**What are limitations in selenium web driver?**

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

3. You need to know at least one of the supported language very well in order to automate your application successfully.  
4. No inbuilt reporting capability so you need plugins like JUnit and Test NG for test reports.  
5. Lot of challenges with IE browser.

**Installing/configure selenium**

Selenium with Java

Step 1 - Install Java on your computer

## Step 2 - Install Eclipse IDE

## Step 3 - Download the Selenium Java Client Driver

## Step 4 - Configure Eclipse IDE with WebDriver

## Download Selenium

## 1) Open url : http://www.seleniumhq.org/download/

## 2) Select java language click on download

## 3) Extract folder - copy folder paste in c driver

## 4) Mozilla fire fox download

## 5) Adding Add on features to Mozilla

## Fire Bug -- open Mozilla - type firebug for Firefox - click on add

## Fire path-- Open Mozilla - type fire path for Firefox

## Selenium IDE-- Open Mozilla - type the selenium ide for Firefox

**What are different ways of locating elements in selenium**

Selenium WebDriver API supports different possibilities to identify elements: by ID, by CLASS, by NAME, by CSS selector, by XPath, by TAG name. Also you define your custom selector in order to interact with the elements.

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

**2. By CLASS** driver.findElement(By.className("element class"))

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

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

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

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

**7. By XPath:** driver.findElement(By.xpath("xpath expression"))

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

The most efficient way and preferred way to locate an element on a web page is **By ID** will be the unique on web page which can be easily identified. IDs are the safest and fastest locator option and should always be the first choice even when there are multiple choices, it is like an Employee Number or Account which will be unique.

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

**Absolute path**: When the xpath starts from html , then it become absolute xpath.

html/body/div[5]/div[2]/div/div[2]/div[2]/h2[1]

**Relative path :** and a relative xpath finds the closed id to the dom element and generates xpath starting from that element.

.//\*[@id='answers']/h2[1]/a[1]

**Different types of waits or synchronization in selenium web driver**

When implementing time synchronization for waiting with Selenium Web Driver technology, we can use two types of waits:

**Implicit Wait:** In Implicit Wait, we define a code to wait for a certain amount of time when trying to find an element or elements. If the Web Driver cannot find it immediately because of its availability, the WebDriver will wait. The default setting is zero. Once we set a time, the Web Driver waits for the period of the WebDriver object instance.

For example:

### public void ImplicitWait(int waitSeconds)

### {

### WebDriver driver = new FirefoxDriver(); driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(waitSeconds)); driver.Url = "http://www.seleniummaster.com"; IWebElement elementToWait = driver.FindElement(By.Id("theElementId"));

### }

**Explicit Wait :** In Explicit Wait, we write a code to define a wait statement for certain condition to be satisfied until the wait reaches its timeout period. If WebDriver can find the element before the defined timeout value, the code execution will continue to next line of code. Therefore, it is important to setup a reasonable timeout seconds according to the system response.

public void ExplicitWait(int timeouseconds)

{

IWebDriver driver = new FirefoxDriver();  
driver.Url = "http://www.seleniummaster.com";  
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(timeouseconds));  
IWebElement elementToWait = wait.Until<IWebElement>((d) =>  
    {  
        return d.FindElement(By.Id("theElementId"));  
    });

}

**How to save screenshots using selenium web driver**

It’s very important to take screenshot when we execute a test script. When we execute huge number of test scripts, and if some test fails, we need to check why the test has failed. It helps us to debug and identify the problem by seeing the screen shot. In selenium web driver, we can take the screen shot using the below command.

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

Check the framework example of  [Taking ScreenShot for ONLY Failed Tests using TestNG](http://seleniumeasy.com/testng-tutorials/how-to-take-screenshot-for-only-failed-tests-using-webdriver)  
The below example explains how to take the screen shot when the test fails.

import java.io.File;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.By;

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 takeScreenShotExample{

public WebDriver driver;

@Test

public void openBrowser() throws Exception {

driver = new FirefoxDriver();

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

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

try{

//the below statement will throw an exception as the element is not found, Catch block will get executed and takes the screenshot.

driver.findElement(By.id("testing")).sendKeys("test");

//if we remove the below comment, it will not return exception and screen shot method will not get executed.

