what are limitations in selenium webdriver

1. Selenium needs very much expertise resources. The resource should also be very well versed in framework architecture.   
   2. Selenium only supports web based application and does not support windows based application.   
   3. It is difficult to test Image based application.   
   4. Selenium need outside support for report generation activity like dependence on TestNG or Jenkins.   
   5. Selenium does not support built in add-ins support.   
   6. Selenium user lacks online support for the problems they face.   
   7. Selenium does not provide any built in IDE for script generation and it need other IDE like Eclipse for writing scripts.   
   8. Selenium Automation Engineers are bit in scarcity these days.   
   9. Selenium script creation time is bit high.   
   10. Selenium does not support file upload facility.   
   11. Selenium partially supports for Dialog boxes.

installing/configure selenium

**In this step,**

1. Right-click on "newproject" and select Properties.
2. On the Properties dialog, click on "Java Build Path".
3. Click on the Libraries tab, and then.
4. Click on "Add External JARs.."

what are different ways of locating elements in selenium

\*Using ID as a Locator

id = id of the element

### **\*Using ClassName as a Locator**

### class = classname of the element

### **\*Using name as a Locator**

### name = name of the element

### **\*Using Link Text as a Locator**

### link = link text of the element

### **\*Using Xpath as a Locator**

**Relative Xpath**

Relative Xpath begins from the current location and is prefixed with a “//”.

For example: //span[@class=’Email’]

**Absolute Xpath**

Absolute Xpath begins with a root path and is prefixed with a “/”.

For example: /html/body/div/div[@id=’Email’]

which is fastest way to identify elements in web page?

1. ID selectors (By.ID – Matches by @id attribute)
   * IDs are the safest, fastest locator option and should always be your first choice
   * IDs should be unique in every page according to W3C website
   * even if the DOM changes, if the ID is still there, then WebDriver can still locate it
   * always try and get extra IDs added into the code, this makes testers life easier
   * fastest locator as it uses the document.getElementById() javascript command which is optimised by many browsers
2. 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
3. XPath locators (Matches with arbitrary XPath expression)
   * most flexible in order to build reliable web element locators
   * very slow locator (particularly in IE) since in order to locate the element it needs to traverse the whole DOM of the page which is a time consuming operation

what is absolute path and relative path in xpath

Relative path: Location path specifies the location of node in XML document. If location path starts with the node that we've selected then it is a relative path.

Following are few examples locating the elements using relative path.

**firstname** − select firstname related to student nodes.

<p><xsl:value-of select = "firstname"/></p>

Absolute path:

Location path specifies the location of node in XML document. This path can be absolute or relative. If location path starts with root node or with '/' then it is an absolute path. Following are few of the example locating the elements using absolute path.

**/class/student** − select student nodes within class root node.

<xsl:for-each select = "/class/student">

**/class/student/firstname** − select firstname of a student node within class root node.

<p><xsl:value-of select = "/class/student/firstname"/></p>

different types of waits or synchronization in selenium webdriver

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();](https://docs.oracle.com/javase/tutorial/essential/concurrency/sleep.html" \t "_blank)*

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

**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.

In Selenium we have implicit Wait and Explicit Wait conditional statements. Check here for [Examples on how to use Webdriver Waits](http://seleniumeasy.com/selenium-tutorials/webdriver-wait-examples)

**1. Implicit Wait.**

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.

Syntax:

driver.manage.TimeOuts.implicitwait(6,Timeunit.SECONDS);

**Example using implicit timeout**

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

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

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

**Explicit Wait:**

We need to define a wait statement for certain condition to be satisfied until the specified timeout period. If the Webdriver finds the element within the timeout period the code will get executed.

Explicit wait is mostly used when we need to Wait for a specific content/attribute change after performing any action, like when application gives AJAX call to system and get dynamic data and render on UI.

Example: Like there are drop-downs Country and State, based on the country value selected, the values in the state drop-down will change, which will take few seconds of time to get the data based on user selection.

Example:

/\*Explicit **wait** **for** **state** dropdown field\*/

    WebDriverWait **wait** = new WebDriverWait(driver, 10);

**wait**.**until**(ExpectedConditions.visibilityOfElementLocated(By.id("statedropdown")));

The above statement waits up to 10 seconds before throwing Exception (TimeoutException - Timed out after 10 seconds waiting for visibility of element) or if it finds the element, it will return in 0 - 10 seconds.

There are different waits that can be used based on the needs which we frequently come across when automating web applications.

