Selenium Documents

Absolute xpath = complete address of that particular element from parent node to child node - Not Recommended -Performance issue, not reliable, can be change any time

Example - div/table/tbody/tr/td/a

Relative xpath =

//div[@class='dropdown']//button[@type='button' and @class='btn btn-secondary dropdown-toggle' and @id='dropdownMenuButton']

++preceding-sibling:

//a[text()='test2 test2']//parent::td[@class='datalistrow']//preceding-sibling::td[@class='datalistrow']//input

++parent & preceding-sibling:

//a[text()='test2 test2']//parent::td[@class='datalistrow']//preceding-sibling::td[@class='datalistrow']//input[@name='contact\_id']

Handle Dynamic Search Using Dynamic XPath - Google Search:

Steps:

1. Google Search – pick a value from suggestions

2. Search using Xpath

3. Get the total count of suggestions

4. Use if condition to match the value

5. Click on matched value

double slace (//) - means total number of child elements

descendant - number of childs(child of child, child of child....)

//ul[@role='rolelist']//li/descendant::div[text()='testing'] - return the list than iterate using for loop

How to handle dynamic WebTable In Selenium

There are two ways of handling WebTable:

Method – 1:

• Iterate row and column and get the cell value.

• Using for loop

• Get total rows and iterate table

• Put if(string matches) then select the respective checkbox

• Lengthy method

Method – 2:

• Using custom XPath

• Using parent and preceding-sibling tags

• No need to write for loop

• No full iteration of table

• Single line statement

• More dynamic

• Efficient and fast

Steps to handle calendar in Selenium Webdriver:

1- Click on calendar

2- Get all td of tables using findElements method?

3- using for loop get text of all elements

4- Using if else condition we will check specific date

5- If date is matched then click and break the loop.

6- Handle NoSuchElementException in case of (31st day)

Handle Calendar/Date Picker using JavaScriptExecutor:

+Use JavaScriptExecutor to inject JavaScript into DatePicker/Calendar field.

+Very Fast approach

+Direct injection of date into DOM - HTML

Selenium cannot change any HTML Dom structure but JavaScript can do

public static void selectDateByJS(WebElement element, WebDriver driver, String dateValue)

{

JavascriptExecutor js = ((JavascriptExecutor) driver);

js.executeScript("arguments[0].setAttribute('value','"+dateValue+"');", element);

}

What is HTML DOM - Document Object Model? Why we get StaleElement Exception in Selenium.

• DOM is an API Interface provided by browser (kind of platform and provided by specific browser)

• When a web page is loaded, browser creates a DOM of the page.

DOM is divided page wise

Although most of the DOM structure will be same in all the browser but these are provided by specific browser

With the document object model, JavaScript gets all the power it needs to create dynamic HTML:

• JavaScript can change all the HTML elements in the page

• JavaScript can change all the HTML attributes in the page

• JavaScript can change all the CSS styles in the page

• JavaScript can remove existing HTML elements and attributes

• JavaScript can add new HTML elements and attributes

• JavaScript can react to all existing HTML events in the page

• JavaScript can create new HTML events in the page

Selenium- Jason Huggins - Thoughtwork - 2004

WebDriver- Simon Stewart- Google - 2006

Selenium WebDriver - 2008

Selenium - have 4 flavour/tool- IDE, RC, WebDriver, and Grid

Because of the JavaScript injection to the browser and because of security reason RC is deprecated officially, most of the browser are not allowing JS injection to their sites, So WebDriver came into picture.

Selenium RC - API

Selenium WebDriver - API, well-designed object-oriented API, and Selenium-WebDriver makes direct calls to the browser using each browser’s native support for automation. Again, it drives the browser directly using the browser’s built in support for automation.

Selenium 2(WebDriver) still runs Selenium 1’s (Selenium RC) interface for backwards compatibility.

Selenium IDE - (Integrated Development Environment) is a prototyping tool for building test scripts. It is a Firefox plugin and provides an easy-to-use interface for developing automated tests. Selenium IDE has a recording feature, which records user actions as they are performed and then exports them as a reusable script in one of many programming languages that can be later executed.

Even though Selenium IDE has a “Save” feature that allows users to keep the tests in a table-based format for later import and execution, it is not designed to run your test passes nor is it designed to build all the automated tests you will need. Specifically, Selenium IDE doesn’t provide iteration or conditional statements for test scripts

Selenium Grid- Selenium Grid allows you to run your tests in parallel, that is, different tests can be run at the same time on different remote machines. Improve the Performance and reduce the Time for execution.

WebDriver and the Selenium-Server

You may, or may not, need the Selenium Server, depending on how you intend to use Selenium-WebDriver. If your browser and tests will all run on the same machine, and your tests only use the WebDriver API, then you do not need to run the Selenium-Server; WebDriver will run the browser directly.

There are some reasons though to use the Selenium-Server with Selenium-WebDriver.

You are using Selenium-Grid to distribute your tests over multiple machines or virtual machines (VMs).

You want to connect to a remote machine that has a particular browser version that is not on your current machine.

You are not using the Java bindings (i.e. Python, C#, or Ruby) and would like to use HtmlUnit Driver

HtmlUnit Driver

This is currently the fastest and most lightweight implementation of WebDriver. As the name suggests, this is based on HtmlUnit. HtmlUnit is a java based implementation of a Web Browser without a GUI. For any language binding (other than java) the Selenium Server is required to use this driver.

WARNING: Do not mix implicit and explicit waits. Doing so can cause unpredictable wait times .

For example setting an implicit wait of 10 seconds and an explicit wait of 15 seconds, could cause a timeout to occur after 20 seconds.WebDriverWait by default calls the ExpectedCondition every 500 milliseconds until it returns successfully. A successful return value for the ExpectedCondition function type is a Boolean value of true, or a non-null object.

Implicit Waits

An implicit wait is to tell WebDriver to poll the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available. The default setting is 0.

Once set, the implicit wait is set for the life of the WebDriver object instance.

When you are working with Chrome Browser or IE browser you need to set the property otherwise will get illegalStateException

public static void main(String[] args)

{ System.setProperty("webdriver.chrome.driver","E:\\Selenium\_Driver\\chromedriver.exe");

WebDriver driver=new ChromeDriver ();

driver.get("https://www.ebay.in/");

}

--------------------------------------------------------------------------------------------------------------------------------------------

When you are working with IE Browser you need to set property otherwise will get illegalStateException

public static void main(String[] args) {

System.setProperty("webdriver.ie.driver", "E:\\IEDriverServer.exe");

WebDriver driver=new InternetExplorerDriver();

driver.get("https://www.ebay.in/");

}

and when working with IE browser -

-Zoom level should be=100% else SessionNotFoundException will appear

-Protected mode should be same for all -otherwise SessionNotFoundException will appear

-xpath changes in IE

-sendkeys performance (64 bit version is worst for IE don’t use- it will type character very slow-sendkeys performance will worst for it)

For the gecko driver also we need to set the property like IE and chrome (working with selenium 3 - than we need gecko driver else not needed)

---------------------------------------------------------------------------------------------------------------------------------

How to solve not connected exception- only will get in Firefox (stable -means properly tested and 2nd is Beta-still testing is going)

For avoid this issue-

-use selenium latest JARS

-use Firefox stable version

-----------------------------------------------------------------------------------------------------------------------------------

Whenever website is Dynamic- go with XPATH or CSS

Difference b/w CSS and XPATH

-CSS is faster and the reason is in XPATH-XPATH engine is evaluate your XPATH and locate element but CSS does not have any engine and it directly identify an object

-CSS will not change with Browser but XPATH may change in IE

-----------------------------------------------------------------------------------------------------------------------------------

How can we find CSS?

1. tagname[attribute='value']

2. tagname[attribute='value'][attribute2='value2']

3. #id

4. .classname -if spaces is present in classname than remove those spaces and give dot on that places

While working with dynamic element

Example-

id=customer365User

Next time

id=customer324User

startwith and endwith plays very important role

5. input[id^='customer'][name='log'] - startwith example

6. input[id$='Login'] - endwith example

7. input[id\*='Email'] - contains example

-----------------------------------------------------------------------------------------------------------------------------------

How to write XPATH

1. Using single attribute

// tagname[@attribute=’value1’]

Example

// a [@href=’http://www.google.com’]

//input[@id=’name’ OR @class='abc']

//input[@name=’username’ AND @class='abc']

//img[@alt=’sometext’]

2.Using multiple attribute

//tagname[@attribute1=’value1’][attribute2=’value2’]

//a[@id=’id1’][@name=’namevalue1’]

//img[@src=’’][@href=’’]

3.Using contains method

//tagname[contains(@attribute,’value1’)]

//input[contains(@id,’’)]

//input[contains(@name,’’)]

//a[contains(@href,’’)][text()='abc']

//img[contains(@src,’’)]

//div[contains(@id,’’)]

4.Using starts-with method

//tagname[starts-with(@attribute-name,’’)]

//id[starts-with(@id,’’)]

//a[starts-with(@href=’’)]

//img[starts-with(@src=’’)]

//div[starts-with(@id=’’)]

//input[starts-with(@id=’’)]

//button[starts-with(@id,’’)]

5.Using Following node

Xpath/following::again-ur-regular-path

//input[@id=’’]/following::input[1]

//a[@href=’’]/following::a[1]

//img[@src=’’]/following::img[1]

6. Using preceding node

Xpath/preceding::again-ur-regular-path

//input[@id=’’]/ preceding::input[1]

//a[@href=’’]/ preceding::a[1]

//img[@src=’’]/ preceding::img[1]

7. Absolute XPath method

/html/head/body/div/input

8. Relative and Absolute XPath method

//parent-xpath/absolute xpath

//input[@id=’section’]/div/input

----------------------------------------------------------------------------------------------------------------------------------

If 1 XPATH matches with 2 or more than 2 node it will always perform operation on 1st matching node only

----------------------------------------------------------------------------------------------------------------------------------

When dealing with Bootstrap Login/popup/window (Flipcart login or redbus login etc.)- remember most of the time window will come under Frame so 1st we need to switch to Frame and then we can perform operations and if it is normal window than we can directly perform operations.

After switching, it is now a separate window /alert-you can directly perform any operations

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

WebDriver driver = new FirefoxDriver();

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

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

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

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

driver.switchTo().frame("authiframe");

driver.findElement(By.xpath(".//\*[@id='authMobile']")).sendKeys("99999999999");

}

------------------------------------------------------------------------------------------------------------------------------------

Bootstrap Dropdown- The bootstrap dropdown is enhanced part of dropdown where you will deal with UL and LI tag of HTML.

To handle this kind of drop-down we have to use findElements method and then we can run a for loop to get specific elements.