//driver.findElement(By.id("gbqfq")).sendKeys("test");

}

catch (Exception e){

System.out.println("I'm in exception");

//calls the method to take the screenshot.

getscreenshot();

}

}

public void getscreenshot() throws Exception

{

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("D:\\screenshot.png"));

}

}

**How to handle multiple windows in selenium web driver**

When we have multiple windows in test automation, all we need to do is switching the focus from one window to another. Let us understand the same in the following way:

Window A has a link "Link1" and we need to click on the link (click event).

Window B displays and we perform some actions.

The entire process can be fundamentally segregated into following steps :

Step 1 : Clicking on Link1 on Window A

A new Window B is opened.

Step 2 : Save reference for Window A

Step 3 : Create reference for Window B

Step 3 : 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

Let us understand the same with a small coding example.

**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**

In WebDriver, We launch FireFox and Internet Explorer by using

***WebDriver driver = new FirefoxDriver(); //this line would launch Firefox  
WebDriver driver = new InternetExplorerDriver(); //this line would launch IE browser***

But when we write below line like FireFox and IE

**WebDriver driver = new ChromeDriver();**  
Then It throws Error and Here I am pasting Error Trace shown in Eclipse

java.lang.IllegalStateException: The path to the driver executable must be set by the webdriver.chrome.driver system property; for more information, see http://code.google.com/p/selenium/wiki/ChromeDriver. The latest version can be downloaded from http://code.google.com/p/chromedriver/downloads/list

But there is one way to resolve this Error and this could be done by using this

1- Download zip file of chromedriver for Windows from [here](http://code.google.com/p/chromedriver/downloads/detail?name=chromedriver_win_26.0.1383.0.zip&can=2&q=)2- Unzip downloaded Chromedriver for Windows and find the absolute path of chromedriver.exe  
3- Now set Property of System by using this line  
**System.setProperty(“webdriver.chorme.driver**,**"path of the exe file\\chromedriver.exe");  
and after this line write your traditional line to launch the browser like this  
WebDriver driver =new ChromeDriver();**

## Sample Program for Launch Chrome Browser using Selenium Webdriver :

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class TestChrome {

public static void main(String[] args) {

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

// Initialize browser

WebDriver driver=new ChromeDriver();

// Open facebook

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

// Maximize browser

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

}

}

**What is desired capabilities in selenium web driver?**

The desired capability is a series of key/value pairs that stores the browser properties like browser name, browser version, the path of the browser driver in the system, etc. to determine the behavior 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 Firefox Driver, Chrome driver, Internet Explorer Driver 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".

getBrowserName()

public java.lang.String getBrowserName()

setBrowserName()

public void setBrowserName(java.lang.String browserName)

getVersion()

public java.lang.String getVersion()

setVersion()

public void setVersion(java.lang.String version)

getPlatform()

public Platform getPlatform()

setPlatform()

public Platform getPlatform()

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)

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 method" in java has the below declarations:**

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**

**Internationalization** is a process of designing a software application so that it can be adapted to various languages and regions without any changes

**Localization** is a process of adapting internationalized software for a specific region or language by adding local specific components and translating text.

If we want to check whether our application is properly internationalized , then we will manually change the language preferences in the browser itself.But if we want to check the same using WebDriver then we have to change the user language preferences.

**Using Firefox Browser:**

FirefoxProfile profile = new FirefoxProfile();  
//setting the locale french : ‘fr’  
profile.setPreference(“intl.accept\_languages”,”fr”);  
driver = new FirefoxDriver(profile);  
driver.get(“[http://google.co.in&#8221](#8221););

**Using Chrome Browser:**

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

ChromeOptions options = new ChromeOptions();  
options.addArguments(“–lang= sl”);  
ChromeDriver driver = new ChromeDriver(options);  
driver.get(“[http://google.co.in&#8221](#8221););

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

There are many cases, where an application displays multiple windows when you open a website. Those are may be advertisements or may be a kind of information showing on popup windows. We can handle multiple windows using Windows Handlers in selenium web driver.

**Step 1:** After opening the website, we need to get the main window handle by using driver.getWindowHandle();  
The window handle will be in a form of lengthy alpha numeric  
**Step 2:** We now need to get all the window handles by using driver.getWindowHandles();  
**Step 3:** We will compare all the window handles with the main Window handles and perform the operation the window which we need.