**Fluent Wait:**

Using FluentWait we can define the maximum amount of time to wait for a condition, as well as the frequency with which to check for the condition.

And also the user can configure to ignore specific types of exceptions such as ["NoSuchElementExceptions"](http://selenium.googlecode.com/git/docs/api/java/org/openqa/selenium/NoSuchElementException.html) when searching for an element. NoSuchElement exception is thrown by findElement(By) and findElements(By). When ever it try to find any element it returns the first matching element on the current page else it throws NoSuchElementException - when no matching elements are found.

Syntax:

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

//Wait for the condition

       .withTimeout(30, TimeUnit.SECONDS)

         // which to check for the condition with interval of 5 seconds.

       .pollingEvery(5, TimeUnit.SECONDS)

     //Which will ignore the NoSuchElementException

       .ignoring(NoSuchElementException.class);

***isElementPresent:***

WebDriverWait **wait** = new WebDriverWait(driver, waitTime); **wait**.**until**(ExpectedConditions.presenceOfElementLocated(locator));

***isElementClickable:***

WebDriverWait **wait** = new WebDriverWait(driver, waitTime); **wait**.**until**(ExpectedConditions.elementToBeClickable(locator));

***isElementVisible:***

WebDriverWait **wait** = new WebDriverWait(driver, waitTime); **wait**.**until**(ExpectedConditions.visibilityOfElementLocated(locator));

***isElementInVisible:***

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

**wait**.**until**(ExpectedConditions.invisibilityOfElementLocated(locator));

***isElementEnabled:***

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

element.isEnabled();

***isElementDisplayed:***

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

element.isDisplayed();

***Wait for invisibility of element:***

WebDriverWait **wait** = new WebDriverWait(driver, waitTime); **wait**.**until**(ExpectedConditions.invisibilityOfElementWithText(by));

***Wait for invisibility of element with Text:***

WebDriverWait **wait** = new WebDriverWait(driver, waitTime); **wait**.**until**(ExpectedConditions.invisibilityOfElementWithText(by, strText));

Synchronization/ Waits can be achieved in many different ways :

1. **Sleep method** of Thread class
2. **Page Load timeout**
3. **Script timeout**
4. **Implicit Wait**
5. **Explicit Wait**

### ****sleep() command :****

The command halts the execution of the script for specified amount of milliseconds.

Example :

WebDriver driver = new FirefoxDriver();driver.get(“http://seleniumatfingertips.wordpress.com”);driver.manage().window().maximize();

WebElement homeLink = driver.findElement(By.linkText(“Home”));

homeLink.click();

try{

           Thread.sleep(5000);

} catch(Exception e){

            System.out.print(e);

}

### ****Page Load timeout :****

We can **set the amount of time to wait for a page load to complete** before throwing an error.

Example :

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

### ****Script timeout :****

We can set the amount of time **to wait for an asynchronous script to finish execution** before throwing any error.

Example :

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

### ****Implicit Wait :****

Implicit wait is a mechanism, which is used **to notify WebDriver instance to wait for specific time**, if **any element is readily not available** on the webpage.

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

### ****Explicit Wait :****

Explicit wait is a technique used to advise the **WebDriver instance to stand by** the **execution** till the time the **certain condition is met** or **the maximum wait time is elapsed**.

WebElement element = driver.findElement(By.linkText(“Home”));WebDriverWait wait = new WebDriverWait(driver, 20);

wait.until(ExpectedConditions.visibilityOf(element));

how to save screen shots using selenium webdriver

import java.io.File;

import java.io.IOException;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.Test;

public class ScreenshootGoogle {

 @Test

 public void TestJavaS1()

{

// Open Firefox

WebDriver driver=new FirefoxDriver();

// Maximize the window

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

// Pass the url

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

// 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());

 }

 }

how to handle multiple windows in selenium webdriver

**import**java.util.List;  
**import**org.junit.After;  
**import**org.junit.Before;  
**import**org.junit.Test;  
**import**org.openqa.selenium.By;  
**import**org.openqa.selenium.WebDriver;  
**import**org.openqa.selenium.WebElement;  
**import**org.openqa.selenium.firefox.FirefoxDriver;  
  