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

WebDriver driver = new FirefoxDriver();

driver.get("https://www.ebay.in/");

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

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

driver.findElement(By.xpath(".//\*[@id='gh-shop-a']")).click();

List<WebElement> dd\_menu=driver.findElements(By.xpath(".//\*[@id='gh-sbc']/tbody/tr/td/ul/li/a"));

System.out.println(dd\_menu.size());

System.out.println("----------------------------------------");

for (int i = 0; i < dd\_menu.size(); i++) {

WebElement element=dd\_menu.get(i);

String innerHTML=element.getAttribute("innerHTML");

if (innerHTML.contentEquals("Digital Cameras")) {

element.click();

break;

}

System.out.println(innerHTML);

}

}

-----------------------------------------------------------------------------------------------------------------------------------

Dropdown- print all the values from dropdown

WebElement element = driver.findElement(By.xpath(".//\*[@id='gh-cat']"));

Thread.sleep(2000);

Select select = new Select(element);

List<WebElement> list = select.getOptions();

System.out.println(list.size());

System.out.println("-------------------------------------------");

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

System.out.println(list.get(i).getText());

}

--------------------------------------------------------------------------------------------------------------------------------------------

Alert - Alert is an Interface

will use switchTo() method for- Alert/Frame/windowHandle

Note– alert is separate window so before using accept() or dismiss() methods we have to switch to alert window using switchTo() method

Example:-

driver.findElement(By.xpath(".//\*[@id='Table\_3']/tbody/tr[1]/td[2]/div/a")).click();

// Switch to alert window and capture the text and print

System.out.println(driver.switchTo().alert().getText());

// Pause testcase for 5 second

Thread.sleep(5000);

// click on ok button

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

// Close browser

driver.quit();

}

---------------------------------------------------------

Now consider a scenario where alert window comes when certain condition true for this we can create method which will check if alert window present then only it will execute otherwise it will skip this part

public static void handleAlert(WebDriver ldriver){

if(isAlertPresent(ldriver)){

Alert alert = ldriver.switchTo().alert();

System.out.println(alert.getText());

alert.accept();

}

}

Important point- If alert in not present in the window and still we try to switchTo alert window then Selenium will throw NoAlertPresentException which will terminate your program so better you should use exception handle also in your script.

public static boolean isAlertPresent(WebDriver ldriver){

try{

ldriver.switchTo().alert();

return true;

}catch(NoAlertPresentException ex){

return false;

}

}

--------------------------------------------------------------------------------------------------------------------------------------------

Mouse hover- print all the links

WebElement element=driver.findElement(By.xpath(".//\*[@id='nav-link-accountList']"));

Actions act=new Actions(driver);

act.moveToElement(element).build().perform();

//XPATH is very very important for this type of problem

List<WebElement>links=driver.findElements(By.xpath(".//\*[@id='nav-flyout-accountList']//a"));

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

System.out.println(links.get(i).getText());

}

--------------------------------------------------------------------------------------------------------------------------------------------

Element is not Clickable Exception-

-Mostly will get in Chrome and in Safari browser

-Try with diffrent XPATH

-mouse hover (best solution- click on middle of the element)

--------------------------------------------------------------------------------------------------------------------------------------------

Frames-

By default selenium will focus on parent window

-you need to perform any operation on Frame than you need to switchTo that frame first and come back to parent window

how to identify Frames ?

ans- Source code (iframe/ frameset)

3 ways to switch to frame-index/id/webelement

if we don’t have any id to find iframe- than so many attribute will be there to find (you will get iframe id or attribute in HTML tag)

in Mozilla- you can directly check there is frame or not but in Chrome/IE you need to find manually

how to handle ?

ans- switch to frame-perform operation-switch back to parent window

how to find no of frames ?

ans-driver.findelements(By.tagname("iframe")).size();

driver.switchTo().frame(indexnumber);

driver.switchTo().frame(“framename”);

driver.switchTo().defaultContent();

Now if you want to find with the help of webelement

try {

WebElement button=driver.findElement(By.xpath(""));

driver.switchTo().frame(button);

}

catch (NoSuchFrameException e)

{

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

}

example-:

int total\_frame=driver.findElements(By.tagName("iframe")).size();

System.out.println(total\_frame);

WebElement my\_frame=driver.findElement(By.xpath("//iframe[@title='selenium\_news']"));

driver.switchTo().frame(my\_frame);

perform operation on frame now and than come back to parent window

--------------------------------------------------------------------------------------------------------------------------------------------

How to Scroll into view in Selenium Webdriver ?

Ans- Selenium does not have inbuilt method which allow us to scroll into view but we can scroll into view in Selenium using JavaScript executor.

public static void main(String[] args) {

// Start browser

WebDriver driver = new FirefoxDriver();

// Maximize browser

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

// Pass application URL

driver.get("http://manos.malihu.gr/repository/custom-scrollbar/demo/examples/complete\_examples.html");

// Create instance of Javascript executor

JavascriptExecutor je = (JavascriptExecutor) driver;

// identify the WebElement which will appear after scrolling down

WebElement element = driver.findElement(By.xpath(".//\*[@id='mCSB\_3\_container']/p[3]"));

// now execute query which actually will scroll until that element is not appeared on page.

je.executeScript("arguments[0].scrollIntoView(true);", element);

// Extract the text and verify

System.out.println(element.getText());

}

What is JavaScript

JavaScript is one of the programming languages of the Web.

JavascriptExecutor is an Interface which is available in package org.openqa.selenium.JavascriptExecutor;

Inside this Interface we have some predefined method called executeScript()- so whatever script you will pass as a String It will be executed by JavascriptExecutor.

Note- This is the most common question in the interview that how to type in Selenium without using the sendKeys method.

Program 1- How to type in Selenium without using sendKeys() method

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.firefox.FirefoxDriver;

public class TestFirefox {

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

// Open Firefox browser

FirefoxDriver driver=new FirefoxDriver();

// Maximize the window

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

// Open applicatiion

driver.get("enter your application URL");

// This will execute JavaScript in your script

((JavascriptExecutor)driver).executeScript("document.getElementById('some id').value='mukesh';");

}

}

Program 2- How to click in Selenium if button or radio button is disable

package seleniumday1;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.firefox.FirefoxDriver;

public class TestFirefox {

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

// Open Firefox browser

FirefoxDriver driver=new FirefoxDriver();

// Maximize the window

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

// Open applicatiion

driver.get("enter your application URL");

// This will execute JavaScript in your script

((JavascriptExecutor)driver).executeScript("document.getElementById('enter your element id').click();");

}

}

Program 3- How to uncheck checkbox in Selenium if checkbox is disable

package seleniumday1;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.firefox.FirefoxDriver;

public class TestFirefox {

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

// Open Firefox browser

FirefoxDriver driver=new FirefoxDriver();

// Maximize the window

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

// Open applicatiion

driver.get("enter your application URL");

// This will execute JavaScript in your script

((JavascriptExecutor)driver).executeScript("document.getElementById('enter element id').checked=false;");

}

}

--------------------------------------------------------------------------------------------------------------------------------------------

How to Scroll Page in Selenium Webdriver using Java Script ?

Selenium handle scrolling page automatically but if want to scroll page using Selenium then yes we can do easily using JavaScript.

We have method scroll(horizontal, vertical) i.e. scroll(0,400)

Note- scroll method is not a method of Webdriver, this is a method of JavaScript.

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

// load browser

WebDriver driver=new FirefoxDriver();

// maximize browser

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

// Open Application

driver.get("http://jqueryui.com");

// Wait for 5 second

Thread.sleep(5000);

// This will scroll page 400 pixel vertical

((JavascriptExecutor)driver).executeScript("scroll(0,400)");

}

--------------------------------------------------------------------------------------------------------------------------------------------

How to highlight elements Selenium Webdriver using JavaScript ?

public class aa {

public static void main(String []args){

WebDriver driver=new FirefoxDriver();

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

driver.get("http://www.facebook.com");

// Inspect element

WebElement username= driver.findElement(By.id("email"));

// Call reuse method

highLightElement(driver,username);

}

// Element highlighter code

public static void highLightElement(WebDriver driver, WebElement element)

{

JavascriptExecutor js=(JavascriptExecutor)driver;

js.executeScript("arguments[0].setAttribute('style', 'background: yellow; border: 2px solid red;');", element);

try

{

Thread.sleep(1000);

}

catch (InterruptedException e) {

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

}

js.executeScript("arguments[0].setAttribute('style','border: solid 2px white');", element);

}

}

--------------------------------------------------------------------------------------------------------------------------------------------

Robot class is not part of Selenium it comes with Java but we can use the same here.We have to use some keyboard event to perform this.

Step 1- We have to copy the file location in system clipboard.

Step 2- We have to click on upload button and use CTR+V and ENTER.

Note- Robot class each key has to press and release respectively

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

// Start browser

WebDriver driver = new FirefoxDriver();

// maximize browser

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

// Specify the file location with extension

StringSelection sel = new StringSelection("C:\\Users\\rahul goru\\Desktop\\TestNG.docx");

// Copy to clipboard

Toolkit.getDefaultToolkit().getSystemClipboard().setContents(sel, null);

System.out.println("selection" + sel);

// Open Monster.com

driver.get("http://my.monsterindia.com/create\_account.html");

Thread.sleep(2000);

// This will scroll down the page

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("scroll(0,350)");

// Wait for 5 seconds

Thread.sleep(5000);

// This will click on Browse button

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

System.out.println("Browse button clicked");

// Create object of Robot class

Robot robot = new Robot();

Thread.sleep(1000);

// Press Enter

robot.keyPress(KeyEvent.VK\_ENTER);

// Release Enter

robot.keyRelease(KeyEvent.VK\_ENTER);

// Press CTRL+V

robot.keyPress(KeyEvent.VK\_CONTROL);

robot.keyPress(KeyEvent.VK\_V);

// Release CTRL+V

robot.keyRelease(KeyEvent.VK\_CONTROL);

robot.keyRelease(KeyEvent.VK\_V);

Thread.sleep(1000);

System.out.println("-------------");

// Press Enter

robot.keyPress(KeyEvent.VK\_ENTER);

robot.keyRelease(KeyEvent.VK\_ENTER);

System.out.println("-----------------------------------------------");

}

--------------------------------------------------------------------------------------------------------------------------------------------

Program to handle calendar in Selenium Webdriver

public static void main(String[] args) {

WebDriver driver = new FirefoxDriver();

driver.get("http://seleniumpractise.blogspot.in/2016/08/how-to-handle-calendar-in-selenium.html");

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

List<WebElement> allDates = driver.findElements(By

.xpath("//table[@class='ui-datepicker-calendar']//td"));

for (WebElement ele : allDates) {

String date = ele.getText();

if (date.equalsIgnoreCase("28")) {

ele.click();

break;

}

}

}

--------------------------------------------------------------------------------------------------------------------------------------------

1. What is Automation Testing?

Automation testing is the process of testing the software using an automation tool to find the defects. In this process, executing the test scripts and generating the results are performed automatically by automation tools. Some most popular tools to do automation testing are HP QTP/UFT, Selenium WebDriver, etc.,

2. What are the benefits of Automation Testing?

• Saves time and money. Automation testing is faster in execution.

• Reusability of code. Create one time and execute multiple times with less or no maintenance.

• Easy reporting. It generates automatic reports after test execution.

• Easy for compatibility testing. It enables parallel execution in the combination of different OS and browser environments.

• Low-cost maintenance. It is cheaper compared to manual testing in a long run.

• Automated testing is more reliable.

• Automated testing is more powerful and versatile.

• It is mostly used for regression testing. Supports execution of repeated test cases.

• Minimal manual intervention. Test scripts can be run unattended.

• Maximum coverage. It helps to increase the test coverage.

