**Note: practise with code**

/

1.It starts selection from the document node

2. It Allows you to create 'absolute' path expressions

3. e.g “/html/body/p” matches all the paragraph elements

 //

1. It starts selection matching anywhere in the document

2. It Allows you to create 'relative' path expressions

3. e.g“//p” matches all the paragraph element

1. **What are limitations in selenium webdriver?**

* It does not support and non web-based applications; it only supports web based applications.
* Its an open source tool so in case of any technical issues you need to rely on the selenium community forums to get your issue resolved.

1. **installing/configure selenium**

Step 1: Install Java on your computer. Step 2: Install Eclipses IDE,

Step 3 – Download the Selenium Java Client Driver,

Step 4: Create a New project in Eclipse

Step 5: Create a New Package

Step 6: Import Selenium Jars

1. **what are different ways of locating elements in selenium**

1. By ID: in Java: driver.findElement(By.id("element id"))

2. By CLASS: in Java: driver.findElement(By.className("element class"))

3. By NAME: in Java: driver.findElement(By.name("element name"))

4. By TAGNAME: in Java: driver.findElement(By.tagName("element html tag name"))

5. By CSS Selector: in Java: driver.findElement(By.cssSelector("css selector"))

6. By Link: in Java: driver.findElement(By.link("link text"))

7. By Xpath: in Java: driver.findElement(By.xpath("xpath expression"))

XPATH: XPath contains the path of the element situated at the web page. Standard syntax for creating XPath is.

Xpath=//tagname[@attribute='value']

* // : Select current node.
* Tagname: Tagname of the particular node.
* @: Select attribute.
* Attribute: Attribute name of the node.
* Value : Value of the attribute.
* Xpath=//\*[contains(@id,'message')] ,
* Xpath=//\*[@type='submit' OR @name='btnReset']

Absolute XPath : It is the direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.

The key characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the element from the root node.

Relative xpath : For Relative Xpath the path starts from the middle of the HTML DOM structure. Its start with the double forward slash (//), which means it can search the element anywhere at the webpage.

You can starts from the middle of the HTML DOM structure and no need to write long xpath.

**CSS Selector: Example:**

WebElement Email = driver.findElement(By.cssSelector("input[id=email]"));

Email.SendKeys("[hello@sampleemail.com](mailto:hello@sampleemail.com)");

1. **Which is fastest way to identify elements in web page?**

CSS Selector is the fastest way to identify the elements.

CSS and Name selectors (Matches by CSS selector or @name attribute)

* faster than XPath
* whenever IDs are not available/ usable, usually @name can be used to identify elements
* CSS and XPath locators are very similar
* not very flexible in identifying elements comparing to Xpath

1. **What is absolute path and relative path in xpath?**

A single slash ‘/’ anywhere in Xpath signifies to look for the element immediately inside its parent element. This is **Absolute Path.**

Example: /html/body/div[2]/div/div/footer/section[3]/div/ul/li[3]/a

A double slash ‘//’ signifies to look for any child or any grand-child element inside the parent element. This is **Relative Path.**

Example: //body//footer/section[3]/div/ul/li[3]/a

1. **Different types of waits or synchronization in selenium webdriver**

## **Implicitly Wait Command:** This means that we can tell Selenium that we would like it to wait for a certain amount of time before throwing an exception that it cannot find the element on the page. We should note that implicit waits will be in place for the entire time the browser is open.

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

## **PageLoadTimeout Command:** Purpose: Sets the amount of time to wait for a page load to complete before throwing an error.

Example: driver.manage().timeouts().pageLoadTimeout(100, SECONDS);

## **Explicit wait:** It is a concept of the dynamic wait which wait dynamically for specific conditions. It can be implemented by WebDriverWait class.

Example: WebDriverWait wait=new WebDriverWait(driver,20);

WebElement element=wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("ur xpath here")));

**Fluent Wait**

The fluent wait is used to tell the web driver to wait for a condition, as well as the frequency with which we want to check the condition before throwing an "ElementNotVisibleException" exception.