  
**public class**MultipleWindowsHandle {  
  
     
     WebDriver driver;    
     @Before    
     **public void**setup() **throws**Exception {    
     driver=**new**FirefoxDriver();    
     String URL="http://www.seleniummaster.com";     
     driver.get(URL);    
     driver.manage().window().maximize();    
     }    
     @Test    
     **public void**test() **throws**Exception {     
     // Opening site    
     driver.findElement(By.xpath("//img[@alt='SeleniumMasterLogo']")).click();    
     // Storing parent window reference into a String Variable    
     String Parent\_Window = driver.getWindowHandle();      
      // Switching from parent window to child window     
     **for**(String Child\_Window : driver.getWindowHandles())    
     {    
     driver.switchTo().window(Child\_Window);    
     // Performing actions on child window    
     driver.findElement(By.id("dropdown\_txt")).click();    
     List  dropdownitems=driver.findElements(By.xpath("//div[@id='DropDownitems']//div"));    
     **int**dropdownitems\_Size=dropdownitems.size();    
     System.out.println("Dropdown item size is:"+dropdownitems\_Size);    
     ((WebElement) dropdownitems.get(1)).click();    
     driver.findElement(By.xpath("//\*[@id='anotherItemDiv']")).click();    
     }    
     //Switching back to Parent Window    
     driver.switchTo().window(Parent\_Window);    
     //Performing some actions on Parent Window    
     driver.findElement(By.className("btn\_style")).click();    
     }    
      @After    
      **public void**close() {    
      driver.quit();    
      }     
     }

how to lanuch webpage using chrome driver

import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;  
public class Chrome {  
WebDriver driver;  
@Before  
public void launchChrome()  
{  
System.setProperty("webdriver.chrome.driver", "E:\\DD MISHRA\\workspace\\chromedriver\_win\_26.0.1383.0\\chromedriver.exe");  
driver = new ChromeDriver();  
}  
@Test  
public void testChrome()  
{  
driver.get("http://www.google.co.in");  
driver.findElement(By.id("gbqfq")).sendKeys("Selenium");  
}  
@After  
public void kill()  
{  
driver.close();  
driver.quit();  
}  
}

what is desired capabilities in selenium webdriver

The desired capability is a series of key/value pairs that stores the browser properties like browsername, browser version, the path of the browser driver in the system, etc. to determine the behaviour of the browser at run time.

* Desired capability can also be used to configure the driver instance of Selenium WebDriver.
* We can configure driver instance like FirefoxDriver, ChromeDriver, InternetExplorerDriver by using desired capabilities.

Desired Capabilities are more useful in cases like:

* In mobile application automation, where the browser properties and the device properties can be set.
* In Selenium grid when we want to run the test cases on a different browser with different operating systems and versions.

**Different types of Desired Capabilities Methods**

Here we will see a different type of desired capabilities methods and see how to use one of this method "**setCapability Method".**

1. **getBrowserName()**

public java.lang.String getBrowserName()

1. **setBrowserName()**

public void setBrowserName(java.lang.String browserName)

1. **getVersion()**

public java.lang.String getVersion()

1. **setVersion()**

public void setVersion(java.lang.String version)

1. **getPlatform()**

public Platform getPlatform()

1. **setPlatform()**

public Platform getPlatform()

1. **getCapability Method**

The getCapability method of the Desired Capabilities class can be used to get the capability that is in use currently in the system.

public java.lang.Object getCapability(java.lang.String capabilityName)

1. **setCapabilityMethod**

The setCapability() method of the Desired Capabilities class can be used to set the device name, platform version, platform name, absolute path of the app under test (the .apk file of the app(Android) under test), app Activity (in Android) and appPackage(java).

setCapability : public void setCapability(java.lang.String capabilityName,boolean value)

setCapability :public void setCapability(java.lang.String capabilityName,java.lang.String value)

setCapability :public void setCapability(java.lang.String capabilityName,Platform value)

setCapability :public void setCapability(java.lang.String key,java.lang.Object value)

how to set language while opening website

private WebDriver driver;

public enum Language {en-us, de}

public WebDriver getDriver(Language lang){

String locale = lang.toString();

FirefoxProfile profile = new FirefoxProfile();

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

driver = new FirefoxDriver(profile);

return driver;

}

@Test

public void TestNumber(){

WebDriver drv = getDriver(Language.en);

drv.get("http://the-site.com");

WebElement el = drv.findElement //... find element

String number = el.getText();

Assert.assertEquals(number, "123.45");

drv.close();

drv = getDriver(Language.de);

drv.get("http://the-site.com");

WebElement el = drv.findElement //... find element

String number = el.getText();

Assert.assertEquals(number, "123,45");

drv.close();

}

how to handle windows based popups (upload and dropdown)

WebDriver offers the users with a very efficient way to handle these pop ups using Alert interface.

