- OOPS in Automation Framework

ABSTRACTION

In Page Object Model design pattern, we write locators (such as id, name, xpath etc.,) in a Page Class. We utilize these locators in tests but we can't see these locators in the tests. Literally we hide the locators from the tests.

Abstraction is the methodology of hiding the implementation of internal details and showing the functionality to the users.

Learn more on Abstraction

INTERFACE

Basic statement we all know in Selenium is WebDriver

driver = new FirefoxDriver();

WebDriver itself is an Interface. So what this means is that WebDriver driver = new FirefoxDriver(); we are initializing Firefox browser using Selenium WebDriver. It also means we are creating a reference variable (driver) of the interface (WebDriver) and creating an Object. Here WebDriver is an Interface as mentioned earlier and FirefoxDriver is a class.

An interface in Java looks similar to a class but both the interface and class are two different concepts. An interface can have methods and variables just like the class but the methods declared in interface are by default abstract.

Learn more on Interface here.

INHERITANCE

We create a Base Class in the Framework to initialize WebDriver interface, WebDriver waits, Property files, Excels, etc., in the Base Class.

We extend the Base Class in other classes such as Tests and Utility Class. Extending one class into other class is known as Inheritance.

Learn more on Inheritance here.

POLYMORPHISM

The combination of overloading and overriding is known as Polymorphism. Polymorphism allows us to perform a task in multiple ways.

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

- DYNAMIC POLYMORHISM - OVERRIDING

- STATIC POLYMORHISM - OVERLOADING

Learn more on Polymorphism here.

METHOD OVERLOADING

We use implicit wait in Selenium. Implicit wait is an example of overloading. In Implicit wait we use different time stamps such as SECONDS, MINUTES, HOURS etc.,

A class having multiple methods with same name but different parameters is called Method Overloading

Learn more on Overloading here.

METHOD OVERRIDING

We use a method which was already implemented in another class by changing its parameters.

Declaring a method in child class which is already present in the parent class is called Method Overriding. Examples are get and navigate methods of different drivers in Selenium.

Learn more on Overriding with examples here

ENCAPSULATION

All the classes in a framework are an example of Encapsulation. In POM classes, we declare the data members using @FindBy and initialization of data members will be done using Constructor to utilize those in methods.

Encapsulation is a mechanism of binding code and data together in a single unit.

Learn more on Encapsulation here

I would like to discuss some other topics which we use in Automation Framework.

WEB ELEMENT:

Web element is an interface used to identify the elements in a web page.

WEBDRIVER:

WebDriver is an interface used to launch different browsers such as Firefox, Chrome, Internet Explorer, Safari etc.,

FIND BY:

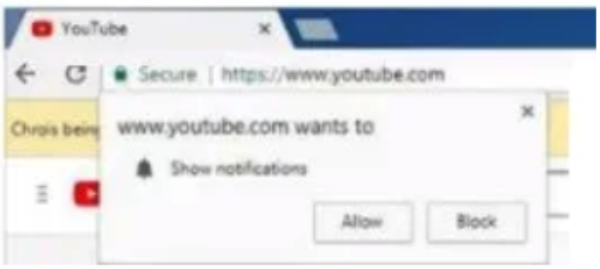
FindBy is an annotation used in Page Object Model design pattern to identify the elements.

FIND ELEMENT:

Find Element is a method in POM to identify the elements in a web page.

Learn Java – A customized post for Selenium Automation Testers

- **How to handle browser (chrome) notifications in Selenium?**



In Chrome, we can use ChromeOptions as shown below.

```
1 ChromeOptions options = new ChromeOptions();
2 options.addArguments("disable-infobars");
3 WebDriver player = new ChromeDriver(options);
```

- **What is the use of @Listener annotation in TestNG?**

TestNG listeners are used to configure reports and logging. One of the most widely used listeners in TestNG is *ITestListener* interface. It has methods like *onTestStart, onTestSuccess*, *onTestFailure*, *onTestSkipped* etc.

Practical Example

- **How to run a group of test cases using TestNG?**

TestNG allows you to perform sophisticated groupings of test methods. Not only can you declare that methods belong to groups, but you can also specify groups that contain other groups. Then TestNG can be invoked and asked to include a certain set of groups (or regular expressions) while excluding another set.

