1. Javascript executor
2. Handling calenders
3. Adding a particular element in cart(Practice 69-73)
4. Wait

Introduction

1. Selenium is an open source Tool
2. Used for web based Application
3. Supports multiple browsers:: Chrome, Firefox,IE,Safari
4. Works in multiple platforms Windows , Mac, Linux
5. Coded in Multiple Languages :: Java,C#,Python,jS,php,Ruby
6. Difference b/w selenium and web Driver

IDE is record n playback tool

WebDriver is a robust Tool. Successor of Selenium RC.

Selenium Architecture

1. After you trigger the test, complete selenium code(Client) which we have written will be converted to JSON format
2. Generated JSON is sent to Browser Driver (Server) through http Protocol

Note:: Each browser contains a separate browser driver

1. Browser driver communicate with the respective browsers and executes the commands by interpreting the JSON which is received on the browser.
2. Browser Driver receives responses back from the browser and it sends JSON response back to Client.

System.*setProperty*("webdriver.chrome.driver","C:\\Study\\chromedriver.exe");

WebDriver driver = **new** ChromeDriver(); 🡺 ChromeDriver is a class

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

System.***out***.println(driver.getTitle());

Webdriver is an interface and it has several in built method in it. ChromeDriver is a class. We are creating an object of ChromeDriver. Driver is the object and as we will be implementing the methods of webdriver interface, the return type of driver is webdriver.

Chrome, Firefox, IE have restricted their API to be invoked by external APIs(Selenium API).Every browser team gave .exe file to selenium and asked selenium to invoke the browser via .exe file.

webdriver.chrome.driver is a property set by selenium for chrome driver.We need to set this property with chrome.driver location.

**Locators in Selenium**

* **ClassName** – A ClassName operator uses a class attribute to identify an object.
* **cssSelector**– CSS is used to create style rules for webpages and can be used to identify any web element.
* **Id** – Similar to class, we can also identify elements by using the ‘id’ attribute.
* **linkText** – Text used in hyperlinks can also locate element
* **name** – Name attribute can also identify an element
* **partialLinkText** – Part of the text in the link can also identify an element
* **tagName** – We can also use a tag to locate elements
* **xpath** – Xpath is the language used to query the XML document. The same can uniquely identify the web element on any page.

**Relative X Path**

<input type="text" class=" inputtext \_55r1 \_6luy

" name="email" id="email" data-testid="royal\_email" placeholder="Email address or phone number" autofocus="1" aria-label="Email address or phone number">

Here

Input 🡺 Tagname

Attribute 🡺 Class

Value 🡺 inputtext \_55r1 \_6luy

Eg:: <input name=” username123”>

//tagname[contains(@attribute,’value’)] 🡺 XPATH

//input[contains(@name,’ username123’]

Tagname[attribute\*=’value’] 🡺 CSS

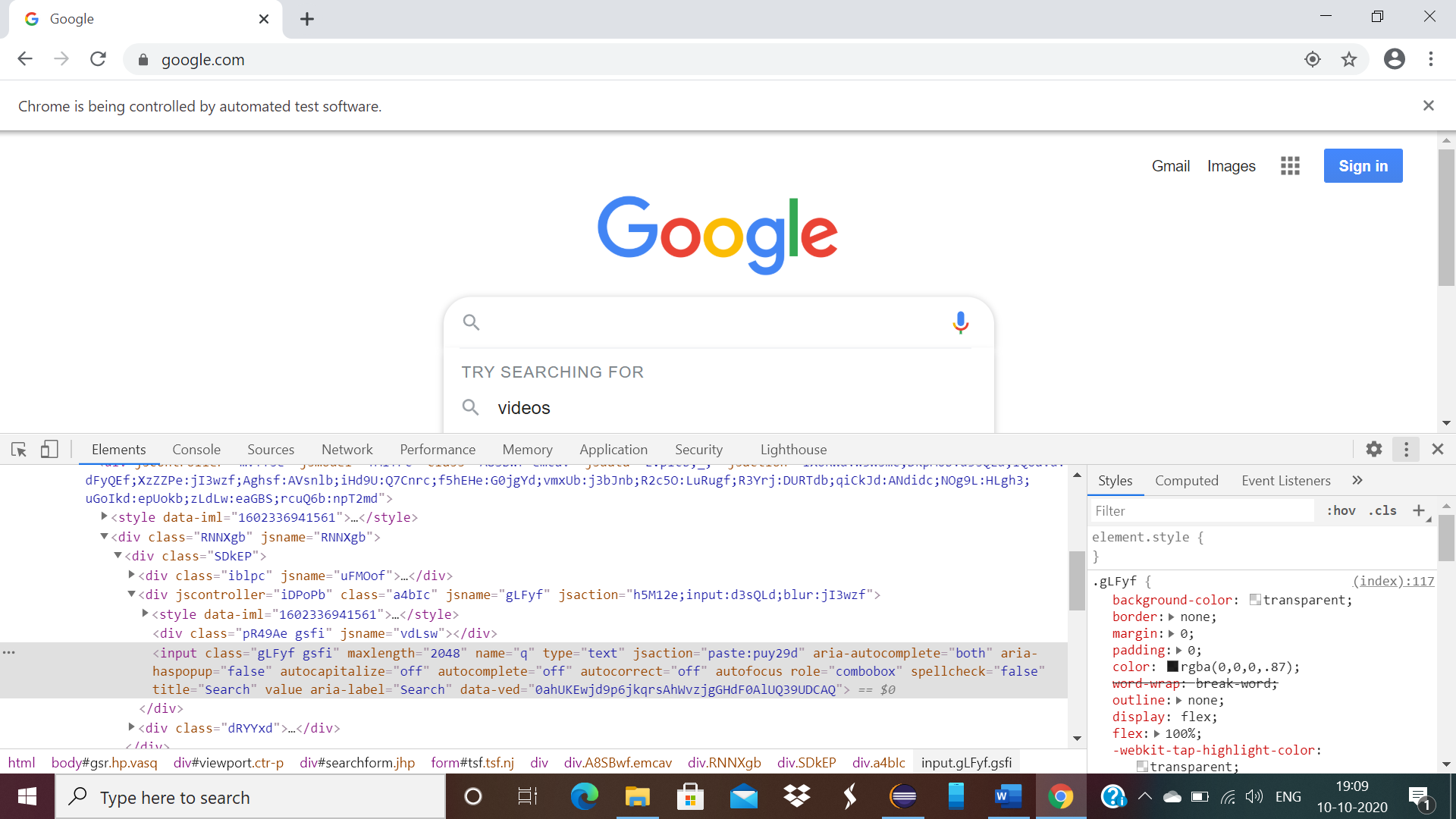
input[name\*=’ username123’]