3. What type of tests have you automated?

Our main focus is to automate test cases to do Regression testing, Smoke testing, and Sanity testing. Sometimes based on the project and the test time estimation, we do focus on End to End testing.

If same test case needs to execute with multiple data sets, test case needs to execute in multiple browser/OS/Environments than these are good candidates for automation.

4. How many test cases you have automated per day?

It depends on Test case scenario complexity and length. I automate 2-5 test cases per day when the complexity is limited. Sometimes just 1 or fewer test scenarios in a day when the complexity is high.

5. What is a Framework?

A framework defines a set of rules or best practices which we can follow in a systematic way to achieve the desired results. There are different types of automation frameworks and the most common ones are:

• Data Driven Testing Framework

• Keyword Driven Testing Framework

• Hybrid Testing Framework

6. Have you created any Framework?

If you are a beginner: No, I didn’t get a chance to create a framework. I have used the framework which is already available.

If you are an experienced tester: Yes, I have created a framework. Or I have involved in the creation of the framework.

7. Can you explain the Framework which you have used in your Selenium Project?

Here we have clearly explained each component of Framework.

8. Why do you prefer Selenium Automation Tool?

• Free and open source

• Have large user base and helping communities

• Cross browser compatibility

• Platform compatibility

• Multiple programming languages support

9. What is Selenium?

Selenium is an open source (free) automated testing suite to test web applications. It supports different platforms and browsers. It has gained a lot of popularity in terms of web-based automated testing.

Selenium is a set of different software tools. Each tool has a different approach in supporting web based automation testing.

It has four components namely,

i Selenium IDE (Integrated Development Environment)

ii Selenium RC (Remote Control) – selenium 1

iii Selenium WebDriver – selenium 2 & 3

iv Selenium Grid

10. What is Selenium IDE?

Selenium IDE (Integrated Development Environment) is a Firefox plugin. It is the simplest framework in the Selenium Suite. It allows us to record and playback the scripts. Even though we can create scripts using Selenium IDE, we need to use Selenium RC or Selenium WebDriver to write more advanced and robust test cases.

11. What is Selenese?

Selenese is the language which is used to write test scripts in Selenium IDE.

12. Which is the only browser that supports Selenium IDE to be used?

Firefox

13. What is Selenium RC?

Selenium RC or Selenium 1. Selenium RC was the main Selenium project for a long time before the WebDriver merge brought up Selenium 2. Selenium 1 is still actively supported (in maintenance mode). It relies on JavaScript for automation. It supports Java, Javascript, Ruby, PHP, Python, Perl and C#. It supports almost every browser out there.

14. What is Selenium WebDriver?

Selenium WebDriver or Selenium 2 is a browser automation framework that accepts commands and sends them to a browser. It is implemented through a browser-specific driver. It controls the browser by directly communicating with it. Selenium WebDriver supports Java, C#, PHP, Python, Perl, Ruby.

15. What is Selenium Grid?

Selenium Grid is a tool to run tests on different machines against different browsers in parallel. That is, running multiple tests at the same time against different machines running different browsers and operating systems.

In simple words, it is used to distribute your test execution on multiple platforms and environments concurrently.

16. When do you use Selenium Grid?

Selenium Grid can be used to execute same or different test scripts on multiple platforms and browsers concurrently so as to achieve distributed test execution

17. What are the advantages of Selenium Grid?

It allows running test cases in parallel thereby saving test execution time.

It allows multi-browser testing

It allows us to execute test cases on multi-platform

18. What is a hub in Selenium Grid?

A hub is a server or a central point that controls the test executions on different machines.

19. What is a node in Selenium Grid?

Node is the machine which is attached to the hub. There can be multiple nodes in Selenium Grid.

20. What are the types of WebDriver APIs available in Selenium?

• Firefox Driver

• Gecko Driver

• InternetExplorer Driver

• Chrome Driver

• HTMLUNIT Driver

• Opera Driver

• Safari Driver

• Android Driver

• iPhone Driver

• EventFiringWebDriver

21. Which WebDriver implementation claims to be the fastest?

The fastest implementation of WebDriver is the HTMLUnitDriver. It is because the HTMLUnitDriver does not execute tests in the browser.

22. What are the Programming Languages supported by Selenium WebDiver?

• Java

• C#

• Python

• Ruby

• Perl

• PHP

23. What are the Operating Systems supported by Selenium WebDriver?

• Windows

• Linux

• Apple

24. What are the Open-source Frameworks supported by Selenium WebDriver?

• JUnit

• TestNG

• CUCUMBER

• JBEHAVE

25. What are the Locators available in Selenium?

Different types of locators are:

1 ID

2 ClassName

3 Name

4 TagName – a, input, img,iframe,table

5 LinkText

6 PartialLinkText

7 XPath

8 CSS Selector

26. What is XPath?

XPath is used to locate the elements. Using XPath, we could navigate through elements and attributes in an XML document to locate web elements such as textbox, button, checkbox, Image etc. in a web page.

27. What is the difference between “/” and “//”

Single Slash “/” – Single slash is used to create absolute path i.e. the XPath would be created to start selection from the document node/start node.

Double Slash “//” – Double slash is used to create relative (Dynamic ) path i.e. the XPath would be created to start selection from anywhere within the document.

28. What is the difference between Absolute Path and Relative Path?

Absolute XPath starts from the root node and ends with desired descendant element’s node. It starts with top HTML node and ends with input node. It starts with a single forward slash (/) as shown below.

/html/body/div[3]/div[1]/form/table/tbody/tr[1]/td/input

Relative XPath starts from any node in between the HTML page to the current element’s node (last node of the element). It starts with a single forward slash (//) as shown below.

//input[@id='email']

29. What is the difference between Assert and Verify in Selenium?

Assert: In simple words, if the assert condition is true then the program control will execute the next test step but if the condition is false, the execution will stop and further test step will not be executed.

Verify: In simple words, there won’t be any halt in the test execution even though the verify condition is true or false.

30. What are Soft Assert and Hard Assert in Selenium?

Soft Assert: Soft Assert collects errors during @Test Soft Assert does not throw an exception when an assert fails and would continue with the next step after the assert statement.

Hard Assert: Hard Assert throws an AssertException immediately when an assert statement fails and test suite continues with next @Test

31. What are the verification points available in Selenium?

In Selenium IDE, we use Selenese Verify and Assert Commands as Verification points

In Selenium WebDriver, there is no built-in features for verification points. It totally depends on our coding style. Some of the Verification points are

To check for page title

To check for certain text

To check for certain element (text box, button, drop down, logo, images etc.)

32. How to launch a browser using Selenium WebDriver?

WebDriver is an Interface. We create Object of a ChromeDriver/FirefoxDriver Class which internally implements WebDriver, because we can’t create object of WebDriver.

Version < 2.53 – no geckodriver

Version > 3.x – geckodriver for FF

To launch Firefox Driver: WebDriver driver = new FirefoxDriver();

To launch Chrome Driver: WebDriver driver = new ChromeDriver();

To launch Internet Explorer Driver: WebDriver driver = new InternetExplorerDriver();

33. Is the FirefoxDriver a Class or an Interface?

FirefoxDriver is a Java class, and it implements the WebDriver interface.

34. What is the super interface of WebDriver?

SearchContext.

35. Explain the line of code Webdriver driver = new FirefoxDriver(); ?

‘WebDriver‘ is an interface and we are creating an object reference of type WebDriver instantiating an object of FirefoxDriver class.

36. We do create a reference variable ‘driver’ of type WebDriver

WebDriver driver = new FirefoxDriver();

instead of creating

FirefoxDriver driver = new FirefoxDriver();

What is the purpose of doing this way?

If we create a reference variable driver of type WebDriver then we could use the same driver variable to work with any browser of our choice such as IEDriver, SafariDriver etc.,

37. What are the different exceptions you have faced in Selenium WebDriver?

• WebDriverException

• TimeoutException

• NoAlertPresentException

• NoSuchWindowException

• NoSuchElementException

• StaleElementReferenceException

• IllegalStateException

38. How to Login Into Any Site If It Is Showing Any Authentication Pop-Up For Username And Password?

To do this we pass username and password with the URL

http://username:password@url

e.g. http://admin:admin123@xyz.com

39. What are the types of waits available in Selenium WebDriver?

In Selenium we could see three types of waits such as Implicit Waits, Explicit Waits and Fluent Waits.

• Implicit Waits

• Explicit Waits

• Fluent Waits

• PageLoadTimeOut

• Thread.sleep() – static wait

40. What is Implicit Wait in Selenium WebDriver?

Implicit waits tell to the WebDriver to wait for a certain amount of time before it throws an exception. Once we set the time, WebDriver will wait for the element based on the time we set before it throws an exception. The default setting is 0 (zero). We need to set some wait time to make WebDriver to wait for the required time.

41. What is WebDriverWait in Selenium WebDriver?

WebDriverWait is applied on a certain element with defined expected condition and time. This wait is only applied to the specified element. This wait can also throw an exception when an element is not found.

42. What is Fluent Wait in Selenium WebDriver?

FluentWait can define the maximum amount of time to wait for a specific condition and frequency (polling time) with which to check the condition before throwing an “ElementNotVisibleException” exception.

43. How to input text in the text box using Selenium WebDriver?

By using sendKeys() method

WebDriver driver = new FirefoxDriver();

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

driver.findElement(By.xpath("xpath")).sendKeys("test");

44. How to input text in the text box without calling the sendKeys()?

JavascriptExecutor JS = (JavascriptExecutor)driver;

JS.executeScript("document.getElementById(‘User').value=test.com'");

45. How to clear the text in the text box using Selenium WebDriver?

By using clear() method

WebDriver driver = new FirefoxDriver();

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

driver.findElement(By.xpath("xpath\_of\_element1")).sendKeys("Software Testing Material Website");

driver.findElement(By.xpath("xpath\_of\_element1")).clear();

46. How to get a text of a web element?

By using getText() method

47. How to get an attribute value using Selenium WebDriver?

By using getAttribute(value);

48. How to click on a hyperlink using Selenium WebDriver?

We use click() method in Selenium to click on the hyperlink

driver.findElement(By.linkText(“Software Testing Material Website”)).click();

49. How to submit a form using Selenium WebDriver?

We use “submit” method on element to submit a form

driver.findElement(By.id("form\_1")).submit();

Alternatively, you can use click method on the element which does form submission

50. How to press ENTER key on text box In Selenium WebDriver?

To press ENTER key using Selenium WebDriver, We need to use Selenium Enum Keys with its constant ENTER.

driver.findElement(By.xpath("xpath")).sendKeys(Keys.ENTER);

51. How to pause a test execution for 5 seconds at a specific point?

By using java.lang.Thread.sleep(long milliseconds) method we could pause the execution for a specific time. To pause 5 seconds, we need to pass parameter as 5000 (5 seconds)

Thread.sleep(5000)

52. Is Selenium Server needed to run Selenium WebDriver Scripts?

When we are distributing our Selenium WebDriver scripts to execute using Selenium Grid, we need to use Selenium Server.

