	• It can maintain the browser history
• Here we can't use forward,backword and refresh button	• Here we can use the forward, backword and refresh button
• Get is slow compared to the navigation because it can wait until the page is fully loaded	• Navigation is faster compared to get because it can't wait until the page is fully loaded

Syntax for Navigation

Navigation refname = driver.navigate ();

Methods in Navigation

1. Navigate.to()→ To load the particular webpage
2. Navigate. Back()→return back to the previous page
3. Navigate. Forward()→Forward to the next page
4. Navigate. Refresh()→refresh the particular page

Sample program for navigations

```
public static void main(String[] args) {
	System.setProperty("webdriver.chrome.driver", “location");

	WebDriver driver = new ChromeDriver();
	driver.manage().window().maximize();

	Navigation navigate = driver.navigate();

	navigate.to("url 1");
	navigate.to("url 2”);
	navigate.to("url 3");

	navigate.back();

}
}
```

Alert

- Alert is an interface
- It is an pop up message without handling the alert we can’t perform any actions
- Alert are not inspected

Switching the Alert

Alert refname = driver.switchto ().Alert ();

Types of Alert

1. **Simple Alert:**
    - The simple alert contains only the OK button
    - We should accept the alert
    - **Refname.accept();**

2. **Confirm Alert;**
    - The confirm alert contains both OK and cancel button
    - We should accept or dismiss the alert
    - **Refname .accept();**

      Or
    - **Refname.dismiss();**

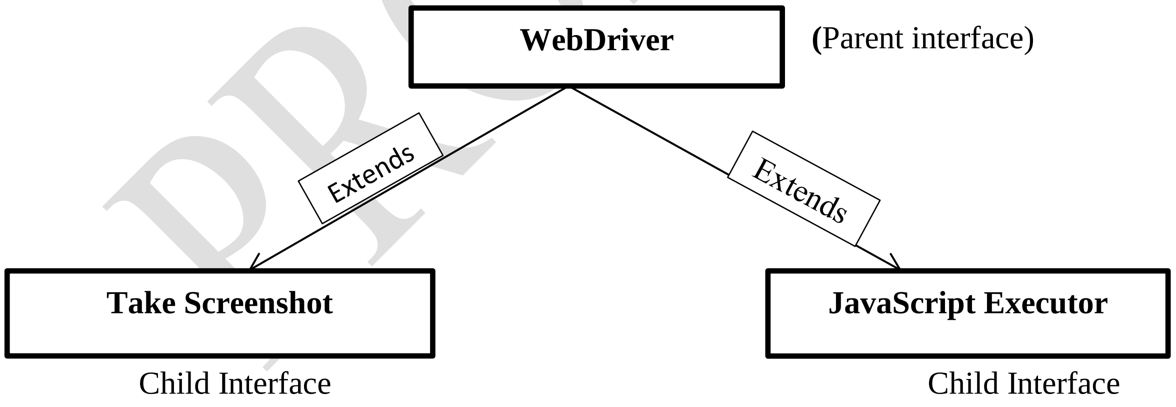
3. **Prompt alert;**
    - The prompt alert contains text box with both ok and cancel button
    - We have to insert the value using sendkeys and accept or dismiss the alert

- **Refname. Sendkeys (“ value”);**
- **Refname. accept();**

  Or
- **Refname. Dismiss ();**

Sample program for prompt alert

```
public static void main(String[] args) {
		System.setProperty("webdriver.chrome.driver", "location");

		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();

		driver.get("String url");

WebElement element2 = driver.findElement(By.xpath("(//a[@class='analystic'])[3]"));
		element2.click();
WebElement ref = driver.findElement(By.xpath("//button[@class='btn btn-info']"));
		ref.click();

		Alert al = driver.switchTo().alert();

		al.sendKeys("value");
		String text = al.getText();
		System.out.println(text);
		al.accept();

	}

}
```



Take Screenshots

- To capture screenshots in selenium one has to utilize the method **Take Screenshots**.
- This notifies WebDriver that it should take a screenshot in selenium and store it

Downcasting

- Assigning parent class object reference to child class object reference.
- It cannot be done automatically, we need to do it manually by following the below syntax

Syntax

Child Interface ref name = (Child Interface) Parent Interface ref name;

- Here the child interface as Take Screenshots and the Parent Interface as WebDriver
- So the syntax is

TakeScreenshot ref name = (TakeScreenshot) WebDriver ref name;

- Here TakeScreenshot is a sub interface of WebDriver under this sub interface we have the method which is named as **getScreenshotAs ().**

- Here to save the screenshot we have to mention the output type the output type will be the file format.it is given as
  Refname. getScreenshotAs (output type. File).
- the return type will be File
  File src Refname = Refname. getScreenshotAs (output type. File).