**There are the four methods that we would be using along with the Alert interface.**

* 1. **void dismiss()** – The dismiss() method clicks on the “Cancel” button as soon as the pop up window appears.

**2) void accept()** – The accept() method clicks on the “Ok” button as soon as the pop up window appears.  
**3) String getText()** – The getText() method returns the text displayed on the alert box.  
**4) void sendKeys(String stringToSend)** – The sendKeys() method enters the specified string pattern into the alert box.

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

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

|  |
| --- |
| import org.openqa.selenium.By; |

|  |  |
| --- | --- |
| 2 | import org.openqa.selenium.WebDriver; |

|  |  |
| --- | --- |
| 3 | import org.openqa.selenium.WebElement; |

|  |  |
| --- | --- |
| 4 | import org.openqa.selenium.firefox.FirefoxDriver; |

|  |  |
| --- | --- |
| 5 |  |

|  |  |
| --- | --- |
| 6 | public class Gmail\_Login { |

|  |  |
| --- | --- |
| 7 | /\*\* |

|  |  |
| --- | --- |
| 8 | \* @param args |

|  |  |
| --- | --- |
| 9 | \*/ |

|  |  |
| --- | --- |
| 10 | public static void main(String[] args) { |

|  |  |
| --- | --- |
| 11 |  |

|  |  |
| --- | --- |
| 12 | // objects and variables instantiation |

|  |  |
| --- | --- |
| 13 | WebDriver driver = new FirefoxDriver(); |

|  |  |
| --- | --- |
| 14 | String appUrl ="https://accounts.google.com"; |

|  |  |
| --- | --- |
| 15 |  |

|  |  |
| --- | --- |
| 16 | // launch the firefox browser and open the application url |

|  |  |
| --- | --- |
| 17 | driver.get(appUrl); |

|  |  |
| --- | --- |
| 18 |  |

|  |  |
| --- | --- |
| 19 | // maximize the browser window |

|  |  |
| --- | --- |
| 20 | driver.manage().window().maximize(); |

|  |  |
| --- | --- |
| 21 |  |

|  |  |
| --- | --- |
| 22 | // declare and initialize the variable to store the expected title of the webpage. |

|  |  |
| --- | --- |
| 23 | String expectedTitle = " Sign in - Google Accounts "; |

|  |  |
| --- | --- |
| 24 |  |

|  |  |
| --- | --- |
| 25 | // fetch the title of the web page and save it into a string variable |

|  |  |
| --- | --- |
| 26 | String actualTitle = driver.getTitle(); |

|  |  |
| --- | --- |
| 27 |  |

|  |  |
| --- | --- |
| 28 | // compare the expected title of the page with the actual title of the page and print the result |

|  |  |
| --- | --- |
| 29 | if (expectedTitle.equals(actualTitle)) |

|  |  |
| --- | --- |
| 30 | { |

|  |  |
| --- | --- |
| 31 | System.out.println("Verification Successful - The correct title is displayed on the web page."); |

|  |  |
| --- | --- |
| 32 | } |

|  |  |
| --- | --- |
| 33 | else |

|  |  |
| --- | --- |
| 34 | { |

|  |  |
| --- | --- |
| 35 | System.out.println("Verification Failed - An incorrect title is displayed on the web page."); |

|  |  |
| --- | --- |
| 36 | } |

|  |  |
| --- | --- |
| 37 |  |

|  |  |
| --- | --- |
| 38 | // enter a valid username in the email textbox |

|  |  |
| --- | --- |
| 39 | WebElement username = driver.findElement(By.id("Email")); |

|  |  |
| --- | --- |
| 40 | username.clear(); |

|  |  |
| --- | --- |
| 41 | username.sendKeys("TestSelenium"); |

|  |  |
| --- | --- |
| 42 |  |

|  |  |
| --- | --- |
| 43 | // enter a valid password in the password textbox |

|  |  |
| --- | --- |
| 44 | WebElement password = driver.findElement(By.id("Passwd")); |

|  |  |
| --- | --- |
| 45 | password.clear(); |

|  |  |
| --- | --- |
| 46 | password.sendKeys("password123"); |

|  |  |
| --- | --- |
| 47 |  |

|  |  |
| --- | --- |
| 48 | // click on the Sign in button |

|  |  |
| --- | --- |
| 49 | WebElement SignInButton = driver.findElement(By.id("signIn")); |