Frequency: Setting up a repeat cycle with the time frame to verify/check the condition at the regular interval of time

Let's consider a scenario where an element is loaded at different intervals of time. The element might load within 10 seconds, 20 seconds or even more then that if we declare an explicit wait of 20 seconds. It will wait till the specified time before throwing an exception. In such scenarios, the fluent wait is the ideal wait to use as this will try to find the element at different frequency until it finds it or the final timer runs out.

//declaring fluent wait with time out of 30 seconds and frequency is set to 5 seconds

Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

.withTimeout(30, TimeUnit.SECONDS)

.pollingEvery(5, TimeUnit.SECONDS)

.ignoring(NoSuchElementException.class);

1. **How to save screen shots using selenium webdriver?**

**publicclass** ScreenShot {

**publicstaticvoid** main(String[] args) **throws** IOException {

// **TODO** Auto-generated method stub

System.*setProperty*("webdriver.gecko.driver","//home//rahul//Downloads//selenium //geckodriver");

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

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

File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.***FILE***);

//The below method will save the screen shot in d drive with name "screenshot.png"

FileUtils.*copyFile*(scrFile, **new** File("/home/rahul/Desktop/screenshot.png")); }}

1. **How to handle multiple windows in selenium webdriver?**

Step 1 : Clicking on Link1 on Window A

A new Window B is opened.

Step 2 : Move Focus from Window A to Window B

Window B is active now

Step 3 : Perform Actions on Window B

Complete the entire set of Actions

Step 4 : Move Focus from Window B to Window A

Window A is active now

Code:

 driver.get("[https://www.irctc.co.in/eticketing/loginHome.jsf](https://www.google.com/url?q=https%3A%2F%2Fwww.irctc.co.in%2Feticketing%2FloginHome.jsf&sa=D&sntz=1&usg=AFQjCNEycP6HYwNTBKKyLwHY47eJZ04ZpA)");  
//saving irctc window  
String mainWindow = driver.getWindowHandle();  
//saving all the popups  
Set<String> popups = driver.getWindowHandles();  
for(String window: popups) //foreach loop  
{  
driver.switchTo().window(window);  
driver.findElement(By.xpath(".//\*[@id='demon\_content']/div[2]/input")).click();  
}  
driver.findElement(By.xpath(".//\*[@id='loginFormId']/div[1]/div[4]/div/ul/li[4]/a")).click();  
Set<String> popup = driver.getWindowHandles();  
for(String window: popup) //foreach loop  
{  
driver.switchTo().window(window);  
System.out.println(driver.getTitle());  
}  
  
//switching back to main window  
driver.switchTo().window(mainWindow);  
driver.findElement(By.id("usernameId")).sendKeys("Lavanya");  
driver.quit();  
  
}  
  
}

1. **How to lanuch webpage using chrome driver**

First Get the Chromedriver.exe file and set System propery as below code

System.*setProperty*("webdriver.gecko.driver","//home//rahul//Downloads//selenium //chromedriver");

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

1. **What are desired capabilities in selenium webdriver?**

Every testing scenario should be executed on some specific testing environment. The testing environment can be a web browser, mobile device, mobile emulator, mobile simulator, etc.

The Desired Capabilities Class helps us to tell the webdriver, which environment we are going to use in our test script. It is used to set the browser properties (Ex. Chrome, IE), Platform Name (Ex. Linux, Windows) that are used while executing the test cases. In the case of mobile automation, as we perform the tests on different varieties of mobile devices, the Mobile Platform (ex. iOS, Android) Platform Version (Ex. 3.x,4.x in Android) can be set.

Code:

//it is used to define IE capability

DesiredCapabilities capabilities = DesiredCapabilities.internetExplorer();

capabilities.setCapability(CapabilityType.BROWSER\_NAME, "IE");

capabilities.setCapability(InternetExplorerDriver.

INTRODUCE\_FLAKINESS\_BY\_IGNORING\_SECURITY\_DOMAINS,true);

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

//it is used to initialize the IE driver

WebDriver driver = newInternetExplorerDriver(capabilities);

Conclusion

The Desired Capabilities class will help to set an environment to define the behaviour of the browser/environment on which the test can be executed.

