SELENIUM WEBDRIVER SCRIPTS

WWW.PAVANTESTINGTOOLS.COM | WWW.PAVANONLINETRAININGS.COM

01. WebDriver Basic Commands - Example

```
package Examples;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;
import org.testng.annotations.Test;

public class click {

	@Test

	public void clickmethod()
	{
		//1. Open the Chrome Browser
		System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chr
omedriver_win32\\chromedriver.exe");
		WebDriver driver = new ChromeDriver();

		//2. Using Implicitly Wait Command
		driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

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

		//Get and store page title in to variable
		String title = driver.getTitle();
		System.out.print(title);

		//Get current page URL
		String CurrentURL = driver.getCurrentUrl();
		System.out.println("My Current URL Is  : "+CurrentURL);

		//Get and store domain name in variable using JavaScript Executor
```

```
		JavascriptExecutor javascript = (JavascriptExecutor) driver;
		String DomainUsingJS=(String)javascript.executeScript("return
document.domain");
		System.out.println("My Current URL Is : "+DomainUsingJS);

		// Checked for search box is enabled or not
		if (driver.findElement(By.xpath("//input[@name='q']")).isEnabled())
		{
			System.out.println("Google search text box Is enabled.");

			// Pass the Test - "WebDriver Test Successful" to search box
			driver.findElement(By.xpath("//input[@name='q']")).sendKey
s("WebDriver Test successful.");

			// clicking the search button
			driver.findElement(By.xpath("//button[@name='btnG']")).clic
k();

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

			// Click the Top most link and open to New Tab
			WebElement link=driver.findElement(By.xpath("//div[@id='ir
es']/ol/div/div[1]/div/h3/a"));
			Actions newTab = new Actions(driver);
			newTab.keyDown(Keys.CONTROL).keyDown(Keys.SHIFT).click(
link).keyUp(Keys.CONTROL).keyUp(Keys.SHIFT).build().perform();

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

		}
	}

}
```

02. Navigating Back And Forward

```
package Others;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
public class testsample
{

 @Test
 public void Navigate_forward_back() throws InterruptedException
 {

  //1. Open the Chrome Browser
  System.setProperty("webdriver.chrome.driver",
"D:\\Softwares\\chromedriver_win32\\chromedriver.exe");
  WebDriver driver = new ChromeDriver();

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

  //navigate on specific software web application page or URL
  driver.navigate().to("http://selenium-venkat.blogspot.com/p/index_4.html");

  //To navigate back (Same as clicking on browser back button)
  driver.navigate().back();

  //To navigate forward (Same as clicking on browser forward button)
  driver.navigate().forward();
 }
}
```


03. Capture Screenshot

Capturing screenshot of software web application page is very easy in selenium webdriver.

As we knows, It is very basic required thing in software automation tools to capture screenshot on test case failure or whenever required during test case execution.

In Selenium WebDriver/Selenium 2, we need to capture screenshot of web page using bellow syntax.

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

Then we have to store it in our local drive using bellow given syntax. You can change/provide your own file destination path and name.

FileUtils.copyFile(screenshot, new File("D:\\screenshot.jpg));
Now we need to import bellow given header files to get support of capture screenshot in 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;

Sample Script:

```
package Others;
import java.io.File;
import java.io.IOException;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import org.apache.commons.io.FileUtils;
public class testsample {

	@Test
	public void Capturing_Screenshot() throws InterruptedException,
IOException
	{

		//1. Open the Chrome Browser
		System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chr
omedriver_win32\\chromedriver.exe");
		WebDriver driver = new ChromeDriver();
```

```
                //Open the URL
                driver.get("http://www.google.com");

                // Cature the screenshot
                File screenshot =
((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
                FileUtils.copyFile(screenshot, new File("D:\\screenshot.jpg"));
                System.out.print("Screenshot is captured and stored in your D:
Drive");
        }
}
```

04. Handling Javascript Alerts, Confirmations And Prompts

Alerts, Confirmation and Prompts are very commonly used elements of any software webpage and you must know how to handle all these popups In selenium webdriver software automation testing tool.

Alert Popup

Generally alert message popup display on page of software web application with alert text and Ok button as shown In bellow given Image.

Confirmation Popup Confirmation popup displays on page of software web application with confirmation text, Ok and Cancel button as shown In bellow given Image.

Prompt Popup Prompts will have prompt text, Input text box, Ok and Cancel buttons.

Selenium webdriver software testing tool has Its own Alert Interface to handle all above different popups.
Alert Interface has different methods like accept(), dismiss(), getText(), sendKeys(java.lang.String keysToSend) and we can use all these methods to perform different actions on popups.

```
package Others;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
public class Alerts
{
	@Test
```

```
	public void Alets_Handling() throws InterruptedException
	{

		//Open the Chrome Browser
		System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chr
omedriver_win32\\chromedriver.exe");
		WebDriver driver = new ChromeDriver();

		//Using Implicitly Wait Command
		driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

		// Click the URL
		driver.get("http://only-testing-
blog.blogspot.in/2014/01/textbox.html");

		// Observe in that page have "3 buttons there" 1. Show Me
Confiramation 2. Show Me Alert 3. Show Me Prompt

		//Alert Pop up Handling - click the Show Me Alert Button
		driver.findElement(By.xpath("//input[@value='Show Me
Alert']")).click();

		//To locate alert.
		Alert A1 = driver.switchTo().alert();

		//To read the text from alert popup.
		String Alert1 = A1.getText();
		System.out.println(Alert1);
		Thread.sleep(2000);

		//To accept/Click Ok on alert popup.
		A1.accept();

		//Confirmation Pop up Handling - CLick the Show Me Confiramation
Button
		driver.findElement(By.xpath("//button[@onclick='myFunction()']")).
click();
		Alert A2 = driver.switchTo().alert();
```

```
                String Alert2 = A2.getText();
                System.out.println(Alert2);
                Thread.sleep(2000);

                //To click On cancel button of confirmation box.
                A2.dismiss();

                //Prompt Pop up Handling - Click the Show Me Prompt button
                driver.findElement(By.xpath("//button[contains(.,'Show Me
Prompt')]")).click();
                Alert A3 = driver.switchTo().alert();
                String Alert3 = A3.getText(); System.out.println(Alert3);

                //To type text In text box of prompt pop up.
                A3.sendKeys("This Is John");
                Thread.sleep(2000);
                A3.accept();
        }
}
```