- Now printing the file means the screenshots are stored as temporary file in the output we can get only the memory location not the taken screenshot

- For that we have to download the common io 2.4 version jar file and build path it

- After downloaded we have copy the file and paste in already created selenium jar file stored folder

- In order to save the taken screenshot we have to create the FileUtils class.

File desfile = new File (“location. Format of screenshot”)

- Now use the copyfiles method the copyfiles method are the static method so we cannot able to create object

- Just use **classname.methodname**
- **FileUtils. Copyfiles (srcfile , desfile);**

Sample program for Screenshot

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

	System.setProperty("webdriver.chrome.driver", "location");

	WebDriver driver = new ChromeDriver();
	driver.manage().window().maximize();

	driver.get("url");
	TakesScreenshot tk = (TakesScreenshot)driver;
	File file = tk.getScreenshotAs(OutputType.FILE);
	System.out.println(file);

File desfile = new File("C:\\Users\\Sathish\\eclipse-workspace\\selenium\\sat.png");
	FileUtils.copyFile(file, desfile);
```

JavaScript Executor

- **JavaScript Executor** is an interface
- It is used to execute javascript through selenium webdriver

Purpose of javascript Executor

- If sendkeys and click method is not working we can go for javascript executor
- It is fastest one
- It is easily interact with hidden elements
- For “element intractable exception” we can go for java script executor
- We can perform scroll up and scroll down
- we can get the attribute value

- In the javascript executor we have the method which is named as **executescript();**

To insert the value

Arguments [0].setattribute (‘value’,’value’)

To click

Arguments [0].click ();

To get the Attribute value

return. Arguments [0].get Attribute (‘value’) ➡ for one webElement refname

return. Arguments [1].get Attribute (‘value’) ➡ for two webElement refname

Syntax

JavascriptExecutor Refname = (JavascriptExecutor) WebDriver Refname;

executescript (“ javascript code”,WebElement Refname)

Sample program for javascript Executor

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

	System.setProperty("webdriver.chrome.driver", "location");

	WebDriver driver = new ChromeDriver();
	driver.manage().window().maximize();
	driver.get("url");
	JavascriptExecutor js = (JavascriptExecutor) driver;
	WebElement element = driver.findElement(By.id("email"));
	js.executeScript("arguments[0].setAttribute('value','abc@123')",element);

Object obj = js.executeScript("return arguments[0].getAttribute('value')",element);

	String s = (String) obj;
	System.out.println(s);


}
}
```

Scroll up:

Pick an element from top of the page

Arguments [0]. ScrollIntoView (true)

Sample program for scroll up

```
public static void main(String[] args) {
		System.setProperty("webdriver.chrome.driver", "location");

		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();

		driver.get("url");

		JavascriptExecutor js = (JavascriptExecutor) driver;
	WebElement up = driver.findElement(By.xpath("//img[@class='_97vu img']"));
		js.executeScript("arguments[0].scrollIntoView(true)", up);


}}
```

Scroll down:

Pick an element from bottom of the page

Arguments[0].ScrollIntoView(false)

Sample program for Scroll down

```
public static void main(String[] args) {
	System.setProperty("webdriver.chrome.driver", "location");

	WebDriver driver = new ChromeDriver();
	driver.manage().window().maximize();

	driver.get("url");

	JavascriptExecutor js = (JavascriptExecutor) driver;
	WebElement up = driver.findElement(By.xpath("//a[text()='Watch']"));
	js.executeScript("arguments[0].scrollIntoView(false)", up);


}}
```

Drop Down

- The **select** class in the selenium webdriver is used for selecting and deselecting option in a dropdown

Drop down syntax

Select refname = new Select(webElement refname)

- It contains variety of methods

1. **Select by index**
    - Here we can select the options by the available index position
      Refname. Select by index(10);

2. **Select by value**
    - Here we can select the option by the value which are seen in blue color in the inspected area
      Refname. Select by value(“2”);