53. What happens if I run this command - driver.get(“www.softwaretestingmaterial.com”) ;

An exception is thrown(fquery is not a function). We need to pass HTTP protocol within driver.get() method.

driver.get("http://www.softwaretestingmaterial.com");

54. What is the alternative to driver.get() method to open an URL using Selenium WebDriver?

Alternative method to driver.get(“url”) method is driver.navigate.to(“url”)

55. What is the difference between driver.get() and driver.navigate.to(“url”)?

driver.get(): To open an URL and it will wait till the whole page gets loaded

driver.navigate.get(): To navigate to an URL and It will not wait till the whole page gets loaded

56. Can I navigate back and forth in a browser in Selenium WebDriver?

We use Navigate interface to do navigate back and forth in a browser. It has methods to move back, forward as well as to refresh a page.

driver.navigate().forward(); – to navigate to the next web page with reference to the browser’s history

driver.navigate().back(); – takes back to the previous webpage with reference to the browser’s history

driver.navigate().refresh(); – to refresh the current web page thereby reloading all the web elements

driver.navigate().to(“url”); – to launch a new web browser window and navigate to the specified URL

57. What are the different types of navigation commands?

Refer above question (Can I navigate back and forth in a browser)

58. How to fetch the current page URL in Selenium?

To fetch the current page URL, we use getCurrentURL()

driver.getCurrentUrl();

59. How can we maximize browser window in Selenium?

To maximize browser window in selenium we use maximize() method. This method maximizes the current window if it is not already maximized

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

60. How to delete cookies in Selenium?

To delete cookies we use deleteAllCookies() method

driver.manage().deleteAllCookies();

61. What are the ways to refresh a browser using Selenium WebDriver?

There are multiple ways to refresh a page in selenium

• Using driver.navigate().refresh()

• Using driver.get(“URL”) on the current URL or using driver.getCurrentUrl()

• Using driver.navigate().to(“URL”) on the current URL or driver.navigate().to(driver.getCurrentUrl());

• Using sendKeys(Keys.F5) on any textbox on the webpage

62. What is the difference between driver.getWindowHandle() and driver.getWindowHandles() in Selenium WebDriver?

driver.getWindowHandle() – It returns a handle of the current page (a unique identifier)

driver.getWindowHandles() – It returns a set of handles of the all the pages available.

63. What is the difference between driver.close() and driver.quit() methods?

Purpose of these two methods (driver.close and driver.quit) is almost same. Both allow us to close a browser but still, there is a difference.

driver.close(): To close current WebDriver instance

driver.quit(): To close all the opened WebDriver instances

64. What is the difference between driver.findElement() and driver.findElements() commands?

The difference between driver.findElement() and driver.findElements() commands is-

• findElement() returns a single WebElement (found first) based on the locator passed as parameter. Whereas findElements() returns a list of WebElements, all satisfying the locator value passed.

• Syntax of findElement()-WebElement textbox = driver.findElement(By.id(“textBoxLocator”));

• Syntax of findElements()-List <WebElement> elements = element.findElements(By.id(“value”));

• Another difference between the two is- if no element is found then findElement() throws NoSuchElementException whereas findElements() returns a list of 0 elements(empty list).

List<WebElement> list = driver.findElements(By.tagName(“a”));

Sop(list.size()); ==40

65. How to find whether an element is displayed on the web page?

WebDriver facilitates the user with the following methods to check the visibility of the web elements. These web elements can be buttons, drop boxes, checkboxes, radio buttons, labels etc.

isDisplayed()

boolean elePresent = driver.findElement(By.xpath("xpath")).isDisplayed();

isSelected()

boolean eleSelected= driver.findElement(By.xpath("xpath")).isSelected();

isEnabled()

boolean eleEnabled= driver.findElement(By.xpath("xpath")).isEnabled();

66. How to select a value in a dropdown?

By using Select class

WebElement mySelectElement = driver.findElement(By.name("dropdown"));

Select dropdown = new Select(mySelectElement);

dropdown.selectByVisibleText(Text);

dropdown.selectByIndex(Index);

dropdown.selectByValue(Value);

67. How to capture Screenshot in Selenium WebDriver?

By using TakesScreenshot Interface

In Selenium 3, we may face few issues while capturing Screenshots. To overcome we use aShot utility.

68. How to mouse hover on a web element using WebDriver?

By using Actions class

WebElement ele = driver.findElement(By.xpath("xpath"));

//Create object 'action' of an Actions class

Actions action = new Actions (driver);

//Mouseover on an element

action.moveToElement(ele).build().perform();

69. How can we handle web based pop-up?

To handle alerts popups we need to do switch to the alert window and call Selenium WebDriver Alert API methods.

70. How can we handle windows based pop up?

Selenium doesn’t support windows based applications. It is an automation testing tool which supports only web application testing. We could handle windows based popups in Selenium using some third party tools such as AutoIT, SIKULI, Robot class etc.

71. How to handle hidden elements in Selenium WebDriver?

It is one of the most important selenium interview questions.

We can handle hidden elements by using javaScriptExecutor

(JavascriptExecutor(driver)).executeScript("document.getElementsByClassName(ElementLocator).click();");

72. How can you find Broken Links in a page using Selenium WebDriver?

Practical Example.

73. How to find more than one web element in the list?

// To store the list

List <WebElement> eleList = driver.findElements(By.xpath("xpath"));

// To fetch the size of the list

int listSize = eleList.size();

//for loop

for (int i=0; i<listSize; i++)

{

// Clicking on each link

links.get(i).click();

// Navigating back to the previous page that stores the links

driver.navigate().back();

}

74. How to read a JavaScript variable in Selenium WebDriver?

By using JavascriptExecutor

// To initialize the JS object.

JavascriptExecutor JS = (JavascriptExecutor) webdriver;

// To get the site title.

String title = (String)JS.executeScript("return document.title");

System.out.println("Title of the webpage : " + title);

75. How do you read test data from excels?

Test data can efficiently be read from excel using JXL or POI API. POI API has many advantages than JXL.

76. Is it possible to automate the captcha using Selenium?

No, It’s not possible to automate captcha and bar code reader.

77. List some scenarios which we cannot automate using Selenium WebDriver?

1. Bitmap comparison is not possible using Selenium WebDriver

2. Automating Captcha is not possible using Selenium WebDriver

3. We cannot read bar code using Selenium WebDriver

4. Windows OS based pop ups

5. Third party calendars/element

6. Image

7. Word/PDF

78. What is Object Repository in Selenium WebDriver?

Object Repository is used to store element locator values in a centralized location instead of hard coding them within the scripts. We do create a property file (.properties) to store all the element locators and these property files act as an object repository in Selenium WebDriver.

79. How can you use the Recovery Scenario in Selenium WebDriver?

By using “Try Catch Block” within Selenium WebDriver Java tests.

try {

driver.get("www.xyz.com");

}catch(Exception e){

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

}

80. How to Upload a file in Selenium WebDriver?

There are two cases which are majorly used to upload a file in Selenium WebDriver such as using SendKeys Method and using AutoIT Script.

Practical Example.

Browser Button – type =“file”

SendKeys (c:\\test\\naveen.jpg);

81. How to Download a file in Selenium WebDriver?

By using AutoIT script, we could download a file in Selenium WebDriver.

82. How to run Selenium WebDriver Test from the command line?

Run Java Program using Command Prompt

Class A{

}

cd c

c: javac A.java

c: java A.java

Run TestNG using Command Prompt

java org.testng.TestNG C:\Users \Desktop\ \workspace\testing\testng.xml

83. How to switch between frames in Selenium?

By using the following code, we could switch between frames.

driver.switchTo().frame();

84. How to connect a Database in selenium?

As we all know Selenium WebDriver is a tool to automate User Interface. We could only interact with Browser using Selenium WebDriver.

We use JDBC Driver to connect the Database in Selenium (While using Java Programming Language).

Practical Example

package com.Database;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class ConnectToMySQL {

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

// Object of Connection from the Database

Connection conn = null;

// Object of Statement. It is used to create a Statement to execute the query

Statement stmt = null;

//Object of ResultSet => 'It maintains a cursor that points to the current row in the result set'

ResultSet resultSet = null;

Class.forName("com.mysql.jdbc.Driver");

// Open a connection

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testdb", "root", "root");

// Execute a query

stmt = conn.createStatement();

resultSet = stmt.executeQuery("select \* from sampletable");

while (resultSet .next()) {

System.out.println(resultSet .getString(1) + " " + resultSet.getString(2) + " " + resultSet.getString(3) + " "

+ resultSet.getString(4) + " " + resultSet.getString(5));

}

if (resultSet != null) {

try {

resultSet.close();

} catch (Exception e) {

}

}

if (stmt != null) {

try {

stmt.close();

} catch (Exception e) {

}

}

if (conn != null) {

try {

conn.close();

} catch (Exception e) {

}

}

}

}

85. How to Resize Browser Window Using Selenium WebDriver?

To resize the browser window to particular dimensions, we use ‘Dimension’ class to resize the browser window.

Practical Example

//Create object of Dimensions class

Dimension d = new Dimension(480,620);

//Resize the current window to the given dimension

driver.manage().window().setSize(d);

86. How To Scroll Web Page Down Or UP Using Selenium WebDriver?

JavaScript scrollBy() method scrolls the document by the specified number of pixels.

87. How To Perform Right Click Action (Context Click) In Selenium WebDriver?

We use Actions class in Selenium WebDriver to do Right-Click (Context Click) action.

action.contextClick(driver.findElement(By.xpsjht()).build().perform();

88. How To Perform Double Click Action In Selenium WebDriver?

We use Actions class to do Double click action in selenium.

89. How To Perform Drag And Drop Action in Selenium WebDriver?

We use Actions class to do Drag And Drop Action

90. How To Highlight Element Using Selenium WebDriver?

By using JavascriptExecutor interface, we could highlight the specified element

-------------------------------------------------------------------------------------------------------------------------------------

Question: What are the different ceremonies in Agile Scrum?

Sprint (or iteration) is the heart of Scrum. It calls for below ceremonies that bring structure to each sprint:

• Sprint Planning: Held at the start of each sprint to define the Sprint Backlog (importing stories from the Product/Release backlog), i.e. items that can be completed in the current sprint. As you might have guessed, the Product Owner drives Sprint Planning as in which stories are highest in priority.

• Daily Scrum : Presided over by the Scrum Master, Daily Scrum is a 15-minute stand-up meeting to synchronize the work of team members, i.e. what’s done on the prior day, what needs to be done today, and identify any impediments. It is also a means to track Sprint progress.

• Sprint Review: Held at the end of each sprint to demonstrate the added functionality. The goal is to get feedback from the product owner and other stakeholders to ensure that the delivered increment met the business need and to revise the Product Backlog based on the feedback.

Also Dev/QA team can provide demo to the PO or Stack holder.

• Sprint Retrospective: Held at the end of each sprint to reflect on the completed sprint and identify opportunities to improve in the next – what went well, what did not and what can be improved. It allows the team to focus on its overall performance and identify strategies for continuous improvement.

---------------------------------------------------------------------------------------------------------------------------------

1.An element has an id "bng\_123" but number is changing how to handle it?

Ans. Using dynamic xpath //tagname[starts-with(@id,'bng')]

2.how to get text from hidden elements?

String script = "return arguments[0].innerHTML";

As defined in WebDriver spec, Selenium WebDriver will only interact with visible elements, therefore the text of an invisible element will always be returned as an empty string.