05. Highlighting element in selenium webdriver & Handling Unexpected alert

```
package Others;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
public class HighlightTest
{
        WebDriver driver;
```

```
	@BeforeTest
	public void setup() throws Exception
	{
		System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chr
omedriver_win32\\chromedriver.exe");
		driver = new ChromeDriver();

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

		driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
		driver.get("http://only-testing-
blog.blogspot.in/2014/06/alert_6.html");
	}
	@AfterTest
	public void tearDown() throws Exception
	{
		driver.quit();
	}
	@Test
	public void Text() throws InterruptedException
	{
		//To handle unexpected alert on page load.
		try
		{
			driver.switchTo().alert().dismiss();
		}
		catch(Exception e)
		{
			System.out.println("unexpected alert not present");
		}
		HighlightMyElement(driver.findElement(By.xpath("//input[@name='
fname']")));
		driver.findElement(By.xpath("//input[@name='fname']")).sendKeys(
"fname");
		HighlightMyElement(driver.findElement(By.xpath("//input[@name='
lname']")));
```

```
		driver.findElement(By.xpath("//input[@name='lname']")).sendKeys(
"lname");
		driver.findElement(By.xpath("//input[@type='submit']")).click();
	}

	public void HighlightMyElement(WebElement element) {
		for (int i = 0; i < 10; i++) {
			JavascriptExecutor js = (JavascriptExecutor) driver;
			js.executeScript(
					"arguments[0].setAttribute('style',
arguments[1]);",
					element, "color: red; border: 3px solid yellow;");
			js.executeScript(
					"arguments[0].setAttribute('style',
arguments[1]);",
					element, "");
		}
	}

}
```


06. Reading Font Properties In Selenium WebDriver Using .getCssValue() Method

Sometimes you need to read font properties like font size, font color, font family, font background color etc.. during WebDriver test case execution. Selenium WebDriver Is very wast API and It has many built In methods to perform very small small operations on web page. Reading font property manually Is very simple task using firebug as shown In bellow given Image.

If you wants to read above shown font property In selenium webdriver then you can do It using .getCssValue() Method. You can provide property name (Example : font-family, font-size, font-weight, etc.) with .getCssValue() method to read Its value.

Example:

```
package Others;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
public class FontProperties
{
	WebDriver driver;
	@BeforeTest
	public void setup() throws Exception
	{
		System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chr
omedriver_win32\\chromedriver.exe");
		driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
```

```
		driver.get("http://only-testing-
blog.blogspot.in/2014/05/login.html");
	}
	@AfterTest
	public void tearDown() throws Exception
	{
		driver.quit();
	}
	@Test
	public void Text() throws InterruptedException
	{
		WebElement text = driver.findElement(By.xpath("//h1[contains(text(
),'Example Login Page')]"));
		//Read font-size property and print It In console.
		String fontSize = text.getCssValue("font-size");
		System.out.println("Font Size -> "+fontSize);

		//Read color property and print It In console.
		String fontColor = text.getCssValue("color");
		System.out.println("Font Color -> "+fontColor);

		//Read font-family property and print It In console.
		String fontFamily = text.getCssValue("font-family");
		System.out.println("Font Family -> "+fontFamily);

		//Read text-align property and print It In console.
		String fonttxtAlign = text.getCssValue("text-align");
		System.out.println("Font Text Alignment -> "+fonttxtAlign);

	}


}
```

07. Generating Mouse Hover Event On Main Menu To Click On Sub Menu

To generate mouse hover event in webdriver software testing tool, We can use advanced user interactions API constructor "Actions" with "moveToElement" method. Bunch of syntax to hover mouse on main menu is as bellow. You can replace xpath of an element as per your requirement in bellow given syntax

Actions actions = new Actions(driver);
WebElement moveonmenu = driver.findElement(By.xpath("//div[@id='menu1']/div"));
actions.moveToElement(moveonmenu);
actions.perform();

Above example will only move mouse on targeted element of software web application. It will not perform click operation. To perform click operation on sub menu, we need to use click() action before perform() as shown in bellow example.


```
package Others;
import java.util.concurrent.TimeUnit;
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.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
public class Dynamicalert
{
	WebDriver driver;
	@BeforeTest
	public void setup() throws Exception
	{
		System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chr
omedriver_win32\\chromedriver.exe");
```

```
		driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
		driver.get("http://only-testing-blog.blogspot.in/p/mouse-
hover.html");
	}
	@AfterTest
	public void tearDown() throws Exception
	{
		driver.quit();
	}
	@Test
	public void Text() throws InterruptedException
	{
		Actions actions = new Actions(driver);
		WebElement moveonmenu = driver.findElement(By.xpath("//div[@i
d='menu1']/div"));
		actions.moveToElement(moveonmenu).moveToElement(driver.findE
lement(By.xpath("//div[@id='menu1choices']/a"))).click().perform();
		WebDriverWait wait = new WebDriverWait(driver, 15);
		wait.until(ExpectedConditions.titleContains("Google"));

	}


}
```


08. Page Scroll different ways using Selenium WebDriver

Using JavaScript

Scroll Down:

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

```
WebDriver driver = new FirefoxDriver();
JavascriptExecutor jse = (JavascriptExecutor)driver;
jse.executeScript("scroll(0, 250)"); //y value '250' can be altered
```

Scroll up:

```
JavascriptExecutor jse = (JavascriptExecutor)driver;
jse.executeScript("scroll(250, 0)"); //x value '250' can be altered
```

Scroll bottom of the Page:

```
JavascriptExecutor jse = (JavascriptExecutor)driver;
jse.executeScript("window.scrollTo(0,Math.max(document.documentElement.scrollHeight,doc
ument.body.scrollHeight,document.documentElement.clientHeight));");
```

(or)

```
Actions actions = new Actions(driver);
actions.keyDown(Keys.CONTROL).sendKeys(Keys.END).perform();
```

Full scroll to bottom in slow motion:

```
for (int second = 0;; second++) {
      if(second >=60){
         break;
      }
         ((JavascriptExecutor) driver).executeScript("window.scrollBy(0,400)", ""); //y value '400'
can be altered
         Thread.sleep(3000);
}
```

(or)

```
JavascriptExecutor jse = (JavascriptExecutor)driver;
for (int second = 0;; second++) {
      if(second >=60){
         break;
      }
         jse.executeScript("window.scrollBy(0,800)", ""); //y value '800' can be altered
         Thread.sleep(3000);
}
```

Scroll automatically to your WebElement:

```
Point hoverItem =driver.findElement(By.xpath("Value")).getLocation();
```

```
((JavascriptExecutor)driver).executeScript("return window.title;");
Thread.sleep(6000);
((JavascriptExecutor)driver).executeScript("window.scrollBy(0,"+(hoverItem.getY())+");");
// Adjust your page view by making changes right over here (hoverItem.getY()-400)
```

(or)

```
((JavascriptExecutor)driver).executeScript("arguments[0].scrollIntoView();",
driver.findElement(By.xpath("Value')]")));
```

Using KeyBoard

We have two options for scrolling in web page.
Using Actions Class

package name : org.openqa.selenium.interactions.Actions
java code :

```
Ctrl+End | Scroll to Bottom of the page
Actions actions = new Actions(driver);
actions.keyDown(Keys.CONTROL).sendKeys(Keys.END).perform();
```