Groups are specified in your `testng.xml` file and can be found either under the `<test>` or `<suite>` tag. Groups specified in the `<suite>` tag apply to all the `<test>` tags underneath.

```
1 @Test (groups = { "smokeTest", "functionalTest" })
2 public void loginTest(){
3 System.out.println("Logged in successfully");
4 }
```

View Complete Post

- **What is Parameterized testing in TestNG?**

Parameterized tests allow developers to run the same test over and over again using different values.

There are two ways to set these parameters:

- *with* `testng.xml` – Practical Example
- *with Data Providers* –

Practical Example

- **How to set test case priority in TestNG?**

We use *priority* attribute to the *@Test* annotations. In case priority is not set then the test scripts execute in alphabetical order.

```
1  package TestNG;
2  import org.testng.annotations.*;
3  public class PriorityTestCase{
4      @Test(priority=0)
5      public void testCase1() {
6          system.out.println("Test Case 1");
7      }
8      @Test(priority=1)
9      public void testCase2() {
10         system.out.println("Test Case 2");
11     }
12 }
```

Output:

```
1 Test Case 1
2 Test Case 2
```

- **How to create and run TestNG.xml?**

In TestNG framework, we need to create T**estNG XML** file to create and handle multiple test classes. We do configure our test run, set test dependency, include or exclude any test, method, class or package and set priority etc in the XML file.

For Complete Post

- **What is TestNG Assert and list out some common Assertions supported by TestNG?**

TestNG Asserts help us to verify the condition of the test in the middle of the test run. Based on the TestNG Assertions, we will consider a successful test only if it is completed the test run without throwing any exception.

Some of the common assertions supported by TestNG are

- assertEqual(String actual,String expected)
- assertEqual(String actual,String expected, String message)
- assertEquals(boolean actual,boolean expected)
- assertTrue(condition)
- assertTrue(condition, message)
- assertFalse(condition)
- assertFalse(condition, message)

- **What are the annotations available in TestNG?**

@BeforeTest
@AfterTest
@BeforeClass
@AfterClass
@BeforeMethod
@AfterMethod
@BeforeSuite
@AfterSuite
@BeforeGroups
@AfterGroups
@Test

- **How to delete Browser Cookies with Selenium Web Driver?**

```
1 driver.Manage().Cookies.DeleteAllCookies();
```

- **How to achieve Database testing in Selenium?**

JDBC is a SQL level API that allows us to execute SQL statements. It creates a connectivity between Java Programming Language and the database.

Using JDBC Driver we can ..

i. Establish a Database connection
ii. Send SQL Queries to the Database
iii. Process the results

- **What is Continuous Integration? CI**

Continuous Integration is a development practice which aims to make sure the correctness of a software. After each commit, a suite of tests run automatically and test the software to ensure whether the software is running without any breaks. If any test fails, we will get immediate feedback say "build is broken" or "Build failed"

In simple words, continuous integration is a process of verifying the correctness of a software.

We can schedule the test suite execution using these CI Tools.

What is desired capabilities?

In Selenium we use desired capabilities to handle SSL certificates in chrome browser

We need to create an instance of DesiredCapabilities

```
public class SauseLabsDemo {

    WebDriver driver;

      public static final String USERNAME = "comal";
      public static final String ACCESS_KEY = "33b65d51-1b82-4ed3-8a21-654d70da8bed";
      public static final String URL = "https://" + USERNAME + ":" + ACCESS_KEY + "@ondemand.saucelabs.com:443/wd/hub";


    @BeforeTest
    public void setUp() throws MalformedURLException {
        DesiredCapabilities caps = DesiredCapabilities.firefox();
        caps.setPlatform(Platform.SIERRA);
        caps.setCapability("version", "latest");

        driver = new RemoteWebDriver(new URL(URL), caps);
    }
```

- **How to Highlight Element Using Selenium WebDriver?**

By using JavaScript Executor interface, we could highlight the specified element

```
    @Test
    public void highlightElement() {
    System.setProperty("webdriver.gecko.driver", "D:\\Selenium Environment\\Drivers\\geckodriver.exe");
        //Instantiating driver object
    WebDriver driver = new FirefoxDriver();
        //To launch gmail.com
    driver.get("https://www.gmail.com");
        //Collects the webelement
    WebElement element = driver.findElement(By.xpath("//*[@id='Email']"));
        //Create object of a JavascriptExecutor interface
    JavascriptExecutor js = (JavascriptExecutor) driver;
        //use executeScript() method and pass the arguments
        //Here i pass values based on css style. Yellow background color with solid red color border.
    js.executeScript("arguments[0].setAttribute('style', 'background: yellow; border: 2px solid red;');", element);
    }
}
```

