Why we need to instantiate chrome driver

Driver = new ChromeDriver();

The purpose of having this piece of code is to control the chrome driver. To give instruction to the chrome browser.

1. **Why do we need to inspect html element**

To access and control it.

1. **Element locator tech**

Id, Name, classname, tagname, css selector, linktext, partial link text and xpath

1. **Example for locator tech**

<input id=”Email” type=”email” autofocus=”” placeholder=”Enter your email” name=”Email” spellcheck=”false” value=””> <input id=”Passwd-hidden” class=”hidden” type=”password” spellcheck=”false” dummy=”email\_123”>

tagName = driver.findElement(By.tagName(“input”))

**cssSelector using ID**

css = driver.findElement(By.cssSelector(“input#Email”))

**cssSelector using className**

css = driver.findElement(By.cssSelector(“input.hidden”))

**Using cascaded class or class with spaces**

Css = tagname.classname1.classname2 or .classname1.classname2

**Using with one tag**

css = input[id=’Email’]

**Using with two attributes**

Css = input[id='Email'][type='email']

**Using contains with the help of \***

Css = input[dummy\*=test]

**Using starts-with with the help of ^**

Css= input[dummy^=email]

**Using ends-with with the help of $**

Css = input[dummy$=123]

**Using comma in css**

Css = Input#Email, input.hidden

1. **How to xPath find**

with contains() and starts-with()

=================================

<input id = ‘iueriRagavendran9898lkllkl’ class = ‘ksjdhf skdj jkkj’ Ragavendran/>

<input id = ‘Ragavendran9898lkllkl’ class = ‘ksjdhf skdj jkkj’ Ragavendran/>

Contains and starts-with can be used along with text / @attribute

For eg.,

//input[contains()] 🡪 //input[contains(text(),’Ragavendran’)]

//input[starts-with()] 🡪 //input[starts-with(@id,’Ragavendran’)]

1. **How will you find the Xpath when the html element is dynamic in nature**

Take the preceding element and find out the element using locator and say following :: tagname[n]

Eg., //input[@id=’email’]/following::input[1] 🡪 which gives the password field locator

1. **Get and Navigate command**

Driver.get() 🡪 used to get the url and open the page

Driver.getTitle() 🡪 used to get the title of the page

Driver.getCurrentURL 🡪 used to get the current url of the page

Driver.navigate().to()🡪 same as driver.get()

Driver.navigate().back() 🡪 take the user to the previous page in the browser

Driver.navigate().forward() 🡪 take the user to the next page in the browser

Driver.navigate().refresh() 🡪 Refresh the current page

1. **Webdriver wait commands 🡪 used for page sync**

Types a. implicit wait and b. page load timeout

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

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

1. **Element not visible exception can be solved using explicitly wait**
2. **How will you use explicit wait?**

WebDriverWait expwait = new WebDriverWait(driver,15);

expWait.until(ExpectedConditions.visibleofElementLocated(By.id(“”))));

expWait*.until(ExpectedConditions.elementToBeClickable(By.xpath(“//div[contains(text(),’COMPOSE’)]”)));*

expWait.until(ExpectedConditions.alertIsPresent()) !=null);

1. **How will you handle dropdown in webpage using selenium?**

Using select class and selectByVisibleText, selectByValue, selectByIndex functions

WebElement element = driver.findElement(By.id(“”));

Select dropdown = new Select(element);

Dropdown.selectByVisibleText(“”);

1. **How will you iterate through the list of dropdown values (all are elements in general)**

To get the list of values in the dropdown = dropdown.getOptions(); //dropdown is used from above statement

To store them in the list

List<WebElements> dropdowncategory = dropdown.getOptions();

To iterate through the list

For(webElement category : dropdowncategory){

System.out.println(category.getText());

}

1. **To practice alert and frame use the following link**

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_alert>

https://www.guru99.com/handling-iframes-selenium.html

driver.switchTo().frame(“framename”);

1. **How to switch between windows**

String parentWindow, childWindow;

Parent window = driver.getWindowHandle().toString(); // return type is string

childWindow = driver.getWindowHandles().toArray()[1].toString(); // return type is set of strings

driver.switchTo().window(childWindow);

----OR----

driver.findElement(By.id("windowButton")).click();

String mainwindow = driver.getWindowHandle();

Set<String> s1 = driver.getWindowHandles();

Iterator<String> i1 = s1.iterator();

while (i1.hasNext()) {

String ChildWindow = i1.next();

if (!mainwindow.equalsIgnoreCase(ChildWindow)) {

driver.switchTo().window(ChildWindow);

WebElement text = driver.findElement(By.id("sampleHeading"));

System.out.println("Heading of child window is " + text.getText());

driver.close();

System.out.println("Child window closed");

}

}

1. **How will you take Screen shot?**

You need two file variables named tempFile, destinationFile;