tagname#id 🡺CSS

tagname.classname 🡺 CSS

**Parent To Child Relationship Xpath**

See Google Search Box



//div[@class='SDkEP']/div[2]/input

**Difference between Relative and Absolute X path?**

**Absolute XPath :-**It starts with the root node or a forward slash (/). The advantage of using absolute is, it identifies the element very fast.  
Disadvantage is, if any thing goes wrong like some other tag added or removed in between, then this path will no longer works.

**Relative Xpath :-**A relative xpath is one where the path starts from the node of your choise – it doesn’t need to start from the root node.

It starts with Double forward slash(//)

**Syntax:**  
//table/tbody/tr/th

**How to traverse from Siblings to parent?**

.//\*[@id=’tablist-tab1’]/parent::u

When we traverse from one sibling to other we use syntax following sibling

Xpath: "//ul/li[contains(text(),'doprep')]/following-sibling::li"

Xpath : "//li[preceding-sibling::li='doprep']"

**How to identify elements using text ?**

//\*[text()=’Soap UI’]

**CSS Selectors**

. for classname

# for Id in CSS

**Difference between driver.close and driver.quit**

Driver.close closes the current window. But sometimes child window is opened so in that case we user driver.quit to close all the windows opened by selenium

**Dropdown**

**Static Dropdown**:: dropdowns having select tag are static dropdown

Select class is given by selenium. We need to create the object of the select class to access all the methods for the select class. In the select class argument we need to pass the web-element of the dropdown.

Select s = **new** Select(driver.findElement(By.*id*("ctl00\_mainContent\_ddl\_Adult")));

s.selectByValue("2");

s.selectByIndex(6);

s.selectByValue("5 Adults");

**Non- static Dropdown**

When dropdowns do not have a select tag they are non static dropdowns.Then we follow the traditional way but we need to take care that the same element may be present twice in dropdowns. So we need to specify the node we are looking for.

driver.findElement(By.*id*("ctl00\_mainContent\_ddl\_originStation1\_CTXT")).click();

driver.findElement(By.*xpath*("//a[@value='BLR']")).click();

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

driver.findElement(By.*id*("ctl00\_mainContent\_ddl\_destinationStation1\_CTXT")).click();

driver.findElement(By.*xpath*("(//a[@value='HYD'])[2]")).click();

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

driver.close();

System.***out***.println("Test Pass");

We can also give parent// child xpath

**Autosuggestive dropdowns**

driver.get("https://rahulshettyacademy.com/dropdownsPractise/");

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

driver.findElement(By.*cssSelector*("#autosuggest")).sendKeys("Ind");

//get all the items in the auto-suggestive dropdown

List<WebElement> options = driver.findElements(By.*xpath*("//li[@class='ui-menu-item']"));

**for**( WebElement option :options)

{

**if** (option.getText().equalsIgnoreCase("India"))

{

option.click();

**break**;

}

**Handling CheckBoxes**

Once asset fails the test stops there itself and do not proceed further

driver.get("http://spicejet.com/");

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

//Assert.assertFalse expects false

Assert.*assertFalse*(driver.findElement(By.*cssSelector*("input[id\*='friendsandfamily']")).isSelected());

System.***out***.println(driver.findElement(By.*cssSelector*("input[id\*='friendsandfamily']")).isSelected());

driver.findElement(By.*cssSelector*("input[id\*='friendsandfamily']")).click();

System.***out***.println(driver.findElement(By.*cssSelector*("input[id\*='friendsandfamily']")).isSelected());

//Assert.assertTrue expects true

Assert.*assertTrue*(driver.findElement(By.*cssSelector*("input[id\*='friendsandfamily']")).isSelected());