Practical Example

- **How To Perform Drag And Drop Action in Selenium WebDriver?**

Selenium provides “Actions” class to handle drag and drop

Practical Example

```
public class ActionsClass {

    @Test
    public void actionsClass() throws InterruptedException{
        System.setProperty("webdriver.chrome.driver", "D:\\Selenium Environment\\Drivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
                //Create an object 'action'
        Actions action = new Actions(driver);
                //navigate to the required url where we could do drag and drop action
        driver.get("http://jqueryui.com/droppable/");
                //WebdriverWait is used to wait for a frame to be available. Once it is availble we switch to the frame to achieve our task
        WebDriverWait wait = new WebDriverWait(driver, 5);
        wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.cssSelector(".demo-frame")));
                //To get source locator
        WebElement sourceLocator = driver.findElement(By.cssSelector("#draggable"));
                //To get target locator
        WebElement targetLocator = driver.findElement(By.cssSelector("#droppable"));
                //dragAndDrop(source, target) method accepts two parameters source and locator.
                //used dragAndDrop method to drag and drop the source locator to target locator
        action.dragAndDrop(sourceLocator, targetLocator).build().perform();
        }
}
```

- **How To Perform Double Click Action In Selenium WebDriver?**

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

Practical Example

```
public class ActionsClass {

    @Test
    public void doubleClick() throws InterruptedException{
        System.setProperty("webdriver.chrome.driver", "D:\\Selenium Environment\\Drivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
                //Open the required URL where you could do double click action
        driver.get("http://api.jquery.com/dblclick/");
                //Maximize the browser
                driver.manage().window().maximize();
                //As per the above URL we need to switch to frame. The targeted element is in the frame
        driver.switchTo().frame(0);
                //Create the object 'action'
        Actions action = new Actions(driver);
        //Find the targeted element
        WebElement ele = driver.findElement(By.cssSelector("html>body>div"));
                //Here I used JavascriptExecutor interface to scroll down to the targeted element
        ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView();", ele);
                //used doubleClick(element) method to do double click action
        action.doubleClick(ele).build().perform();
                //Once clicked on the element, the color of element is changed to yellow color from blue color
                //driver.close();
    }
}
```

- **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.

Practical Example

```
public class ActionsClass {

    @Test
    public void textInCaps() throws InterruptedException{
                //Instantiating the WebDriver interface.
        System.setProperty("webdriver.chrome.driver", "D:\\Selenium Environment\\Drivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
                //Open the required URL
        driver.get("http://swisnl.github.io/jQuery-contextMenu/demo.html");
                //To maximize the browser
        driver.manage().window().maximize();
                //Create an object 'action' of an Actions class
        Actions action = new Actions(driver);
        By locator = By.cssSelector(".context-menu-one");
                //Wait for the element. Used Explicit wait
                WebDriverWait wait = new WebDriverWait(driver, 5);
                wait.until(ExpectedConditions.presenceOfElementLocated(locator));
                WebElement rightClickElement=driver.findElement(locator);
                //contextClick() method to do right click on the element
                action.contextClick(rightClickElement).build().perform();
                WebElement getCopyText =driver.findElement(By.cssSelector(".context-menu-icon-copy"));
                //getText() method to get the text value
                String GetText = getCopyText.getText();
                //To print the value
                System.out.println(GetText);
                //To close the browser
                driver.close();
    }
}
```

- **How To Scroll Web Page Down Or UP Using Selenium WebDriver?**

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

Practical Example

```
package softwareTestingMaterial;

import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.Test;

public class HandleScroll {

    @Test
    public void scrollDown(){
        System.setProperty("webdriver.gecko.driver","D://Selenium Environment//Drivers//geckodriver.exe");
        WebDriver driver = new FirefoxDriver();
        driver.navigate().to("https://www.softwaretestingmaterial.com");
        //to perform Scroll on application using  Selenium
        JavascriptExecutor js = (JavascriptExecutor) driver;
        js.executeScript("window.scrollBy(0,250)", "");
    }
}
```

- **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.

```
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxOptions;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.testng.annotations.Test;

public class ResizeBrowser {

    @Test
    public void launchBrowser() {
        System.setProperty("webdriver.gecko.driver","D://Selenium Environment//Drivers//geckodriver.exe");
        WebDriver driver = new FirefoxDriver();
        driver.navigate().to("http://www.SoftwareTestingMaterial.com");
        System.out.println(driver.manage().window().getSize());
        //Create object of Dimensions class
        Dimension d = new Dimension(480,620);
        //Resize the current window to the given dimension
        driver.manage().window().setSize(d);
        System.out.println(driver.manage().window().getSize());
     }
}
```