Then an interface TakesScreenshot myScreen;

destinationFile = new File(“path//filename.PNG”):

myScreen = (TakesScreenshot)driver; //cast the driver to TakesScreenshot

tempFile = myScreen.getScreenShotas(outputType.file);

FileUtils.copyFile(tempFile,DestinationFile);

1. **Keyboard and Mouse events**

Actions myAction, Action seriesofActions;

myAction = new Actions(driver);

seriesofAction = myAction.moveToElement(driver.FindElement(By.id(“”)).click().sendKeys(“”).sendKeys(Keys.Tab).build();

seriesofAction.perform();

**How to perform double click:**

Actions action = new Actions(driver);

WebElement element = driver.findElement(By.id(“”));

Action.doubleClick(element).perform();

**How to perform right click:**

Actions action = new Actions(driver);

WebElement element = driver.findElement(By.id(“”));

Action.contextClick(element).perform();

**How to perform drag and drop:**

Actions action = new Actions(driver);

WebElement element = driver.findElement(By.id(“”));

WebElement element1 = driver.findElement(By.id(“”));

Action.dragAndDrop(element,element1).build().perform();

**How to perform drag and drop:**

Actions action = new Actions(driver);

WebElement element = driver.findElement(By.id(“”));

action.moveToElement(element).perform();

1. **TestNG**
2. Add @Test annotation to run the method without main method
3. Add @Test(priority = 1) to run the methods with priority based on needs. By default, all methods as 0 priority

Note: Priority works on the following order. When no priority is mentioned then it works on the methods name say method name with capital letter runs first (A to Z order) and the lower case. When priority is given then negative takes the precedence.

1. Add dependency by @Test(dependsOnMethods = {“”})

Note: When something has dependsOnMethods annotation, then that will execute at the last with priority. Multiple depends on can be mentioned using (dependsOnMethods = {"b","d"})

1. Disable a method by saying @Test(enabled = false)
2. To group the methods use groups tag as @Test(group = “A”)
3. To execute a particular method for more than one time then use @Test(invocationCount=X)
4. To make the method run with in particular time then use @Test(invocationCount = X, invocationTimeout=X)
5. If the dependsOnMethods fails then the method which has this attribute will get skipped
6. To make the method run for the above scenario, then add a attribute alwaysRun = true as

@Test(dependsOnMethods=”f2”, alwaysRun=true)

1. **Order of testNG annotation** 🡪 BeforeSuite, BeforeTest, BeforeClass, BeforeMethod, Test, AfterMethod (runs after every method annotated with @test), AfterClass, AfterTest (run after every test tag in testing.xml file for e.g., test can have multiple class and class can have multiple methods… so this will run after all methods in each class and completes all classes), AfterSuite.
2. When you want to run the test cases in parallel, then use the attribute parallel at suite level as

<Suite parallel = “tests” thread-count=”2”>

//Note: if you want to use classes then say parallel=”classes”. The default thread count is 5.

<test>

<classes>

<class> //when parallel = test, then have two diff test cases and if parallel = class then have two diff class under one test tags

</class>

</classes>

</test>

<test>

<classes>

<class>

</class>

</classes>

</test>

</suite>

1. TestNG xml – to “include or exclude” a particular method in a class. Use include when you want to run only 2 out of 100 test case and use exclude when you want to skip only 2 test case out of 100.
2. Create a folder named config in the project
3. Create a file named testing.xml
4. In the newly create file go to source panel and write the order of tags
5. <!DOCTYPE suite SYSTEM <http://testng.org/testng-1.0.dtd>>

<Suite name = “xxx”>

<test name =”xxx”>

<classes>

<class name = “package.classname”>

**<methods>**

**<include name=”testcaseName.\*> // regex pattern**

**Note:** When to use regex pattern, when there 100 test case like 20 for car loan, 20 for bike loan, 20 for home loan and so on. The first step is we need to create test case with respective module name at first like carloan\_test1, carloan\_test2 etc., Now if we asked to exclude or include one particular module then use regex pattern as above like

**<include name=”carloan.**\*> which runs only test method those start with car

**<exclude name=”method1”>**

**</method>**

</class>

</classes>

</test>

</Suite>

1. TestNG groupings
2. Create a new XML file under the project 🡪 config folder
3. Create a testNG class with group tags as in @Test(groups = “A”)
4. If you want to include multiple value then @Test(groups={“A”,”B”})
5. Add the group tags between test tags / suite tags
6. If group tags is added after suite level, then it is common for all the test tags
7. If it is added between the test tags, then it is applicable only to that class
8. To check whether Groups tags are allowed or not, press ctrl + space between the tags
9. The order of the tags will be as follows

