**1.How to verify Alert is present or not ?**

Public Boolean isAlertPresent(){

Try{

Driver.switchTo.Alert();

Return true;

}

Catch{

Return false;

}

}

**2. Find the position of the X and Y co-ordinate .**

driver.manage().window().getPosition().getX();

driver.manage().window().getPosition().getY();

**3. Explain the different exceptions in Selenium WebDriver.**

An exception in Java in an event, which can disturb our program flow or you, can say it can terminate our program.

In Java, we can handle exception using try- catch block. Try can have multiple catch blocks.

Try must have catch block or finally block.

Note- Make Sure Parent Exception should come in last catch block otherwise we will face code, not reachable error.

**illegalstateexception** - when you are working with 3rd party browser that time you need to set the location of the driver else you get this exception.

**NoSuchElementException** - This exception occurs when WebDriver is unable to identify the elements during run time. Due to wrong selector or selector which is not exist.

**ElementNotVisibleException** - This Exception occurs when the element presence is in DOM, it is not visible

Example:-

Hidden Elements, which has presence in DOM and it is not visible. Visibility means the height and width should be greater than zero. Hidden Elements are defined in HTML using of type=”hidden”.

**NoSuchFrameException** - This Exception occurs when the driver is switching to an invalid frame, which is not available.

**NoAlertPresentException** - This Exception occurs when the driver is switching to an invalid Alert, which is not available.

**NoSuchWindowException** - This Exception occurs when the driver is switching to an invalid Window, which is not available.

**WebDriverException** - This Exception occurs when the driver is performing the action after immediately closing the browser.

Example:-

driver.close();

driver.findElement(By.id("username")).sendKeys("Mukesh");

**SessionNotFoundException** - This Exception occurs when the driver is performing the action after immediately quitting the browser.

**InvalidStateException**- when there is button but we want to type or there is a texbox(disable) but we want to write that time will get this exception.

**StaleElementReferenceException** - This Exception occurs when the Element belongs to a different frame/window than the current one. The user has navigated away to another page.

Example:-

WebElement element=driver.findElement(By.id("username"));// Element is available in parent window

driver.switchTo().window(Child\_Window);//Switch to Child Window

element.sendKeys("Mukesh");//perform the action on the element which is not visible in the child window

**How to solve stale element reference exception in selenium**

**Solution 1–**

You can refresh the page and again try for the same element.

Example- If you are trying to click on link and getting the exception then try in below format

Driver.navigate().refersh();

Driver.findElement(By.id(“ur element id”)).click();

**Solution 2-**

Sometimes it takes the time to attach element on Dom so you can retry using for loop and try catch.

**for**(**int** i=0i<=2;i++)

{

**try**{

Driver.findElement(By.id()).click();

**break**;

}

**catch**(Exception e)

{

Sysout(e.getMessage());

}

**Reasons for ElementNotVisibleException**

**Reason 1- Duplicated XPATH**

While writing xpath for your application, you might have taken xpath that is matching with more than 1 element, in this case, Selenium will throw Element, not the visible exception.

**Reason 2-** If you are trying to access some particular element on Webpage that is not currently visible, in this case also you will get the Element, not visible exception.

**First Solution:** Try to write unique XPATH that matches with a single element only.

**Second Solution:** Use Explicit wait feature of Selenium and wait till the element is not visible. Once it is visible then you can perform your operations.

**Third solution:**

int ok\_size=driver.findElements(By.xpath("//button[text()='OK']")).size();

driver.findElements(By.xpath("//button[text()='OK']")).get(ok\_size-1).click();

**WebDriverException: f.QueryInterface is not a function**

While passing URL in Selenium we have to pass protocol as well so that Selenium can understand which protocol to use while communicating.

There are so many protocols available some of them are

FTP, SMTP, HTTP etc.

Solution:

While passing URL we have to mention HTTP also.

driver.get(“http://www.learn-automation.com”);

**The reason for the element is not clickable at point(x,y) exception.**

Some of my observation was

It mostly happens in Chrome so if you are mostly working with Firefox or IE then you will not be getting this exception.