//get the number of checkboxes

List<WebElement> countOfCheckboxes = driver.findElements(By.*cssSelector*("input[type='checkbox']"));

countOfCheckboxes.size();

System.***out***.println(countOfCheckboxes.size());

**To check if the checkbox is enabled**

Sometimes in the Ui , the checkbox is greyed out by changing one of the attributes though its enabled. Tester needs to find the attribute which is changing between two radio buttons to confirm the checkbox is enabled or disabled.

driver.get("http://spicejet.com/");

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

//driver.findElement(By.xpath("//input[@id='ctl00\_mainContent\_view\_date1']")).click();

driver.findElement(By.*id*("ctl00\_mainContent\_ddl\_originStation1\_CTXT")).click();

driver.findElement(By.*xpath*("//a[@value='BLR']")).click();

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

driver.findElement(By.*id*("ctl00\_mainContent\_ddl\_destinationStation1\_CTXT")).click();

driver.findElement(By.*xpath*("(//a[@value='HYD'])[2]")).click();

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

driver.findElement(By.*xpath*("//a[@class ='ui-state-default ui-state-highlight ui-state-hover']")).click();

driver.findElement(By.*xpath*(" //\*[@for='ctl00\_mainContent\_rbtnl\_Trip\_0']"));

System.***out***.println(driver.findElement(By.*xpath*("//\*[@for='ctl00\_mainContent\_rbtnl\_Trip\_0']")).getAttribute("class"));

String attribute = driver.findElement(By.*xpath*("//\*[@for='ctl00\_mainContent\_rbtnl\_Trip\_0']")).getAttribute("class");

**if** (attribute == "select-label")

{

Assert.*assertTrue*(**true**);

}

**else**

Assert.*assertFalse*(**false**);

System.***out***.println ("Test Pass");

**How to handle Alerts**

Page is switched to alerts via switch to alert method

driver.get("https://rahulshettyacademy.com/AutomationPractice/");

driver.findElement(By.*id*("alertbtn")).click();

System.***out***.println(driver.switchTo().alert().getText());

driver.switchTo().alert().accept();

driver.findElement(By.*id*("confirmbtn")).click();

driver.switchTo().alert().dismiss();

System.***out***.println("Test Pass");

}

**How to format a code?**

Cont+shift+F

Classname first Character is Capital letter

Variable name should start with small letter andsubsequent letters should start with Caps

**Add Items in the cart**

Arraylist supports a method contains. Array saves memory . So add items to arrey and then get it converted to arreylist in runtime to save memory.

Arraylist does not support break statement

**Actions Class**

When we need to mouse hover, we make use of actions class.

We need to create an object of the actions class and make use of it.

1. We can mouse hover
2. We can double click
3. We can right click
4. We can change caps lock
5. We can drag and drop

driver.get("https://www.amazon.com/");

Actions a= **new** Actions(driver);

a.moveToElement(driver.findElement(By.*cssSelector*("div[class='nav-line-1-container']"))).build().perform();

.build.perform is used for composite actions

driver.get("https://www.amazon.com/");

Actions a = **new** Actions(driver);

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

WebElement move = driver.findElement(By.*cssSelector*("div[class='nav-line-1-container']"));

// How to input capital caps using shift

a.moveToElement(driver.findElement(By.*xpath*("//\*[@id='twotabsearchtextbox']"))).click()

.keyDown(Keys.***SHIFT***).sendKeys("hello").build().perform();

// right click on the element

// Move the mouse to a specific element

a.moveToElement(driver.findElement(By.*cssSelector*("div[class='nav-line-1-container']"))).build().perform();

a.moveToElement(move).contextClick().build().perform();

System.***out***.println("Test Pass");

**Window Handling**

By default Webdriver only understands the parent window that is opened. getwindowhandles is a method that gives all the ids for the windows that are opened.

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

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

driver.findElement(By.*xpath*("//\*[text()='Help']")).click();

System.***out***.println(driver.getTitle());

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

When the set is created, the parent id is always stored at first and then the child 1, child2… The set always works in first come, first serve basis.

So using the iterator method.. first we navigate to the parent and then navigate to the corresponding child.

Iterator is a java method and we are creating an object it and traversing the string .

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

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

driver.findElement(By.*xpath*("//\*[text()='Help']")).click();

System.***out***.println(driver.getTitle());

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

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

String parentid=it.next();

String childid=it.next();

driver.switchTo().window(childid);

System.***out***.println("After Switching");

driver.switchTo().window(parentid);

System.***out***.println("Switching back to parent");

System.***out***.println(driver.getTitle());

**Frame Handling**

Frames are components that is hosted in the webpage. In the context of a **web** browser, a **frame** is a part of a **web** page or browser window which displays content independent of its container, with the ability to load content independently.

