**Selenium (ThoughtWorks)**

**Naveen Automation Labs**

<https://github.com/naveenanimation20>

**Selenium WebdDiver (3.x) – 20MB**

1. **Download**
2. Site: [www.seleniumhq.org](http://www.seleniumhq.org) has good docs
3. Selenium WebDriver Java Doc : [www.seleniumhq.org](http://www.seleniumhq.org) >> Downloads secton >> latest Javadoc

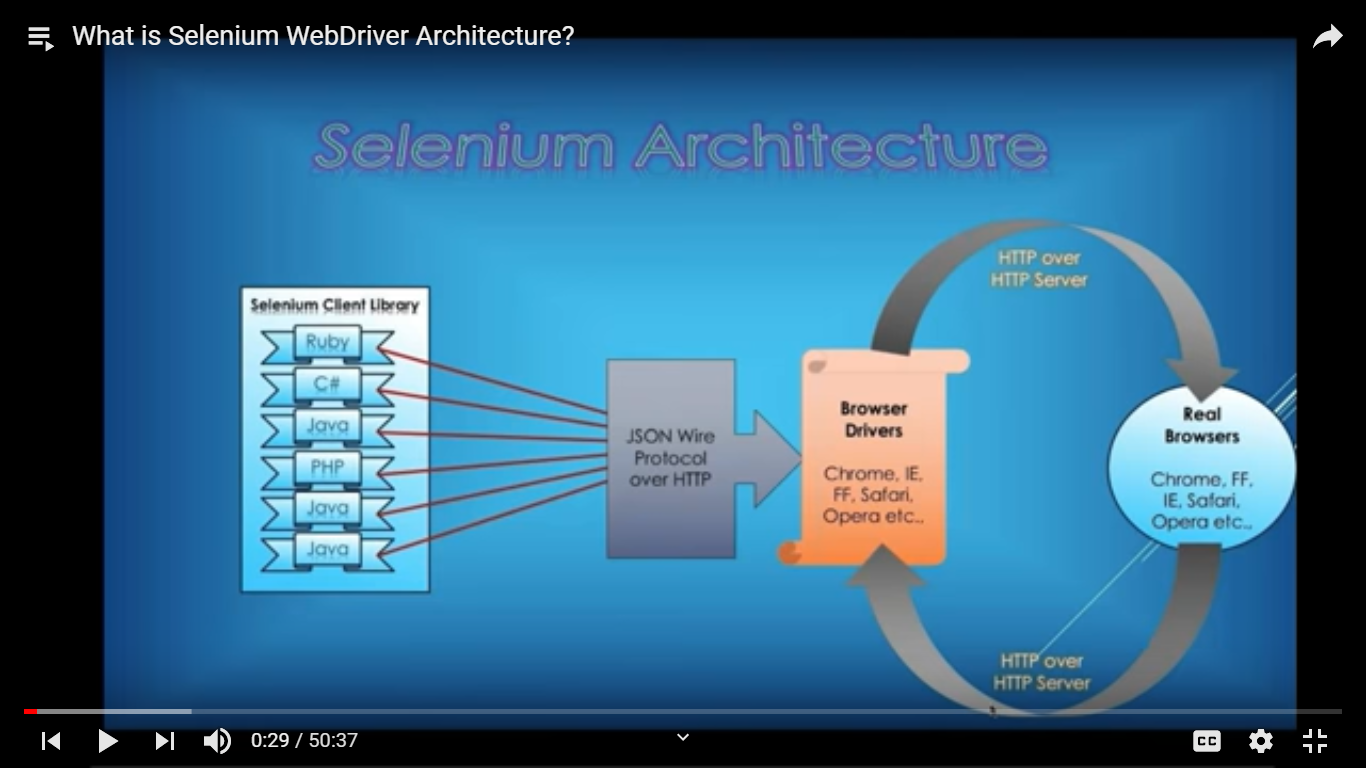
To check internal Interfaces and Classes in Selenium WebDriver API - to check implementation of different methods, packages, polymorphism

1. Download > Extract> There will be many JAR files
2. Open Eclipse > Import Selenium: Create Project > RC> Properties > Java build Path > Libraries > Add External JARs > Go to the folder in 1 (nodeps.jar) and lib files
3. Project > RC > should be able to see the ‘Referenced Libraries’
4. This support Selenium codes & Java codes

Note: Object is a class (default class) and superclass of all classes

1. To comment use: Ctrl + Shift + /
2. Create Package and then Class within it
3. Useful sites for testing: [www.freecrm.com](http://www.freecrm.com) (naveenk/test123); [www.popuptest.com](http://www.popuptest.com)

**Selenium Architecture**



Everything is happening via REST Webservice API >> over internet send request and get response

+for each Selenium command, a HTTP request is created and sent to the browser driver

+the browser driver uses a HTTP server for getting the HTTP requests the HTTP server determines the steps needed for implementing the Selenium command

+the implementation steps are executed on the browser

+the execution status is sent back to the HTTP server

+the HTTP server sends the status back to the automation script

<https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol> >> Command Summary

1st component >> jar files

From Eclipse (for Java) we are sending a command over HTTP – Browser recognized

System.setProperty (“webdriver.chrome.driver”, “exe.path /User/Jo/Downloads/chromedriver”);

WebDriver driver = new ChromeDriver(); // 3rd component, Sel create JSON & browser initialized REST APIs

driver.get(“https://www.google.com”) //4th component – this request created a JSON file (key & value format) – once successful Chrome along same path 4-3-2-1

Download Postman to practice JSON

1. **Launch**

Src > RC > New > Package > TestApp1> Finish

TestApp1 > RC > New > Class > Test1 > checkbox – public static void main (String[] args) & Inherited abstract methods> Finish

Code Appears as:

**Package**

**Public class**

**Public static void**

**Open Browser - Chrome**

Download exe – geckodriver for fireFox / chromedriver download tar file with win64

Copy path of executable and paste in exe.path (below) e.g. C:\\Download\\chromedriver.exe

**Package** Test App1

**Public class** Test1 {

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

**System.setProperty (“webdriver.chrome.driver”, “exe.path /User/Jo/Downloads/chromedriver”);**

**WebDriver driver = new ChromeDriver();** // import object by RC>> Chrome will be launched

**driver.get**(“https://www.google.com”) //method to enter URL

driver.getPageSource(); // get the entire HTML page source of that page

String title = driver.getTitle(); //to get title

System.out.println(title); //to show ‘Goog le’ on console

// To check if title is correct or no – verification point

if(title.equals(“Google”)){

System.out.println(“correct title”);

}

else{

System.out.println(“incorrect title”);

}

//to get current URL

System.out.println(driver.getCurrentUrl());

// to close the browser

driver.quit();

**}**

**}**

**Notes:**

WebDriver >> interface

driver >> Webdriver reference variable

New ChromeDriver() >> is object of this particular class

ChromeDriver() >> is a class

1. **Different Locators in WebDriver (ID,Xpath,CSSSelector etc)**

There are 8 locaters: Xpath, ID, Name, linkText, partialLinkText, CSSSelector, className, Tag

Order of preference: ID – Name – Xpath (Relative) – linkText - CSSSelector

Download Plugins – These help to recognize the elements i.e. text/dropdown, Image, Date, Button, Checkbox etc.