Without Using Actions Class
java code :

```
for(int i=0;i<10;i++)
{
    driver.findElement(By.tagName("body")).sendKeys(Keys.DOWN
);
}
```

Example On - **Full scroll to bottom in slow motion**

```
package Others;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
```

```
import org.testng.annotations.Test;
public class Dynamicalert
{
	WebDriver driver;
	@BeforeTest
	public void setup() throws Exception
	{
		System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chr
omedriver_win32\\chromedriver.exe");
		driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
		driver.get("http://selenium-venkat.blogspot.com/p/index_4.html");
	}
	@AfterTest
	public void tearDown() throws Exception
	{
		driver.quit();
	}
	@Test
	public void Text() throws InterruptedException
	{
		for (int second = 0;; second++) {
			if(second >=60){
				break;
			}
			((JavascriptExecutor) driver).executeScript("window.scrollBy(
0,400)", ""); //y value '400' can be altered
			Thread.sleep(3000);
		}
	}
}
```


09. Extracting All Links From Page

```
package Others;
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
public class ExtractLink
{
	WebDriver driver;
	@BeforeTest
	public void setup() throws Exception
	{
		System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chr
omedriver_win32\\chromedriver.exe");
		driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
		driver.get("http://www.google.com");
	}
	@AfterTest
	public void tearDown() throws Exception
	{
		driver.quit();
	}
	@Test
	public void Text() throws InterruptedException
	{
		try
		{
			List<WebElement> no = driver.findElements(By.tagName("a")
);
			int nooflinks = no.size();
			System.out.println(nooflinks);
```

```
            for (WebElement pagelink : no)
            {
                String linktext = pagelink.getText();
                String link = pagelink.getAttribute("href");
                System.out.println(linktext+" ->");
                System.out.println(link);
            }
        }catch (Exception e){ System.out.println("error "+e); } }

}
```

10. Read CSV Files Using Selenium Webdriver

What is CSV files.

CSV stands for comma separated values. Sometimes in your application you have to take data from existing csv files as well. Here is how csv files looks.

How to read CSV files

In this post we will use some third party API called opencsv, we can download this as jar file and can use existing methods to read file

download opencsv using below link

http://www.java2s.com/Code/Jar/o/Downloadopencsv23jar.htm

This will come as rar file extract this then you will find jar

Add that jar into project

How to add jar -Right click on project > Select Build path > Select configure build path> Add external jar> now Select the jar which we downloaded

Steps How to read-

1- We have predefined class called CSVReader, create an object and pass the csv file path

2- call readAll() method which will return the csv content in List<String[]>

3-using Iterator, you can iterate all the values and use according to application

Program – How to read CSV files using Java

```
package Others;
import java.io.FileReader;
import java.util.Iterator;
import java.util.List;
```

```
import au.com.bytecode.opencsv.CSVReader;

public class csv {

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

    @SuppressWarnings("resource")
    CSVReader reader = new CSVReader(new FileReader("D:\\WebDriver\\Input\\abhi.csv"));

    List<String[]> li=reader.readAll();
    System.out.println("Total rows which we have is "+li.size());

    Iterator<String[]>i1= li.iterator();

    while(i1.hasNext())
    {
        String[] str=i1.next();
        System.out.print(" Values are ");
        for(int i=0;i<str.length;i++)
        {
            System.out.println(" "+str[i]);
        }
        System.out.println("   ");

    }

}
}
```


11. Read excel file in selenium using JExcel

As we, all know Selenium support only browser automation, so for reading and writing with Excel files we have to user third party API like JExcel and Apache POI.
JExcel- JExcel is free API to work with Excel files; it comes with predefined method, classes and interface, which we can directly use.

Limitation of JExcel- It only support .xls files it means files whose extension is .xls files
some useful classes and interface, which we will be using
Workbook (Class) – Will handle workbook.
Sheet (Interface) – Which will handle sheets
Cell (Interface) – Which will handle cell

Download JExcel API

http://www.java2s.com/Code/Jar/j/Downloadjxljar.htm

```
package Others;
import java.io.File;
import java.io.IOException;
import jxl.Cell;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;
import org.testng.annotations.Test;
public class excel {

	@Test
	public void TestReadData(){

		// You need to use File class which will ask for file location.I specified  base// directory //using dot(.) operator then inside data folder I have testdata.xls// stored

		File src=new File("D:\\excelsample.xls");

		try {
			// Workbook is a class in Jexcel which will take file as an argument and getWork//book is a predefined method which will read the workbook and will return the w//Workbook object

			Workbook wb=Workbook.getWorkbook(src);

			// Workbook is loaded now we have to load sheet so using workbook object (wb) we// can call getSheet method which will take index as an argument and will load t//he sheet, we can also specify the sheetname also

			Sheet sh1=wb.getSheet(0);

			// Sheet is loaded then we have to read cell so using sh1 object call getCell me//thod which we take two arguments getCell(column,row)

			Cell c1=sh1.getCell(0,0);
```

```
                    //Cell is loaded then using getContents method we have  to
extract the data usin//g getContents() methods
                    // this method will always return you String.
                    // now you are done

                    String data1=c1.getContents();

                    System.out.println(data1);

                    System.out.println("Sheet data is "+data1);

                } catch (BiffException e) {

                    e.printStackTrace();

                }
                catch (IOException e){
                    e.printStackTrace();
                }
        }
}
```


12. Read/Write excel files in Selenium using Apache POI

Step 1- Download apache poi jar file as below

http://www.apache.org/dyn/closer.lua/poi/dev/bin/poi-bin-3.14-beta1-20151223.zip

All jar files will come in zip files, Extract it and you will get final jar folder
Step 2- How to add Jar files
Select project then Right click on project > Build path > Configure build path > select jar Files

Note: Do not forget to add the jar file inside of the “ooxml-lib” folder.

Precondition- Create a xlsx file and enter some data to read and save file at perticuler location.

```
package Others;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public class excel_poi {

	public static void main(String []args){

		try {
			// Specify the path of file
			File src=new File("D:\\sample.xlsx");

			// load file
			FileInputStream fis=new FileInputStream(src);

			// Load workbook

			XSSFWorkbook wb=new XSSFWorkbook(fis);

			// Load sheet- Here we are loading first sheetonly
			XSSFSheet sh1= wb.getSheetAt(0);

			// getRow() specify which row we want to read.

			// and getCell() specify which column to read.
			// getStringCellValue() specify that we are reading String data.


			System.out.println(sh1.getRow(0).getCell(0).getStringCellValu
e());

			System.out.println(sh1.getRow(0).getCell(1).getStringCellValu
e());

			System.out.println(sh1.getRow(1).getCell(0).getStringCellValu
e());
```

```
                System.out.println(sh1.getRow(1).getCell(1).getStringCellValu
e());

                System.out.println(sh1.getRow(2).getCell(0).getStringCellValu
e());

                System.out.println(sh1.getRow(2).getCell(1).getStringCellValu
e());

                // here createCell will create column

                // and setCellvalue will set the value

                 sh1.getRow(0).createCell(2).setCellValue("2.41.0");

                 sh1.getRow(1).createCell(2).setCellValue("2.5");

                 sh1.getRow(2).createCell(2).setCellValue("2.39");

                // here we need to specify where you want to save file

                 FileOutputStream fout=new FileOutputStream(new File("D:\\
sample.xlsx"));

                // finally write content
                 wb.write(fout);

                // close the file
                 fout.close();


            } catch (Exception e) {

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

            }

    }
```

}

13. How to Generate XSLT Report in Selenium Webdriver

Generate XSLT Report (Advanced HTML report) in Selenium

XSLT Report

XSLT stands for XML Style-sheet language for transformation, It provide very rich formatting report using TestNG framework

To generate XSLT report in Selenium be ready with below precondition.