However, in some cases, one may find it useful to get the hidden text, which can be retrieved from element's textContent, innerText or innerHTML attribute, by calling element.attribute('attributeName') or injecting JavaScript like return arguments[0].attributeName.

3.Difference between Selenium-IDE & Selenium RC , WebDriver

Ans. Selenium has started in 2004 and it is designed by ThoughtWorks (they have one of the branch in Koramangala Bangalore). Later on it is OpenQA has taken and maintaining till now.

Selenium Started in with Add-on in firefox which is known as Selenium IDE and trust me it is one of the most popular Addon in market.

It gives you flexibility to convert your code into multiple languages and use it accordingly.

Now you will get plenty of Add-on that you can integrate with Selenium IDE itself and you can use it.

Some plugin that I used with IDE is screenshot on failure, highlighter, export code into excel and so on.

But Selenium IDE has some of the limitation like

1- It is only available in Firefox so we can record your script in firefox only.

2- Selenium IDE does not have good reporting feature which generally we use for reporting to managers and lead or Team.

3- Selenium IDE does not support parallel execution which is one of the most important features of Automation.

4- Selenium IDE does not support remote execution as well.

**Selenium 1/Selenium Rc**

Selenium team was totally aware of all this limitations so in 2007 they came with another version which covers almost all the limitation of Selenium IDE.

Selenium team released new version called Selenium RC. Here RC stands for remote control.

Selenium RC is not another tool or plugin it is just library which contains several packages classes interfaces and methods.

Selenium RC had so many features like

1- It supports almost all browser which is available in the market.

2- We can easily integrate Selenium RC with TestNG which makes Selenium more powerfull.

3- Selenium RC having very good reporting feature with the help of TestNG.

4- Now we can do execution in parallel which the help of TestNG. We can pass multiple parameters, data driven and all the features which TestNG have.

5- Selenium RC came with Remote execution which we can achieve through Selenium grid.

Limitations of Selenium RC

It seems like Selenium RC was very powerfull and does not have any limitations but no Selenium RC also had some limitations which were

1- It generally interact with Server it means everytime you run script it will send request to server then server will communicate with browsers.

Performance was an issue ere.

2- Since 2010 Mobile industry is leading in market so Selenium RC was not capable to perform.elenium Webdriver came in 2011 and It supports all the feature of Selenium RC and additionally they have following benefits.

1- They removed server part from it so performance not an issue in Webdriver. It means simply write your code and it will directly communicate with browsers.

2- Selenium Webdriver supports Mobile Automation as well which make Selenium more powerfull. Currently Selendroid and Appium are present which allow us to automate IOS and Android application.

Soon Webriver will launch API for other platforms too.

Selenium Grid – 2.0

If you are using Grid with Selenium Webdriver.

10. What is WebElement & explain all the Mtds available in WebElement

86. What is SVN or GitHub.

At a high level, GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code. To understand exactly what GitHub is, you need to know two connected principles:

Version control

Git

What Is Version Control?

Version control helps developers track and manage changes to a software project’s code. As a software project grows, version control becomes essential. Take WordPress…

At this point, WordPress is a pretty big project. If a core developer wanted to work on one specific part of the WordPress codebase, it wouldn’t be safe or efficient to have them directly edit the “official” source code.

Instead, version control lets developers safely work through branching and merging.

With branching, a developer duplicates part of the source code (called the repository). The developer can then safely make changes to that part of the code without affecting the rest of the project.

Then, once the developer gets his or her part of the code working properly, he or she can merge that code back into the main source code to make it official.

All of these changes are then tracked and can be reverted if need be.

What Is Git?

Git is a specific open-source version control system created by Linus Torvalds in 2005.

Specifically, Git is a distributed version control system, which means that the entire codebase and history is available on every developer’s computer, which allows for easy branching and merging.

---------------------------------------------------------------------------------------------------------------------------------

Selenium Interview Questions

1. What are the components of Selenium?

Selenium IDE

Selenium RC-Remote control

Selenium WebDriver

Selenium Grid

2. Which version you are using and what addition features you see from the previous version?

Selenium WebDriver- open sorce, support multiple languages, multiple browser, lot of community support but only for browser automation, no direct reporting, we can integrate with Mavn, TestNG, Apache POI, Jenkins,GIT.

3. What is a WebDriver?

WebDriver is an interface,which have multiple methods but for implementation of those methods we need a classes- FirefoxDriver(),ChromeDriver() etc.

4. How do you select you drivers to launch a url?

driver.get("http:www.google.com")

driver.navigate().to("http://www.google.com")

5. Which method is used to fetch the driver?

get()

getCurrentUrl();

getTitle()

findElements()

findElement()

getPageSource()

close()

quit()

30. Give one example of method overloading concept used in Selenium?

Ans : Switching to frames

Q. What are some common exceptions in Selenium Webdriver?

• ElementNotVisibleException : Although an element is present in the DOM, it is not visible (cannot be interacted with). E.g. Hidden Elements – defined in HTML using type=”hidden”.

• ElementNotSelectableException : Although an element is present in the DOM, it may be disabled (cannot be clicked/selected).

• InvalidSelectorException : Selector used to find an element does not return a WebElement. Say XPath expression is used which is either syntactically invalid or does not select WebElement.

• NoSuchElementException : WebDriver is unable to identify the elements during run time, i.e. FindBy method can’t find the element.

• NoSuchFrameException : WebDriver is switching to an invalid frame, which is not available.

• NoAlertPresentException : WebDriver is switching to an invalid alert, which is not available.

• NoSuchWindowException : WebDriver is switching to an invalid window, which is not available.

• StaleElementReferenceException : The referenced element is no longer present on the DOM page (reference to an element is now Stale). E.g. The Element belongs to a different frame than the current one OR the user has navigated away to another page.

• SessionNotFoundException : The WebDriver is performing the action immediately after ‘quitting’ the browser.

• TimeoutException : The command did not complete in enough time. E.g. the element didn’t display in the specified time. Encountered when working with waits.

• WebDriverException : The WebDriver is performing the action immediately after ‘closing’ the browser.

Q #1) Explain Cucumber shortly.

Ans: Cucumber is a Framework that is based on Behaviour Driven Development (BDD) methodology.

The main aim of Behaviour Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers etc., understand the application without diving deep into the technical aspects.

Q #2) what language is used by Cucumber?

Ans: Gherkin is the language that is used by the Cucumber tool. It is a simple English representation of the application behaviour. Gherkin language uses several keywords to describe the behaviour of application such as Feature, Scenario, Scenario Outline, Given, When, Then etc.

Q #3) What is meant by a feature file?

Ans: A feature file must provide a high-level description of an Application Under Test (AUT). The first line of the feature file must start with the keyword ‘Feature’ following the description of the application under test.

A feature file may include multiple scenarios within the same file. A feature file has the extension .feature.

Q #4) What are the various keywords that are used in Cucumber for writing a scenario?

Ans: Mentioned below are the keywords that are used for writing a scenario:

Given

When

Then

And

Q #5) What is the purpose of Scenario Outline in Cucumber?

Ans: Scenario outline is a way of parameterization of scenarios. This is ideally used when the same scenario needs to be executed for multiple sets of data, however, the test steps remain the same. Scenario Outline must be followed by the keyword ‘Examples’, which specify the set of values for each parameter.

Q #6) What programming language is used by Cucumber?

Ans: Cucumber tool provides support for multiple programming languages such as Java, .Net, Ruby etc. It can also be integrated with multiple tools such as Selenium, Capybara etc.

Q #7) What is the purpose of Step Definition file in Cucumber?

Ans: A step definition file in Cucumber is used to segregate the feature files from the underlying code. Each step of the feature file can be mapped to a corresponding method on the Step Definition file.

While feature files are written in an easily understandable language such as Gherkin, Step Definition files are written in programming languages such as Java, .Net, Ruby etc.

Q #8) What are the major advantages of Cucumber framework?

Ans: Given below are the advantages of Cucumber Gherkin framework that make Cucumber an ideal choice for rapidly evolving agile methodology in today’s corporate world.

Cucumber is an open source framework.

Plain Text representation makes it easier for non-technical users to understand the scenarios.

It bridges the communication gap between various project stakeholders such as Business Analysts, Developers, and Quality Assurance personnel.

Automation test cases developed using the Cucumber tool are easier to maintain and understand as well.

Easy to integrate with other tools such as Selenium and Capybara.

Q #9) Provide an example of a feature file using the Cucumber framework.

Ans: Following is an Example of a feature file for the scenario ‘Login into the application’:

Feature: Login to the application under test.

Scenario: Login to the application.

Given Open Chrome browser and launch the application.

When User enters the username onto the UserName field.

And User enters the password into the Password field.

When User clicks on the Login button.

Then Validate if the user login is successful.

Q #10) Provide an example of Scenario Outline using Cucumber framework.

Ans: The following is an Example of Scenario Outline keyword for the scenario ‘Upload a file’. Number of parameter values to be included in the feature file is based on the tester’s choice.

Scenario Outline: Upload a file

Given that the user is on upload file screen.

When a user clicks on the Browse button.

And user enters <filename> onto the upload textbox.

And user clicks on the enter button.

Then verify that the file upload is successful.

Examples:

|filename|

|file1|

|file2|

Q #11) What is the purpose of Behaviour Driven Development (BDD) methodology in the real world?

Ans: BDD is a methodology to understand the functionality of an application in simple plain text representation.

The main aim of Behaviour Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers, Support Team understand the application without diving deep into the technical aspects.

Q #12) What is the limit for the maximum number of scenarios that can be included in the feature file?

Ans: A feature file can contain a maximum of 10 scenarios, but the number can vary from project to project and from one organization to another. But it is generally advisable to limit the number of scenarios included in the feature file.

Q #13) What is the use of Background keyword in Cucumber?

Ans: Background keyword is used to group multiple given statements into a single group. This is generally used when the same set of given statements are repeated in each scenario of the feature file.

Q #14) What symbol is used for parameterization in Cucumber?

Ans: Pipe symbol (|) is used to specify one or more parameter values in a feature file.

Q #15) What is the purpose of Examples keyword in Cucumber?

Ans: Examples keyword is used to specify values for each parameter used in the scenario. Scenario Outline keyword must always be followed by the keyword Examples.

Q #16) What is the file extension for a feature file?

Ans: File Extension for a feature file is .feature. A feature file is ideally written in a notepad file and is saved with the extension feature.

Q #17) Provide an example of step definition file in Cucumber.

Ans: Step definition corresponding to the step “Open Chrome browser and launch the application” may look like the code mentioned below:

@Given("^Open Chrome browser and launch the application$")

public void openBrowser()

{

driver = new ChromeDriver();

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

driver.get("www.facebook.com");

}

Q #18) What is the purpose of Cucumber Options tag?

Ans: Cucumber Options tag is used to provide a link between the feature files and step definition files. Each step of the feature file is mapped to a corresponding method on the step definition file.

Below is the syntax of Cucumber Options tag:

@CucumberOptions(features="Features",glue={"StepDefinition"})

Q #19) How can Cucumber be integrated with Selenium WebDriver?

Ans: Cucumber can be integrated with Selenium webdriver by downloading the necessary JAR files.