<Suite>

<groups>

<run>

<include name=””>

</run>

</groups>

<test>

<classes>

<class></class>

</classes>

</test>

</Suite>

1. How to test parameterization:

To use the parameterization in the program use the following annotations

@Test

@Parameters({“XXX”,”XXX”})

To pass the value for the parameters use them at config.xml file as

<Suite>

<Test>

<Parameters name=”XXX” value =”XXX”/> // press Ctrl + space to get the parameter tag after test tag

<classes>

<class>

</class>

</classes>

</test>

</Suite>

1. How to use data providers

Create a data provider by

@dataprovider(name = “ABC”)

Public Object[][] values(){

Object[][] values = null;

Values = new Object[][]{{“raghavdce@gmail.com”,”password”},{ “raghavdce@gmail.com”,”password”}}

return values;

}

To use the above data provider in any method then

@Test(dataprovider=”ABC”)

Public void logintoFB(){

}

1. Suppose you want to use the same data provider to diff methods say one method expects 2 columns and other expects 1 column, then use Method function as an argument in dataprovider

@dataprovider(name = “ABC”)

Public Object[][] values(Method m){ //use method belongs to java.lang.reflect

If (m.getName().equals(“Login”)){

Object[][] values = null;

Values = new Object[][]{{“raghavdce@gmail.com”,”password”},{ “raghavdce@gmail.com”,”password”}}

}

Else if (m.getName().equals(“print”)){

Values = new object[][]{{“Ragavendran”}}

}

Return values;

}

1. How to use data providers from different class

Say you have dataprovider defined in a different class(say testDP) and your test methods are in different class then in test method use the dataprovider as

@Test(dataprovier=”ABC”, dataproviderclass = testDP.class)

Public void logintoFB(){

}

1. **How to run the test cases based on package level instead of class using XML**

<suite>

<test>

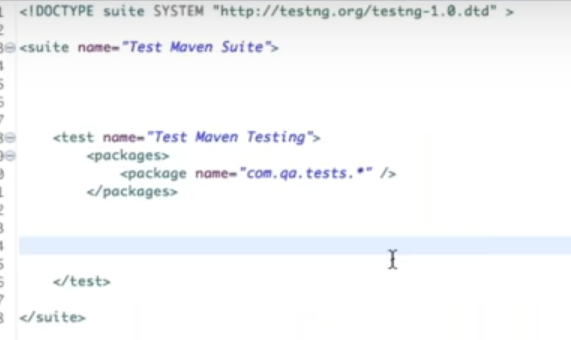
<packages>

<package name=”packageName”/>

</packages>

</test>

</suite>



1. Assertions

Hard assert 🡪 makes the test case failure and if there any other line of code after this command, it won’t run…. The syntax is

Assert.assertEquals(“Gogle”,driver.getTitle()) // use google page and the syntax will verify like (expected, actual)

Soft Assert🡪 **marks the test case as pass only if user has not used assertAll() at the end** though the assertion fails and also it allows the remaining code to get executed. Soft assert can be used by importing the softassert from testNG

SoftAssert myAssert = new SoftAssert();

myAssert.assertEquals(“Gogle”,driver.getTitle())

myAssert.assertAll(); //if we don’t use this line, test case will be marked as pass though condition fails

**assertAll() 🡪 it will be used with softassert and it will mark the test case as fail** if any of the assertion fails

**Maven:**

1. xlDownload the maven from google and install it
2. Add environment variables for maven
3. Verify whether maven is installed properly using mvn –version command in command prompt
4. Then go back to Eclispe and create a new mavenproject 🡪 New🡪 Other🡪 Maven 🡪 Maven project
5. Once Maven project is created, go to pom.xml file
6. Add the required dependencies from selenium org and testng org website under <dependencies> tag
7. Add the following plugins under <build> tag

<Plugins>

<plugin>

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

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

<version>2.20</version>

<configuration>

<suiteXmlFiles>

<suietXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

The above can also be copied from <http://maven.apache.org/surefire/maven-surefire-plugin/examples/testng.html>

1. Write your code under src/java/test folder as same as writing for testNG i.e., using Before class, Afterclass and test methods with annotation
2. Once done with the code, run the program using pom.xml by right clicking it and select RunAs🡪Maven Test
3. Also, we can run the same using RunAs🡪 Configuration…. While doing so, select the Maven build in the pop up which comes after you click configuration… And in Goals text box, type clean test (phases that are there in MAVEN clean, install, Test,package ) and click apply and then click test
4. Package 🡪 is used to convert your project into jar file