Precondition-
1- Ant should be installed-

2- We should have some test case should be executed by TestNG (i.e. -output directory should be available in home directory)

Install ANT-

What is Ant

1-Apache Ant is a Java based build tool from Apache.

2-Apache Ant's build files are written in XML .

3-Open Source tool

This post will cover you how to Install Apache ANT to automate the build and deployment process in simple and easy steps.

Let us Generate XSLT Report in Selenium

Step 1- Navigate to below mention url

http://ant.apache.org/bindownload.cgi

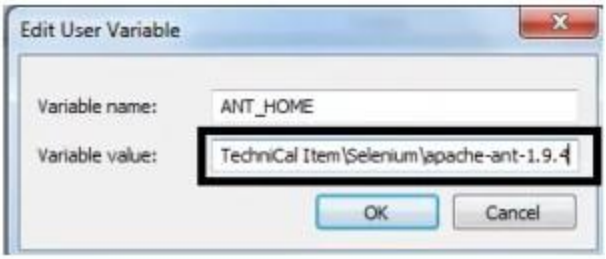
Step 2- Navigate to Current release of ant and download the zip file

Step 3- Extract zip file

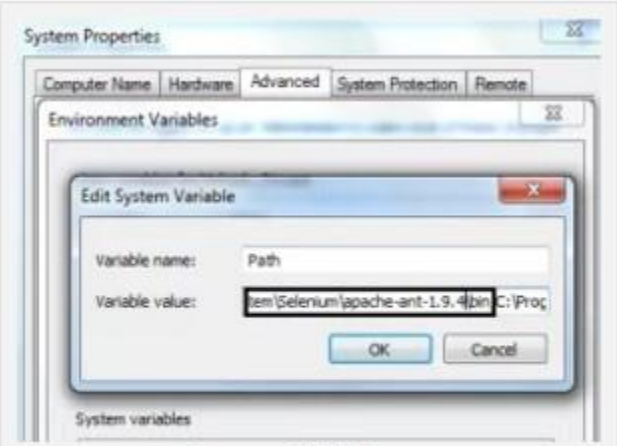
Step 4- Once we extract the zip file then we need to set environment variable

Right click on My computer and select the properties and click on Advanced system setting

Add the user variable – Here give the name ANT_HOME and in value section specify the path till home directory of ant



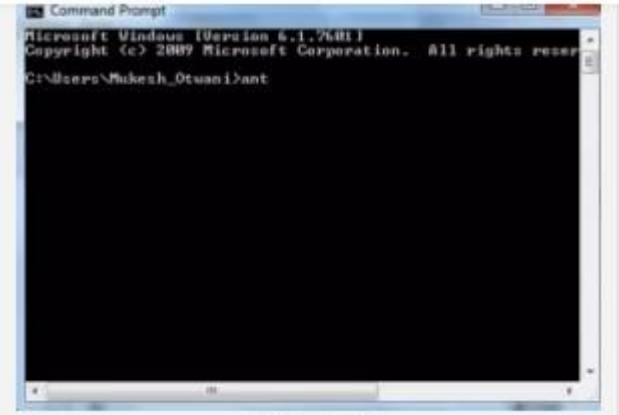
Add system variable- In this we need to edit existing system path click on edit , go till last and give the location till bin and save



Generate XSLT Report

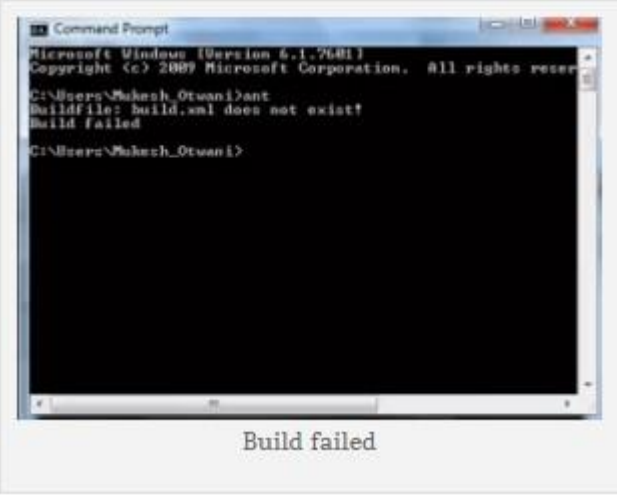
Note- Please do not edit other path- it may crash your system.

Step 5- Now verify that Ant is installed properly- Open CMD and type run and hit enter.



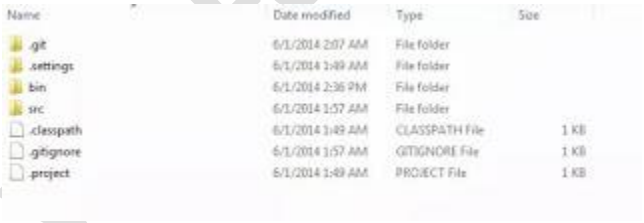
Ant install

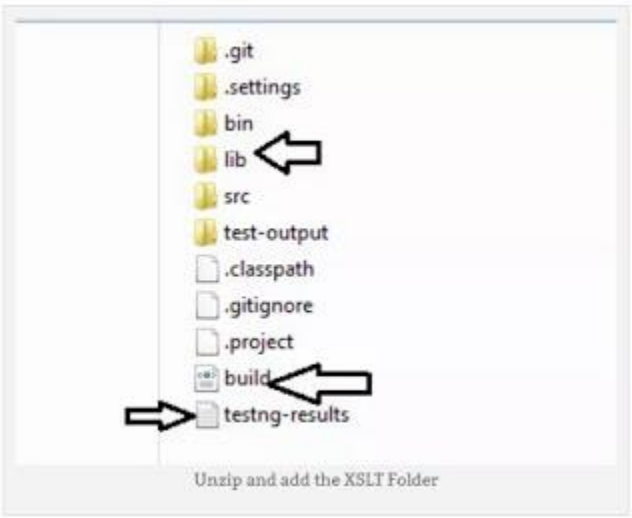
Note- if it is install properly then in output console we will get build.xml not found-build failed