Given below are the list of JAR files that are to be downloaded for using Cucumber with Selenium web driver:

cucumber-core-1.2.2.jar

cucumber-java-1.2.2.jar

cucumber-junit-1.2.2.jar

cucumber-jvm-deps-1.0.3.jar

cucumber-reporting-0.1.0.jar

gherkin-2.12.2.jar

Q #20) When is Cucumber used in real time?

Ans: Cucumber tool is generally used in real time to write acceptance tests for an application. It is generally used by non-technical people such as Business Analysts, Functional Testers etc.

Q #21) Provide an example of Background keyword in Cucumber.

Ans: Background: Given user is on the application login page.

Q #22) What is the use of Behavior Driven Development in Agile methodology?

Ans: The advantages of Behavior Driven Development are best realized when non-technical users such as Business Analysts use BDD to draft requirements and provide the same to the developers for implementation.

In Agile methodology, user stories can be written in the format of feature file and the same can be taken up for implementation by the developers.

Q #23) Explain the purpose of keywords that are used for writing a scenario in Cucumber.

Ans: “Given” keyword is used to specify a precondition for the scenario. When a keyword is used to specify an operation to be performed. The keyword is used to specify the expected result of a performed action. “And” keyword is used to join one or more statements together into a single statement.

Q #24) What is the name of the plugin that is used to integrate Eclipse with Cucumber?

Ans: Cucumber Natural Plugin is the plugin that is used to integrate Eclipse with Cucumber.

Q #25) What is the meaning of TestRunner class in Cucumber?

Ans: TestRunner class is used to provide the link between feature file and step definition file. Below is the sample representation of how TestRunner class will look like. A TestRunner class is generally an empty class with no class definition.

Q #26) Provide an example of TestRunner class in Cucumber.

Ans:

Package com.sample.TestRunner

importorg.junit.runner.RunWith;

importcucumber.api.CucumberOptions;

importcucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)

@CucumberOptions(features="Features",glue={"StepDefinition"})

public class Runner

{

}

Q #27) What is the starting point of execution for feature files?

Ans: When integrated with Selenium, the starting point of execution must be from TestRunner class.

Q #28) Should any code be written within TestRunner class?

Ans: No code should be written under the TestRunner class. It should include the tags @RunWith and @CucumberOptions.

Q #29) What is the use of features property under Cucumber Options tag?

Ans: Features property is used to let Cucumber framework identify the location of the feature files.

Q #30) What is the use of glue property under Cucumber Options tag?

Ans: Glue property is used to let Cucumber framework identify the location of step definition files.

Q #31) What is the maximum number of steps that are to be written within a scenario?

Ans: The maximum number of steps to be written in a scenario is 3-4 steps.

How do I explain Java-Selenium automation project in the interview?

We can start as mentioned below.

1. We are using Page Object with Page Factory framework with functional/structural implementation.

>> Make sure you know what is functional/structural implementation in this framework.

2. We have maintained a page class for every page in our application and a page test class to maintain test for that pages. e.g. Product listing page,Add to cart page,Payment page,Invoice generation page.

>> Make sure you know we maintain different page and page test class and all different annotations in page factory

3. We have maintained separate package for page and page test e.g.

com.companyName.page1 com.companyName.pageTest1

>> Maintaining different packages is always a good practice to follow.

4. We also have a base page class for common functions use by all the pages.

>> Make sure you know why we have Base Page class in page object

5. We also have library package to maintain common functions related to Selenium/waits/directory creations etc

We are using JAVA/Ruby as our binding language

>> We use java because it is known to most people when we started automation.

6. We are using standardized maven project for build,execution & dependency management.

>> Make sure you know about a build tool like ant/maven

7. For handling data driven cases we are passing data using java properties file/xls file /csv file.

>> Make sure you know about libraries like openCSV,JXL/APACHE POI/Java Properties class

8. For ordering tests we are using testng framework.

>> Prepare testng questions and make sure you know how to run testng.xml using maven

9. We are using log4j library to maintain logging of our project. We are using all kinds of logging statements like INFO,DEBUG,ERROR etc. We have maintained a separate class for it in com.companyName.main package

>> Make sure you know this library usage in java, we can use log4j by mentioning properties of this framework in a xml file or a properties file and putting that file on build path.

10. We are using Extent Report for reporting purpose. It is a third party report and it is easily available at maven central repo.

We are using maven postman plugin / JAVA API to send generated extent reports as an attachment to client Distribution list.

>> Make sure you know about this plugin of maven or Java API

We checkin our code into client repository using a version controlling tool git bash on windows sytem.

Q. Used TestNG with Selenium? What are some of the annotations used?

A. Annotations are lines of code that help in passing the configuration information. They are always preceded by @ symbol.

@BeforeSuite : Once, before all Tests in the suite.

@BeforeTest : Once, Prior to the first Test case in the TestNG file.

@BeforeClass : Once before the first Test method in the current class.

@BeforeMethod : Before each Test method.

@Test: Actual Test case, the business logic.

@AfterMethod : After each Test method. @AfterClass : Once after all the Test methods in the current class.

@AfterTest : Once, after all Test cases in the TestNG file.

@AfterSuite : Once, after all Tests in the suite.

@BeforeGroups : The list of groups that this method will run before.

@AfterGroups : The list of groups that this method will run after.

@Parameters : to pass parameters in Test methods.

@DataProvider : Marks a method as supplying data for a Test method.

@Factory : Marks a method as a factory that returns objects that will be used by TestNG as Test classes.

@Listeners : Defines listeners on a Test class, helpful for logging purpose.

--------------------------------------------------------------------------------------------------

**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() { |

}

**Ques - Why you chose Selenium WebDriver for your automation?**

In my project we were working on web based application and for the same reason selenium is the best option and have multiple advantages like

1. Selenium is pure open source, freeware and portable tool.   
2. Selenium supports variety of languages that include Java, Perl, Python, C#, Ruby, Groovy, Java Script, and VB Script. etc.   
3. Selenium supports many operating systems like Windows, Macintosh, Linux, Unix etc.   
4. Selenium supports many browsers like Internet explorer, Chrome, Firefox, Opera, Safari etc.   
5. Selenium can be integrated with ANT or Maven kind of framework for source code compilation.   
6. Selenium can be integrated with TestNG testing framework for testing our applications and generating reports.   
7. Selenium can be integrated with Jenkins or Hudson for continuous integration.   
8. Selenium can be integrated with other open source tools for supporting other features.   
9. Selenium can be used for Android, IPhone, Blackberry etc. based application testing.   
10. Selenium supports very less CPU and RAM consumption for script execution.   
11. Selenium comes with different component to provide support to its parent which is Selenium IDE, Selenium Grid and Selenium Remote Control (RC).

**Ques - Selenium alone can perform full web based automation?**

Selenium can perform web based automation but not completely (silver light application-you tube, captha, graph) and for strong and clean framework we need a TestNG , POM ,sometimes support from JavaScript as well.

**Ques- What is Cucumber? How you will Implement? How you will map? Syntax? Scenario? Feature file? Gherkin keyword? TestNG or Cucumber which one you like?**

TestNG-TDD framework

Cucumber is BDD(Behavior driven framework). It totally works or deal with plain English.

Implementation-

1. Feature file –

Feature:

Scenario:

Given Then When And Then …etc – Gherkin keywords

1. Step definition class – class file – Map feature file with class file using annotation

@Given (“^ $”)

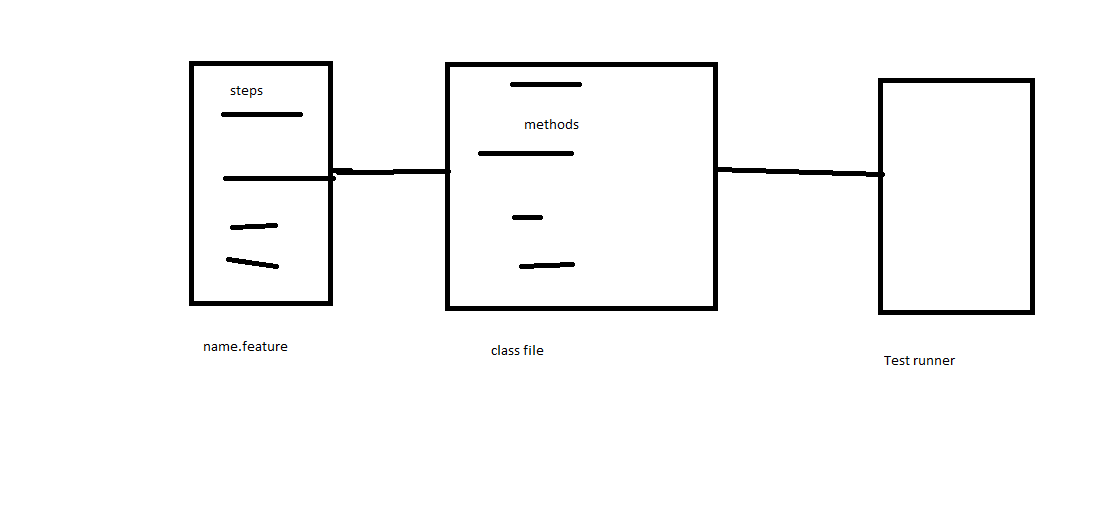
@Then(“^ $”)

@And(“^ $”)

1. Test runner – will run this class file

@RunWith(Cucumber.CLASS)

@cucumberOptions()



**Pain of an automation tester-**

When I used to create automation script

* My script used to work fine sometimes and sometimes it used to fail.
* When I run the script in my local machine it works fine but when I run in remote or another browser it fails.
* Scripts work fine in one browser and it fails in other browsers.
* Scripts work fine for current release, but it fails when new release come for my applications.
* Earlier I used to create scripts without any framework so even if small changes in the application then I used to make the changes in each script.

**Best practice to overcome this pain.**

* Start writing your own xpath using xpath methods.
* Start using a design/Framework-You can directly start writing your automation script without any framework that fine but maintenance task will be a big headache once test case size will grow even if a small change in the application then you have to modify all script.
* You can start using any framework, but it should have below characteristic. -

1. Reusable 2- Easy to use 3- Easy to maintain 4- Robust in nature

* Execute your script multiple times -It may take some time but your script will be highly stable and you will get good ROI from your script.
* Use Smart wait in your script-90 % automation script failure reason is Sync issue and locator changes only.

**IE browser** is very much concerned about security you will find a couple of Issues with IE browser

If you are using Selenium 3 then to work with Firefox browser, you need to use separate driver which will interact with Firefox browser.

**How to execute Selenium Webdriver in Chrome Browser?**

Your test case will fail and you will get IllegalStateException which says we need to specify the chrome driver path where it resides. If you notice Selenium also gives a very meaningful message that we need to add some chrome variable also while running the script.

**Variable name is – webdriver.chrome.driver**

In Java to set variable we use setProperty method of System class so let us add the same in our program.

**System.setProperty("webdriver.chrome.driver", "path of the exe file\\chromedriver.exe");**

# How to remove Disable Developer Mode Extension in Selenium?

// Create object of ChromeOptions class

ChromeOptions options = new ChromeOptions();

// add parameter which will disable the extension

options.addArguments("--disable-extensions");

// Start the chrome session

WebDriver driver = new ChromeDriver(options);

**Some key point while working with IE Browser in Selenium.**

* IE browser is slow as compared to other browsers.
* Your browser zooming level should be set to 100 % otherwise, you will get an exception.
* You have to check your security setting also in IE. While running IE browser in Selenium your all zones should be either enabled or disabled. If not, then again you will get an exception and your test cases will fail.