**How to identify the iframe:**

1. Right click on the element, If you find the option like 'This **Frame**' then it is an iframe. ...
2. Right click on the page and click 'View Page Source' and Search with the 'iframe', if you can find any tag name with the 'iframe' then it is meaning to say the page consisting an iframe.

driver.get("https://jqueryui.com/droppable/");

//First identify the frame and then see the count of frame in the webpage

System.***out***.println(driver.findElements(By.*tagName*("iframe")).size());

//Frames can be navigated via index or id

driver.switchTo().frame(driver.findElement(By.*cssSelector*("iframe[class='demo-frame']")));

//when frame size is 1 then the index to be used is always 0

//driver.switchTo().frame(0);

Actions a =**new** Actions(driver);

WebElement source = driver.findElement(By.*id*("draggable"));

WebElement target = driver.findElement(By.*id*("droppable"));

a.dragAndDrop(source, target).build().perform();

//always switch to default content once the activities inside the frame is done.

driver.switchTo().defaultContent();

**Link Assignment**

/\*driver.get("https://rahulshettyacademy.com/AutomationPractice/");

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

//get the count of all the links

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

int count = driver.findElements(By.tagName("a")).size();

System.out.println(count);

for(int i = 0;i<=count;i++)

{

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

}\*/

///////////////// this is a issue/////////////

//To get the list of Links in the footer region

driver.get("https://rahulshettyacademy.com/AutomationPractice/");

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

**int** count = driver.findElements(By.*tagName*("a")).size();

System.***out***.println(count);

WebElement footerDriver = driver.findElement(By.*xpath*("//\*[@class=' footer\_top\_agile\_w3ls gffoot footer\_style']"));

System.***out***.println(footerDriver.findElements(By.*tagName*("a")).size());

//to get the list of elements in the column

WebElement columnDriver = driver.findElement(By.*xpath*("//\*[@id='gf-BIG']/table/tbody/tr/td[1]/ul"));

System.***out***.println(columnDriver.findElements(By.*tagName*("a")).size());

**for**(**int** i =1; i<columnDriver.findElements(By.*tagName*("a")).size();i++)

{

String clickOnLinkTab = Keys.*chord*(Keys.***CONTROL***,Keys.***ENTER***);

columnDriver.findElements(By.*tagName*("a")).get(i).sendKeys(clickOnLinkTab);

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

//To get the window Tile

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

Iterator it = abc.iterator();

**while**(it.hasNext()) // it tells whether the next index is present or not

{

driver.switchTo().window(it.next());

System.***out***.println(driver.getTitle());//it.next actually moves to next window

}

}

**Table in Selenium**

driver.get("https://www.cricbuzz.com/live-cricket-scorecard/30469/kxip-vs-dc-38th-match-indian-premier-league-2020");

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

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

WebElement table = driver.findElement((By.*cssSelector*("div[class='cb-col cb-col-100 cb-scrd-hdr-rw']")));

//We are restricting the scope to elements inside the table

**int** rowCount = table.findElements(By.*cssSelector*("div[class='cb-col cb-col-27']")).size();

System.***out***.println(rowCount);

//here we are targeting the 3rd child of the row ,hence passing tagname:nth-child(index)

**int** count = table.findElements(By.*cssSelector*("div[class='cb-col cb-col-27'] div:nth-child(3)")).size();

System.***out***.println(count);

**for**(**int** i = 0;i<count-2;i++)

{

System.***out***.println(table.findElements(By.*cssSelector*("div[class='cb-col cb-col-27'] div:nth-child(3)")).get(i).getText());

}

// Find the siblings

//sum the value

System.***out***.println("Test Pass");

Count and rowcount coming as 0. Need to get it confirmed.

**Practice:: Autosuggestive dropdown session 106**

driver.get("https://www.makemytrip.com/");

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

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

WebElement source = driver.findElement(By.*xpath*("//\*[@id='fromCity']"));

source.click();

source.sendKeys("DEL");

Thread.*sleep*(2000);

source.sendKeys(Keys.***ENTER***);

destination.sendkeys(Keys.ARROW.DOWN)

destination.sendkeys(Keys.ENTER);

**JavaScript Executor**

Javascript DOM can extract the hidden elements as selenium cannot identify hidden elements (Ajax Implementation)

Also it helps to investigate the properties of object if it has any hedden texts

DOM🡺 Document Object Model

Practice lecture 107

**SSL Certifications**

Desired capabilities is a class that helps to customize the chrome browser. Pass the object of capabilities to chrome browsers.

//general chrome profile-user has designed a customized chrome profile

DesiredCapabilities ch = DesiredCapabilities.*chrome*();

ch.setCapability(CapabilityType.***ACCEPT\_INSECURE\_CERTS***,**true**);

ch.setCapability(CapabilityType.***ACCEPT\_SSL\_CERTS***, **true**);

//we can also write as below

ch.acceptInsecureCerts();

//local chrome browser-

ChromeOptions c = **new** ChromeOptions();

c.merge(ch);

System.*setProperty*("webdriver.chrome.driver", "C:\\Study\\chromedriver.exe");

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

**Maximize browser and delete cookies**

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

manage() is a method

driver.manage().deleteAllCookies();

//if we need to delete a particular cooki , then pass the cooki name

driver.manage().deleteAllCookies(“abcd”);

//add a particular cooki

**Taking Screenshots in Selenium**

We are casting the driver object to the takescreenshot method object.

driver.get("https://www.makemytrip.com/");

TakesScreenshot scrShot =((TakesScreenshot)driver);

File srcScreenshot= scrShot.getScreenshotAs(OutputType.***FILE***);

FileUtils.*copyFile*(srcScreenshot, **new** File("C://Personal//screenshot.png"));

**How to handle broken links**

**Selenium Grid**

Selenium – Grid allows you to run your tests on different machines against different browsers.