Chrome does not calculate the exact location of element

Chrome always click in the middle of Element.

Sometimes you will get this exception due to Sync issue also.

**WebElement elementToClick = driver.findElement(By.xpath("Your xpath"));**

**((JavascriptExecutor)driver).executeScript("window.scrollTo(0,"+elementToClick.getLocation().y+")");**

**elementToClick.click();**

**OR**

**WebElement elementToClick = driver.findElement(By.xpath("Your xpath"));**

**((JavascriptExecutor)driver).executeScript("window.scrollTo(0,"+elementToClick.getLocation().x+")");**

**elementToClick.click();**

**4. What is exception test in Selenium?**

An exception test is an exception that you expect will be thrown inside a test class. If you have written a test case in such way that it should throw an exception, then you can use the **@Test** annotation and specify which exception you will be expecting by mentioning it in the parameters. Take a look at the example below:

**@Test(expectedException = NoSuchElementException.class)**

Do note the syntax, where the exception is suffixed with .class

This is used for Negative test cases.

**5. Why and how will you use an Excel Sheet in your project?**

The reason we use Excel sheets is because it can be used as data source for tests. An excel sheet can also be used to store the data set while performing DataDriven Testing. These are the two main reasons for using Excel sheets.

When you use the excel sheet as **data source**, you can store the following:

* **Application URL for all environments**: You can specify the URL of the environment in which you want to do the testing like: development environment or testing environment or QA environment or staging environment or production/ pre-production environment.
* **User name and password credentials of different environments:**You can store the access credentials of the different applications/ environments in the excel sheet. You can store them in encoded format and whenever you want to use them, you can decode them instead of leaving it plain and unprotected.
* **Test cases to be executed**: You can list down the entire set of test cases in a column and in the next column, you can specify either Yes or No which indicates if you want that particular test case to be executed or ignored.

When you use the excel sheet for **DataDriven Test**, you can store the data for different iterations to be performed in the tests. For example, while testing a web page, the different sets of input data that needs to be passed to the test box can be stored in the excel sheet.

**6. How can you redirect browsing from a browser through some proxy?**

Selenium provides a PROXY class to redirect browsing from a proxy. Look at the example below:

|  |  |
| --- | --- |
|  | String PROXY = “199.201.125.147:8080”;  Proxy proxy = new Proxy();  proxy.setHTTPProxy(Proxy)   .setFtpProxy(Proxy)   .setSslProxy(Proxy)  DesiredCapabilities cap = new DesiredCapabilities();  cap.setCapability(CapabilityType.PROXY, proxy);  WebDriver driver = new FirefoxDriver(cap); |

**7. What is POM (Page Object Model)? What are its advantages?**

1- It is design pattern in which will help you to maintain the code and code duplication, which is a crucial thing in Test automation.

2- You can store all locators and respective methods in the separate class and Call them from the test in which you have to use. So the benefit from this will be if any changes in Page then you do not have to modify the test simply modify the respective page and that all.

3- You can create a layer between your test script and application page, which you have to automate.

4- In other words, it will behave as Object repository where all locators are saved.

#### **Program for Page Object Model using Selenium Webdriver without Page factory**

**public** **class** LoginPage {

WebDriver driver;

By username = By.id("user\_login");

By password = By.xpath(".//\*[@id='user\_pass']");

By loginButton = By.name("wp-submit");

**public** LoginPage(WebDriver driver) {

**this**.driver = driver;

}

**public** **void** loginToWordpress(String userid,String pass)

{

driver.findElement(username).sendKeys(userid);

driver.findElement(password).sendKeys(pass);

driver.findElement(loginButton).click();

}

### **Code for Page Object Model Using Selenium Webdriver using Page Factory**

**public** **class** LoginPageNew {

WebDriver driver;

**public** LoginPageNew(WebDriver ldriver) {

**this**.driver = ldriver;

}

@FindBy(id = "user\_login") WebElement username;

@FindBy(how = How.XPATH, using = ".//\*[@id='wp-submit']")

@CacheLookup

WebElement submit\_button;

**public** **void** login\_wordpress(String uid,String pass)

{

username.sendKeys(uid);

password.sendKeys(pass);

submit\_button.click();

}

**8. What is Page Factory?**

Page Factory gives an optimized way to implement Page Object Model. When we say it is optimized, it refers to the fact that the memory utilization is very good and also the implementation is done in an object oriented manner.