## Challenges with IE browser in Selenium Webdriver

You will get the different type of Exception while working with IE Browser

**Issues 1-**

### openqa.selenium.NoSuchWindowException

This is a common issue with Selenium and you can avoid this by doing some IE setting, which we are going to discuss now.

**Issue 2-**

### sendKeys works very slow it takes 1-2 second to type each character.

This is a known issue with Selenium and it only happens once you work with IE 64 bit driver.

Solution- You can download IE Driver 32 bit and start using it, even you are working with 64 bit OS this 32 bit IE driver works every time.

**Issue 3-**

### **Unexpected error launching Internet Explorer. Protected Mode must be set to the same value**

When I started working with IE this was the first exception, which I used to get, and I was sure that this related to some browser setting.

**Issue 4-**

**Unexpected error launching Internet Explorer. Browser zoom level was set to 0%**

By the name itself, you can see that we have to set the zoom level to 100 % to make it work.

**Issue 5:**

**Handle Untrusted SSL certificate error in IE browser in different ways  
Solution: IE is the product of Microsoft and IE is much worried about security so when you start working with some https application you will get a untrusted certificate.**

**Selenium has so many ways to handle this, but we will see 2 ways which work all the time for me.**

**First:**Open the application for which SSL certificate is coming so use below code after passing the URL.  
driver.get(“ur app URL”);  
driver.navigate().to(“javascript:document.getElementById(‘overridelink’).click()”);  
// you can use your code now

**Second:**

You can handle this certificate using Desired Capabilities as well.

# How to Automate Radio button and Checkbox in Selenium webdriver

Before performing click action, sometimes we need to verify some activity as well, take some example

* You need to verify whether radio button or checkbox is enabled.
* You need to verify whether radio button or checkbox is Displayed on UI or not.
* You need to verify whether checkbox and radio button is default selected or not.

Above validations are must use in script because automation is all about validation only.

we can easily [***verify***](http://learn-automation.com/capture-error-message-in-selenium/)this using some predefined method in Selenium.

|  |
| --- |
| isDisplayed();  isEnabled();  isSelected(); |

WebElement male\_radio\_button=driver.findElement(By.id("u\_0\_e"));

**boolean** status=male\_radio\_button.isDisplayed();

System.out.println("Male radio button is Displayed >>"+status);

**boolean** enabled\_status=male\_radio\_button.isEnabled();

System.out.println("Male radio button is Enabled >>"+enabled\_status);

**boolean** selected\_status=male\_radio\_button.isSelected();

System.out.println("Male radio button is selected >>"+enabled\_status);

# How to Handle Dropdown in Selenium WebDriver

WebElement month\_dropdown=driver.findElement(By.id("month"));

Select month=**new** Select(month\_dropdown);

month.selectByIndex(4);

WebElement month\_dropdown=driver.findElement(By.id("month"));

Select month=**new** Select(month\_dropdown);

month.selectByValue(“5”);

WebElement month\_dropdown=driver.findElement(By.id("month"));

Select month=**new** Select(month\_dropdown);

month.selectByVisibleText("Aug");

**Get Selected option from Dropdown.**

WebElement month\_dropdown=driver.findElement(By.id("month"));

Select month=**new** Select(month\_dropdown);

WebElement first\_value=month.getFirstSelectedOption();

String value=first\_value.getText()

**Get All option from dropdown**

WebElement month\_dropdown=driver.findElement(By.id("month"));

Select month=**new** Select(month\_dropdown);

List<WebElement> dropdown=month.getOptions();

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

String drop\_down\_values=dropdown.get(i).getText();

System.***out***.println("dropdown values are "+drop\_down\_values);

}

Explanation- getOptions() is a method of Select class which will return List of WebElement then we can iterate using for loop or iterator and using getText() method we can extract values.

getAllOptions().size()-1 which will always select last value from Dropdown.

# How to Handle Bootstrap Dropdown in Selenium WebDriver

**First one – What is bootstrap dropdown**

**The second one- How to Select values from the bootstrap dropdown.**

The bootstrap dropdown is enhanced part of dropdown where you will deal with UL and LI tag of HTML.To handle this kind of drop-down we have to use findElements method and then we can run a for loop to get specific elements.

// Dropdown items are coming in <a> tag so below xpath will get all

// elements and findElements will return list of WebElements

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

// We are using enhanced for loop to get the elements

**for**(WebElement ele:list)

{

// for every elements it will print the name using innerHTML

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

// Here we will verify if link (item) is equal to Java Script

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

ele.click();

// break the loop or come out of loop

**break**;

}

}

// here you can write rest piece of code

}

You can also select the values directly using [xpath](http://learn-automation.com/write-dynamic-css-selector-in-selenium/)  and [CSS](http://learn-automation.com/write-dynamic-css-selector-in-selenium/) but that approach is not recommended because direct xpath might change.

In above approach, we can pass a parameter directly so based on test data it will select the values from the list.

# Upload file in Selenium webdriver using Robot class

**There are multiple ways to upload files in Selenium Webdriver**

* You can directly use sendKeys(“File path”) method but sometime this methods does not work.
* We can use Robot class to upload files in Selenium.
* ***We can upload files using AutoIT as well.***

While working with robot class I faced so many issues like it works on Latest window only

***Robot class is a separate class in Java which will allow us to perform multiple tasks based on our requirement. It generally will throw AWT exception so we need to deal with it accordingly.***

Using Robot class, we can simulate keyboard event in Selenium.

To use keyboard events you have to use to a method of Robot class.

## Robot Class in Selenium Webdriver

keyPress()

keyRelease()

Each key has to be press and release respectively-

// Create object of Robot class

  Robot r=new Robot();

   // Press Enter

  r.keyPress(KeyEvent.VK\_ENTER);

   // Release Enter

  r.keyRelease(KeyEvent.VK\_ENTER);

**Actions Class-**

For all advance activity in Selenium Webdriver, we can perform easily using Actions class like Drag and Drop, mouse hover, right click, Click and Hold and so on.  
We have predefined method called **dragAndDrop(source, destination)** which is a method of Actions class.

Approach- Find the xpath of the Source  and find the xpath of destination.

Both source and destination in form of WebElement.

Note- Any method of Actions class we need to call perform () method otherwise we will get anexception. If we have series of action in our script using Actions class then we have to call build().perform() method.

// Create object of actions class

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();

We can also perform drag and drop using x and y coordinate but this will be only applicable when destination is not given.

moveToElement(WebElement)-- Mouse Hover

contextClick()-- Right click on page

contextClick(WebElement)-- Right click on specific Element

sendKeys(KEYS.TAB)--For keyboard events

clickAndHold(WebElement)--Click on element and hold until next operation

release() Release the current control

Actions act=new Actions(driver);

act.contextClick(driver.findElement(By.linkText(“Gujarati”))).perform();

Actions act=new Actions(driver);

 act.contextClick(driver.findElement(By.linkText("Gujarati"))).sendKeys(Keys.ARROW\_DOWN).sendKeys(Keys.ARROW\_DOWN).sendKeys(Keys.ENTER).build().perform();

**// Verify error message**

Assert.assertEquals(actual\_msg, expect);

Note- If text does not match then TestNG will throw AssertionError and if we do not use Exception handling then it will simply terminate our application/ program.

# Handle Multiple Windows in Selenium Webdriver

If you have to switch between tabs then also you have to use the same approach.

In Selenium, we have the feature that we can get the window name of the current window.In Selenium, we have the getWindowName method that will return current window name in String form.

We also have getWindowNames, which will return Set<String> it means the set of window name then we can iterate using Iterator. The set is part of Java collection which allows us to handle multiple sets of data dynamically.

**How to handle alert in Selenium Webdriver**

Web-Based alert and Java Script alerts are same so do not get confused.

How to Handle alert pop up in selenium Webdriver

To handle alert window in Selenium Webdriver we have predefined Interface known as Alert .

1- accept()- Will click on the ok button when an alert comes.

2- dismiss()- Will click on cancel button when an alert comes.

Note– Since alert is separate window so before using these methods we have to switch to alert window using switchTo() method

Now consider a scenario where alert window comes when certain condition true for this we can create method which will check if alert window present then only it will execute otherwise it will skip this part

public static void handleAlert(WebDriver ldriver){

if(isAlertPresent(ldriver)){

Alert alert = ldriver.switchTo().alert();

System.out.println(alert.getText());

alert.accept();

}

}

public static boolean isAlertPresent(WebDriver ldriver){

try{

ldriver.switchTo().alert();

return true;

}catch(NoAlertPresentException ex){

return false;

}

}

Important point- If alert in not present in the window and still we try to switchTo alert window then Selenium will throw NoAlertPresentException which will terminate your program so better you should use exception handle also in your script.

## Programs to handle frames in selenium

**Syntax 1-**

In this scenario, if you know the total number of frames in the web page then using the index, you can easily switch.

The index generally starts with zero so if you have only one frame then the index will be zero. If you don’t know the total number of frames in the page then you can use findElementBytagname method.

try {

driver.switchTo().frame(indexnumber);

      }

catch (NoSuchFrameException e)

     {

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

      }

We have enclosed our code with try and catch if now frame will not available this throw exception NoSuchFrameException

**Syntax 2-**

In this scenario, if you know the name  of frames in web page then using name also, you can easily switch

 try {

driver.switchTo().frame(“framename”);

      }

catch (NoSuchFrameException e)

    {

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

    }

**Syntax 3-**

In this scenario

try {

WebElement button=driver.findElement(By.xpath(""));

driver.switchTo().frame(button);

}

catch (NoSuchFrameException e)

     {

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

     }

**Until you are in frames you can not perform any operation so once we are don with frame then switchTo parent window**

|  |  |
| --- | --- |
| driver.switchTo().defaultContent(); |  |

**How to capture Screenshot in Selenium webdriver**

For taking  screenshots Selenium has provided TakesScreenShot interface in this interface you can use getScreenshotAs method which will capture the entire screenshot in form of file then using FileUtils we can copy screenshots from one location to another location

// Take screenshot and store as a file format

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

try {

 // now copy the  screenshot to desired location using copyFile //method

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

}

catch (IOException e)

 {

  System.out.println(e.getMessage());

 }

Now consider a scenario where you have to take multiple screenshots then above code will be repetitive so for this we will create a small method which captures screenshots and will call this method from our script.

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

**public** **static** **void** captureScreenShot(WebDriver ldriver) {

// Take screenshot and store as a file format

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

**try** {

// now copy the screenshot to desired location using copyFile method

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

}

**catch** (IOException e)

{

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

}

}

# How to capture screenshot for failed test cases in Selenium Webdriver

### **Generally, scripts fail in 2 situations.**

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)

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

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

// 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);

// Copy files to specific location here it will save all screenshot in our project home directory and

// result.getName() will return name of test case so that screenshot name will be same

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());

}

## Limitation of screenshots by Selenium.

1- When any alert comes on screen and if you call screenshot method then it will fail because the alert is windows activity.

If you are not aware of Alert in Selenium and how to handle, then check out below article to get more info.