|  |  |
| --- | --- |
| 50 | SignInButton.click(); |

|  |  |
| --- | --- |
| 51 |  |

|  |  |
| --- | --- |
| 52 | // close the web browser |

|  |  |
| --- | --- |
| 53 | driver.close(); |

|  |  |
| --- | --- |
| 54 | System.out.println("Test script executed successfully."); |

|  |  |
| --- | --- |
| 55 |  |

|  |  |
| --- | --- |
| 56 | // terminate the program |

|  |  |
| --- | --- |
| 57 | System.exit(0); |

|  |  |
| --- | --- |
| 58 | } |

|  |  |
| --- | --- |
| 59 | } |

how to select items from dropdown/select box

WebElement mySelectElement = driver.findElement(By.id("mySelect"));

Select dropdown= new Select(mySelectElement);

dropdown.selectByVisibleText("Italy");

WebElement option = dropdown.getFirstSelectedOption();

System.out.println(option.getText()); //output "Italy"

### Or to select an option you can do:

By text:

dropdown.selectByVisibleText("Italy");

By index:

dropdown.selectByIndex(2);

By value:

dropdown.selectByValue("option2");

how to know if checkbox is checked or not in webpage

boolean checked = driver.findElement(By.xpath("//xpath of the checkbox")).isSelected();

**or**

**boolean** actualValue = e.isSelected();

//

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

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

**import** org.openqa.selenium.support.ui.Select;

**public** **class** first {

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

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

System.*setProperty*("webdriver.chrome.driver", "C:\\Selenuim\\chromedriver2.3.exe");

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

**try**{

driver.get("http://register.rediff.com/register/register.php");

Thread.*sleep*(2000);

WebElement e = driver.findElement(By.*name*("chkemail"));

**boolean** actualValue = e.isSelected();

**if** (actualValue)

       System.*out*.println("Checkbox is selected");

**else**

       System.*out*.println("Checkbox is not selected");

Thread.*sleep*(2000);

}

**catch**(Exception ex){

       System.*out*.println("Exception " + ex.getMessage());

              }

**finally**{

                     driver.close();

                     driver.quit();

              }

       }

}

//

tell me code to pass values from parent window to child window

String mainwindow=driver.getWindowHandle();

driver.findElement(By.xpath("//img[contains(@src,'/i/lov\_16x16.gif')]")).click();

System.out.println(mainwindow);

for (String winHandle : driver.getWindowHandles())

driver.switchTo().window(winHandle);

{ if( driver.getTitle().equals("List of Values Search"))

System.out.println("You are in required Window");

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

driver.findElement(By.xpath("html/body/form/div[1]/input[1]")).sendKeys("583 - WALNUT DISTRIBUTION CENTER");

driver.findElement(By.cssSelector("input[type='button'][value='Search']"));

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

driver.switchTo().window(mainwindow);

write code to find out if all links are working or not

**package** selenium.tests;

**import** **java.util.List**;

**import** **java.util.concurrent.TimeUnit**;

**import** **org.openqa.selenium.\***;

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

**import** **org.openqa.selenium.support.ui.ExpectedConditions**;

**import** **org.openqa.selenium.support.ui.WebDriverWait**;

**public** **class** **TestAllLinks** {

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

String baseUrl = "http://www.qaautomated.com/";

System.setProperty("webdriver.chrome.driver",

 "C:\\Users\\chromedriver\_win32\\chromedriver.exe");

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

String notWorkingUrlTitle = "Under Construction: QAAutomated";

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

driver.get(baseUrl);

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

String[] linkTexts = **new** String[linkElements.size()];

**int** i = **0**;

//extract the link texts of each link element

**for** (WebElement elements : linkElements) {

linkTexts[i] = elements.getText();

i++;

}

//test each link

**for** (String t : linkTexts) {

driver.findElement(By.linkText(t)).click();

**if** (driver.getTitle().equals(notWorkingUrlTitle )) {

System.out.println("\"" + t + "\""

+ " is not working.");

} **else** {

System.out.println("\"" + t + "\""

+ " is working.");

}

driver.navigate().back();

}

driver.quit();

}

}

write code on how to use javascriptexecutor?