Page Factory is used to initialize the elements of the Page Object or instantiate the Page Objects itself. Annotations for elements can also be created (and recommended) as the describing properties may not always be descriptive enough to differentiate one object from the other.

The concept of separating the Page Object Repository and Test Methods is followed here also. Instead of having to use ‘FindElements’, we use annotations like: **@FindBy** to find WebElement, and **initElements** method to initialize web elements from the Page Factory class.

**@FindBy** can accept **tagName**, **partialLinkText**, **name**, **linkText**, **id**, **css**, **className**& **xpath**as attributes.

**9. What are the different types of WAIT statements in Selenium WebDriver? *Or the question can be framed like this:* How do you achieve synchronization in WebDriver?**

There are basically two types of wait statements: **Implicit Wait** and **Explicit Wait**.

Implicit wait instructs the WebDriver to wait for some time by polling the DOM. Once you have declared implicit wait, it will be available for the entire life of the WebDriver instance. By default, the value will be 0. If you set a longer default, then the behavior will poll the DOM on a periodic basis depending on the browser/ driver implementation.

Explicit wait instructs the execution to wait for some time until some condition is achieved. Some of those conditions to be attained are:

* elementToBeClickable
* elementToBeSelected
* presenceOfElementLocated

**10. Write a code to wait for a particular element to be visible on a page. Write a code to wait for an alert to appear.**

We can write a code such that we specify the XPath of the web element that needs to be visible on the page and then ask the WebDriver to wait for a specified time. Look at the sample piece of code below:

|  |  |
| --- | --- |
|  | **WebDriverWait wait=new WebDriverWait(driver, 20);**  **Element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath( “<xpath”)));** |

Similarly, we can write another piece of code asking the WebDriver to wait until an error appears like this:

|  |  |
| --- | --- |
|  | **WebDriverWait wait=new WebDriverWait(driver, 20);**  **Element = wait.until(ExpectedConditions.alertIsPresent());** |

**11. What is the use of JavaScriptExecutor?**

We have used Java in our script and we implemented almost all feature but some features we can’t handle using Java so we need scripting language also which can control server side or client side scripting so we will use JavaScript in our Selenium script.

some functionality cannot handle by Selenium (scrolling, forcefully data sending, resizing a window) so we need a **JavaScriptExecutor**

**JavaScriptExecutor** is an interface which provides a mechanism to execute JavaScript through the Selenium WebDriver. It provides “**executescript**” and “**executeAsyncScript**” methods, to run JavaScript in the context of the currently selected frame or window. An example of that is:

|  |  |
| --- | --- |
|  | JavascriptExecutor js = (JavascriptExecutor) driver;  js.executeScript(Script,Arguments); |

**12. How to scroll down a page using JavaScript in Selenium?**

We can scroll down a page by using window.scrollBy() function. Example:

|  |  |
| --- | --- |
|  | ((JavascriptExecutor) driver).executeScript("window.scrollBy(0,500)"); |

**13. How to scroll down to a particular element?**

To scroll down to a particular element on a web page, we can use the function **scrollIntoView()**. Example:

|  |  |
| --- | --- |
|  | ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView();", element); |

**14. How to handle keyboard and mouse actions using Selenium?**

We can handle special keyboard and mouse events by using **Advanced User Interactions API**. The Advanced User Interactions API contains the Actions and the Action Classes that are needed for executing these events. Most commonly used keyboard and mouse events provided by the Actions class are in the table below:

|  |  |
| --- | --- |
| **Method** | **Description** |
| clickAndHold() | Clicks (without releasing) the current mouse location. |
| dragAndDrop() | Performs click-and-hold at the location of the source element, moves. |
| source, target() | Moves to the location of the target element, then releases the mouse. |