Selenium Grid support distributed test execution

You can also run the tests in parallel in multiple machines on Selenium Grid.

**HUB::**

**The hub is the machine that will receive all the test requests and distribute them to the right nodes.**

**Only one hub is present in a grid**

**The machine containing the hub is the one where the tests will be triggered but the tests will be run in nodes**

**NODE::**

**Nodes are the selenium instances that will execute the tests that you loaded in the hub**

**Nodes can be launched on multiple machines with different platforms and browsers.**

Steps::

1. Download Selenium Server Jar
2. To invoke the jar we use command java -jar selenium-server-standalone-3.8.1.jar
3. Register the hub🡺 Navigate to the place where the selenium server jar is stored 🡺 Open command prompt there🡺 java -jar selenium-server-standalone-3.8.1.jar -role hub 🡺 now this machine serves as selenium hub
4. Now login to the machine which needs to be registered as node. Node and hub cannot be same.
5. Download the selenium server file should also be downloaded in the node machine
6. Register the node🡺 Navigate to the place where the selenium server jar is stored 🡺 Open command prompt there🡺 java -jar selenium-server-standalone-3.8.1.jar -role Webdriver hub (Provide the IP address of the hub/grid/register) port 5666(port number of node can be any)🡺 now this machine serves as selenium node
7. All the properties needed to run a test case is decided by desired capabilities. When test cases are needed to be run remotely , desired capabilities class are needed. It decides which browser or which OS code needs to be executed. Cross browser testing is also supported by desired capabilities.

DesiredCapabilities dc = new DesiredCapabilities();

dc.setBrowserName(“Chrome”);

dc.setPlatform(Platform.WINDOWS);

WebDriver driver = new RemoteWebDriver(new URL(“http://localhost:4444/wd/hub”),dc);

Driver.get(“http://google.com”);

Here RemoteDriver accepts 2 arguments. One is the IP address of the server ie Hub which we can also mention as localhost and then the object of desired capabilities to understand in which browser or which OS or which browser version , the test cases needs to be run.

**TestNG**

Install the TestNG plugin🡺 Goto eclipse Help🡺 Install new software🡺mention the url mentioned in testing portal

@Test 🡺 Method that follows the annotation refers to be a testing Test. Main method is not needed in TestNG

We can run multiple tests from the single class.

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Loan Department"*>

<test name =*"Personal Loan"*>

<classes>

<class name=*"TestNGPackage.TestNGTestsSet2"*/>

<class name=*"TestNGPackage.TestNgTests"*/>

</classes>

</test>

<test name =*"Car Loan"*>

<classes>

<class name=*"TestNGPackage.TestNGTestsSet3"*/>

<class name=*"TestNGPackage.TestNGTestsSet4"*>

<methods>

<exclude name = *"Test2"*/>

</methods>

</class>

</classes>

</test>

</suite> <!-- Suite -->

Here Test2 method is excluded in xml

If we want to include only the test cases then we make use of include in XML

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Loan Department"*>

<test name =*"Personal Loan"*>

<classes>

<class name=*"TestNGPackage.TestNGTestsSet2"*>

<methods>

<include name = *"Test2"*/>

</methods>

</class>

<class name=*"TestNGPackage.TestNgTests"*/>

</classes>

</test>

<test name =*"Car Loan"*>

<classes>

<class name=*"TestNGPackage.TestNGTestsSet3"*/>

<class name=*"TestNGPackage.TestNGTestsSet4"*>

<methods>

<exclude name = *"Test2"*/>

</methods>

</class>

</classes>

</test>

</suite> <!-- Suite -->

In the similar way we can include and exclude packages as well.

@BeforeTest annotation will execute at the beginning of Any [test.@Aftertest](mailto:test.@Aftertest) will run end of the test folder but if a different test folder is present , then the test test will execute in the middle. For that another annotation is present.

@BeforeSuit (Used to load environment variable)[. @AfterSuit](mailto:.@AfterSuit) will run at the end of the entire test execution.

@BeforeClass,@BeforeMethod,@AfterClass, @AfterMethod

TestNG Groups (153)

Groups::

**public** **class** TestNGTestsSet5 {

@Test(groups=("Smoke"))

**public** **void** Demo() {

System.***out***.println("Hello5");

}

@Test

**public** **void** Test2()

{

System.***out***.println("Bye5");

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Loan Department"*>

<test name=*"Regression"*>

<groups>

<run>

<include name=*"Smoke"* />

</run>

</groups>

<classes>

<class name=*"TestNGPackage.TestNGTestsSet3"* />

<class name=*"TestNGPackage.TestNGTestsSet4"* />

<class name=*"TestNGPackage.TestNgTests"* />

<class name=*"TestNGPackage.TestNGTestsSet2"* />

<class name=*"TestNGPackage.TestNGTestsSet5"* />

</classes>

</test>

</suite> <!-- Suite -->

We can also excluede a group and include a group

Depend On method Attribute🡺 it provides interdependency on methods. It can also include multiple methods

Enabled method helps prevention of execution of tests not needed may be due to a detected bug.