**Jenkins:**

1. Continuous integration tool
2. Download the Jenkins war file
3. Go to command prompt and change the drive where the Jenkins war file is downloaded
4. Install Jenkins by saying java –jar “path of the file” (you can drag and drop the file)
5. Once Jenkins is installed (by default it will take 8080 port)
6. Open localhost:8080 in browser and say enter
7. If it asks for user name (admin) and password (will get it from command prompt secrets/initial admin password)
8. Change the password if required
9. Login to Jenkin and move to Manage Jenkins….
10. Config system 🡪 and just click save
11. Manage Jenkins 🡪 Global Tool Config 🡪 Go to Maven Section 🡪Specify where you have the maven home (user variable path with out bin) and click save
12. Click New Item
13. Name the project / work space and use Maven project as the project type
14. Go to Build section and change the Root pom 🡪 pass the pom.xml path from your project where you created it
15. Give Goal and options as Clean Test and click save
16. Click Build Now to run the project
17. Using Configure you can set the time on which day / time you want to run the project using

* \* \* \* \* \*

**How to Create a batch file to run the normal project in Jenkins**

1. Create a java project
2. Have the code in it (as TestNG class)
3. Add the jars under build path
4. Add a folder named lib and have all the jars again
5. Create testNg.xml by convert the project to TestNG
6. Navigate to the project folder in the system and create a batchfile by creating a normal note pad
7. Type the following command java –cp bin;lib/\* org.testng.TESTNG testNG.xml and save the file with .bat extension in double quotes such as “run.bat”
8. To test whether it runs properly just double click it and see
9. To run the same in Jenkins, create a free style project and name it
10. Go the project and click advanced setting (under General tab) and add the path where you have the run.bat under use custom workspace textbox
11. In Build select “Execute Windows Batch Command” add run.bat and click save

**Selenium Grid:**

The purpose of selenium Grid is you can run the same project in different systems as parent – child / hub – node relationship

1. You must have min of 2 laptops
2. Make sure all the laptops have selenium stand-alone jars
3. Have the same name file convention and file location hierarchy across all laptops
4. To create a hub and node, make use of the command prompt
5. Open first command prompt and change the directory where you have the stand-alone jars
6. Then say java – jar “drag and drop the jar” –role hub and press enter
7. Make note of the node should be register here line
8. Open the 2nd command prompt and say java –Dwebdriver.chrome.driver = path of chromedriver.chromedriver.exe –jar selenium-server-standalone-3.5.2.jar –role node –hub paste the node’s address from hub (get it from line 7)
9. Once both the hub and nodes are up and running start creating a project
10. Create a package and have 2 testNG class one for node and other for hub
11. DesiredCapabilities capabilities = DesiredCapabilities.chrome();
12. Capabilities.setPlatform(platform.Win8.1);
13. Capabilities.setBrowser(“chrome”);
14. Driver = new RemoteWebDriver(new URL(“http://192.168.1.4:5555/wd/hub”),capabilities);
15. No need to write the system property and new chrome driver() line since it is done at cmd prompt

**How to find the broken links inside a webpage:**

**==========================================**

1. Launch the web page where you want to find the no. of broken links.
2. Add them inside a list using driver.findElements(By.tagName(“a”));
3. Declare a new array list which takes webelements
4. Iterate through the list1 with the help of size() and inside the loop use an if condition to filter the link that is not blank and not contains javascripts as in

**For(int i=0;i<list1.size();i++){**

**If(list1.get(i).getAttribute(“href”)!=null && !(list1.get(i).getAttribute(“href”).contains(“javascript”))){**

**List2.add(list1.get(i));**

**}**

**};**

1. Now list2 has webelements that is not null or does not have javascript
2. Iterate through the list2 with the help of size() to identify the links that are broken based on the response code
3. For(int j0;j<list2.size();j++){