**Example-**

**Actions act=new Actions(driver);**

**// find element which we need to drag**

**WebElement drag=driver.findElement(By.xpath(".//\*[@id='draggable']"));**

**// find element which we need to drop**

**WebElement drop=driver.findElement(By.xpath(".//\*[@id='droppable']"));**

**// this will drag element to destination**

**act.dragAndDrop(drag, drop).build().perform();**

**15. What are different types of frameworks?**

The different types of frameworks are:

* Data Driven Framework:-  
  When the entire test data is generated from some external files like Excel, CSV, XML or some database table, then it is called Data Driven framework.
* Keyword Driven Framework:-  
  When only the instructions and operations are written in a different file like an Excel worksheet, it is called Keyword Driven framework.
* Hybrid Framework:-  
  A combination of both the Data Driven framework and the Keyword Driven framework is called Hybrid framework.

**16. Which files can be used as data source for different frameworks?**

Some of the file types of the dataset can be: excel, xml, text, csv, etc.

**17. How can you fetch an attribute from an element? How to retrieve typed text from a textbox?**

We can fetch the attribute of an element by using the **getAttribute()** method. Sample code:

|  |  |
| --- | --- |
|  | **WebElement eLogin = driver.findElement(By.name(“Login”);**  **String LoginClassName = eLogin.getAttribute("classname");** |

Here, I am finding the web page’s login button named ‘Login’. Once that element is found, getAttribute() can be used to retrieve any attribute value of that element and it can be stored it in string format. In my example, I have retrieved ‘classname’ attribute and stored it in LoginClassName.

Similarly, to retrieve some text from any textbox, we can use getText() method. In the below piece of code I have retrieved the text typed in the ‘Login’ element.

|  |  |
| --- | --- |
|  | **WebElement eLogin = driver.findElement(By.name(“Login”);**  **String LoginText = Login.getText ();** |

**18.How to send ALT/SHIFT/CONTROL key in Selenium WebDriver?**

**When we generally use ALT/SHIFT/CONTROL keys, we hold onto those keys and click other buttons to achieve the special functionality. So it is not enough just to specify keys.ALT or keys.SHIFT or keys.CONTROL functions.**

**For the purpose of holding onto these keys while subsequent keys are pressed, we need to define two more methods: keyDown(modifier\_key) and keyUp(modifier\_key)**

**Parameters: Modifier\_key (keys.ALT or Keys.SHIFT or Keys.CONTROL)**

**Purpose: Performs a modifier key press and does not release the modifier key. Subsequent interactions may assume it’s kept pressed.**

**Parameters: Modifier\_key (keys.ALT or Keys.SHIFT or Keys.CONTROL)**

**Purpose: Performs a key release.**

**Hence with a combination of these two methods, we can capture the special function of a particular key.**

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

{

String baseUrl = “https://www.facebook.com”;

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

driver.get("baseUrl");

WebElement txtUserName = driver.findElement(By.id(“Email”);

Actions builder = **new** Actions(driver);

Action seriesOfActions = builder.moveToElement(txtUerName).click()

.keyDown(txtUserName, Keys.SHIFT)

.sendKeys(txtUserName, “hello”)

.keyUp(txtUserName, Keys.SHIFT)

.doubleClick(txtUserName)

.contextClick()

.build();

seriesOfActions.perform();

## }

## ****19. How to take screenshots in Selenium WebDriver?****

You can take a screenshot by using the **TakeScreenshot** function. By using **getScreenshotAs()** method you can save that screenshot. Example:

File src = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

**try** {

FileUtils.copyFile(src, **new** File("C:/selenium/error.png"));

}

**catch** (IOException e) {

System.***out***.println(e.getMessage());

}

## How to Capture screenshot in selenium for failed test cases

1-If script has some issue (some locator has been changed or application has some changes)- In this case, we need to maintain our Selenium script.