Timeout method in TestNG 🡺 the test will wait for some time before its executed

Parameters

@Parameters({"URL"})

@Test

**public** **void** Test2(String urlname)

{

System.***out***.println("Bye5");

System.***out***.println(urlname);

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Loan Department"*>

<parameter name = *"URL"* value = *"qaClickAcademy.com"*/>

<test name=*"Regression"*>

<classes>

<class name=*"TestNGPackage.TestNGTestsSet3"* />

<class name=*"TestNGPackage.TestNGTestsSet4"* />

<class name=*"TestNGPackage.TestNgTests"* />

<class name=*"TestNGPackage.TestNGTestsSet2"* />

<class name=*"TestNGPackage.TestNGTestsSet5"* />

</classes>

</test>

</suite> <!-- Suite -->

**DataProvider**

@Test(dataProvider="getData")

**public** **void** TestDataprovider(String username, String password)

{

System.***out***.println("Dataprovider Test");

System.***out***.println(username);

System.***out***.println(password);

}

@DataProvider

**public** Object[][] getData()

{

//Need 3 sets of username and password-3 rows and 2 columns.Arrays always starts with 0th index

Object[][] data =**new** Object[3][2];

//1st Set

data[0][0] ="1stuserN";

data[0][1] ="1pw";

//2nd Set

data[1][0] ="2stuserN";

data[1][1] ="2pw";

//3rd Set

data[2][0] ="3stuserN";

data[2][1] ="3pw";

**return** data;

}

**TestNG Listeners**

**package** TestNGPackage;

**import** org.testng.ITestContext;

**import** org.testng.ITestNGListener;

**import** org.testng.ITestResult;

//ITestListener interface which implements TestNG Listeners

**public** **class** Listeners **implements** ITestNGListener {

**public** **void** onStart(ITestContext context) {

System.***out***.println("onStart method started");

}

**public** **void** onFinish(ITestContext context) {

System.***out***.println("onFinish method started");

}

**public** **void** onTestStart(ITestResult result) {

System.***out***.println("New Test Started" +result.getName());

}

**public** **void** onTestSuccess(ITestResult result) {

System.***out***.println("onTestSuccess Method Pass" );

}

//result.getname gives the name of the test

**public** **void** onTestFailure(ITestResult result) {

System.***out***.println("onTestFailure Method" +result.getName());

}

**public** **void** onTestSkipped(ITestResult result) {

System.***out***.println("onTestSkipped Method" +result.getName());

}

**public** **void** onTestFailedButWithinSuccessPercentage(ITestResult result) {

System.***out***.println("onTestFailedButWithinSuccessPercentage" +result.getName());

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Loan Department"*>

<listeners>

<listener class-name = *"TestNGPackage.Listeners"*/>

</listeners>

<parameter name = *"URL"* value = *"qaClickAcademy.com"*/>

<test name=*"Regression"*>

<classes>

<class name=*"TestNGPackage.TestNGTestsSet3"* />

<class name=*"TestNGPackage.TestNGTestsSet4"* />

<class name=*"TestNGPackage.TestNgTests"* />

<class name=*"TestNGPackage.TestNGTestsSet2"* />

<class name=*"TestNGPackage.TestNGTestsSet5"* />

</classes>

</test>

</suite> <!-- Suite -->

@Listeners(TestNGPackage.Listeners.**class**)

**public** **class** TestNGTestsSet5 {

**Running tests Parallely**

Information to be passed in suit level. In output the results will make no difference but time of execution is increased. Leason 159

TestNG reporting🡺 Index .html in the output file

**Properties File**

**public** **class** Properties {

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

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

Properties prop =**new** Properties();

FileInputStream fis = **new** FileInputStream("C:\\Users\\debol\\eclipse-workspace\\TestNGCode\\src\\TestNGPackage\\data.properties");

prop.load(fis);

System.***out***.println(prop.getProperty("browser"));

prop.setProperty("browser","Firefox");

System.***out***.println(prop.getProperty("browser"));

//to write the data back to the file we use fileOutput Stream

FileInputStream fos = **new** FileInputStream("C:\\Users\\debol\\eclipse-workspace\\TestNGCode\\src\\TestNGPackage\\data.properties");

prop.store(fos,**null**);

**Maven**

It is a Software project management and build management tool for java framework.

Maven project gives a dummy project

* We can avoid adding jars, maven will take care of it. User just needs to add the maven dependency. Central repository of dependencies
* Maintaining common structure across the organization
* Flexibility in integrating with CI tools
* Plugins for Test framework execution(supports TestNG, Junit .. etc)

**Terminologies**

**ArtifactID** :: An artifact is a file, usually a JAR , that gets deployed to a maven repository.

**GroupID**:: GroupID will identify your project uniquely across all projects

**Archetype:generate** : Generates a new project from an archetype

Maven Sure Fire plugin is used to run the tests in maven

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M5</version>

</plugin>

Clean🡺 mvn clean

Compile🡺mvn compile

Test🡺mvn test

Maven classes should always end with test keyword

**For integrating testing we need to add the suite level parameter in the surefire plugin**

1. <configuration>
2. <suiteXmlFiles>
3. <suiteXmlFile>testng.xml</suiteXmlFile>
4. </suiteXmlFiles>
5. </configuration>

Use profiles in maven to segregate regression, smoke etc. command is mvn test -p regression

**Jenkins**

We can create jobs and it can run anytime. Also we can run the tests without any jars needed. Its centralized as well.