1. **How to set language while opening website**

FireFox Browser

System.*setProperty*("webdriver.gecko.driver","//home//rahul//Downloads//selenium //geckodriver");

//setting the language preference before opening the browser

FirefoxProfile profile=**new** FirefoxProfile();

//setting the local language to the french: fr

profile.setPreference("intl.accept\_languages", "fr");

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

driver.get("https://twitter.com");

For Chrome Browser: System.setProperty(“webdriver.chrome.driver”,”D:/DollarArchive/chromedriver.exe”);  
 ChromeOptions options = new ChromeOptions();  
 options.addArguments(“–lang= sl”);  
 ChromeDriver driver = new ChromeDriver(options);  
 driver.get(“http://google.co.in&#8221;);

1. **How to handle windows based popups (upload and dropdown)**

write code to verify any application login page is working or not

(u should write code to use textbox, button click events)

Ex: write code

System.*setProperty*("webdriver.gecko.driver","//home//rahul//Downloads//selenium //geckodriver");

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

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

driver.findElement(By.*id*("Email")).sendKeys("rahul.gandhi484@gmail.com");

driver.findElement(By.*id*("next")).click();

//driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

WebDriverWait wait=**new** WebDriverWait(driver,20);

WebElement element=wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*(".//\*[@id='Passwd']"))); driver.findElement(By.*xpath*(".//\*[@id='Passwd']")).sendKeys("Azmeer@1993");

driver.findElement(By.*id*("signIn")).click();

WebDriverWait hh=**new** WebDriverWait(driver,20);

WebElement elements=hh.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*(".//\*[@id=':j']/span")));

String s=driver.findElement(By.*xpath*(".//\*[@id=':j']/span")).getText();

**if**(s=="Gmail") {

System.***out***.println("Login SucessFul");

}**else** {

System.***out***.println("Login Failed");

}

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

}

1. **How to select items from dropdown/select box**

**public class** Handling Dropdown {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

System.*setProperty*("webdriver.gecko.driver","//home//rahul//Downloads//selenium //geckodriver");

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

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

// this is to make the wait tille the required elemnt is visible

WebDriverWait wait=**new** WebDriverWait(driver,20);

WebElement element=wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*id*("month")));

//use select class to handle the drop downs

Select dropdown=**new** Select(driver.findElement(By.*id*("month")));

dropdown.selectByIndex(3);

//to get the selected element value we need to blow method

WebElement option=dropdown.getFirstSelectedOption();

System.***out***.println(option.getText());

Select day=**new** Select(driver.findElement(By.*id*("day")));

day.selectByIndex(9);

WebElement val=day.getFirstSelectedOption();

System.***out***.println(val.getText());

Select year=**new** Select(driver.findElement(By.*id*("year")));

year.selectByIndex(30);

WebElement vall=year.getFirstSelectedOption();

System.***out***.println(vall.getText());

}

Select class only works for elements with <select> tags.

getOptions( ) : List<WebElement>–This gets the all options belonging to the Select tag. It takes no parameter and returns List<WebElements>.

Select oSelect = new Select(driver.findElement(By.id("yy\_date\_8")));

List <WebElement> elementCount = oSelect.getOptions();

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

Print all the Options

Code – To get the Count of the total elements inside SELECT and to Print the text value of every element present in the SELECT.

Select oSelect = new Select(driver.findElement(By.id("yy\_date\_8")));

List <WebElement> elementCount = oSelect.getOptions();

int iSize = elementCount.size();

for(int i =0; i>iSize ; i++){

String sValue = elementCount.get(i).getText();

System.out.println(sValue);

}

isMultiple( ) : boolean – This tells whether the SELECT element support multiple selecting options at the same time or not. This accepts nothing by returns boolean value(true/false).

Command– oSelect.isMultiple();

1. **How to know if check box is checked or not in webpage**

driver.get("http://www.pittstate.edu/");

Thread.*sleep*(3000);

driver.findElement(By.*id*("right-panel-link")).click();

driver.findElement(By.*xpath*(".//\*[@id='right-panel']/ul[1]/li[2]/a")).click();

Thread.*sleep*(3000);

WebElement checkbox1;

//this will get the information of the box

checkbox1=driver.findElement(By.*id*("kmsiInput"));

//hear the is selected method is used to check if the box is selected or not

**if**(!checkbox1.isSelected()){

checkbox1.click();

}

1. **Tell me code to pass values from parent window to child window**
2. **Write code to find out if all links are working or not**

**publicclass** LinksValidating {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

System.*setProperty*("webdriver.gecko.driver","//home//rahul//Downloads//selenium //geckodriver");

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

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

//crea the list array method to collect all the links from the web page.