2-Due to application issue- In this case, we need to inform to respective point of contact (Manual Tester or Developer)

Here I will be using two new topics which will help us to achieve the same.

1-We will use ITestResult Interface which will provide us the test case execution status and test case name.

Please refer official doc for **ITestResult**

2- @AfterMethod is another annotation of TestNG which will execute after every test execution whether test case pass or fail @AfterMethod will always execute.

## Program to Capture screenshot in selenium for failed test cases

// It will execute after every test execution

@AfterMethod

**public** **void** tearDown(ITestResult result) {

// Here will compare if test is failing then only it will enter into if condition

**if** (ITestResult.FAILURE == result.getStatus()) {

**try** {

// Create refernce of TakesScreenshot

TakesScreenshot ts = (TakesScreenshot) driver;

// Call method to capture screenshot

File source = ts.getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(source, **new** File("./Screenshots/" + result.getName() + ".png"));

System.***out***.println("Screenshot taken");

} **catch** (Exception e) {

System.***out***.println("Exception while taking screenshot " + e.getMessage());

}

}

|  |  |
| --- | --- |
|  |  |

## ****20. How to set the size of browser window using Selenium?****

To maximize the size of browser window, you can use the following piece of code:  
**driver.manage().window().maximize();** – To maximize the window

To resize the current window to a particular dimension, you can use the **setSize()** method. Check out the below piece of code:

|  |  |
| --- | --- |
|  | System.out.println(driver.manage().window().getSize());  Dimension d = new Dimension(420,600);  driver.manage().window().setSize(d); |

To set the window to a particular size, use **window.resizeTo()** method. Check the below piece of code:

|  |  |
| --- | --- |
|  | **((JavascriptExecutor)driver).executeScript("window.resizeTo(1024, 768);");** |

## ****21. How to handle a dropdown in Selenium WebDriver? How to select a value from dropdown?****

The most important detail you should know is that to work with a dropdown in Selenium, we must always make use of this html tag: **‘select’**. Without using ‘select’, we cannot handle dropdowns. Look at the snippet below in which I have written a code for a creating a dropdown with three options.

|  |  |
| --- | --- |
|  | <select id="mySelect">  <option value="option1">Cars</option>  <option value="option2">Bikes</option>  <option value="option3">Trains</option>  </select> |
|  |  |

|  |  |
| --- | --- |
|  | WebElement mySelectElement = driver.findElement(By.id("mySelect"));  Select dropdown = new Select(mySelectElement); |

Now to select an option from that dropdown, we can do it in either of the three ways:

1. dropdown.selectByVisibleText(“Bikes”); → Selecting an option by the text that is visible
2. dropdown.selectByIndex(“1”); → Selecting, by choosing the Index number of that option
3. dropdown.selectByValue(“option2”); → Selecting, by choosing the value of that option

## ****BootStrap Dropdown****

**public** **class** BootStrap {

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

FirefoxDriver driver = **new** FirefoxDriver();

driver.get("http://seleniumpractise.blogspot.in/2016/08/bootstrap-dropdown-example-for-selenium.html");

driver.findElement(By.xpath(".//\*[@id='menu1']")).click();

List<WebElement> list = driver.findElementsByXPath("//ul[@class='dropdown-menu']//li/a");

**for** (WebElement ele : list)

{

System.***out***.println("Values " + ele.getAttribute("innerHTML"));

**if** (ele.getAttribute("innerHTML").contains("JavaScript")) {

ele.click();

**break**;

}

## }

## ****22. How to switch to a new window (new tab) which opens up after you click on a link?****

If you click on a link in a web page, then for changing the WebDriver’s focus/ reference to the new window we need to use the **switchTo()** command. Look at the below example to switch to a new window:  
**driver.switchTo().window();**

Here, ‘windowName’ is the name of the window you want to switch your reference to.