Build failed

Step 1- Download XSLT from my Google driver account.
Step 2- Unzip XSLT Folder and copy all the files and paste into project home directory.
Refer below screen-shot

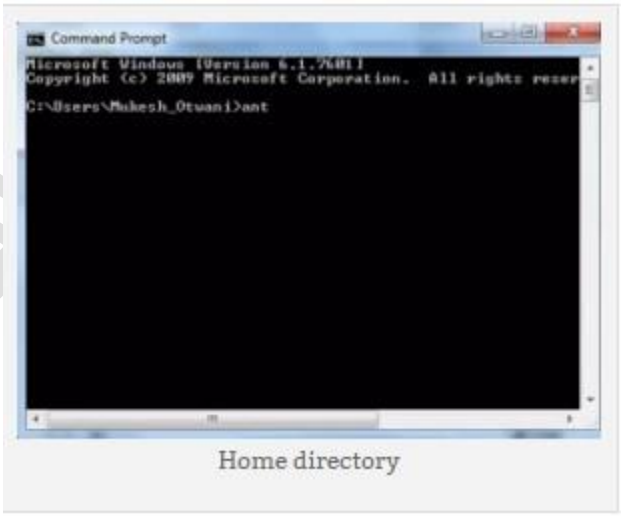




Unzip and add the XSLT Folder

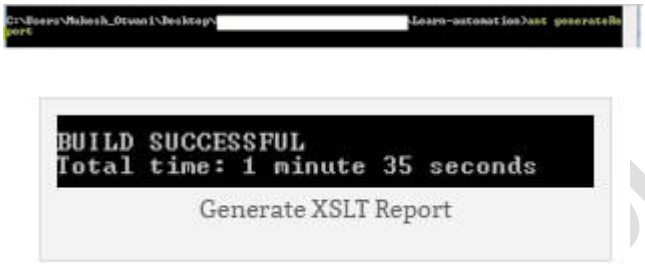
Step 3- Now execute build.xml file using Ant – for running build.xml file

Open command prompt and go till project home directory and type run and hit enter.

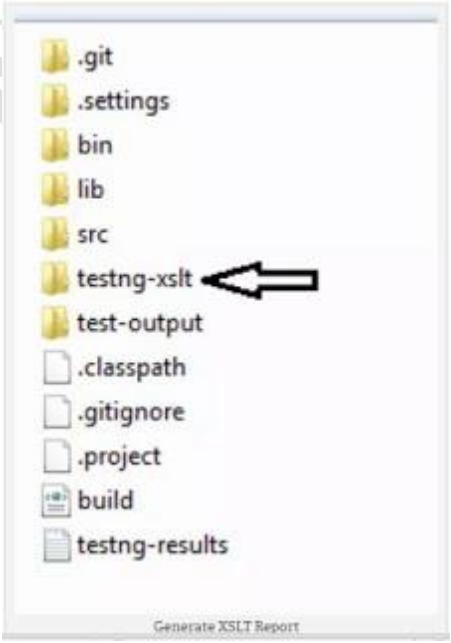


Home directory

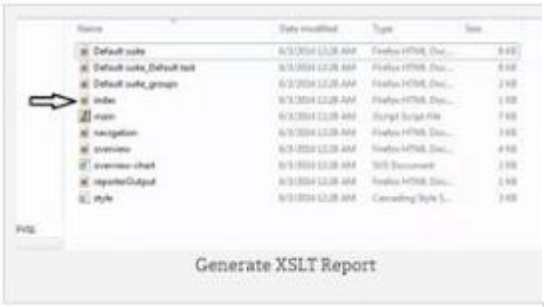
Step 4- Once build is successful then write ant generateReport and hit enter.



Step 5- After build is successful navigate to project directory and you will get testng-xslt folder



Inside testng-xslt you will get index.html (this is the main report) open in Firefox or in Chrome browser which support JavaScript



14. Object Repository in Selenium Webdriver

Whenever you talk about repository by the name itself you can think that it is kind of storage. Object repository is the collection of object and object here is locator.

Here **locator means** web element id, name, CSS, XPath,class name etc.

Object Repository in Selenium Webdriver

To understand an importance of Object repository, we will take real time scenario.

The scenario is you have 100 test cases and in all test cases, you have login scenario. If your application changes now and some locator changes like id changes from email to login email then you have to modify all the test cases and you have to change the id.

This process does not make any sense so to overcome with this we will move all locator in a separate file and we will link all test cases to this file. In case any changes happen in our locator we will simply change in that file, not our test cases.

This will increase the modularity of test cases and I will strongly suggest that you should use Object Repository in Automation.

Object Repository in Selenium Webdriver- Selenium Tutorial

Precondition

1- Project should be created and Selenium jars should be added

2- Now create a new file and give file name as Object_Repo (depends on you)with extension **.properties**

For this right click on project > create a new file > Specify name

Select folder and specify name & extension

3- Now file is created > Double click on file > File will open in Edit mode

object.properties:

Google_URL=http://www.google.com

Search_box=//input[@name='q']

Button=//button[@name='btnG']

Sample Code:

```
package Others;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
public class objectrepositry {

@Test
public void TestOR() throws IOException{

// Specify the file location I used . operation here because
//we have object repository inside project directory only
File src=new File("D:\\WebDriver\\object.properties");

// Create  FileInputStream object
FileInputStream fis=new FileInputStream(src);

// Create Properties class object to read properties file
Properties pro=new Properties();

// Load file so we can use into our script
pro.load(fis);
```

```
System.out.println("Property class loaded");


// Open ChromeBrowser

System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chromedriver_win32\\chrome
driver.exe");

WebDriver driver = new ChromeDriver();


// Maximize window

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


// Pass application

driver.get(pro.getProperty("Google_URL"));


// find the text box and enter the text

driver.findElement(By.xpath(pro.getProperty("Search_box"))).sendKeys("Properties");


driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

// click the search button

driver.findElement(By.xpath(pro.getProperty("Button"))).click();


driver.quit();


}


}
```

15. Handling Basic Authentication Using Webdriver

Here is post which explains you how to handle basic authentication.

Problem:

Some of the applications that are secured with Basic Authentication. If you want to access those applications first you need to pass credentials. Those applications will launch a system level pop-up which cant not be handled by selenium.