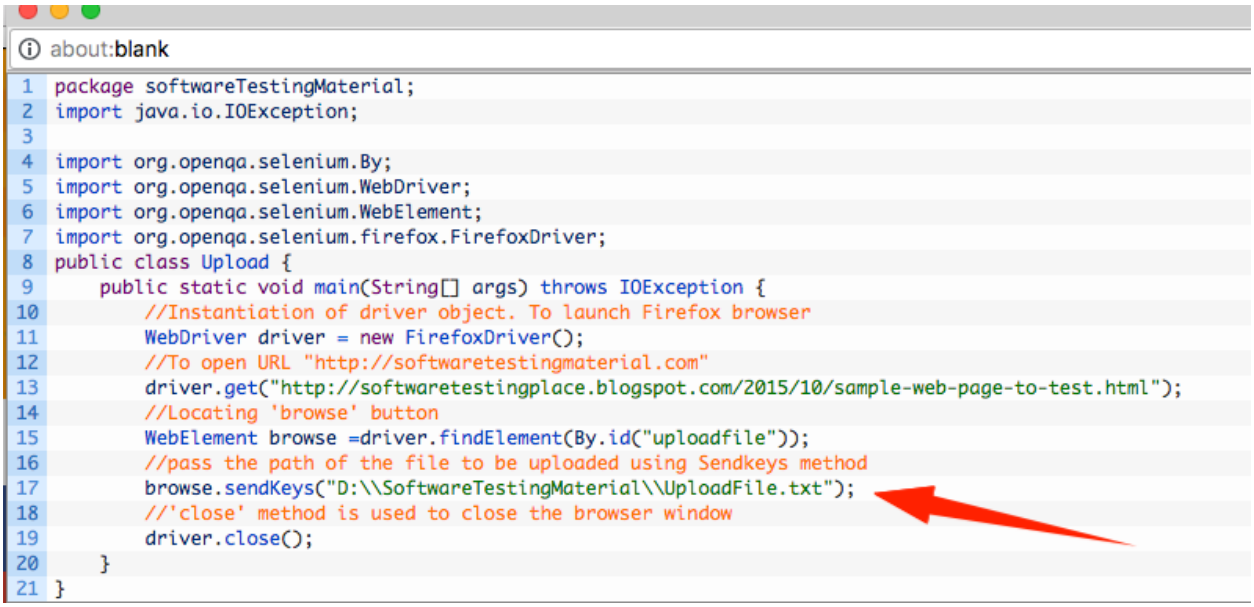
- **How to switch between frames in Selenium?**

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

```
driver.switchTo().frame();
```

- **How to Upload a file in Selenium WebDriver?**

By Using send keys method



- **How can you use the Recovery Scenario in Selenium WebDriver?**

  By using "*Try Catch Block*" within Selenium WebDriver Java tests.

```
try {
      driver.get("www.SoftwareTestingMaterial.com");
}catch(Exception e){
      System.out.println(e.getMessage());
}
```

- **What are the advantages of Page Object Model Framework?**

Code reusability – We could achieve code reusability by writing the code once and use it in different tests.

Code maintainability – There is a clean separation between test code and page specific code such as locators and layout which becomes very easy to maintain code. It enhances test maintenance and reduces code duplication.

Object Repository – Each page will be defined as a java class. All the fields in the page will be defined in an interface as members. The class will then implement the interface.

Readability – Improves readability due to clean separation between test code and page specific code

- **What is the difference between Page Object Model (POM) and Page Factory?**

Page Object is a class that represents a web page and hold the functionality and members.

Page Factory is a way to initialize the web elements you want to interact with within the page object when you create an instance of it.

- **What is Page Factory?**

As you know 'Page Object Model' is a way of representing an application in a test framework. For every 'page' in the application, we create a Page Object to reference the 'page' in the other hand 'Page Factory' is one way of implementing the 'Page Object Model'.

- **What is Page Object Model in Selenium? POM**

Page object model (POM) can be used in any kind of framework
A page object is an object-oriented class that serves as an interface to a page of your Application Under Test(AUT). The tests then use the methods of this page object class whenever they need to interact with the User Interface (UI) of that page. The benefit is that if the UI changes for the page, the tests themselves don't need to change, only the code within the page object needs to change.