2- When running cross browser testing if need to verify that test is running on which browser then you won’t be able to verify because it captures only web view par

### Create highlight element method for reuse

public static void highLightElement(WebDriver driver, WebElement element)

{

JavascriptExecutor js=(JavascriptExecutor)driver;

js.executeScript("arguments[0].setAttribute('style', 'background: yellow; border: 2px solid red;');", element);

try

{

Thread.sleep(500);

}

catch (InterruptedException e) {

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

}

js.executeScript("arguments[0].setAttribute('style','border: solid 2px white');", element);

}

### **What is Base Class in Selenium**

* Base class in the main class which will take care of Browser setup, loading [**configuration file**](http://learn-automation.com/object-repository-in-selenium-webdriver/)and other reusable methods like [**screenshot**](http://learn-automation.com/how-to-capture-screenshot-for-failed-test-cases-in-selenium-webdriver/), handling [**sync issues**](http://learn-automation.com/best-way-to-handle-synchronization-in-selenium-webdriver/)and many more.
* Using Base class we can avoid code duplication.
* Reuse code as much we can.

### How Base class works in Selenium

1-When we create base class and if TestCases extends BaseClass then we can use all the methods of Baseclass.

2- Before calling actual @Test Base class methods will get executed and Depends on annotations it will call the respective methods.

3- We can extend this class in all test cases and we can call custom methods as well directly.

----------------------------------------------------------------------------------------------------------------------------

Selenium always opens new profile/fresh instance of browser by default along with no addons/extensions. So there is no need to clear history separately.

**is it not possible to set zooming level 100% and security settings in ie using selenium?**

DesiredCapabilities caps = DesiredCapabilities.internetExplorer();

caps.setCapability(“EnableNativeEvents”, false);

caps.setCapability(“ignoreZoomSetting”, true);

WebDriver driver = new InternetExplorerDriver(caps);

--------------------------------------------------------------------------------------------------------------------------------

DesiredCapabilities cap=DesiredCapabilities.chrome();

// Set ACCEPT\_SSL\_CERTS variable to true

cap.setCapability(CapabilityType.ACCEPT\_SSL\_CERTS, true);

System.setProperty("webdriver.chrome.driver","Chrome driver path");

WebDriver driver=new ChromeDriver(cap);

--------------------------------------------------------------------------------------------------------------------------------

// Create object of DesiredCapabilities class

DesiredCapabilities cap=DesiredCapabilities.internetExplorer();

// Set ACCEPT\_SSL\_CERTS  variable to true

cap.setCapability(CapabilityType.ACCEPT\_SSL\_CERTS, true);

System.setProperty("webdriver.ie.driver","IE driver path");

WebDriver driver=newInternetExplorerDriver(cap);

--------------------------------------------------------------------------------------------------------------------------------

//It create firefox profile

FirefoxProfile profile=new FirefoxProfile()

// This will set the true value

profile.setAcceptUntrustedCertificates(true);

// This will open firefox browser using above created profile

WebDriver driver=new FirefoxDriver(profile);

driver.get("pass the url as per your requirement");

text() function always looks for exact matching of text while contains(text(),”) doesn’t look for exact matching text.

**I have following scenario :**

**Click on + Add button then it will add new row in the table.Now here whatever new row added in the table then its hard to find xpath. Second thing i want to add multiple dynamic xpath and want to enter text.**

If you are able to locate + button using xpath then after click on button, try dynamic xpath like //button[text()=’+’]/following::tr[1] to get first row. In similar fashion, you can proceed with other rows. This is just an example to navigate. Actual xpath on your application might differs.

--------------------------------------------------------------------------------------------------------------------------------

Google verification and Capcha code cant be automated using Selenium.

--------------------------------------------------------------------------------------------------------------------------------

**Can we use combination of xpath and css?**

NO

--------------------------------------------------------------------------------------------------------------------------------

**Please check whether the locators are correct. Moreover in some applications, sendKeys for input box doesn’t work in one shot, so first do click inside input box and then try with sendKeys action.**

**When the dropdown tag element is not a ‘select’ but a ‘button’**

Selenium by default supports only select tag. If you are seeing other than select then you need to go for some workaround.

If we are using drop down but having div class instead of select class what needs to be done

In this case you can use findElements() method which will return list of elements then you can select the item.

--------------------------------------------------------------------------------------------------------------------------------

**If we have multiple dropdown and all dropdown is depend on previous one selection dropdown. How we will execute second one or how to write the script in webdriver.**

Ex:- one dropdrown for Mobile Brand name.Second dropdown for Mobile Model so mobile model name depend on Brand name. ?

generally use explicit wait to handle this scenario

-------------------------------------------------------------------------------------------------------------------------------

**I actually finding different scenario like drag from local machine and drop in website. such as go to and drag a object from machine and drop on specified area. is there any method available in action or other class?? without using any third party tool.**

Actions class comes from Selenium APIs’ and selenium is built for performing actions on webpage only. Therefore, usage of third party tool is must if you want to drag and drop an item from your machine to webpage.

Selenium works with browsers only so if you want to work with local drive then you have to check other tools like Sikuli and AutoIT.

**What is difference between build() and perform() method?**

build() is called when you have to compile more than one action in a single step or in simple words it like adding sequence of actions to buffer and later we use perform() to execute it.

If you have single operation then its ok but when you have multiple operation then build().perform() is must.

--------------------------------------------------------------------------------------------------------------------------------

**Default timeout is ZERO. Default polling time is 250 millisecond if implicit wait given.**

--------------------------------------------------------------------------------------------------------------------------------

My code works fine with implicit wait on firefox. When I run the same code in chrome, I get No such element exception. I increased the implicit wait time to 100 seconds but still it throws no such element exception as soon as it logs in before waiting for 100secs (implicit wait time). The only way I could get my code work in chrome is by adding explicit wait on each element. Can you please let me know why is implicit wait not working in chrome in my case? Is there any other solution than adding explicit wait to each element to all my scripts?

If you are working with cross browsers, then sometime xpath will change. Try to use CSS which will remain same for all browsers.

--------------------------------------------------------------------------------------------------------------------------------

I am just wandering how your code is working without using System.SetProperty(“webdriver.chrome.driver”, “chromedriver.exe path”).

Yes, we can do if you set the path in Env variable then You don’t have to define the path in every script.

--------------------------------------------------------------------------------------------------------------------------------

**I have below doubts:**

**1. If we do not use any types of wait statements in program, what is default time for which webdriver will search for an element?**

**2. Is there any timeout for web page loading in selenium?**

**3. Can we increase wait time at run time means can we make webdriver to wait for some extra time to find out en element?**

1- by default wait is zero and if some wait are given then by default polling is 250 mili second.

2- Yes page load timeout is present if you want to set page load timeout. If no timeout given then page will wait until full page loaded.

3- At run time we can’t change the time out.

7. How to maximixe & minimize the browser

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

driver.manage().window().setPosition(new Point(0, 200)); - minimize

21. How to get the webelemnt height & width

WebElement element=driver.findElement(By.xpath(""));

element.getRect().getHeight();

element.getRect().getWidth();

element.getSize().getHeight();

element.getSize().getWidth();

Question on WebDriver wait statement

22. What is Synchronization

It is a mechanism which involves more than one components to work parallel with Each other.

Generally in Test Automation, we have two components

1. Application Under Test

2. Test Automation Tool.

Both these components will have their own speed. We should write our scripts in such a way that both the components should move with same and desired speed, so that we will not encounter "Element Not Found" errors which will consume time again in debugging.

Synchronization can be classified into two categories:

1. Unconditional

2. Conditional Synchronization

Unconditional :

In this we just specify timeout value only. We will make the tool to wait until certain amount of time and then proceed further.

Examples: Wait() and Thread.Sleep();

The main disadvantage for the above statements are, there is a chance of unnecessary waiting time even though the application is ready.

The advantages are like in a situation where we interact for third party systems like interfaces, it is not possible to write a condition or check for a condition. Here in this situations, we have to make the application to wait for certain amount of time by specifying the timeout value.

Conditional Synchronization:

We specify a condition along with timeout value, so that tool waits to check for the condition and then come out if nothing happens.

It is very important to set the timeout value in conditional synchronization, because the tool should proceed further instead of making the tool to wait for a particular condition to satisfy.

24. Which wait statement will be used to wait till page load

driver.manage().timeouts().pageLoadTimeout(20, TimeUnit.SECONDS);

25. How to handle dynamic object

using dynamic xpath

Question on Multiple Browser

58. What is IllegalState Exception

when we are worming with 3rd party browser than we need to define exact location of exe file of those browser otherwise will get IllegalState Exception.

59. How to work with IE , Chrome browser

System.setProperty("webdriver.chrome.driver", "c://chromedriver.exe");

System.setProperty("webdriver.ie.driver", "c://iedriver.exe");

60. How to write xpath in IE & chrome browser

using developer option

TestNG interview Questions

64. What is TestNG , why it is required

Ans : Unit testing framework Tool , used for parallel , grouping , parallel execution , Assertion , HTML mREport

65. With OUT TESTNG , what all the challenges you faced

66. Why TestNG , Why not JUNIT

1. Additional annotation

2. HTMl reporting

3. Grouping

4. Parameterziation

5. Support both java , .net

6. Parallel execution

67. What is Annotation, explain all the annotation with real time EG:

@beforeClass // global config like Launch browser

@AfterClass // close browser

@beforeMethod //login

@AfterMEthod //logout

@test

//actual test script

68. What is the use @beforeTest , @afterTest in testing

@BeforTest will be executed , Before executing all the <Classes> available with Test-Runner

RealTime useage : in case of cross browser parallel execution , we do use before test annotation to set the browser

69. What is the use @beforeSuite , @afterSuite in testing

70. Explain the hierarchy of testNG annotation

@beforeSuite

@beforeTest

@beforeClass

@beforeMethod

@test

71. What is batch execution,& how to achieve batch execution

Collection multiple test is called batch , execute multiple through testing.xml

72. Write Syntax of Xml

<suite>

<test>

<classes>

<class>

73. What is grouping execution,& how to achieve group execution

74. What is parallel execution, & how to achieve parallel execution

75. How to achieve CROSS browser testing using Selenium

@parameter

76. How to disable the testing test scripts

@test(enabled=false)

77. How to execute same test with multiple times

@test(invocation count= 10)

78. What is ASSERTION, & how many assertion you used in real-time selenium testscripts , explain with real time EG

Assert.assertEquals(“actcomNAme”, “expComNAme”)

Assert.assertTrute(logoStatus)

79. What is @parameter annotation in testNG , how to send parameter to testNG test

80. How to execute same test with multiple data

@dataprrovider

81. What is the @Listner annotation in TestNG

82. Difference between testNG-Listner & webdriver Listner

testNG Lis : ITestListner

WebDriver List : EventFiringWebDriver

52 How to execute only failed test only , when batch execution is Done

After the batch execution “refresh the folder” than automatically we get testing-falied.xml (inside test-output), just the that xml file

83. How to execute dependent test-Scripts

2 ways

Priority

dependencies

84. How to execute failed test multiple times

ITestREtry

85. When ever we get build which test-scripts , you will execute first

Using grouping concept , we will execute smokeTest first

95. How to Execute TEstNG.xml in Maven POM.xml