If you access below url it will ask for authentication.

http://the-internet.herokuapp.com/basic_auth

Solution:

By specifying userName and password in URL when accessing the page we can avoid system level dialog. This approach will work for HTTP and HTTPS pages.


```
package Others;

import java.util.concurrent.TimeUnit;


import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;


public class authentication {

    public WebDriver driver;
```

```
    @Test
    public void testBasicAuth() {
        //UserName --admin
        //Password --admin
        //URL --- http://the-internet.herokuapp.com/basic_auth
        //Updated URl -- http://admin:admin@the-internet.herokuapp.com/basic_auth
        driver.get("http://admin:admin@the-internet.herokuapp.com/basic_auth");
        //assert Text
        String actualText=driver.findElement(By.xpath("//div[@class='example']/p")).getText();
        Assert.assertEquals(actualText, "Congratulations! You must have the proper credentials.");
        System.out.println("Test passed");
    }
    @BeforeClass
    public void beforeClass() {
        System.setProperty("webdriver.chrome.driver","D:\\Softwares\\chromedriver_win32\\chromedriver.exe");
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    }

    @AfterClass
    public void afterClass() {
        driver.quit();
    }


}
```

16. Handling web tables in Selenium Webdriver

Handling webtables

This post explains about Handling Webtables.

Get Number of Columns

Get Number of Rows

Get Content of a specific cell based on Row and Column value.

Below is the code:

```
package Others;
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class webtables {
	public WebDriver driver;
 @Test
 public void testWebtable_Test() throws Exception {

		driver.get("http://the-internet.herokuapp.com");
```

```
        driver.findElement(By.linkText("Sortable Data Tables")).click();
        //get number of columns and print column names

        List<WebElement> columns=driver.findElements(By.xpath("//table[@id='table1']//tr//th"));
        System.out.println("NUmber of columns : "+columns.size());
        for (WebElement col : columns) {
            System.out.println("Columns are : "+col.getText());
        }

        //get number of rows
        List<WebElement> rows=driver.findElements(By.xpath("//table[@id='table1']/tbody/tr"));
        System.out.println("Number of rows : "+rows.size());
        for (WebElement row : rows) {
            System.out.println("Rows are : "+row.getText());
        }

        //get content
        System.out.println("Cell content is " +selectTableContent(2, 3));
         Thread.sleep(3000);
 }

 //method to get table content.
 public String selectTableContent(int row, int column)
 {
        String content=driver.findElement(By.xpath("//table[@id='table1']//tr["+row+"]/td["+column+"]")).getText();
```

```
        return content;
 }
 @BeforeClass
 public void beforeClass() {
        System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chromedriver_win32\\
chromedriver.exe");
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
 }

 @AfterClass
 public void afterClass() {
        //close browser
        driver.quit();
 }

}
```


17. Simulate pressing of Multiple Keys (using chord)

One of the nice feature in selenium webdriver is it allows us to simulate pressing of multiple keys using a method called "chord"

Here is the explanation of chord method:

Simulate pressing many keys at once in a "chord". Takes a sequence of Keys.XXXX or strings; appends each of the values to a string, and adds the chord termination key (Keys.NULL) and returns the resultant string.

In the below example i am going to show how to perform Copy and Paste using Selenium webdriver using Actions + Keys + chord.

Below is scenario:
1. Open https://accounts.google.com/signup
2. Type something in FirstName
3. Copy the content in FirstName field (CTRL+c)
4. Paste same thing in LastName field (CTRL+v

Note: Similar way you can use different keywords.


```
package Others;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.interactions.Actions;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;


public class chord_keyword {

	public WebDriver driver;

	@Test

	public void testKeys_Chord() throws Exception {

		driver.get("https://accounts.google.com/signup");
```

```
        //Element definitions for FirstName and LastName
        WebElement txtFName=driver.findElement(By.name("FirstName"));
        WebElement txtLName=driver.findElement(By.name("LastName"));
        //Type "webdriver" in First Name
        txtFName.sendKeys("webdriver");
        //Create an object for Actions Class
        Actions a = new Actions(driver);
        //select the value which is typed in FirstName field
        a.sendKeys(txtFName, Keys.chord(Keys.CONTROL,"a")).perform();
        //Performing copy action using CTRl+C
        a.sendKeys(Keys.chord(Keys.CONTROL,"c")).perform();
        //Performing paste action using CTRL+V in LastName field
        a.sendKeys(txtLName, Keys.chord(Keys.CONTROL,"v")).perform();
        Thread.sleep(2000);
    }

    @BeforeClass
    public void beforeClass()
    {
        System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chromedriver_wi
n32\\chromedriver.exe");
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    }

    @AfterClass
```

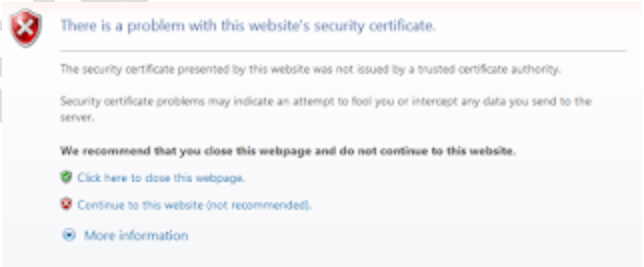
```
	public void afterClass() throws Exception
	{
		driver.quit();
	}

}
```

18. Handling HTTPS sites Or Handling Security Certificate Errors using Selenium WebDriver

Selenium webdriver is used to automate web applications.
Web applications generally starts with http://www.example.com.....But some websites like internal websites or banking or secured web sites will starts with https://www.example.com.

Below is one of the approach to handle **https** site while you are running from InternetExplorer.



Here we need to click the second option to go in to website.

Here we are taking help of Java script to click the second link

"Continue to this website (not recommended). "

Below is the code for handling https site..

```
package Others;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.ie.InternetExplorerDriver;
import org.testng.Assert;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
public class numberofproducts {
	WebDriver driver;
	@Test
	public void httpsTest() throws Exception {
		driver.get("https://www.example.com.");
		//Java script to click the link
		driver.navigate().to("javascript:document.getElementById('overridelink').click()");
		Thread.sleep(5000);
		//assert the title of the page
		Assert.assertEquals(driver.getTitle(), "Example Domain");
		System.out.println("asssert successfull");
		Thread.sleep(5000);
	}
	@BeforeTest
	public void beforeTest() {
		//launch Internet explorer
```

```
		System.setProperty("webdriver.ie.driver", "D:\\Softwares\\IEDriverServer_Win32_2.51.0\\IEDriverServer.exe");
		driver=new InternetExplorerDriver();
		driver.manage().window().maximize();

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

	@AfterTest
	public void afterTest() {
		driver.close();
		driver.quit();
	}

}
```

19. isDisplayed vs isEnabled

isDisplayed vs isEnabled

isDisplayed

boolean isDisplayed()

Is this element displayed or not? This method avoids the problem of having to parse an element's "style" attribute.

Returns:

Whether or not the element is displayed

Method used to verify presence of a web element within the webpage. The method is designed to result a Boolean value with each success and failure. The method returns a “true” value if the specified web element is present on the web page and a “false” value if the web element is not present on the web page.

isEnabled

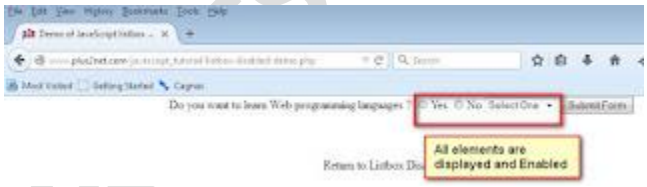
boolean isEnabled()

Is the element currently enabled or not? This will generally return true for everything but disabled input elements.