- **How you build Object Repository in your project?**
-

When a user records a test, the objects and its properties are captured by default in an Object Repository. QTP uses this Object Repository to play back the scripts. Coming to Selenium, there is no default Object Repository concept. It doesn't mean that there is no Object Repository in Selenium. Even though there is no default one still we could create our own. In Selenium, we call objects as locators (such as ID, Name, Class Name, Tag Name, Link Text, Partial Link Text, XPath, and CSS). Object repository is a collection of objects. One of the ways to create Object Repository is to place all the locators in a separate file (i.e., properties file). But the best way is to use Page Object Model. In the Page Object Model Design Pattern, each web page is represented as a class. All the objects related to a particular page of a web application are stored in a class.

- **List some scenarios which we cannot automate using Selenium WebDriver?**

1. Bitmap comparison is not possible using Selenium WebDriver(photos = Bitmap)
2. Automating Captcha is not possible using Selenium WebDriver
3. We can not read bar code using Selenium WebDriver

- **How to handle Ajax calls in Selenium WebDriver?**

AJAX sends HTTP requests from the client to server and then process the server's response without reloading the entire page.

When you click on a submit button, the required information may appear on the web page without refreshing the browser. Sometimes it may load in a second and sometimes it may take longer. We have no control over loading time. The best approach to handle this kind of situations in selenium is to use dynamic waits

1. **titleIs()** – The expected condition waits for a page with a specific title.

```
1 wait. Until(ExpectedConditions.titleIs("Deal of the Day"));
```

2. **elementToBeClickable()** – The expected condition waits for an element to be clickable i.e. it should be present/displayed/visible on the screen as well as enabled.

```
1 wait.until(ExpectedConditions.elementToBeClickable(By.xpath("xpath")));
```

3. **alertIsPresent()** – The expected condition waits for an alert box to appear.

```
1 wait.until(ExpectedConditions.alertIsPresent()) !=null);
```

4. **textToBePresentInElement()** – The expected condition waits for an element having a certain string pattern.

```
1 wait.until(ExpectedConditions.textToBePresentInElement(By.id("title'"), "text to be found"));
```

- **Is it possible to automate the captcha using Selenium?**

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

- **How do you read test data from excels?**

To handle excel files we use Apache POI in Selenium WebDriver. As we all know Selenium supports only Web browser automation. We

need to get the help of third party API like Apache POI to handle (read and write) excel files using Selenium WebDriver.

List of JAVA INTERFACES AND CLASSES IN APACHE POI FOR READING XLS AND XLSX FILE

INTERFACE	XLS CLASS	XLSX CLASS
Workbook	HSSFWorkbook	XSSFWorkbook
Sheet	HSSFSheet	XSSFSheet
Row	HSSFRow	XSSFRow
Cell	HSSFCell	XSSFCell

- **What is JavaScriptExecutor and in which cases JavaScriptExecutor will help in Selenium automation?**

Let me give you an example in general, we click on an element using click () method in Selenium.

For example:

1 - driver. findElement(By.id("Id Value")).click();

Sometimes web controls don't react well against selenium commands and we may face issues with the above statement (click ()). To overcome such kind of situation, we use JavaScriptExecutor interface.

Package:

```
1 import org.openqa.selenium.JavascriptExecutor;
```

Syntax:

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

Some scenarios we could handle using this Interface are :

1. To type Text in Selenium WebDriver without using send Keys() method
2. To click a Button in Selenium WebDriver using JavaScript
3. To handle Checkbox
4. To generate Alert Pop window in selenium
5. To refresh browser window using Javascript
6. To get innertext of the entire webpage in Selenium
7. To get the Title of our webpage
8. To get the domain
9. To get the URL of a webpage
10. To perform Scroll on an application using Selenium
11. To click on a SubMenu which is only visible on mouse hover on Menu
12. To navigate to different page using Javascript

- 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);
```

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

```
1  // To store the list
2  List <WebElement> eleList = driver.findElements(By.xpath("xpath"));
3  // To fetch the size of the list
4  int listSize = eleList.size();
5  //for loop
6  for (int i=0; i<listSize; i++)
7  {
8      // Clicking on each link
9      links.get(i).click();
10     // Navigating back to the previous page that stores the links
11     driver.navigate().back();
12 }
```

- How to handle hidden elements in Selenium WebDriver?

about:blank

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

We can handle hidden elements by using JavaScript Executor.

- **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, Robot class until now I have not used those tools but I am looking forward to learning if needed.

- **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.

we use ***Alert*** Interface. The ***Alert*** Interface provides some methods to handle the popups

We need to Import a package
org.openqa.selenium.Alert
to handle the alerts in Selenium.

driver.switchTo().alert();

To get a handle to the open alert:

```
1 Alert alert = driver.switchTo().alert();
```

To Click on OK button:

```
1 alert.accept();
```

To click on Cancel button.