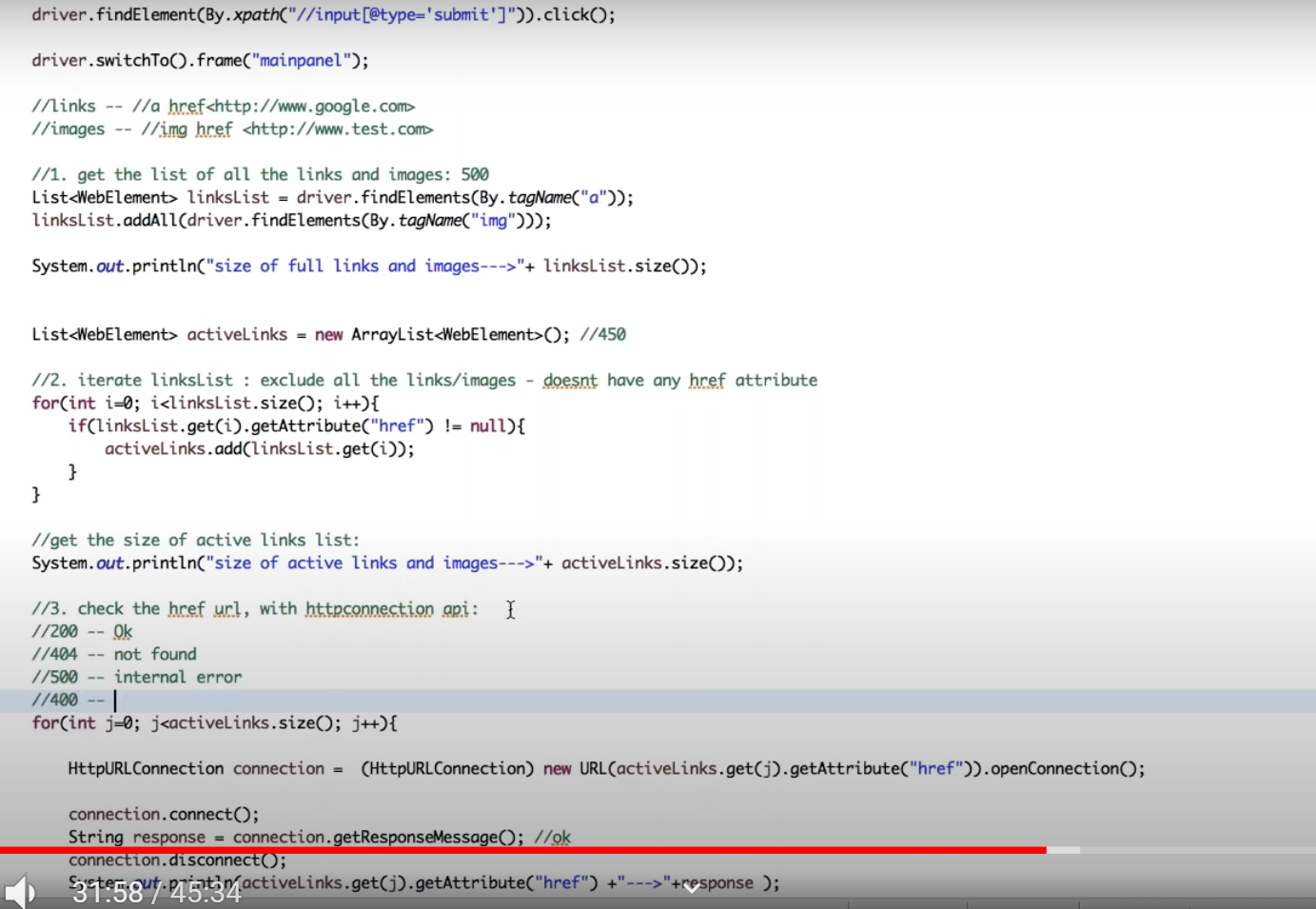
HttpURLConnection **connection** = (HttpURLConnection)newURL(list2.get(j).getAttribute(“href”)).openConnection();

connection.connect();

String response = connection.getResponseMessage();

connection.disconnect();

sysout(list2.get(j)+”🡪”+response);