Returns:

True if the element is enabled, false otherwise.

Method used to verify if the web element is enabled or disabled within the webpage. Like isDisplayed() method, it is designed to result a Boolean value with each success and failure. The method returns a “true” value if the specified web element is enabled on the web page and a “false” value if the web element is not enabled (state of being disabled) on the web page



After Executing code :

Sample Code:

```
package Others;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class isEnableDisable {
	public WebDriver driver;
	@Test
	public void testIsDiplayed_Enabled() throws Exception {
		driver.get("http://www.plus2net.com/javascript_tutorial/listbox-disabled-demo.php");
		//verify if drodpdown is displayed or not
		//as it is returning True or False i will print the status
		System.out.println("Dropdown is displayed : " +driver.findElement(By.name("Category")).isDisplayed());
		//select radio button so Dropdown will be disabled
		driver.findElement(By.xpath("//form[@id='f1']/input[2]")).click();
		//Now dropdown is disable
		//as it will return true or false i will print status
		System.out.println("Dropdown is Enabled : " + driver.findElement(By.name("Category")).isEnabled());
		//still the dropdown is displayed but it is in disabled status
```

```
		System.out.println("Dropdown is displayed :
" +driver.findElement(By.name("Category")).isDisplayed());


		Thread.sleep(3000);

	}


	@BeforeClass

	public void beforeClass() {

		System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chromedriver_wi
n32\\chromedriver.exe");

		driver = new ChromeDriver();

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

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

	}


	@AfterClass

	public void afterClass() throws Exception {

		driver.quit();

	}


}
```

Output:

```
Dropdown is displayed : true
Dropdown is Enabled : false
Dropdown is displayed : true
```

20. Get Values from Dropdown

Get Values from Dropdown (Birthday field in Gmail registration)

Below is the sample script to get values or options from a dropdown.

Below is the image of Birth Month dropdown

The Birth Month field is not a Drop down. If you want to get all the values from dropdown first we need to click on the arrow mark and then we can get all the values of dropdown.

Here in the below example. First click on the drop down arrow then all the elements (Month options) will be visible and then you can get all the values of dropdown.

```
package Others;
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
public class dropdown {
	public WebDriver driver;
	@BeforeTest
	public void setUp() throws Exception {
		//Specify the browser
		System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chr
omedriver_win32\\chromedriver.exe");
		driver = new ChromeDriver();
		//declare globally wait
```

```
		driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
		//maximize the window
		driver.manage().window().maximize();
	}
	@AfterTest
	public void tearDown() throws Exception {
		//close the browser
		driver.quit();
	}
	@Test
	public void testGmail_Reg() throws Exception {
		driver.get("https://accounts.google.com/SignUp");
		//click on the arrow mark
		driver.findElement(By.xpath("//label[@id='month-
label']/span/div/div")).click();
		//get all the vlaues of dropdown
		List<WebElement> x=driver.findElements(By.xpath("//div[@class='g
oog-menu goog-menu-vertical']/div"));
		System.out.println("Size of the dropdown : "+x.size());
		//print dropdown options
		for (int i = 0; i < x.size(); i++) {
			System.out.println(x.get(i).getText());
		}
		Thread.sleep(5000);

	}


}
```


21. Select Month from Birthday field in Gmail registration page (New UI)

Below is the Sample script to select month from Gmail Registration page

Below is the image of Birth Month

The Birth Month field is not a Drop down. By using normal select option we cannot select a value from that drop down. To select value from these kind of fields. We need to use click command.

Here in the below example. First click on the drop down arrow then all the elements (Month options) will be visible and then click on the option which you want to select.

```
package Others;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
public class dropdown {
   public WebDriver driver;
   @Test
   public void testSelectBirthMonth() {
         driver.findElement(By.id("FirstName")).sendKeys("Selenium");
      driver.findElement(By.id("LastName")).sendKeys("Webdriver");
      driver.findElement(By.id("GmailAddress")).sendKeys("seleniumwebdriverxyz"
);
      driver.findElement(By.id("Passwd")).sendKeys("testingnow");
      driver.findElement(By.id("PasswdAgain")).sendKeys("testingnow");
      //Click on the Arrow mark
      driver.findElement(By.xpath("//label[@id='month-
label']/span/div/div")).click();
      //Select value from the list
      driver.findElement(By.xpath("//label[@id='month-
label']/span/div[2]/div[@id=':5']")).click();
      driver.findElement(By.id("BirthDay")).sendKeys("16");
      driver.findElement(By.id("BirthYear")).sendKeys("1999");
   }
   @BeforeClass
   public void beforeClass() throws Exception {
         System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chromedr
iver_win32\\chromedriver.exe");
```

```
            driver = new ChromeDriver();
        driver.get("https://accounts.google.com/SignUp");
    }
    @AfterClass
    public void afterClass() {
        driver.quit();
    }

}
```

22. Selecting a date from Datepicker using Selenium WebDriver

Selecting a date from Date picker using Selenium Web Driver

Calendars look pretty and of course they are fancy too. So now a days most of the websites are using advanced jQuery Datepickers instead of displaying individual dropdowns for month, day, year

If we look at the Date picker, it is just a like a table with set of rows and columns.To select a date ,we just have to navigate to the cell where our desired date is present.

Here is a sample code on how to pick a 13th date from the next month.

```
package Others;
import java.util.List;
import java.util.concurrent.TimeUnit;
```

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
public class dropdown {
   public WebDriver driver;
   @Test
   public void testSelectBirthMonth() {


         driver.switchTo().frame(0);
         driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);
         //Click on textbox so that datepicker will come
         driver.findElement(By.id("datepicker")).click();
         driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);
         //Click on next so that we will be in next month
         driver.findElement(By.xpath(".//*[@id='ui-datepicker-
div']/div/a[2]/span")).click();

         /*DatePicker is a table.So navigate to each cell
          * If a particular cell matches value 13 then select it
          */
         WebElement dateWidget = driver.findElement(By.id("ui-datepicker-div"));
         List<WebElement> rows=dateWidget.findElements(By.tagName("tr"));
         List<WebElement> columns=dateWidget.findElements(By.tagName("td"))
;

         for (WebElement cell: columns){
                  //Select 13th Date
                  if (cell.getText().equals("13")){
                  cell.findElement(By.linkText("13")).click();
                  break;
                  }
                 }
```

```
    }
    @BeforeClass
    public void beforeClass() throws Exception {
          System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chromedr
iver_win32\\chromedriver.exe");
                driver = new ChromeDriver();
       driver.get("http://jqueryui.com/datepicker/");
    }
    @AfterClass
    public void afterClass() {
      driver.quit();
    }

}
```

23. Selecting a date from Datepicker using Selenium WebDriver

Selecting a date from Date picker using Selenium Web Driver

Calendars look pretty and of course they are fancy too. So now a days most of the websites are using advanced jQuery Datepickers instead of displaying individual dropdowns for month, day, year

If we look at the Date picker, it is just a like a table with set of rows and columns.To select a date ,we just have to navigate to the cell where our desired date is present.

Here is a sample code on how to pick a 13th date from the next month.