Firebug plugin >> right click anywhere on a page and choose "Inspect element" from the menu.

Chropath >> <https://browsee.io/blog/chropath-a-quick-way-to-get-and-verify-xpath-and-css-selectors/>

Don’t spy label, spy field to get Locators >> Highlighted Blue line in code >> HTML code

HTML Tag <input = input field; Dropdown = select; Image = img

HTML Properties: Name/size /value/max length/type

e.g. <input\_id= “firstname” name = “firstname” size = “32” value = “ ” type=“text”/>

Step1: Launch URL

Step2:

**Using Xpath locator: //\*[@id=’firstname’]**

Xpath means address of element. Below is example of Relative Xpath

driver.findElement(By.xpath(“//\*[@id=’firstname’]”)).sendKeys(“Tom”) >> to find element and enter value

Copy value from xpath field – don’t copy (.)

Create your own: By.xpath(“//input[@type=’submit’]”) >> get from HTML code – i/p is tag & type if present

**Customised xpath: Makes Script very fast**

Sometimes xpath is not correct, other HTML parameters are also not available

driver.findElement(By.xpath

Next right your own xpath >> Syntax: //\*[@id=’firstname’]

Spy html tag and if input “//input” then [] then pick any HTML property like say class (“//input[@class=]”) then ‘value’ i.e. (“//input”[@class=’firstname’])

So driver.findElement(By.xpath(“//input[@**class**=’firstname’]”)).sendKeys(“abc”);

driver.findElement(By.xpath(“//input[@**name**=’query’]”)).sendKeys(“abc”);

name/class are HTML properties. Alternately using **contains** functions in xpath library

driver.findElement(By.xpath(“//input[**contains**(@class**,**’firstname’)]”)).sendKeys(“abc”); >> here use , inctead of =

driver.findElement(By.xpath(“//input[**starts-with**(@id,’test\_’)]”)).sendKeys(“abc”); if id is dynamic i.e. id = test\_123, test\_456

driver.findElement(By.xpath(“//input[**ends-with**(@id,’\_test’)]”)).sendKeys(“abc”); if id is dynamic i.e. id = 123\_test, 456\_test

For links: All links are represented by HTML tag ‘a’ i.e. <a>

driver.findElement(By.xpath(“//a[**contains**(text(),’My Account’]”)).click(); text is a function, My Account is name of link before /a>

**You should verify your Customised Xpath**

WebPage > Inspect > Ctrl F

Some Good Examples:

++**link**: Features

//a[text()='Features']

//a[contains(text(),'Features')] >> this text can be part of any HTML properties

--recommended ++**button**:

//button[@type='button' and @class='btn'] >> when there are more than 2 buttons having type = button using multiple ands

//button[contains(text(),'Sign Up')]

//div[@class='dropdown']

//button[@type='button' and @class='btn btn-secondary dropdown-toggle' and @id='dropdownMenuButton'] //button[@id='dropdownMenuButton']

**For tables:**

You want to click checkbox against a name – spy on name – go to parent – then go to sibling – then the sibling has input as checkbox

++**preceding**-sibling: or forward-sibling

//a[text()='test2 test2']//parent::td[@class='datalistrow']//preceding-sibling::td[@class='datalistrow']//input [@name='contact\_id']

**Using ID locator:**

driver.findElement(By.id(“firstname”)).sendKeys(“Tom”) >> to find element and enter value

Copy value from input\_id = *value*

If id not available use xpath

**Using Name locator:**

driver.findElement(By.name(“firstname”)).sendKeys(“Tom”); >> to find element and enter value

Copy value from name = *value*

Repeat for all fields

If id not available use xpath/ID

**Using linkText locator: used only for links**

Can use Xpath too but ensure its not a hierarchy based one (e.g. of hierarchy html/body/div[1]/ div[5]/ div[2]/a) as hierarchy depends on position of link on screen. Also known as Absolute Xpath

Instead Look for a tag <a …… </a>

id & name will be missing mostly

driver.findElement(By.linkText(“Sign In”)).click(); >> to find element and enter value

Copy value before </a>

**Using partialLinkText locator: not useful**

For long link text e.g. how to pick a great user id

driver.findElement(By.partialLinkText(“how to pick”)).click(); >> to find element and enter value

Copy value before </a>

**Using CSSSelector locator:**

driver.findElement(By.cssSelector(“#firstname”)).sendKeys(“Tom”); >> to find element and enter value

Go to HTML code of the element > RC > Copy CSS Selection > paste

**Using className locator: not useful**

Class name can be duplicate for 2 elements

driver.findElement(By.className(“ancAAB”)).click(); >> to find element and enter value

Can be used only if class = is present in HTML code

**Using tag locator: TBU**

**For Dropdown**

Select element > RC > Spy > ID or Name

Left triangle > Expand > see options

Select select = new Select (); (>>RC > Import select from selenium support.ui)

Create object of select class. This class is already available in Sel so import from support.

Inside select class identify element where you want to perform action – spy and by id/name