**How to Read barcode using zxing api:**

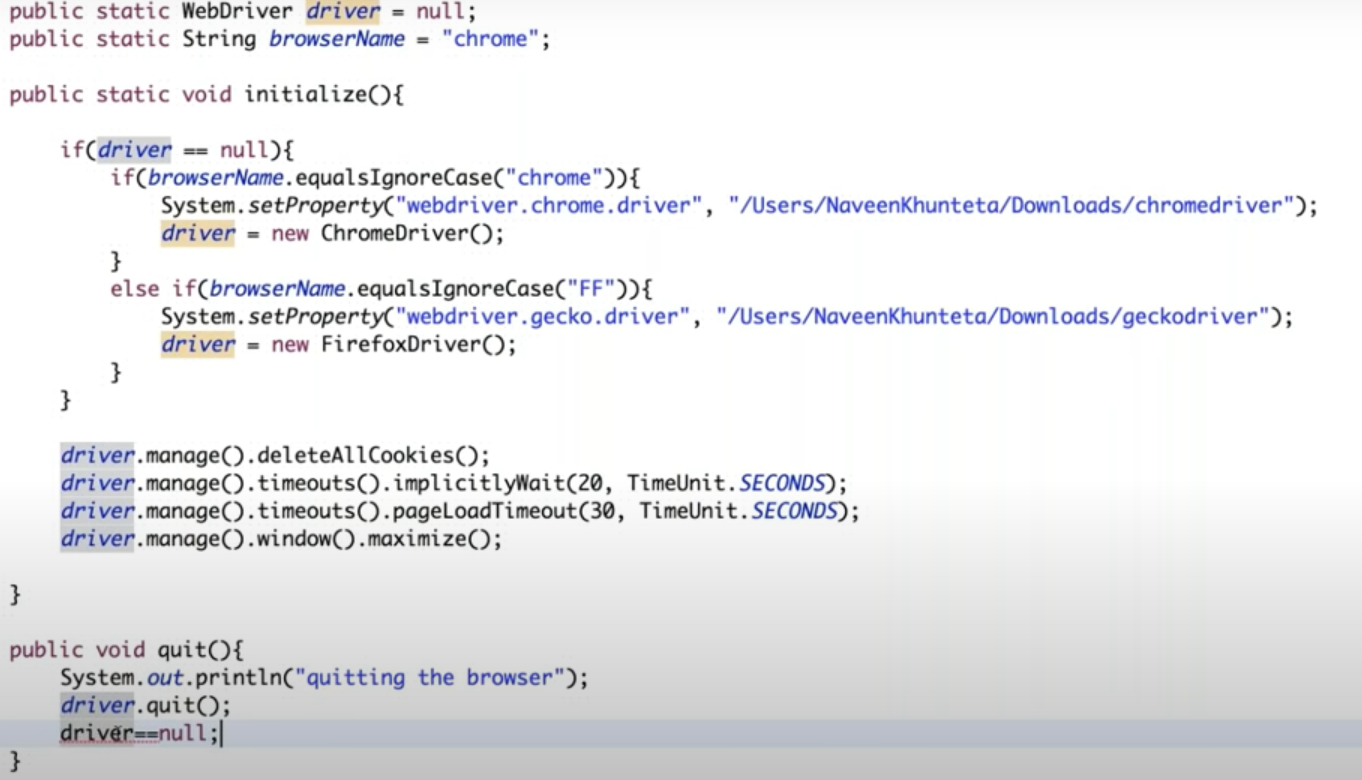
**=================================**



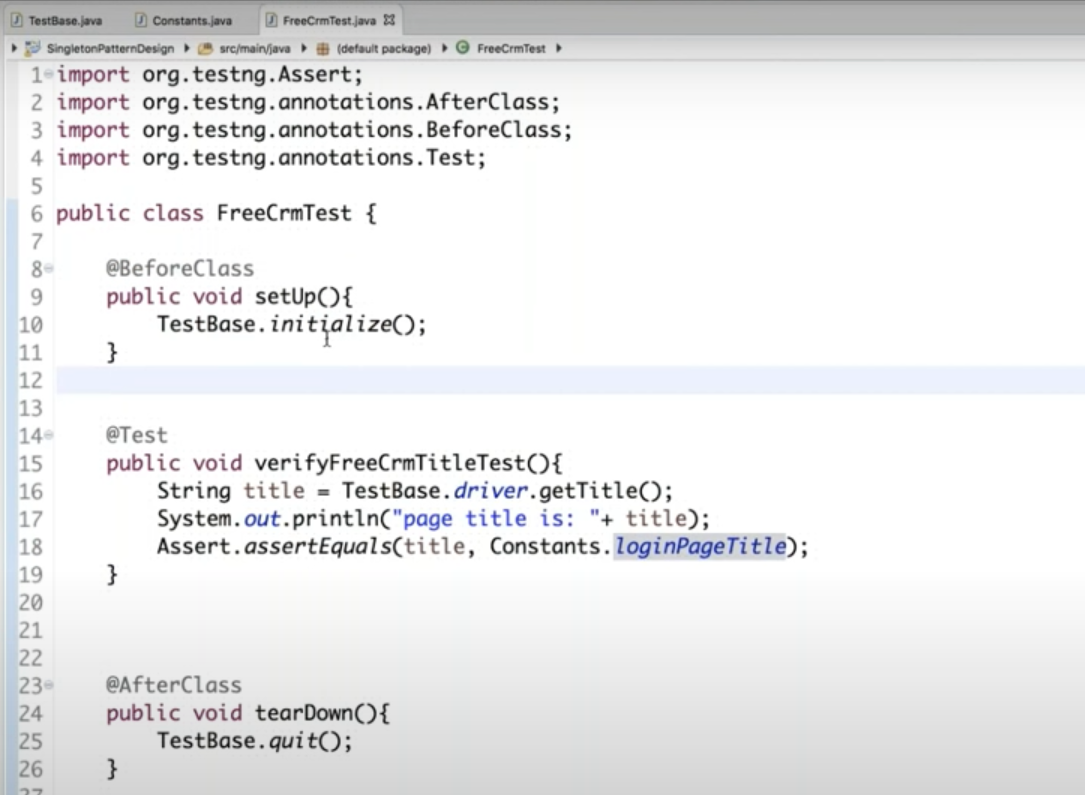
**How to use singleton design pattern to restrict the webdriver instantiation to once in F/W:**

**============================================================================**

1. In Base class, set the webdriver driver = null;
2. Then create a method where mention a condition to check if driver==null and if yes, then do the system.setproperty and driver=new ChromeDriver();
3. After that create one more method to close the browser and set driver again to null so that after each test execution the driver instantiation will set back to null