```
1 alert.dismiss()
```

To get the text which is present on the Alert.

```
1 alert.getText();
```

To enter the text into the alert box

```
1 alert.sendkeys(String stringToSend);
```

To Authenticate by passing the credentials

```
1 alert.authenticateUsing(Credentials credentials)
```

Practical Example.

- **How to mouse hover on a web element using WebDriver?**

By using Actions class

```
1 WebElement ele = driver.findElement(By.xpath("xpath"));
2 //Create object 'action' of an Actions class
3 Actions action = new Actions(driver);
4 //Mouseover on an element
5 action.moveToElement(ele).perform();
```

- **How to capture Screenshot in Selenium WebDriver?**

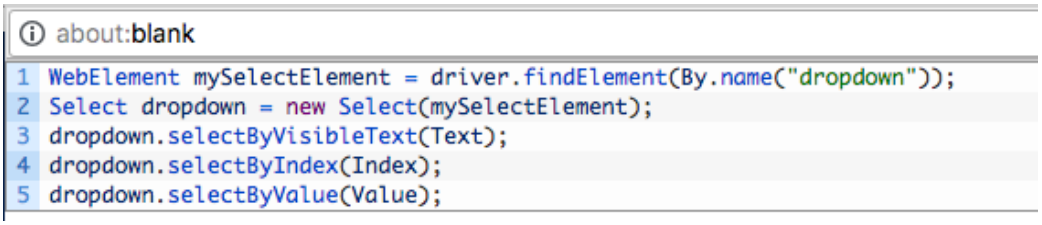
Selenium provides an interface called *TakesScreenshot* which has a method *getScreenShotAs* which can be used to take a screenshot of the application under test.

about:blank

```
1  package softwareTestingMaterial;
2
3  import java.io.File;
4  import org.apache.commons.io.FileUtils;
5  import org.openqa.selenium.OutputType;
6  import org.openqa.selenium.TakesScreenshot;
7  import org.openqa.selenium.WebDriver;
8  import org.openqa.selenium.firefox.FirefoxDriver;
9  import org.testng.annotations.Test;
10
11 public class CaptureScreenshot {
12
13     @Test
14     public static void captureScreenMethod() throws Exception{
15         System.setProperty("webdriver.gecko.driver","D://Selenium Environment//Drivers//geckodriver.exe");
16         WebDriver driver = new FirefoxDriver();
17         driver.manage().window().maximize();
18         driver.get("https://www.softwaretestingmaterial.com/capture-screenshot-using-selenium-webdriver");
19         File screenshotFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
20         FileUtils.copyFile(screenshotFile, new File("D:\\SoftwareTestingMaterial.png"));
21         driver.close();
22         driver.quit();
23     }
24 }
```

How to select a value in a dropdown?

By using *Select* class



Types of Select Methods:

i. **selectByVisibleText** Method
ii. **selectByIndex** Method
iii. **selectByValue** Method

Types of DeSelect Methods:
i. deselectByVisibleText Method
ii. deselectByIndex Method
iii. deselectByValue Method
iv. deselectAll Method

- **How to find whether an element is displayed on the web page?**

In selenium WebDriver to verify if an element is displayed like buttons, drop boxes, checkboxes, radio buttons, labels we use…

1. isDisplayed()

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

2. isSelected()

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

3. isEnabled()

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

- **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.

- **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

- **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.

- **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()* command as mentioned in the question 45
- 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

- Difference between Test Strategy and Test Plan

Test Plan	Test Strategy
• A test plan for software project can be defined as a document that defines the scope, objective, approach and emphasis on a software testing effort	• Test strategy is a set of guidelines that explains test design and determines how testing needs to be done

• Components of Test plan include- Test plan id, features to be tested, test techniques, testing tasks, features pass or fail criteria, test deliverables, responsibilities, and schedule, etc.	• Components of Test strategy includes- objectives and scope, documentation formats, test processes, team reporting structure, client communication strategy, etc.
• Test plan is carried out by a testing manager or lead that describes how to test, when to test, who will test and what to test	• A test strategy is carried out by the project manager. It says what type of technique to follow and which module to test
• Test plan narrates about the specification	• Test strategy narrates about the general approaches
• Test plan can change	• Test strategy cannot be changed
• Test planning is done to determine possible issues and dependencies in order to identify the risks.	• It is a long-term plan of action.You can abstract information that is not project specific and put it into test approach
• A test plan exists individually	• In smaller project, test strategy is often found as a section of a test plan

• It is defined at project level	• It is set at organization level and can be used by multiple projects

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();

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

- How to fetch the current page URL in Selenium?

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