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class JavaScriptExecuter {

public static void main(String[] args) {

WebDriver driver = new FirefoxDriver();

//Launching the browser application

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

//Adding wait

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

//Maximize window

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

//Creating the Javascriptexecutor interface object by Type casting

JavascriptExecutor js = (JavascriptExecutor)driver;

//Fetching the Domain Name

String sDomain = js.executeScript("return document.domain;").toString();

System.out.println("Domain = "+sDomain);

//Fetching the URL

String sURL = js.executeScript("return document.URL;").toString();

System.out.println("URL = "+sURL);

//Fetching the Title

String sTitle = js.executeScript("return document.title;").toString();

System.out.println("Title = "+sTitle);

//Vertical scroll - down by 200 pixels

js.executeScript("window.scrollBy(0,200)");

System.out.println("Successfully did the vertical scroll by 200px");

}

}

difference between assert and verify?

**Assert command in selenium:**

* When an "assert" command fails then test execution will be aborted.
* Assert is best used when the check value has to pass for the test to be able to continue to run. Like a log in.

**Verify command in selenium:**

* When a "verify" command fails then test will continue executing and logging the failure.
* Verify is best used to check non critical things. Like hte presence of a headline element.

difference between driver.close and driver.quit methods?

driver.close and driver.quit are two different methods for closing the browser session in Selenium WebDriver.

* driver.close – It closes the the browser window on which the focus is set.
* driver.quit – It basically calls driver.dispose method which in turn closes all the browser windows and ends the WebDriver session gracefully.

**Code for driver.close():**

**public** **class** BrowserClose(){

*//Declare Selenium WebDriver variables*

**public** **static** WebDriver driver;

**public** **static** String baseUrl;

@BeforeTest

**public** **void** setup(){

driver = **new** FirefoxDriver();

baseUrl="http://www.wikishown.com";

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

}

@Test

**public** **void** wikiShown(){

*//Navigate to above website urk*

driver.Navigate().to(baseUrl);

driver.manage().timeouts().implecitlyWait(30,TimeUnit.SSECONDS);

}

@AfterTest

**public** **void** tearDown(){

driver.close();

}

}

**Code for driver.quit():**

**public** **class** BrowserClose(){

*//Declare Selenium WebDriver variables*

**public** **static** WebDriver driver;

**public** **static** String baseUrl;

@BeforeTest

**public** **void** setup(){

driver = **new** FirefoxDriver();

baseUrl="http://www.wikishown.com";

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

}

@Test

**public** **void** wikiShown(){

*//Navigate to above website urk*

driver.Navigate().to(baseUrl);

driver.manage().timeouts().implecitlyWait(30,TimeUnit.SSECONDS);

}

@AfterTest

**public** **void** tearDown(){

driver.quit();

}

}

common exceptions in selenium?

**NoSuchElement** : An element could not be located on the page using the given search parameters.

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

**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.

**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).

how to handle Ajax calls in selenium?

Ajax calls in selenium can be handled using the following wait methods,

1. **Thread.Sleep()**
2. **Implicit Wait()**
3. **Explicit Wait()**
4. **WebdriverWait**
5. **Fluent Wait**

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:

How to assign the value to textbox other than sendkeys method?

There are two ways:

Execute javascript

WebDriver driver = new FirefoxDriver();

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

JavascriptExecutor myExecutor = ((JavascriptExecutor) driver);

myExecutor.executeScript("document.getElementsByName('q')[0].value='Kirtesh'", searchbox);

driver.quit();

Passing webElement as an argument to the javascript

WebDriver driver = new FirefoxDriver();

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

WebElement searchbox = driver.findElement(By.xpath("//input[@name='q']"));

JavascriptExecutor myExecutor = ((JavascriptExecutor) driver);

myExecutor.executeScript("arguments[0].value='Kirtesh';", searchbox);

driver.quit();

Selenium grid, how to execute scripts on multiple browser

package myPackage;

import java.net.MalformedURLException;

import java.net.URL;

import org.openqa.selenium.By;

import org.openqa.selenium.Platform;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.remote.DesiredCapabilities;

import org.openqa.selenium.remote.RemoteWebDriver;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class TestParallel extends BaseClass {

    @Test

    public void test\_01() throws InterruptedException, MalformedURLException{

        try{

            getDriver().get("http://www.w3schools.com/");

            getDriver().findElement(By.xpath("html/body/div[2]/div/a[4]")).click();

            //Wait intentially added to show parallelism execution

            Thread.sleep(10000);

            getDriver().findElement(By.xpath("//\*[@id='gsc-i-id1']")).sendKeys("test");

            Thread.sleep(5000);

        }

        catch(Exception e){

            System.out.println(e);

        }

    }

}