The below example shows how to handle multiple windows and close all the child windows which are not need. We need to compare the main window handle to all the other window handles and close them.

package com.pack;

import java.util.Set;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.Test;

public class WindowExamples {

static WebDriver driver;

@Test

public void test\_CloseAllWindowsExceptMainWindow() {

driver = new FirefoxDriver();

// It will open Naukri website with multiple windows

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

// To get the main window handle

String windowTitle= getCurrentWindowTitle();

String mainWindow = getMainWindowHandle(driver);

Assert.assertTrue(closeAllOtherWindows(mainWindow));

Assert.assertTrue(windowTitle.contains("Jobs - Recruitment"), "Main window title is not matching");

}

public String getMainWindowHandle(WebDriver driver) {

return driver.getWindowHandle();

}

public String getCurrentWindowTitle() {

String windowTitle = driver.getTitle();

return windowTitle;

}

//To close all the other windows except the main window.

public static boolean closeAllOtherWindows(String openWindowHandle) {

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

for (String currentWindowHandle : allWindowHandles) {

if (!currentWindowHandle.equals(openWindowHandle)) {

driver.switchTo().window(currentWindowHandle);

driver.close();

}

}

driver.switchTo().window(openWindowHandle);

if (driver.getWindowHandles().size() == 1)

return true;

else

return false;

}

}

**Write code to verify any application login page is working or not**

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

We will create a Project called “Gmail” , Package as “login”, Class as “Login1”. Now we will write the code as below

package login;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class Login1 {

public static void main(String[] args) {

**// Create a new instance of the Firefox driver**

WebDriver driver = new FirefoxDriver();

**//  Wait For Page To Load**

**// Put a Implicit wait, this means that any search for elements on the page**

**could take the time the implicit wait is set for before throwing exception**

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

**// Navigate to URL**

driver.get("https://mail.google.com// Maximize the window.

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

**// Enter UserName**

driver.findElement(By.id("Email")).sendKeysER NAME");

**// Enter Password**

driver.findElement(By.id("Passwd")).sendKeys("YOUR Wait For Page To Load

driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);

**// Click on 'Sign In' button**

driver.findElement(By.id("signIn")).clickk on Compose Mail.

driver.findElement(By.xpath("//divss='z0']/div")).click();

**// Click on the image icon present in the top right navigational Bar**

driver.findElement(By.xpath("//divss='gb\_1 gb\_3a gb\_nc gb\_e']/div/a")).click();

**//Click on 'Logout' Button**

driver.findElement(By.xpath("//\*[@id='gb\_71']")).click();

**//Close the browser.**

driver.close();

}

}

**how to select items from dropdown/select box**

<select id="mySelectID">

<option value="Value">Option</option>

<option value="NotValue">Not Option</option>

</select>

## Select by option name

Java:

WebElement mySelectElm = driver.findElement(By.id("mySelectID"));

Select mySelect= new Select(mySelectElm);

selMySelect.selectByVisibleText("Option");

## Select by option value

Java:

## WebElement mySelectElm = driver.findElement(By.id("mySelectID"));

## Select mySelect= new Select(mySelectElm);

## selMySelect.selectByValue("Value");

## Select by index

Java:

WebElement mySelectElm = driver.findElement(By.id("mySelectID"));

Select mySelect= new Select(mySelectElm);

selMySelect.selectByIndex(0);

## Get the selected option

Java:

WebElement mySelectElm = driver.findElement(By.id("mySelectID"));

Select mySelect= new Select(mySelectElm);

WebElement option = mySelect.getFirstSelectedOption();

System.out.println(option.getText()); //prints "Option"

## Get the list of options

## Java:

## WebElement mySelectElm = driver.findElement(By.id("mySelectID"));

## Select mySelect= new Select(mySelectElm);

## List<WebElement> options = mySelect.getOptions();

## for (WebElement option : options) {

## System.out.println(option.getText()); //Prints "Option", followed by "Not Option"

## }

**How to know if checkbox is checked or not in webpage?**

We can use the below methods to check or uncheck web checkbox.

**//Checking**

public void CheckingChkbox(WebElement chkbx1){  
boolean checkstatus;  
checkstatus=chkbx1.isSelected();  
if (checkstatus==true){  
System.out.println("Checkbox is already checked");   
}  
else  
{  
chkbx1.click();  
System.out.println("Checked the checkbox");  
}  
}  
   
**//Unchecking**

public void UnCheckingChkbox(WebElement chkbx1){  
boolean checkstatus;  
checkstatus=chkbx1.isSelected();  
if (checkstatus==true) {  
chkbx1.click();  
System.out.println("Checkbox is unchecked");  
}  
else  
{  
System.out.println("Checkbox is already unchecked");   
}  
}