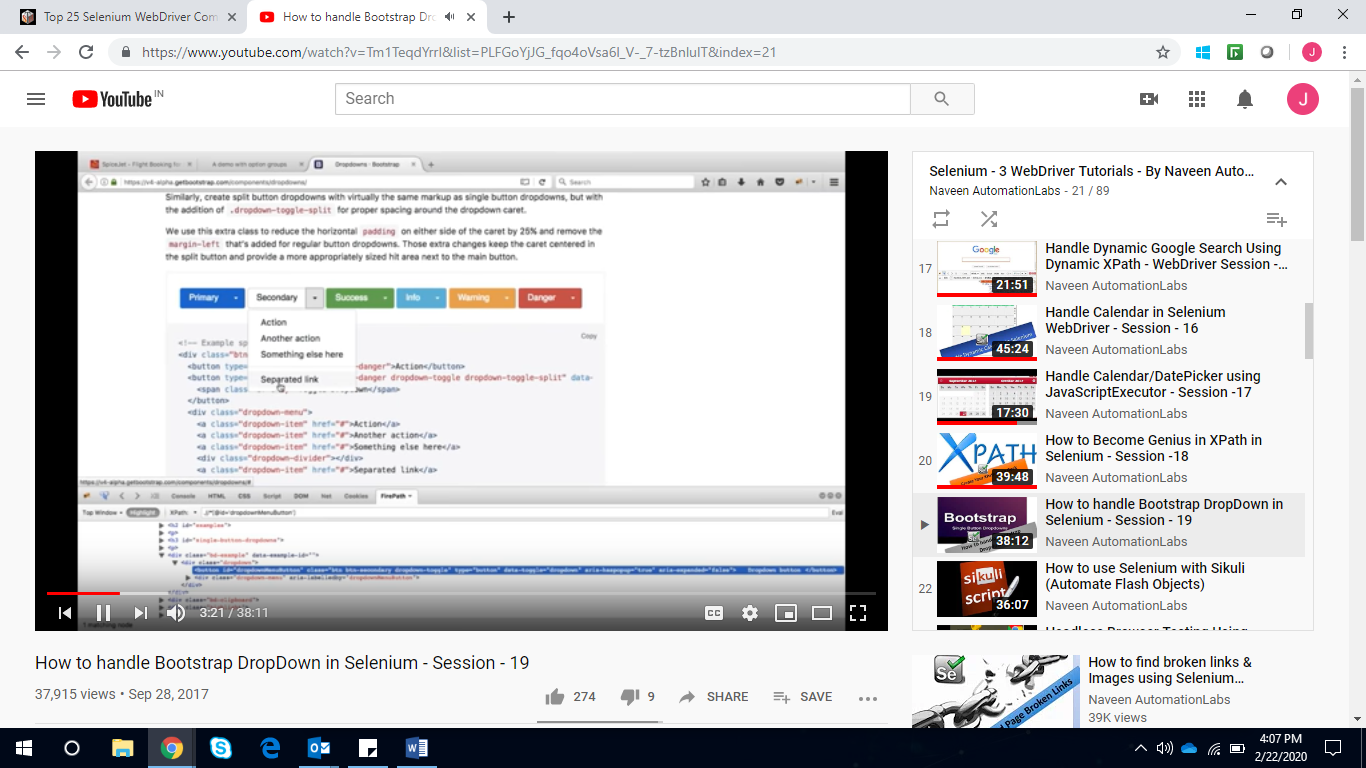
Select select = new Select (driver.findElement(By.id(“state”)));

select.selectByVisibleText(“California”);

Next you have method above (2nd line) and pass value you want in the drop down

How to handle **Bootstrap DropDown** in Selenium : Sometimes there is no select tag in dropdown so above wont work

See SS:



* 1. Click on the given dropdown by using findElement
  2. For all options you will see, ul. If you mouse over, the whole drpdn will get highlighted
  3. Expand ul > you will see many li(s)
  4. Get all the tags available under "li" tag
  5. Get the text of each "li//a" OR "div//a"
  6. Match the text and click on it
  7. Code same as Google Search scenario (below)

List <WebElement> l = driver.findElements(By.xpath(“//ul[contains(@class,“abc”)] //li//a//label”));

//li//a//label >> path of tags to drill to label

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

For (i=0; i<l.size(); i++){

System.out.println(l.get(i).getText());

If (l.get(i).getText().contains(“*your text*)){

l.get(i).click(); >> click on the option

break; >> terminate for loop - no need to search for other text

}

}

**For Button/Checkbox**- find element and click();

1. **Handling JavaScript Alert /Popup**

Somethings cannot be handled by Locators e.g. Pop ups (Error message/Confirmation) – you can’t even spy this

Create Class > select main method (chkbox – public static void main)

Website> Go to Submit/go button > spy > HTML > Name/ID

driver.findElement(By.name(“proceed”)).click(); >> to find element i.e go button & click on it. Pop should appear on click

Alert alert = driver.switchTo().alert(); >> After popup there are 2 screens, background app and message on fore, you have to switch to popup. Use ‘switch to’ method - driver.switchTo()).alert();. Alert class is available in Sel. Remember this is Java Script popup

System.out.println(alert.getText()) >> get text on popup [alert.getText()] & print using [system.out.println] on o/p window

alert.accept() >> click on ok button. If cancel, alert.dismiss()

String text = alert.getText();

if(text.equals(“Pls enter username”)){

System.out.println(“correct alert”)}

Else {

System.out.println(“incorrect alert”)} >> to add validation point

**To wait in between actions**

Thread.sleep(5000); >> 5000 millsec = 5 sec

**FileUpload PopUp**

Browse/Upload/Attach button i.e Type = “file” > Window pop up> Select File > Ok

Sel cannot understand the path whether downloads folder/documents folder/drives etc.