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

//get the total number of links by using size

System.***out***.println(links.size());

// get travese through all the links

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

//store every link in the every element

WebElement ele=links.get(i);

//store the total href total elements in the url string and make it work in method

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

//for this we have to create the method

*vrifylinkactive*(url);

}

}

**privatestaticvoid** vrifylinkactive(String linkUrl) {

// **TODO** Auto-generated method stub

**try**{

// cret URL class object

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

//open the the url connection which you have got from the link

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

//set the time out for the url to connect in maximum value

httpUrlConnect.setConnectTimeout(3000);

//make object to connect

httpUrlConnect.connect();

//make condition to check if it is equal to 200 or not

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

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

}

//check if it is equal to http not found so it is about to fail

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

System.***out***.println(linkUrl+"-"+httpUrlConnect.getResponseMessage()+"-"+HttpURLConnection.***HTTP\_NOT\_FOUND***);

}

}**catch**(Exception e){

}

}

}

1. **write code on how to use javascriptexecutor?**
2. **difference between assert and verify?**

When an “assert” fails, the test will be aborted.

Where if a “verify” fails, the test will continue executing and logging the failure.

So when Assertion fails all test steps after that line of code are skipped 

To resolve this we write this command in try catch block 

**publicclass** AssertANdVerify {

@Test

**publicvoid** assertion(){

System.*setProperty*("webdriver.gecko.driver","//home//rahul//Downloads//selenium //geckodriver");

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

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

String Title="Googles";

String actualtitle=driver.getTitle();

//here assert method check for validation

// Assert.assertEquals(Title, actualtitle);

//if assert is passe then the below code will execute if not it will be stopped

//System.out.println("passed");

//using try catch method will give the verify method

**try**{

Assert.*assertEquals*(Title, actualtitle);

}

// here Throwable will give make the below code code to run even after failing the case

**catch**(Throwable e){

System.***out***.println("it still runs after failing ");

}

}

}

1. **Difference between driver.close and driver.quit methods?**

quit(): Quits this driver, closing every associated window that was open.

close() : Close the current window, quitting the browser if it's the last window currently open. driver.quit();

WebDriver.Dispose() This method closes all Browser windows and safely ends the session

1. **Common exceptions in selenium?**

http://learn-automation.com/exceptions-in-selenium-webdriver/

NoSuchElement : An element could not be located on the page using the given search parameters. (hidden element is there )

NoSuchFrame : A request to switch to a frame could not be satisfied because the frame could not be found.

driver.switchTo().frame(invalidindex);

(or)

driver.switchTo().frame("frame\_z");

StaleElementReference : An element command failed because the referenced element is no longer attached to the DOM.

Firefox Not Connected : Firefox browser upgraded toop new version.

ElementIsNotSelectable : An attempt was made to select an element that cannot be selected.

UnknownCommand : The requested resource could not be found, or a request was received using an HTTP method that is not supported by the mapped resource.

ElementNotVisible : An element command could not be completed because the element is not visible on the page.

InvalidElementState : An element command could not be completed because the element is in an invalid state (e.g. attempting to click a disabled element).

UnknownError : An unknown server-side error occurred while processing the command.

JavaScriptError : An error occurred while executing JavaScript code.

XPathLookupError : An error occurred while searching for an element by XPath.

Timeout : An operation did not complete before its timeout expired.

NoSuchWindow : A request to switch to a different window could not be satisfied because the window could not be found.

InvalidCookieDomain : An illegal attempt was made to set a cookie under a different domain than the current page.

UnableToSetCookie : A request to set a cookie’s value could not be satisfied.

UnexpectedAlertOpen : A modal dialog was open, blocking this operation

NoAlertOpenError : An attempt was made to operate on a modal dialog when one was not open.

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

ScriptTimeout : A script did not complete before its timeout expired.

InvalidElementCoordinates : The coordinates provided to an interactions operation are invalid.