```
package Others;
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
public class dropdown {
   public WebDriver driver;
   @Test
   public void testSelectBirthMonth() {


         driver.switchTo().frame(0);
         driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);
         //Click on textbox so that datepicker will come
         driver.findElement(By.id("datepicker")).click();
         driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);
         //Click on next so that we will be in next month
         driver.findElement(By.xpath(".//*[@id='ui-datepicker-
div']/div/a[2]/span")).click();

         /*DatePicker is a table.So navigate to each cell
          * If a particular cell matches value 13 then select it
          */
         WebElement dateWidget = driver.findElement(By.id("ui-datepicker-div"));
         List<WebElement> rows=dateWidget.findElements(By.tagName("tr"));
         List<WebElement> columns=dateWidget.findElements(By.tagName("td"))
;

         for (WebElement cell: columns){
               //Select 13th Date
               if (cell.getText().equals("13")){
               cell.findElement(By.linkText("13")).click();
```

```
                    break;
                    }
                   }
    }
    @BeforeClass
    public void beforeClass() throws Exception {
          System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chromedr
iver_win32\\chromedriver.exe");
                    driver = new ChromeDriver();
       driver.get("http://jqueryui.com/datepicker/");
    }
    @AfterClass
    public void afterClass() {
      driver.quit();
    }

}
```

24. Handling iFrames using Selenium Webdriver

In this tutorial we will learn ***handling iFrames using Selenium Webdriver***. iFrame is a HTML document embedded inside an HTML document. iFrame is defined by an *<iframe></iframe>* tag in HTML. With this tag you can identify an iFrame while inspecting the HTML tree.

Here is an sample HTML code of a HTML page which contains two iFrames.

You can find iFrame's test page here ***http://toolsqa.com/iframe-practice-page/***. We will use this to learn iFrame handling logic. Before starting we have to understand that to work with different iFrames on a page we have to switch between these iFrames. To Switch between iFrames we have to use the driver's ***switchTo().frame*** command. We can use the *switchTo().frame()* in three ways:

- ***switchTo.frame(int frameNumber)*****:** Pass the frame index and driver will switch to that frame.
- ***switchTo.frame(string frameNameOrId)*****:** Pass the frame element Name or ID and driver will switch to that frame.
- ***switchTo.frame(WebElement frameElement)*****:** Pass the frame web element and driver will switch to that frame.

Here is the image showing all the three overloads:

How to find total number of iFrames on a webpage

There are two ways to find total number of iFrames in a web page. First by executing a JavaScript and second is by finding total number of web elements with a tag name of iFrame. Here is the code using both these methods:


```
package Others;

import java.util.List;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class dropdown {

   public WebDriver driver;

   @Test

   public void testSelectBirthMonth() {


      // By finding java script executor

     JavascriptExecutor exe = (JavascriptExecutor) driver;

            Integer numberOfFrames = Integer.parseInt(exe.executeScript("return
window.length").toString());

            System.out.println("Number of iframes on the page are " + numberOfFrames);
```

```
        //By finding all the web elements using iframe tag
        List<WebElement> iframeElements = driver.findElements(By.tagName("iframe"));
        System.out.println("The total number of iframes are " + iframeElements.size());

    }
    @BeforeClass
    public void beforeClass() throws Exception {
        System.setProperty("webdriver.chrome.driver", "D:\\Softwares\\chromedriver_win32\\chromedriver.exe");
        driver = new ChromeDriver();
        driver.get("http://toolsqa.com/iframe-practice-page/");
    }
    @AfterClass
    public void afterClass() {
      driver.quit();
    }
}
```


Switch to Frames by Index

Index of an iFrame is the position at which it occurs in the HTML page. In the above example we have found total number of iFrames. In the sample page we have two IFrames, index of iFrame starts from 0. So there are two iFrames on the page with index 0 and 1. Index 0 will be the iFrame which exists earlier in the HTML page. Refer to the image below:

to switch to *0th iframe* we can simple write **driver.switchTo().frame(0)**. Here is the sample code:

```
public static void main(String[] args) throws InterruptedException {
		WebDriver driver = new FirefoxDriver();
		driver.get("http://toolsqa.com/iframe-practice-page/");

		//Switch by Index
		driver.switchTo().frame(0);
		driver.quit();
	}
```

Switch to Frames by Name

Now if you take a look at the HTMLcode of iFrame you will find that it has **Name** *attribute*. Name attribute has a value iframe1. We can switch to the iFrame using the *name* by using the command**driver.switchTo().frame("iframe1?)**.

Here is the sample code:

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

```
driver.get("http://toolsqa.com/iframe-practice-page/");

//Switch by frame name
driver.switchTo().frame("iframe1");
driver.quit();
```

Switch to Frame by ID

Similar to the name attribute in the iFrame tag we also have the *ID attribute*. We can use that also to switch to the frame. All we have to do is pass the id to the *switchTo* command like this**driver.SwitchTo().frame("IF1?)**.

Here is the sample code:

```
WebDriver driver = new FirefoxDriver();
driver.get("http://toolsqa.com/iframe-practice-page/");

//Switch by frame ID
driver.switchTo().frame("IF1");
driver.quit();
```

Switch to Frame by WebElement

Now we can switch to an iFrame by simply passing the iFrame WebElement to the**driver.switchTo().frame()** command. First find the iFrame element using any of the locator strategies and then passing it to *switchTo* command.

Here is the sample code:

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

```
driver.get("http://toolsqa.com/iframe-practice-page/");
//First find the element using any of locator strategy
WebElement iframeElement = driver.findElement(By.id("IF1"));

//now use the switch command
driver.switchTo().frame(iframeElement);
driver.quit();
```

Switching back to Main page from Frame

There is one very important command that will help us to get back to the main page. Main page is the page in which two iFrames are embedded. Once you are done with all the task in a particular iFrame you can switch back to the main page using the **switchTo().defaultContent()**.

Here is the sample code which switches the driver back to main page.

```
WebDriver driver = new FirefoxDriver();
driver.get("http://toolsqa.com/iframe-practice-page/");
//First find the element using any of locator strategy
WebElement iframeElement = driver.findElement(By.id("IF1"));

//now use the switch command
driver.switchTo().frame(0);

//Do all the required tasks in the frame 0
//Switch back to the main window
driver.switchTo().defaultContent();

driver.quit();
```