Note: HTML spy – Type should be file

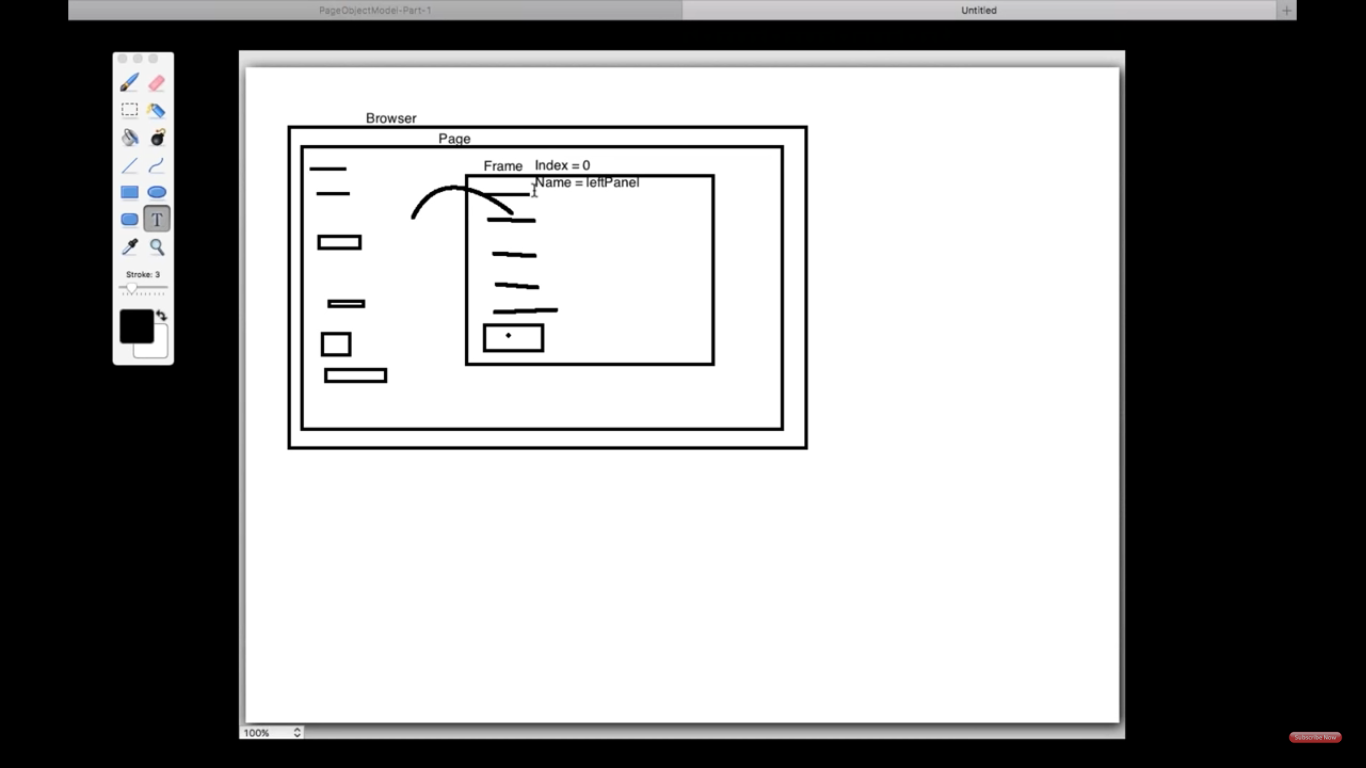
Launch Chrome > enter url >

Spy Browse button **(Do not click!!)**

driver.findElement(By.xpath(“//\*[@id=’file upload’]”)).~~click~~sendKeys(“C:\Joseanne\04. ING\ToDo.xls”); >> don’t use click cos if you click the pop up action is lost. Instead, use sendkeys and directly send the path of file to be uploaded

Go to file > RC > Get Path e.g. C:\Joseanne\04. ING\ToDo.xls

1. **Handling Frames**



[**www.freecrm.com**](http://www.freecrm.com) (good site for Sel prac)

Launch Chrome > enter url > Enter uname/pwd

Click on Contacts by creating your own xpath

driver.findElement(By.xpath(“//a[contains(text(), ‘Contacts’)]”)).click();

Run the code - Error msg in o/p window: Unable to locate element

Above Contacts > RC > Frame section appears > means there are frames

RC > View Page source > number of frames present will appear under ‘frameset’

So we have to use switch to frame for the same

driver.switchTo().frame(“mainpanel”); >> get **name** of frame from View Page source OR driver.switchTo().frame(0); using **index**

Thread.sleep(2000); >> wait for 2 sec

driver.findElement(By.xpath(“//a[contains(text(), ‘Contacts’)]”)).click();

1. **MouseMovement, Drag & Drop and ImplicitWait**

Hover over Menu > SubMenu displayed> Select a SubMenu

Launch Chrome > enter url > Enter uname/pwd > home page

Actions action = new Actions (driver); >> Hover over Menu; import from Sel

action.moveToElement (driver. findElement(By.linkText(“About Us”))).build().perform() >> action is object, move to Element is method which will take you to target, we are not clicking the target only **hovering** - build().perform() will perform the action and we will be able to see sub menu

Next to click on one submenu say Fleet, get xpath/Id etc of the element. But add a few secs otherwise the next action will be very early

Thread.sleep(3000); >> RC Throws Declaration

driver.findElement(By.linkText(“Fleet”)).click();

**Best Practices**

Maximize screen > We are asking driver to pls manage the window to maximize – Add it immediately after launching a browser

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

If we want to clear cache every time we open a browser:

driver.manage().deleteAllCookies();

**Drag and Drop Concept**

Launch Chrome > enter url > Enter uname/pwd > home page

Click & hold object to be dragged – move to target location – release

Actions action = new Actions(driver);

action.clickAndHold(driver.findElement(By.xpath(“//\*[@id=’draggable’]”))) >> method to click & hold an element. Find element-using xpath

In same code, add move action

.moveToElement(driver.findElement(By.xpath(“//\*[@id=’draggable’]”)))

Finally release

.release().build().perform()

**All together**

Actions action = new Actions(driver);

action.clickAndHold(driver.findElement(By.xpath(“//\*[@id=’draggable’]”))) .moveToElement(driver.findElement(By.xpath(“//\*[@id=’draggable’]”))) .release().build().perform();

**Implicit Wait**

Or Dynamic wait for heavy site and takes time to load

driver.manage().timeouts().pageLoadTimeout(40, TimeUnit.SECONDS);

After 40s if page doesn’t load, error in o/p

But if page loads in 2s, Sel will ignore balance 40s

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

Implicitly wait is applicable for **all** elements on the page, after the page is loaded. If loaded earlier, the remaining sec will be ignored

Static wait: Thread.sleep(5000); is not recommended as script is totally paused for the duration

**Standard Best Practice:**

Launch Browser – Open URL – Maximize - Delete Cookies – Dynamic wait

**Package** Test App1

**Public class** Test1 {

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

System.setProperty (“webdriver.chrome.driver”, “/User/Jo/Downloads/chromedriver”);

WebDriver driver = new ChromeDriver();

driver.get(“https://www.google.com”);

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

driver.manage().deleteAllCookies();

driver.manage().timeouts().pageLoadTimeout(40, TimeUnit.SECONDS);

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

**}**

**}**

1. **Driver.findElements Concept**

Total Count of all links on page and name/text of each link

driver.findElements(By.tagName(“a”)); >> you are asking driver pls give me all elements with tag a (links) using findElements method

Tag here is HTML tag i.e. input for text /a for links/button for buttons

This gives you a list

List <WebElement> linkList = driver.findElements(By.tagName(“a”));

List is a java utility

WebElement is a class in Sel

linkList is object name of type List that will have all the Webelements

So return type for **findElements** is List and hence you have to use **List <WebElement> anyobjectname**

Similar to Arraylist

To get size of arraylist

System.out.println(linkList.size()); >> gives total count of links available on linkList

To display all the link text use for loop

for (i=0; i< linkList.size(); i++){

linkList.get(i).getText(); >> needs to be stored in string so String linkText = linkList.get(i).getText();

System.out.println(linkText);

}

1. **Back & Forward (Browser)**

Launch Chrome > Go to google.com >

Next open facebook

driver.navigate().to(“www.facebook.com”);

Now,go back to google

driver.navigate().back();

Thread.sleep(2000);

Now,go back to facebook

driver.navigate().forward();

Thread.sleep(2000);

**Refresh Page**

driver.navigate().refresh();

**Screenshot**

Incase of errors for evidence – take screenshot and save file in a folder

Use  method

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

FileUtils.copyFile(src, new File(“/pathe.g./User/Jo/SSFolder/pic.jpg”)

1. **Handle Browser Pop Up Window  (**[**www.popuptest.com**](http://www.popuptest.com) **)**

Browser Window pop / not Java pop up

Spy/HTML code will not work

Browser > Link > Click > New Win PopUp > 2 windows> **Get Win ID for parent and child window > Switch** > Enter Details > close > **Return** to parent browser

driver.getWindowHandles(); >> to get Win ID of PopUp

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

Iterator<String> it = handler.iterator();