In case you do not know the name of the window, then you can use the **driver.getWindowHandle()** command to get the name of all the windows that were initiated by the WebDriver. Note that it will not return the window names of browser windows which are not initiated by your WebDriver.

**public** **class** WindowHandle3 {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\chromedriver.exe");

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

driver.manage().window().maximize();

driver.get("https://www.naukri.com/");

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

String parentWindow = driver.getWindowHandle();

Set<String> s1 = driver.getWindowHandles();

**for** (String child : s1) {

**if** (!parentWindow.equals(child)) {

driver.switchTo().window(child);

driver.close();

}

}

ArrayList<String>list=**new** ArrayList<>(s1);

## list.get(1);

## }

## }

## The reason why we use Set in window handling is Set cannot contains duplicate but list can.

## ****23. How do you upload a file using Selenium WebDriver?****

To upload a file we can simply use the command **element.send\_keys(file path).**But there is a prerequisite before we upload the file. We have to use the html tag: **‘input’**and attribute type should be **‘file’**. Take a look at the below example where we are identifying the web element first and then uploading the file.

|  |  |
| --- | --- |
|  | <input type="file" name="uploaded\_file" size="50" class="pole\_plik">  element = driver.find\_element\_by\_id(”uploaded\_file")  element.send\_keys("C:\myfile.txt") |

## ****24. Can we enter text without using sendKeys()?****

Using DOM method of, identification of an element, we can go to that particular document and then get the element by its ID (here login) and then send the text by value. Look at the sample code below:

|  |  |
| --- | --- |
|  | **JavascriptExecutor jse = (JavascriptExecutor) driver;**  **jse.executeScript("document.getElementById(‘Login').value=Test text without sendkeys");** |

## ****25. Explain how**** ****you will login into any site if it is showing any authentication popup for username and password?****

Since there will be popup for logging in, we need to use the explicit command and verify if the alert is actually present. Only if the alert is present, we need to pass the username and password credentials. The sample code for using the explicit wait command and verifying the alert is below:

|  |  |
| --- | --- |
|  | **WebDriverWait wait = new WebDriverWait(driver, 10);**  **Alert alert = wait.until(ExpectedConditions.alertIsPresent());**  **alert.authenticateUsing(new UserAndPassword(\*\*username\*\*, \*\*password\*\*));** |

## ****26. Explain how can you find broken links in a page using Selenium WebDriver?****

This is a trick question which the interviewer will present to you. He can provide a situation where in there are 20 links in a web page, and we have to verify which of those 20 links are working and how many are not working (broken).

Since you need to verify the working of every link, the workaround is that, you need to send http requests to all of the links on the web page and analyze the response. Whenever you use driver.get() method to navigate to a URL, it will respond with a status of **200 – OK**. 200 – OK denotes that the link is working and it has been obtained. If any other status is obtained, then it is an indication that the link is broken.

But how will you do that?

First, we have to use the anchor tags <a> to determine the different hyperlinks on the web page.

For each <a> tag, we can use the attribute ‘href’ value to obtain the hyperlinks and then analyze the response received for each hyperlink when used in **driver.get()** method.

find broken links using selenium it means we need to check the link which is pointing to wrong URL or invalid URL.

 404 page not found an issue in most of the application which is called **broken link.**

**Approach 1-**

Manual Process- Go to each link and verify the link is working or not.

**Approach 2-**

Smart work- Write a code which will check all the link and will verify the status as well.

**public** **class** VerifyLinks {

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

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

driver.manage().window().maximize();

driver.get("http://www.google.co.in/");

List<WebElement> links = driver.findElements(By.tagName("a"));

System.***out***.println("Total links are " + links.size());

**for** (**int** i = 0; i < links.size(); i++) {

WebElement ele = links.get(i);

String url = ele.getAttribute("href");

*verifyLinkActive*(url);

}

}

**public** **static** **void** verifyLinkActive(String linkUrl) {

**try** {

URL url = **new** URL(linkUrl);

HttpURLConnection httpURLConnect = (HttpURLConnection) url.openConnection();

httpURLConnect.setConnectTimeout(3000);

httpURLConnect.connect();