**Apache POI**

POI API is used to drive data from Excel to java/selenium

* Create an object for XSSFWorkbook class present in POI APi to get an access to the excel sheet
* Need to get access to the sheet required(Collection of Rows)
* Get an access to all the rows in the sheet
* Access to specific rows which user is interested.
* Access to each and every cell of the row
* Access the data from excel to Array

**DB Connection**

create database QaDB; //Create a Database

use demo; //use command points to the database we are using

// Create a table

create table employeeinfo(name varchar(20),id int,location varchar(20), age int);

describe employeeinfo // gives the information of the table

insert table employeeinfo values(‘sam’,1,’USA’,21);

// varchar characters needs to be inserted in single quotes

**Integration od database with JDBC API**

Connectio URL:: “jdbc:mysql://”+host+”:”+port + “/databasename”;

We have created a table named Credentials and then the table has columns named scenario , username and password. Database name is Demo

String host = “localhost”;

String port = “3306”;

Connection con = DriverManager.getConnection (“jdbc:mysql://”+host+”:”+port + “Demo”);

Statement s = con.create statement();

ResultSet rs = s.executeQuery(Select \* from Credentials where Scenarios =”testing”);

While(rs.next()) // this brings the cursor to the first row. Else will throw error

{

System.out.println(rs.getString(“Username”));

System.out.println(rs.getString(“password”));

}

//The **DriverManager class** acts as an **interface** between user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. Connection is a class. getConnection is a method.

**createStatement Method** ()

Creates a SQLServerStatement object for sending SQL statements to the database.

A **ResultSet** object is a table of data representing a database **result set**, which is usually generated by executing a statement that queries the database.

**Generate Reports**

**Cucumber Framework**

Behaviour Driven Development::

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering), **behavior-driven development** (**BDD**) is an [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development) process that encourages collaboration among developers, QA and non-technical or business participants in a software project.[[1]](https://en.wikipedia.org/wiki/Behavior-driven_development#cite_note-IntroToBDD_DanNorth-1)[[2]](https://en.wikipedia.org/wiki/Behavior-driven_development#cite_note-BDD_Def_BehaviourDriven-2)[[3]](https://en.wikipedia.org/wiki/Behavior-driven_development#cite_note-IntroBDD-3) It encourages teams to use conversation and concrete examples to formalize a shared understanding of how the application should behave.

BDD is largely facilitated through the use of a simple [domain-specific language](https://en.wikipedia.org/wiki/Domain-specific_language) (DSL) using natural-language constructs (e.g., English-like sentences) that can express the behaviour and the expected outcomes. Test scripts have long been a popular application of DSLs with varying degrees of sophistication.

Advantages::

* Can be used as a standard Template for defining test cases.
* Each scenario reflects a business value
* Test coverage can be estimated
* Can be used for both manual and automation testing

Download the cucumber plugin

Naturals Plugin

Cucumber supports quick start template for maven

Jar information needs to be added to POM.xml(Cucumbet.java and cucumber.junit)

3 main things needed::

* Feature File
* StepDefinition
* Junit Test Runner

TidyGherkin is needed to create automatically , select java steps and the template is given to us

Reuse of steps using regular expressions

Given users enters “abcUsername” and “abcpassword”

In the method

Given users enters /“$%^&\*((” and /“@#$%^^&\*\*”

Here the username and password is replaced by regular expressions and we need to pass the arguments in the method

Parameterise The test

Scenario Outline: Positive test case

Given user enters <username> and <password>

Examples:

|username |password|

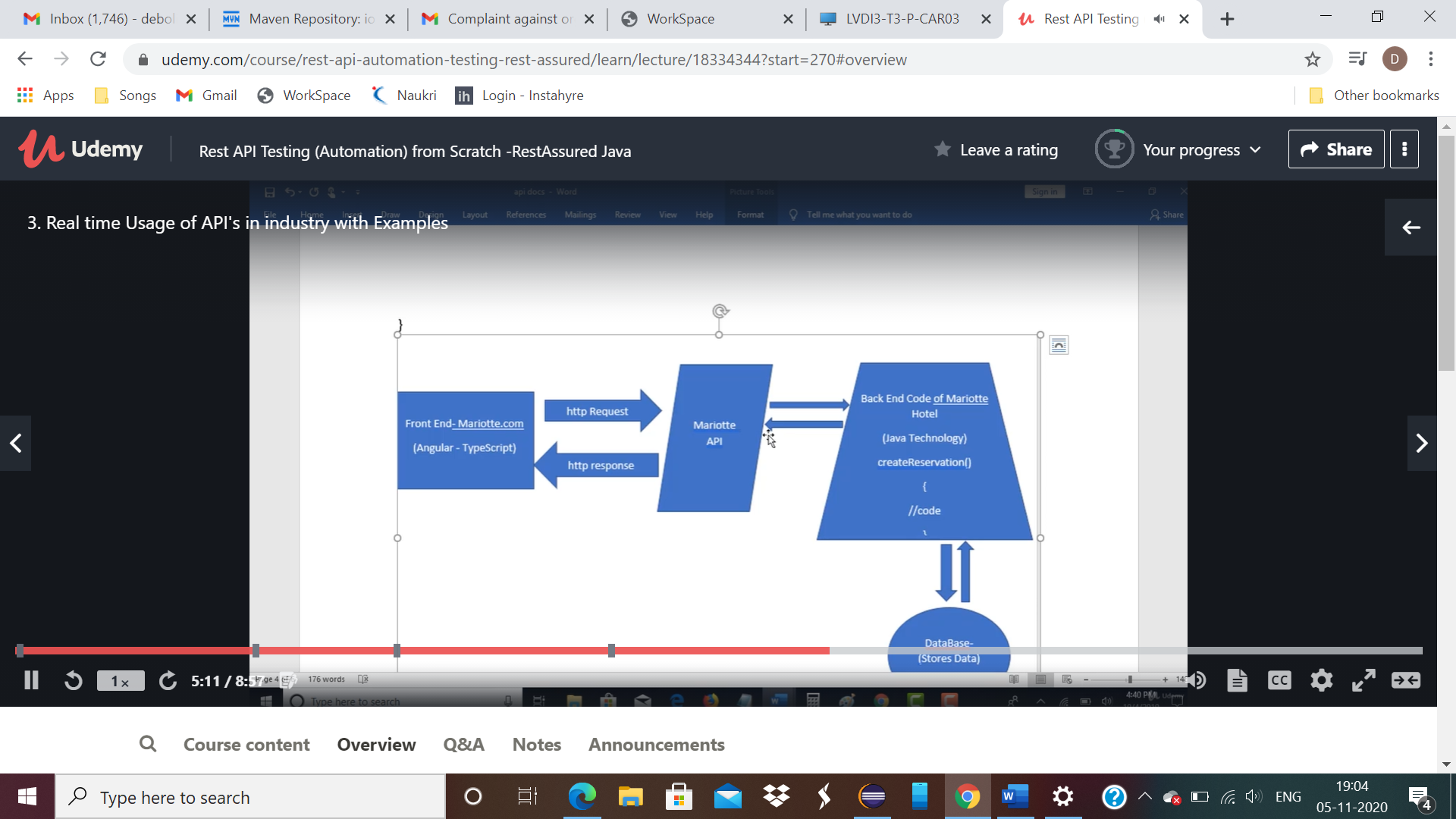
|testusername |testpawd|

Automation project is there for cucumber

**Log 4J**

**Page Object Model**

**API Testing**



**Rest API methods:: (CRUD Operations)**

**GET::** The HTTP **GET method** is used to \*\*read\*\* (or retrieve) a representation of a resource. In the “happy” (or non-error) path, **GET** returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST). Extract information from server

**POST::** The **POST method** is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. It essentially means that **POST** request-URI should be of a collection URI. PUT **method** is idempotent. Used to send data to the server.

**PUT::** Replaces all current representations of the target resource with the uploaded content.

**DELETE::** Removes all current representations of the target resource given by a URI

The PUT and DELETE methods are defined to be **idempotent**. From a RESTful service standpoint, for an operation (or service call) to be **idempotent**, clients can make that same call repeatedly while producing the same result.

**End Point**:: Address where API is hosted on the server

HTTP methods which are commonly used to communicate with Rest API’s are

**GET, POST, PUT, and DELETE**

GET- The GET method is used to extract information from the given server using a given URI. While using GET request, it should only extract data and should have no other effect on the data. No Payload/Body required

**How to send input data in GET?**  
Ans: Using Query Parameters

POST- A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.

**How to send input data in POST?**  
Ans: Using Form Parameters /Body Payload

PUT- Replaces all current representations of the target resource with the uploaded content.

DELETE- Removes all current representations of the target resource given by a URI.

**Resources:  
Resources represent API/Collection which can be accessed from the Server**

Google.com/maps  
google.com/search  
google.com/images

**Path Parameters:**  
**Path parameters** are variable parts of a URL path. They are typically used to point to a specific resource within a collection, such as a user identified by ID

<https://www.google.com/Images/1123343>  
<https://www.google.com/docs/1123343>  
<https://amazon.com/orders/112>

<https://www.google.com/search?q=newyork&oq=newyork&aqs=chrome..69i57j0l7.2501j0j7&sourceid=chrome&ie=UTF-8>

**Query Parameters:**  
Query Parameter is used to sort/filter the resources.

Query Parameters are identified with?””

https://amazon.com/orders?sort\_by=2/20/2020

**Headers/Cookies**:

Headers represent the meta-data associated with the API request and response. In layman terms, we were sending Additional details to API to process our request.  
Example : Authorization details

**End Point Request URL can be constructed as below**  
Base URL/resource/(Query/Path)Parameters

In automation via Rest see lecture below::

How to integrate postman with Jenkins below::

<https://dzone.com/articles/continuous-integration-and-testing-using-postman-w>

Git is the software using which users use git commands to access Github

Files are first added to stash and then committed. Only the files that are in stash are committed.