Set doesn’t have index (specific location) and hence we have to use iterator, forloop will not work

String parentWindow = it.next(); >> window Id is available in the form of string

System.out.println(“ParentWinID: “ + parentWindow);

String childWindow = it.next();

System.out.println(“ChildWinID: “ + childWindow);

Swtich

driver.switchTo().window(childWindow) >> control now on child window

driver.close(); >> to close pop up – driver.quit() is used to close browser i.e. both windows

driver.switchTo().window(parentWindow) >> to get control of parent browser

Verify if it is actually parent window id

System.out.println(“Parent Window Title: “ +driver.getTitle());

1. **Object repository/Properties file**

To create Auto framework > Properties file (separate component)

Package > RC > Other (not class) > General > File > Name = abc.**properties** > Finish

Data here is stored in Key & Value format. This will be for all generic things in the project

If were are using some details very frequently in our Sel code then user properties

name = Tom

age = 25

Browser = Chrome

URL = [www.google.com](http://www.google.com)

username = [test@gmail.com](mailto:test@gmail.com)

password = test123

can also define xpath

**Adv**: If there is a changes, just change in one place. All scripts will get run with new details

To comment, add # #name = Tom, don’t delete

How to use this info in Sel?

Next create class > use Property Class in Java, import from Java util

Properties prop = new Properties(); >> Create object ‘prop’ of properties class

FileInputStream ip = new FileInputStream(); >> Create object ‘ip’ of FIS class of Java

Inside this, point where is abc.properties file RC > path of prop file - FileinputStream ip = new FileinputStream(‘/path’) >> this way FIS class will connect to our .properties file i.e it will create i/p stream between Java code and abc.properties. Once this stream is established, we have to load the file

prop.load(ip);

Read the properties

System.out.println(prop.getProperty(“name”)) >>case sensitive

System.out.println(prop.getProperty(“age”))

Webdriver driver; >>declare as global variable

System.setProperty (“webdriver.chrome.driver”, “/User/Jo/Downloads/chromedriver”);

driver = new ChromeDriver();

driver.get(URL)

1. **Headless Browser Testing – HTMLUnitDriver**

Write script – script runs in front of you

Headless means Browser will not be displayed but o/p will show in console

// WebDriver driver = new ChromeDriver();

WebDriver driver = new HtmlUnitDriver();

HTMLUnitDriver jar file needs to be downloaded – then follow steps

Open Eclipse > Project > RC> Properties > Java build Path > Libraries > Add External JARs > Go to the folder > Ok

Also called Ghost Driver / Very fast

**For Chrome**

Pre-Req: Chrome verison should be > 60

Following chrome options are required for Chrome HeadLess Browser:

ChromeOptions options = new ChromeOptions();

options.addArguments("window-size=1400,800"); >> imp

options.addArguments("headless");

Before Webdriver driver =new ChromeDriver(**options**);

Launch Chrome > URL >

Useful when you want to run scripts and parallelly do some other work (script will run in the background)

1. **JavaScriptExecutor**

Tonic to Sel script to boost performance – in addition to click button etc. we can scroll up/down or highlight some button

1. **To make an element Blink**

Write method

public static void **flash**(WebElement element, WebDriver driver) { >> pass element & driver instance

javascriptExecutor js = ((JavascriptExecutor) driver); >> javascriptExecutor class is available in Sel to handle JS code/calls so RC & import. Next cast your driver into jsE

String bgcolor = element.getCssValue(“backgroundColor”); >> method CSSValue to get bckgrnd color of element and store in string bgcolor

For (int i = 0; i<10; i++ ){ >> blink the element 10 times

changeColor(“rgb(0,200,0)”, element,driver); >> changeColor method (below) called and color of element will change to chosen color

changeColor(bgcolor, element, driver); >> default color restored again then go to prev and element color will change to chosen col. This will repeat 10 times.

}

}

public static void **changeColor**(String color, WebElement element, WebDriver driver) { >> pass element & driver instance

javascriptExecutor js = ((JavascriptExecutor) driver); >> javascriptExecutor class is available in Sel to handle JS code/calls so RC & import. Next cast your driver into jsE

js.executeScript (“arguments[0].style.backgroundColor = ‘”+color+”’, element); >> with js use method executeScript. It will call changeColor method and element will be highlighted 10 times in chosen color

try{

Thread.sleep(3000);

} catch (InterruptedException e){

}

}

**To call above method in your script, say to highlight a button**

Class > Launch Chrome (Best Prac) > URL > write basic script>Get xpath of button

WebElement loginBtn = driver.findElement(By.xpath(“//input[contains(@type, ‘submit’)]));

flash (loginBtn, driver) >> Call flash method

1. **To make a border on an element & take screenshot : helpful while raising a defect to point exact issue**

Write method

public static void **drawBorder**(WebElement element, WebDriver driver) { >> pass element & driver instance

javascriptExecutor js = ((JavascriptExecutor) driver); >> javascriptExecutor class is available in Sel to handle JS code/calls so RC & import. Next cast your driver into jsE

js.executeScript (“arguments[0].style.border = ‘3px solid red’“, element); >> for a red border

try{

Thread.sleep(3000);

} catch (InterruptedException e){

}

**To call above method in your script, to draw a border on button**

Class > Launch Chrome (Best Prac) > URL > write basic script>Get xpath of button

WebElement loginBtn = driver.findElement(By.xpath(“//input[contains(@type, ‘submit’)]));

drawBorder (loginBtn, driver) >> Call method

for screenshot:

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

FileUtils.copyFile(src, new File(“/pathe.g./User/Jo/SSFolder/element.jpg”)

1. **To create an alert to draw attn**

Write method

public static void **generateAlert**(WebDriver driver, String message) { >> pass driver instance & msg

javascriptExecutor js = ((JavascriptExecutor) driver); >> javascriptExecutor class is available in Sel to handle JS code/calls so RC & import. Next cast your driver into jsE

js.executeScript (“alert(‘”+message+”’)”); >> alert is the JS method

**To call above method in your script, to draw a border on button**

Class > Launch Chrome (Best Prac) > URL > write basic script>Get xpath of button

WebElement loginBtn = driver.findElement(By.xpath(“//input[contains(@type, ‘submit’)]));

generateAlert(driver, “There is an issue”); >> Call method

use alert.dismiss() to remove alert

1. **To click on a btn using JS (used if you are having trouble with xpath)**

Write method

public static void **clickElementbyJS**(WebElement element, WebDriver driver) { >> pass element & driver instance

javascriptExecutor js = ((JavascriptExecutor) driver); >> javascriptExecutor class is available in Sel to handle JS code/calls so RC & import. Next cast your driver into jsE

js.executeScript (“arguments[0].click();”, element); >> for a red border

**To call above method in your script, to draw a border on button**

Class > Launch Chrome (Best Prac) > URL > write basic script

WebElement loginBtn = driver.findElement(By.xpath(“//input[contains(@type, ‘submit’)]));