**if** (httpURLConnect.getResponseCode() == 200) {

System.***out***.println(linkUrl + " - " + httpURLConnect.getResponseMessage());

}

**if** (httpURLConnect.getResponseCode() == HttpURLConnection.HTTP\_NOT\_FOUND) {

System.***out***.println(linkUrl + " - " + httpURLConnect.getResponseMessage() + " - "

+ HttpURLConnection.HTTP\_NOT\_FOUND);

}

} **catch** (Exception e) {

}

}

}

OR

**public** **class** BrokenLinksExample {

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

WebDriver driver;

String exePath = "D:\\chromedriver.exe";

System.*setProperty*("webdriver.chrome.driver", exePath);

driver = **new** ChromeDriver();

driver.manage().window().maximize();

driver.get("http://www.google.co.in/");

List<WebElement> links = driver.findElements(By.*tagName*("a"));

System.***out***.println("Total links are " + links.size());

**int** brokenLink = 0;

**for** (**int** i = 0; i < links.size(); i++) {

String url = links.get(i).getAttribute("href");

**if** (url != **null** && !url.contains("javascript")) {

System.***out***.println("Working link:" + url);

} **else** {

brokenLink++;

System.***out***.println(url);

}

}

System.***out***.println("Total no. of broken links are " + brokenLink);

}

}

**Broken Images**

**public** **class** BrokenImageExample {

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

WebDriver driver;

String exePath = "D:\\chromedriver.exe";

System.*setProperty*("webdriver.chrome.driver", exePath);

driver = **new** ChromeDriver();

driver.manage().window().maximize();

driver.get("http://ruchi-myseleniumblog.blogspot.in");

List<WebElement> images = driver.findElements(By.*tagName*("img"));

System.***out***.println("Total images are " + images.size());

**int** brokenImages = 0;

**for** (**int** i = 0; i < images.size(); i++) {

String url = images.get(i).getAttribute("src");

**if** (url != **null**) {

System.***out***.println("Working Images:" +url);

} **else** {

brokenImages++;

System.***out***.println(url);

}

}

System.***out***.println("Total no. of broken Images are " + brokenImages);

}

}

## ****27. Which technique should you consider using throughout the script “if there is neither frame id nor frame name”?****

If neither frame name nor frame id is available, then we can use **frame by index**.

Let’s say, that there are 3 frames in a web page and if none of them have frame name and frame id, then we can still select those frames by using frame (zero-based) index attribute. Each frame will have an index number. The first frame would be at index “0”, the second at index “1” and the third at index “2”. Once the frame has been selected, all subsequent calls on the WebDriver interface will be made to that frame.

|  |  |
| --- | --- |
|  | driver.switchTo().frame(int arg0); |

## ****28. What is the significance of testng.xml?****

Since Selenium does not support report generation and test case management, we use TestNG framework with Selenium. TestNG is much more advanced than JUnit, and it makes implementing annotations easy. That is the reason TestNG framework is used with Selenium WebDriver.

But have you wondered where to define the test suites and grouping of test classes in TestNG?

It is by taking instructions from the testng.xml file. We cannot define a test suite in testing source code, instead it is represented in an XML file, because suite is the feature of execution. The test suite, that I am talking about is basically a collection of test cases.

So for executing the test cases in a suite, i.e a group of test cases, you have to create a testng.xml file which contains the name of all the classes and methods that you want to execute as a part of that execution flow.

Other advantages of using testng.xml file are:

* It allows execution of multiple test cases from multiple classes
* It allows parallel execution
* It allows execution of test cases in groups, where a single test can belong to multiple groups

## ****29. What is parameterization in TestNG? How to pass parameters using testng.xml?****

Parameterization is the technique of defining values in testng.xml file and sending them as parameters to the test class. This technique is especially useful when we need to pass multiple login credentials of various test environments. Take a look at the code below, in which “myName” is annotated as a parameter.