1. Then in the test class, first call the initialize method as the first line of code so that the driver instantiation will be done as below



**How to use Hashmap in selenium:**

==============================

1. First create a class say TestData and create hashmap to store user credentials inside a static method for example like

Public static Hashmap credentials(){

HashMap<String,String> map = new HashMap<>();

Map.put(“admin”,”Raghav\_Rumble);

Return map;

}

1. In test method where we want to use the credentials, declare the following variable as public so that all methods can use it

Public String credentials; // to get the value of the key

Public String credentialsInfo[]; // split the value based on “-“ and store it

1. After instantiating the webdriver, use the above two variables to get the desired value from the hashmap as in

credentials = TestData.credentials().get(“admin”);

credentialsInfo= credentials.split(“\_”);

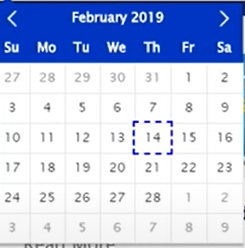
1. Then in sendkeys, use the array index to get the respective value as in

Driver.findElement(By.id(“username”).sendKeys(credentialsInfo[0]);

**How to automate calendar in selenium:**

**===================================**

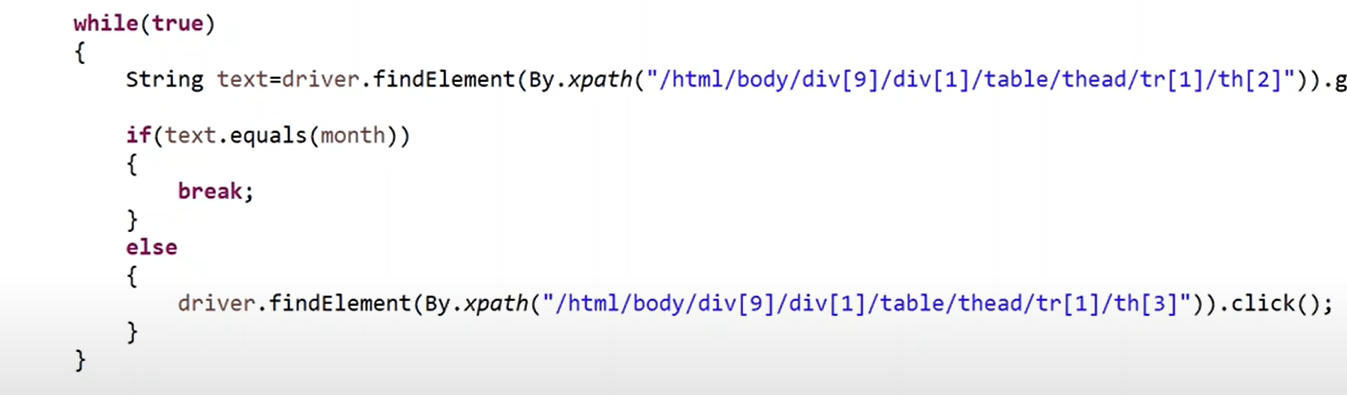
There are many types of calendar and for our understanding we are going to automate the below model



i..e, month and year together and date are from the grid.

1. Get the text of the Month and Year(February 2019), using xpath and getText() method

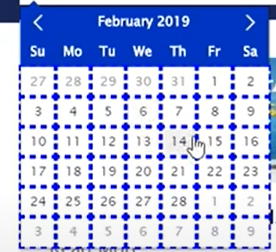
2. Create a while loop with true condition and check whether the text is as same as what we expected say “March 2019” (it can be string variable)



3. First find the xpath of a particular date and then we can customize it as per our needs i.e, parameterize row (tr) and column(td). Removing the value of Tr and td makes our customized xpath to select the entire grid







1. Now customize the xpath further to select a particular date like

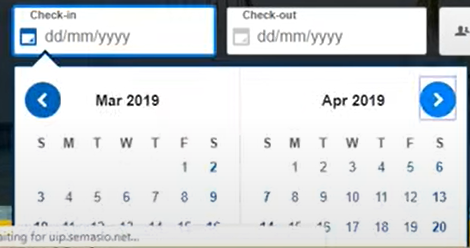




1. Also, we can modify the above code further just by making the hard-coded date to parameterized just by declaring a int variable with our desired date of the month and have the same above code as