IMENotAvailable : IME was not available.  
IMEEngineActivationFailed : An IME engine could not be started.

InvalidSelector : Argument was an invalid selector (e.g. XPath/CSS).

1. **How to handle Ajax calls in selenium?**

Main important in ajax handling are

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

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

acts.contextClick(els).build().perform();

acts.keyDown(Keys.***SHIFT***).moveToElement(els).sendKeys("rahul").build().perform();

to hover to any element:

Take action class and make this code so it will work and only in chrome

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

we have webtable, need to click on second row from table.

when we click on child will be populated. first column in primary column for each row.

tell me steps to verify child form has proper data or not

Ex:

Handling columns:

**publicclass** HandlingWebTables {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

System.*setProperty*("webdriver.chrome.driver", "//home//rahul//Downloads//chromedriver");

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

driver.get("http://only-testing-blog.blogspot.in/2013/09/test.html");

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

**int**rowcount=driver.findElements(By.*xpath*("//\*[@id='post-body-6522850981930750493']/div[1]/table/tbody/tr")).size();

**int**colcount=driver.findElements(By.*xpath*("//\*[@id='post-body-6522850981930750493']/div[1]/table/tbody/tr[1]/td")).size();

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

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

String firstpath="//\*[@id='post-body-6522850981930750493']/div[1]/table/tbody/tr[";

String Secondpath="]/td[";

String Thirdpath="]";

**for**(**int**i=1;i<rowcount;i++)

{

**for**(**int**j=1;j<colcount;j++){

String finapath=firstpath+i+Secondpath+j+Thirdpath;

System.***out***.println(driver.findElement(By.*xpath*(finapath)).getText());

}

}

}

}

1. **How to assign the value to textbox other than sendkeys method?**

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

//by using java script executor we can send the keys as an alternate to the send keys method

JavascriptExecutor jse = (JavascriptExecutor) driver;

jse.executeScript("document.getElementById('twotabsearchtextbox').value='Virender Testing sending'");

1. **Selenium grid, how to execute scripts on multiple browser**

Example: first download the Jar stnadalone server

then from command prompt go to the jar location and run the below command in the Command prompt

java -jar Sandaloneserver.jar -role hub (to run the server machine )

To add the nodes we have to run the below command, where the ip address is the ip of the hub machine and port number (any number ) and make it register.

java -jar selenium-server-standalone-3.1.jar -role webdriver -hub http://192.168.0.17:4444/grid/register -port 2244

1. **Write code for drag/drop in selenium**

WebElement fromElement= driver.findElement(By.xpath("SourceElement"));

WebElement toElement=driver.findElement(By.xpath("TragetElement"));

Actions action = newActions(WebDriver);

Action dragDrop = action.dragAndDrop(fromElement, toElement).build();

dragDrop.perform();

example: driver.get("http://jqueryui.com/resources/demos/droppable/default.html");

WebElement drag=driver.findElement(By.*id*("draggable"));

WebElement drop=driver.findElement(By.*id*("droppable"));

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

drdp.dragAndDrop(drag, drop).build().perform();

Other ways to perform same thing:

Actions builder = newActions(driver);

Action dragAndDrop = builder.clickAndHold(someElement)

.moveToElement(otherElement)

.release(otherElement)

.build();

dragAndDrop.perform();

1. **Write code for right click in seelnium**

In Selenium WebDriver to perform right click on any web element is not a complex process. All you have to do is create an instance of Action class and call the methods defined in it to perform the actions you want. Follow the below steps in order to perform right click on any web element :

//ContextClick() is a method to use right click

driver.get("[http://www.amazon.com](http://www.amazon.com/)");

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

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

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

acts.contextClick(els).build().perform();

**26. Write code for scroll to specific element**

driver.get("http://www.pittstate.edu");

//this java script executor will help in scrolling the webpage

JavascriptExecutor je=(JavascriptExecutor)driver;

//pass the element with is only visible when the page is scrolled WebElement el=driver.findElement(By.*xpath*(".//\*[@id='quick-links']/ul[2]/li[3]/a"));

//this will execute until the web element is visible

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

//after scrolling down the the element value is printed

System.***out***.println(el.getText());