**clickElementbyJS** (loginBtn, driver); >> Call method

1. **To refresh browser using JS**

Write method

public static void **refreshBrowserbyJS**(WebDriver driver) { >> pass driver instance

javascriptExecutor js = ((JavascriptExecutor) driver); >> javascriptExecutor class is available in Sel to handle JS code/calls so RC & import. Next cast your driver into jsE

js.executeScript (“history.go(0)”); >> for refresh

**To call above method in your script, to draw a border on button**

**refreshBrowserbyJS** (driver); >> Call method

1. **To get page title using JS**

Write method

public static **String** **getTitlebyJS**(WebDriver driver) { >> pass driver instance + remove Void and add String as this method is returning string.

javascriptExecutor js = ((JavascriptExecutor) driver); >> javascriptExecutor class is available in Sel to handle JS code/calls so RC & import. Next cast your driver into jsE

String title = js.executeScript (“return document.title;”).toString(); >> doc.title will capture title and pass to String (here object is title

return title;

**To call above method in your script, to draw a border on button**

getTitlebyJS(driver); >> Call method

system.out.println(getTitlebyJS(driver));

1. **To get page inner text using JS**

Write method

public static **String** **getPageInnerTextbyJS**(WebDriver driver) { >> pass driver instance + remove Void and add String as this method is returning string.

javascriptExecutor js = ((JavascriptExecutor) driver); >> javascriptExecutor class is available in Sel to handle JS code/calls so RC & import. Next cast your driver into jsE

String pageText = js.executeScript (“return document.documentElement.innerText;”).toString(); >> doc.title will capture title and pass to String (here object is title

return pageText;

**To call above method in your script, to draw a border on button**

getPageInnerTextbyJS(driver); >> Call method

system.out.println(getPageInnerTextbyJS (driver));

1. **To scroll page using JS**

Write method

public static **void scrollDownbyJS**(WebDriver driver) { >> pass driver instance + remove Void and add String as this method is returning string.

javascriptExecutor js = ((JavascriptExecutor) driver); >> javascriptExecutor class is available in Sel to handle JS code/calls so RC & import. Next cast your driver into jsE

js.executeScript (“window.scrollTo(0,document.body.scrollHeight)”); >> starting from 0 to vertical height

**To call above method in your script, to draw a border on button**

scrollDownbyJS (driver); >> Call method

1. **To scroll into view using JS >>scroll till a Btn/link is visile**

Write method

public static **void scrollIntoViewbyJS**(WebElement element, WebDriver driver) {

javascriptExecutor js = ((JavascriptExecutor) driver); >> javascriptExecutor class is available in Sel to handle JS code/calls so RC & import. Next cast your driver into jsE

js.executeScript (“arguments[0].scrollIntoView(true);”, element); >> keep scrolling till element is visible

**To call above method in your script, say to find Forgot Password link**

WebElement forgotPassword = driver.findElement(By.xpath(“//a(contains(text(), “Forgot Password))”));

**scrollIntoViewbyJS** (forgotPassword, driver); >> Call method and incl the element you are looking for

1. **Difference b/w isDisplayed, isEnabled and isSelected Methods in Sel**

Useful for validations

Create class > Open URL >

1. **isDisplayed()** is the method used to verify the presence of a web element within the web page. 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 webpage.

It is capable to check for the presence of all kinds of web elements available.

driver.findElement(By.id(“Submit Button”)).isDisplayed(); >> will return a T/F value, pls store it in an obect

boolean b1 = driver.findElement(By.id(“Submit Button”)).isDisplayed();

system.out.println(b1);

2. **isEnabled()** is the method used to verify if the web element is enabled or disabled within the web page It is primarily used with buttons.

E.g. Submit button not enabled till I agree chkbox is not checked

boolean b2 = driver.findElement(By.id(“Submit Button”)).isEnabled();

system.out.println(b2);

3. **isSelected() i**s the method used to verify if the web element is selected or not. It is predominantly used with radio buttons, dropdowns and checkboxes.

boolean b3 = driver.findElement(By.name(“agreeTerms”)).isSelected();

system.out.println(b3);

1. **Handle Dynamic WebTable**

Example: If one a page, you keep adding data and it reflects in a table below

Now if you want to select a specific data, and modify/delete

Today this row may be on row 10, tmrw 100 – you can’t use xpath as table is dynamic and is always changing

There are two ways of handling WebTable:

Method – 1: each row is read and hence performance is slow

• Iterate row and column and get the cell value.

• Using for loop

• Get total rows and iterate table

• Put if(string matches) then select the respective checkbox

• Lengthy method

Method – 2: **preferred!!**

• Using custom XPath

• Using parent and preceding-sibling tags

• No need to write for loop

• No full iteration of table

• Single line statement

• More dynamic

• Efficient and fast

Class > URL > uname/pwsd > Go to desired page > Use custom xpath

Spy value you want to select on table

HTML code will get highlighted

Xpath:

//a[contains(text(),’test2’)] >> test2 is any option you want to select

/parent::td >> go 1 level up i.e. parent of this element

//preceeding-sibling::td >>moving across column and getting near the checkbox

//input(@name=’contact\_id’)

Combine:

Driver.findElement(By.xpath(“*from above”*)).click(); >> click checkbox

driver.findElement(By.xpath(“//a[contains(text(),’test2’)]/parent::td//preceeding-sibling::td//input(@name=’contact\_id’)”)).click();

1. **Handle Dynamic Google Search Using Dynamic XPath (Example of Bootstrap ddn)**

Example: Google > Search > testing > suggestions > you want the 4th suggestions

Steps:

1. Google Search – pick a value from suggestions
2. Search using Xpath
3. Get the total count of suggestions
4. Use if condition to match the value
5. Click on matched value

Class > URL google > spy Search field by id > sendkeys(“Search value”)

Search options will appear, the whole thing is a box, try to select the entire box, all the suggestions and Google Search / I’m Feeling lucky btns will appear in the box

In the Firepath, it will appear in ul tag

Expand it to see l1,l2……last 2 are for buttons (Google Search / I’m Feeling lucky)

All li elements are child of ul tag

So here use, findElements concept to get # of elements

driver.findElements(By.xpath(“//ul[@role=listBox]”)) >> to get to parent

//li >> to get all the li elements i.e child under parent ul

/descendant:: >> to come to the child details (descendant of divs – there are multiple divs)

Div(@class=’sbqs\_c’); >> to get to the li s where search option is available i.e. testing, pick the class value from there