```
1 driver.getCurrentUrl();
```

- 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

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

- **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")*

- **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)

```
1 Thread.sleep(5000)
```

- **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.

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

- **How to submit a form using Selenium WebDriver?**

We use "submit" method on element to submit a form

```
1 driver.findElement(By.id("form_1")).submit();
```

- **How to click on a hyperlink using Selenium WebDriver?**

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

```
1 driver.findElement(By.linkText("Software Testing Material Website")).click();
```

- **How to get an attribute value using Selenium WebDriver?**

By using getAttribute(value);

It returns the value of the attribute passed as a parameter.

HTML:

```
1 <input name="nameSelenium" value="valueSelenium">SoftwareTestingMaterial</input>
```

Selenium Code:

```
1 String attributeValue = driver.findElement(By.name("nameSelenium")).getAttribute("value");
2 System.out.println("Available attribute value is :"+attributeValue);
3 Output: valueSelenium
```

- **How to get a text of a web element?**

By using getText() method

about:blank

```
package softwareTestingMaterial;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
public class TestTestTest {

    @Test
    public void testmethod(){
        System.setProperty("webdriver.chrome.driver", "D:\\Selenium Environment\\Drivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.google.com");
        String availableText = driver.findElement(By.xpath("//*[@id='gbw']/div/div/div[1]/div[1]/a")).getText();
        System.out.println("Text Available is :"+availableText);
    }
}
```

- **How to clear the text in the text box using Selenium WebDriver?**

By using clear() method

about:blank

```
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();
```

- **How to input text in the text box without calling the sendKeys()?**

```
// To initialize js object
JavascriptExecutor JS = (JavascriptExecutor)webdriver;
// To enter username
JS.executeScript("document.getElementById('User').value='SoftwareTestingMaterial.com'");
// To enter password
JS.executeScript("document.getElementById('Pass').value='tester'");
```

- **How to input text in the text box using Selenium WebDriver?**

By using sendKeys() method

about:blank

```
1 WebDriver driver = new FirefoxDriver();
2 driver.get("https://www.gmail.com");
3 driver.findElement(By.xpath("xpath")).sendKeys("Software Testing Material Website");
```

- **What is Fluent Wait In Selenium WebDriver?**

In My experience FluentWait can define the maximum amount of time to wait for a specific condition and frequency with which to check the condition before throwing an "*ElementNotVisibleException*" exception.

Syntax

```
1  Wait wait = new FluentWait(WebDriver reference)
2  .withTimeout(timeout, SECONDS)
3  .pollingEvery(timeout, SECONDS)
4  .ignoring(Exception.class);
5
6  WebElement foo=wait.until(new Function<WebDriver, WebElement>() {
7  public WebElement applyy(WebDriver driver) {
8  return driver.findElement(By.id("foo"));
9  }
10 });
```

Example

```
1 Wait wait = new FluentWait<WebDriver>(driver)
2 .withTimeout(45, TimeUnit.SECONDS)
3 .pollingevery(5, TimeUnit.SECONDS)
4 .ignoring(NoSuchElementException.class);
```

- **What is WebDriver Wait 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.

This are some Conditions that can be used in Explicit Wait

1. alertIsPresent()
2. elementSelectionStateToBe()
3. elementToBeClickable()
4. elementToBeSelected()
5. frameToBeAvaliableAndSwitchToIt()
6. invisibilityOfTheElementLocated()
7. invisibilityOfElementWithText()
8. presenceOfAllElementsLocatedBy()
9. presenceOfElementLocated()
10. textToBePresentInElement()
11. textToBePresentInElementLocated()
12. textToBePresentInElementValue()
13. titleIs()
14. titleContains()
15. visibilityOf()
16. visibilityOfAllElements()
17. visibilityOfAllElementsLocatedBy()
18. visibilityOfElementLocated()

about:blank