**Another model of calendar:**





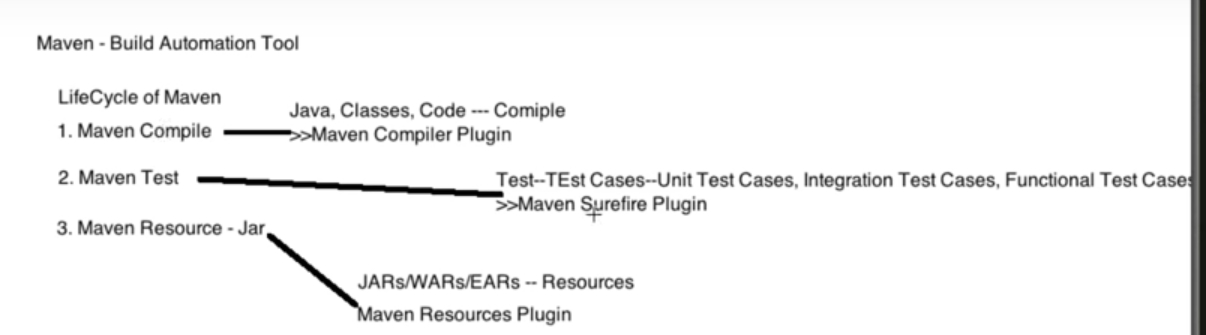
In this model we are handling in a bit different approach. i.e., all dates are found using findElements and store it in a List

Iterate through the list and get the text out of it. Then use the split concept using new line notation (since the getText() method enters the text as Month in 1 line and date in next line)

Then using if condition, check for the expected date and click it and break the “if” condition as well as the “for” each loop. Note: Break is meant for “If” condition” and since the work is done, it will exit the for each loop as well

**Maven Life Cycle:**

**================**



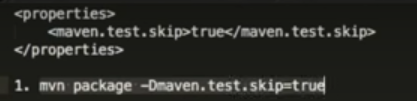
**Commands used to run test cases via Maven:**

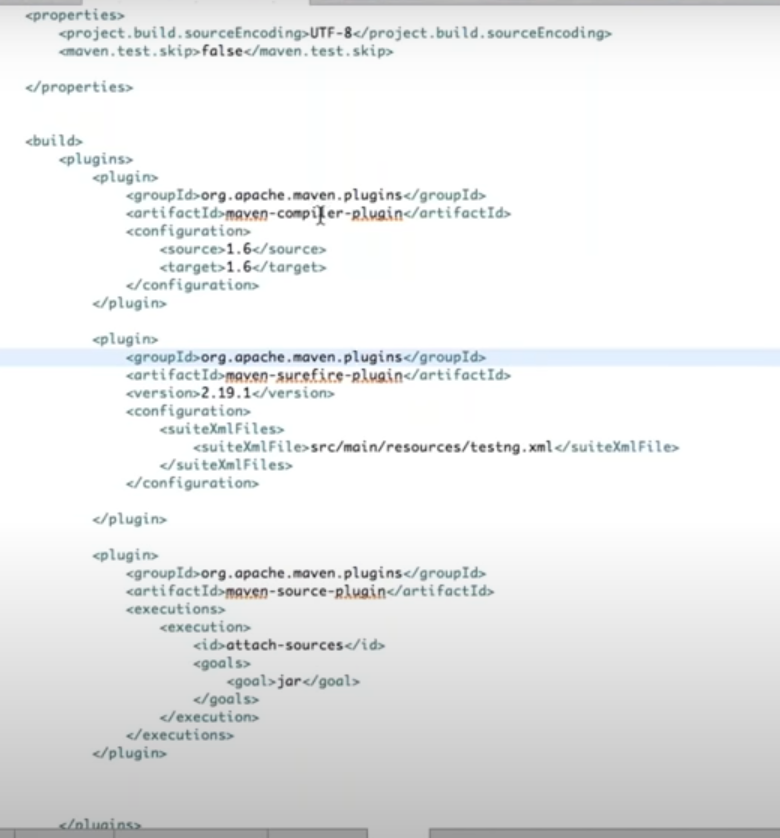
**======================================**

1. mvn clean install – OR-
2. mvn test
3. To build the project and not to run test case then use

mvn package –DskipTests

1. The above part can also be achieved using the following properties in pom.xml and command





**How to perform automation on a manually opened browser:**

First open the command prompt and type the following command and press Enter

**Note**: 1. No spaces between 2 hypens and chrome executable should have been added to run environment

chrome.exe - -remote-debugging-port=9222 - - user-data-dir=”C:\Selenium\Automaionprofile” and press enter

The above command will open the google chrome and to check whether the opened chrome is from the command prompt then open a new chrome and enter the url as “IPaddressOfTheSys:9222” and press enter. Click inspectable page. Both browsers will show the same screen.

In Eclipse:

ChromeOptions option = new ChromeOptions();

Option.setExperimentalOption(“debuggerAddress”,”ipaddress:9222”);

WebDriver driver = new Chromedriver(option);

**How to handle SSL:**

For ref 🡪 <https://www.guru99.com/ssl-certificate-error-handling-selenium.html>

for ref in selenium 4 🡪 <https://toolsqa.com/selenium-webdriver/ssl-certificate-in-selenium/>