As a result of this, **all** the suggestions will be picked – 4 or 8 or 10 suggestions

This is the dynamic customized xpath

driver.findElements(By.xpath(“//ul[@role=”listBox”]//li/descendant::div(@class=’sbqs\_c’)”));

Output will be a list and so store it in list object

List <WebElement> l = driver.findElements(By.xpath(“//ul[@role=”listBox”]//li/descendant::div(@class=’sbqs\_c’)”));

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

Print all suggestions

For (i=0; i<l.size(); i++){

System.out.println(l.get(i).getText());

If (l.get(i).getText().contains(“*your text*)){

l.get(i).click(); >> click on the option

break; >> terminate for loop - no need to search for other text

}

}

1. **Handle Calendar**

Steps to handle calendar in Selenium Webdriver:

1. Click on calendar
2. Get all td of tables using findElements method
3. using for loop get text of all elements
4. using if else condition we will check specific date
5. If date is matched then click and break the loop. 6- Handle NoSuchElementException in case of (31st day)

Very complex script using Sel

Instead use JS

Launch URL > Go to date field

public static **void selectDatebyJS**(WebElement element, String dateVal, WebDriver driver) { >> element where we have to perform action i.e date field, string is the date we want to select in the date field (element) .

javascriptExecutor js = ((JavascriptExecutor) driver);

js.executeScript (“argument[0].setAttribute(“value”, “+dateVal+”)”, element); >> starting from 0 to vertical height

**To call above method in your script, to draw a border on button**

First define element

WebElement date = driver.findElement(By.id(“date”)); >> spy date field

String dateVal = “30-12-2019”; >> what is the value to be entered in date field. Format should be identical to what is predefined in the field

selectDatebyJS (date, dateVal, driver); >> Call method

1. **Selenium with Sikuli (Automate Flash Objects)**

Sikuli is an image-based open source tool to automate the GUI and can be used on any platform like Windows/Linux/Mac/Mobile.

Sikuli uses a technique called image recognition to identify and control GUI components.

YouTube URL > Video > Pause/Mute/Speed control/Star/Stop. These are Flash Objects.

Download Sikuli jar file (SikuliX API). You can even create Maven project & add Maven dependencies: pom.xml > Copy dependency from net and paste under <Dependencies> > Save >> Under Maven Dependencies find that Sikuli jar file is added

Limitations: It won’t open browsers, URL hence we need Sel

So integ Sel scripts with Sikuli

Launch YT URL (Sel code)

Take screenshot as Sikuli uses images, like Sel uses xpath/Id i.e. locators. Take SS using Print Screen and store it as .png. Rename files appropriately. E.g. Youtube\_Settings.png

Copy and paste under Eclipse project – using copy & paste

Sikuli will read the image & perform click option

//create object of screen class of Sikuli

Screen s =new Screen(); >> import from Sikuli

// create pattern class object

Pattern pauseImg = new Pattern(“filename sayPause.png”);

s.wait (pauseImg, 2000); >> for image to be visible for 2s

s.click(); >> after waiting click on image

s.click(); >> click once again

Repear for say Play

Pattern playImg = new Pattern(“filename sayPlay.png”);

s.wait (playImg, 2000); >> for image to be visible for 2s

s.click(); >> after waiting click on image

s.click(); >> click once again

1. **HTML DOM**

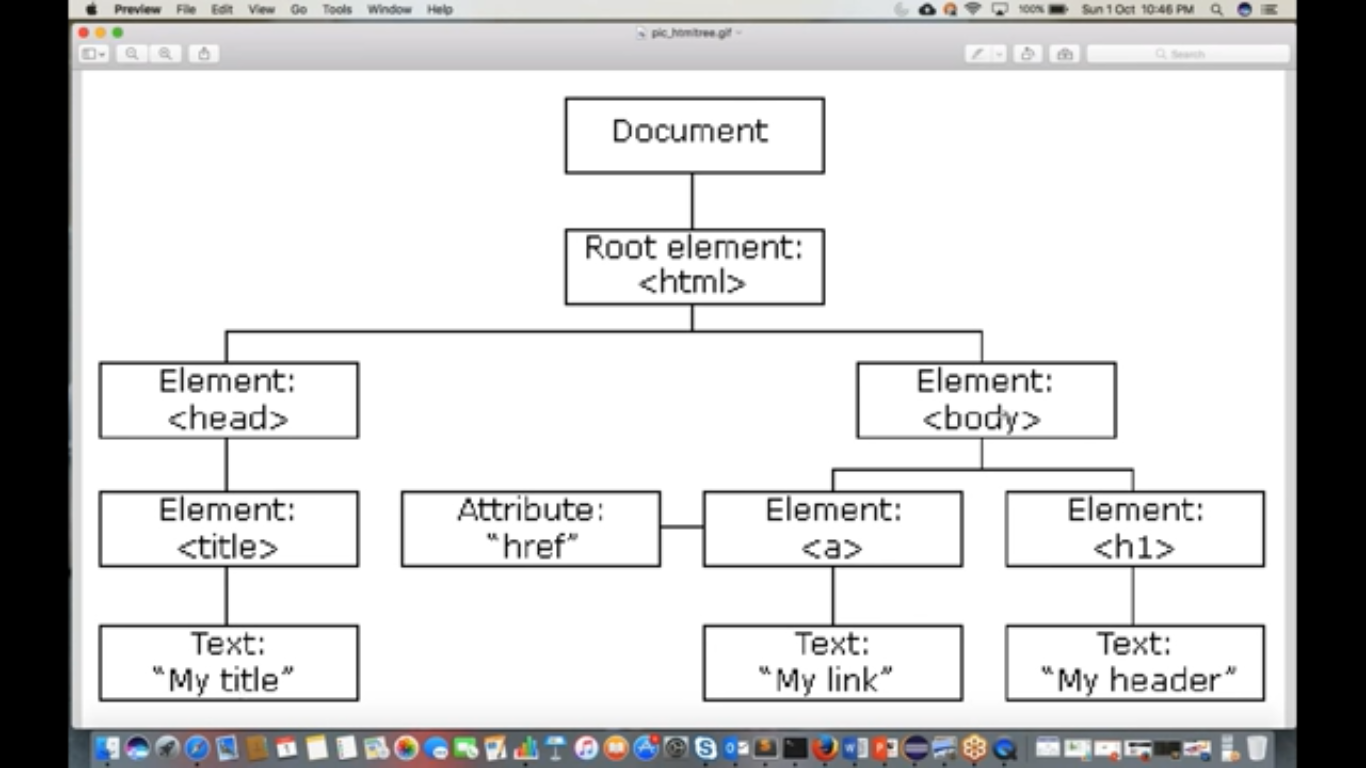
What is HTML DOM - Document Object Model?