```
//WebDriverWait wait = new WebDriverWait(WebDriverRefrence,TimeOut);
WebDriverWait wait = new WebDriverWait (driver, 20);
wait.until(ExpectedConditions.VisibilityofElementLocated(By.xpath(""//button[@value='Save Changes']"")));
```

```
package waits;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
public class ExplicitWaits {
    public static void main(String[] args) {
        //To create a new instance of Firefox Driver
        WebDriver driver = new FirefoxDriver();
        //To open a website "Software Testing Material"
        driver.get("http://www.SoftwareTestingmaterial.com");
        //To maximize the browser window
        driver.manage().window().maximize();
        //This waits up to 15 seconds before throwing a TimeoutException or if it finds the element will return it in 0 - 15 seconds
        WebDriverWait wait = new WebDriverWait (driver, 15);
        //Title of the webpage is "Software Testing Material - A site for Software Testers"
        wait.until(ExpectedConditions.titleIs("Software Testing Material - A site for Software Testers"));
        //If the above condition met then the browser will be closed
        //To close the browser
        driver.close();
        //Change the title "Software Testing Material - A site for Software Testers" as "xyz" in the script and try
        //You will face an execption - Exception in thread "main" org.openqa.selenium.TimeoutException: Timed out after 20 seconds waiting for title to be "Software Testing
    }
}
```

Practical example

- **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.

Practical example

about:blank

```
1 driver.manage().timeouts().implicitlyWait(TimeOut, TimeUnit.SECONDS);
```

```
package waits;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class ImplicitWaits {
    public static void main(String[] args) {
        //To create a new instance of Firefox Driver
        WebDriver driver = new FirefoxDriver();
        //Implicit Wait - Here the specified Implicit Wait time frame is 15 seconds.
        //It waits 15 seconds of time frame for the element to load.
        //It throws an exception, if the element is not loaded within the specified ti
        driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
        //To open a website "Software Testing Material"
        driver.get("http://www.SoftwareTestingMaterial.com");
        //To maximize the browser window
        driver.manage().window().maximize();
        //To close the browser
        driver.close();
    }
}
```

- **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 – Click to view detailed post
- Explicit Waits – Click to view detailed post
- Fluent Waits – Click to view detailed post

- **What are the different exceptions you have faced in Selenium WebDriver?**

Some of the exceptions I have faced in my current project are

1. ElementNotVisibleException
2. StaleElementReferenceException

Element Not visible Exception:

This exception will be thrown when you are trying to locate a particular element on webpage that is not currently visible even though it is present in the DOM. Also sometimes, if you are trying to locate an element with the xpath which associates with two or more element.

Stale Element Reference Exception:

A stale element reference exception is thrown in one of two cases, the first being more common than the second.

The two reasons for Stale element reference are

1. The element has been deleted entirely.
2. The element is no longer attached to the DOM.

We face this stale element reference exception when the element we are interacting is destroyed and then recreated again. When this happens the reference of the element in the DOM becomes stale. Hence we are not able to get the reference to the element.

Some other exceptions we usually face are as follows:

- WebDriverException
- IllegalStateException
- TimeoutException

- NoAlertPresentException
- NoSuchWindowException
- NoSuchElementException

- **Explain the line of code *Webdriver driver = new FirefoxDriver();* ?**

```
1 Webdriver driver = new FirefoxDriver();
```

`*WebDriver*` is an interface and we are creating an object of type WebDriver instantiating an object of FirefoxDriver class.

- **What are the verification points available in Selenium?**

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, etc.)

- **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`

- **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.

- **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.

```
1 /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 double forward slash(//) as shown below.

```
1 //input[@id='email']
```

- **What is the difference between "/" and "//"**

Single Slash "/" – Single slash is used to create XPath with 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 XPath with relative path i.e. the XPath would be created to start selection from anywhere within the document.

- **What is an 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.

- **What are the Locators available in Selenium?**

In Selenium WebDriver, there are 8 different types of locators:

1. ID – Practical example
2. ClassName – Practical example
3. Name – Practical example
4. TagName – Practical example
5. LinkText – Practical example
6. PartialLinkText – Practical example
7. XPath – Practical example
8. CSS Selector – Practical example

Click here to see the detailed post on Locators.

- **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

- **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

- **What is a hub in Selenium Grid?**

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

- **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.

-

-