3. **Select by visible text**
    - Here we can select the options by the visible text which are seen in black color in the inspected area
    - Refname.Select by visibletext (“India”);

4. **Get options**
    - In this get options method we can get all the options which are available

5. **Is multiple**
    - This option is used to check whether drop down is single selected or multi selected

6. Deselect by index

- In this option we can deselect the selected option by the index position
  Refname. Deselect by index (10);

7. **Deselect by value**
    - In this option we can deselect the selected option by the value
      Refname. Deselect by value(“10”);

8. **Deselect by visibletext**
    - In this option we can deselect the selected options by the visibletext
      Refname. Deselect by visibletext (“India”);

9. **Get all selected options**
    - In this option we can get all the selected options by using index, value and the visibletext

10. Get first selected options

- In this options we can get the first selected option

Frames

- When we inspect an image we can get only “no such element exceptions” because element is present inside of frame
- HTML document embedded with another HTML document
- It contains two Tagname
    - **Iframes**
    - **Frameset**

Switch to frames

- Driver.switchto().frame(String id);
- Driver.switchto().frame(String name);
- Driver.switchto().frame(int index);
- Driver.switchto().frame(WebElement refname);

Windows Handling

- In amazon search box search iPhone it will going to open in the new window so we can get the “no such element exception”

- Here “no such element” occurred because the element is present in another window

- The first window is the parent window and the second window is the child window

Syntax

- Driver.switchto().windows();

Two methods

- Get window handle()
- Get window handles()

1. Get window Handle()

- The get window handle method is used to get the parent window id
- The return type will be String

2. **Get window handles()**
   - The get window handles is the method which is used to get the all window id including parent window id
   - The return type will be set<String>

- ➢ Here windows id are the alpha numeric one it contains both alphabetical and numerical function so the return type will be string

- ➢ If parent window id and all window id are same means we no need to switch the control

- ➢ If parent window id and all window id are different means we need to switch the control

- ➢ In order to return back to the parent window means we can use the method default content()
  Driver. switch to().default content();

- ➢ The default content method is not possible so we can use
- ➢ **Driver. Switch to().window(parent window id).**

Sample program for windows handling

```
    public static void main(String[] args) {
          System.setProperty("webdriver.chrome.driver", "location");

          WebDriver driver = new ChromeDriver();
          driver.manage().window().maximize();
          driver.get("url");

    WebElement iphonesrch = driver.findElement(By.id("twotabsearchtextbox"));
          iphonesrch.sendKeys("iphone",Keys.ENTER);

WebElement link = driver.findElement(By.xpath("//span[text()='Apple iPhone 14 128GB
Starlight']"));
          linkiphone.click();

    String parentwindowid = driver.getWindowHandle();
    System.out.println(parentwindowid);

    Set<String> allwindowid = driver.getWindowHandles();
    System.out.println(allwindowid);

    for (String eachwindowid: allwindowid) {
          if (!parentwindowid.equals(eachwindowid)) {
                driver.switchTo().window(eachwindowid);

          }
    }
    WebElement findElement = driver.findElement(By.id("add-to-cart-button"));
```

```
        findElement.click();

                driver.switchTo().window(parentwindowid);
```

}

WebTable

- The table which contains row,header,data which are present in webpage are webtable
- In the webpage contains lot of table we need to inspect the particular table
- The method used are **findelements()**
- At first we can fetch table rows from the table
- From the table row we can get the table header
- And finally from the table header we can get the table data

Sample program for webtable

```
public static void main(String[] args) {
                // TODO Auto-generated method stub

                System.setProperty("webdriver.chrome.driver", "location");

                WebDriver driver = new ChromeDriver();
                driver.manage().window().maximize();
                driver.get("url");

                WebElement element = driver.findElement(By.id("customers"));
                List<WebElement> tablerows = element.findElements(By.tagName("tr"));
                for (int i = 0; i < tablerows.size(); i++) {
                        WebElement tablerow = tablerows.get(i);

                List<WebElement> tabledatas = tablerow.findElements(By.tagName("td"));
                    for (int j = 0; j < tabledatas.size(); j++) {
                                WebElement tabledata = tabledatas.get(j);
                                String text = tabledata.getText();
                                System.out.println(text);

                                        System.out.println("row no:" +(i+1));
                                        System.out.println("column no:" +(j+1));

                        }


                }
                }
```