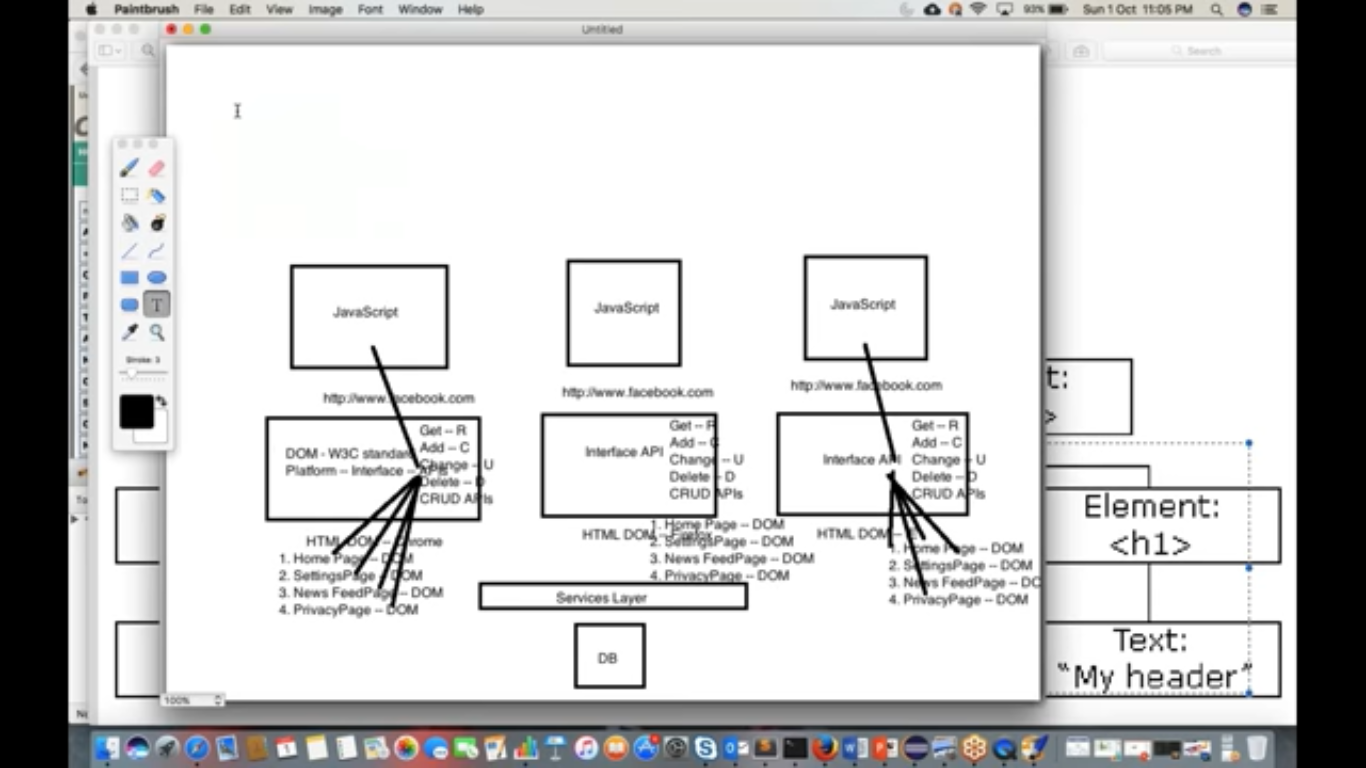
DOM is an API Interface provided by browser. JavaScript calls HTML DOM APIs

When a web page is loaded, browser creates a DOM of the page.

There are 3 types: HTML/XML/Core DOM

W3C standard, It is a platform/interface





Different Exceptions in Selenium:

a. StaleElement Exception

b. ElementNotFound Exception

c. NoElementReference Exception

With DOM model, JavaScript gets all the power it needs to create dynamic HTML:

• JavaScript can change all the HTML elements in the page

• JavaScript can change all the HTML attributes in the page

• JavaScript can change all the CSS styles in the page

• JavaScript can remove existing HTML elements and attributes

• JavaScript can add new HTML elements and attributes

• JavaScript can react to all existing HTML events in the page

• JavaScript can create new HTML events in the page

1. **How to find broken links & Images**

\*\*For checking the broken links, you will need to do the following steps.

1. Collect all the links in the web page based on "a" and "img" tags.

2. Send HTTP request for the link and read HTTP response code.

3. Find out whether the link is valid or broken based on HTTP response code.

4. Repeat this for all the links captured.

Ex. If there are 100 pages of a website & each page has 50-70 links & images

If some images or links ar broken – we get 404 or 500 error

Imagine clicking all links manually to see if any are broken….time consuming!

Instead, create utility (basic Sel script)

* Images are denoted by HTML **img** & Links by **a** ; there will be some ref **href**
* First you collect all links & images on the page i.e. get list of all elements
* Next, you check for validity

Class > Launch Chrome > Enter URL >

list <WebElement> linksList = driver.findElements(By.tagname(“a”)); >> as all links will have a tag

linksList.addAll(driver.findElements(By.tagname(“img”)); >> Images will also get stored in the same List linksList

system.out.println (“Size of all Links & Images” +linksList.size());

//Look for href

* If not available, ignore (out of scope)
* If available, test for no break (in scope)

Create one more list

List <WebElement> activeLinks = new ArrayList<WebElement>(); >>import from utill;this list will have only links & images with href

linksList has all imgs & links so we will iterate

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

if (linksList.get(i).getAttribute(“href”) !=null){ >>to exclude all img/link without href

activeLinks.add (linksList.get(i)); >> activeLinks list is ready

}

}

system.out.println (“Size of active Links & Images” +activeLinks.size()); >> get size of activeLinks list

// check href URL with Java’s **httpconnection api**

for (int j=0; j< activeLinks.size(); j++){

HttpURLConnection connection = (HttpURLConnection)new URL(activeLinks .get(j).getAttribute(“href”)).openConnection();

Connection.connect(); >> connect method; to connect to URL

String response = Connection.getResponseMessage(); >> will return ok if link is working else error

Connection.disconnect(); >> method to disconnect URL

System.out.println (activeLinks.get(j).getAttribute(“href”) + “--🡪”+response);

}

Breaking above code

Use Java httpconnection api to check if href is correct or not, nothing will be clicked

URL class > active links have all the links/img with href: new URL(activeLinks .get(j).getAttribute(“href”)).

Cast this into HttpURLConnection: (HttpURLConnection)new URL(activeLinks .get(j).getAttribute(“href”))

Get properties of href & open the connection, internally it will chk if element is correct or not: openConnection();

Store this in object: HttpURLConnection connection =

Add throws exception

Sample response:

200 – Ok; 404 – Not Found; 500 – Internal error; 400 – Bad Request