**public** **class** ParameterizedTest1 {

@Test

@Parameters("myName")

**public** **void** parameterTest(String myName) {

System.***out***.println("Parameterized value is : " + myName);

}

}

To pass parameters using testng.xml file, we need to use ‘parameters’ tag. Look at the below code for example:

|  |  |
| --- | --- |
|  | <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "<http://testng.org/testng-1.0.dtd>" >   <suite name=”CustomSuite">    <test name=”CustomTest”>     <parameter name="myName" value=”John"/>      <classes>       <class name="ParameterizedTest1" />      </classes>    </test>   </suite> |

## ****30. Explain DataProviders in TestNG using an example. Can I call a single data provider method for multiple functions and classes?****

DataProvider is a TestNG feature, which enables us to write DataDriven tests. When we say, it supports DataDriven testing, then it becomes obvious that the same test method can run multiple times with different data-sets. DataProvider is in fact another way of passing parameters to the test method.

**@DataProvider** marks a method as supplying data for a test method. The annotated method must return an Object[] where each Object[] can be assigned to parameter list of the test method.

To use the DataProvider feature in your tests, you have to declare a method annotated by **@DataProvider** and then use the said method in the test method using the ‘dataProvider‘ attribute in the Test annotation.

As far as the second part of the question is concerned, Yes, the same DataProvider can be used in multiple functions and classes by declaring DataProvider in separate class and then reusing it in multiple classes.

## ****31. How to skip a method or a code block in TestNG?****

If you want to skip a particular test method, then you can set the ‘enabled’ parameter in test annotation to false.  
@Test(enabled = false)

By default, the value of ‘enabled’ parameter will be true. Hence it is not necessary to define the annotation as true while defining it.

## ****32. What is soft assertion in Selenium? How can you mark a test case as failed by using soft assertion?****

Soft Assertions are customized error handlers provided by TestNG. Soft Assertions do not throw exceptions when assertion fails, and they simply continue to the next test step. They are commonly used when we want to perform multiple assertions.

To mark a test as failed with soft assertions, call **assertAll()** method at the end of the test.

## ****33. Explain what is Group Test in TestNG?****

In TestNG, methods can be categorized into groups. When a particular group is being executed, all the methods in that group will be executed.  We can execute a group by parameterizing it’s name in group attribute of**@Test** annotation. Example: @Test(groups={“xxx”})

|  |  |
| --- | --- |
|  | @Test(groups={“Car”})  public void drive(){  system.out.println(“Driving the vehicle”);  }  @Test(groups={“Car”})  public void changeGear() {  system.out.println("Change Gears”);  }  @Test(groups={“Car”})  public void accelerate(){  system.out.println(“Accelerating”);  } |

## ****34. How does TestNG allow you to state dependencies? Explain it with an example.****

**Dependency**is a feature in TestNG that allows a test method to depend on a single or a group of test methods. Method dependency only works if the “depend-on-method” is part of the same class or any of the inherited base classes (i.e. while extending a class). Syntax:  
**@Test(dependsOnMethods = { “initEnvironmentTest” })**

|  |  |
| --- | --- |
|  | @Test(groups={“Car”})  public void drive(){  system.out.println(“Driving the vehicle”);  }    @Test(dependsOnMethods={“drive”},groups={cars})  public void changeGear() {  system.out.println("Change Gears”);  }  @Test(dependsOnMethods={“changeGear”},groups={“Car”})  public void accelerate(){  system.out.println(“Accelerating”);  } |

## ****35. Explain what does @Test(invocationCount=?) and @Test(threadPoolSize=?) indicate.****

**@Test(invocationCount=?)** is a parameter that indicates the number of times this method should be invoked.  
**@Test(threadPoolSize=?)** is used for executing suites in parallel. Each suite can be run in a separate thread.

To specify how many times @Test method should be invoked from different threads, you can use the attribute**threadPoolSize** along with **invocationCount**. Example:

|  |  |
| --- | --- |
|  | @Test(threadPoolSize = 3, invocationCount = 10)  public void testServer() { |

}