Selenium Practice-

Code Alignment in Eclipse – Ctrl+Shift+F 🡪 formats the code properly in eclipse.

Camelcase Standard –

ClassNames. E.g. – PractiseSelenium

Debug 🡪 Double click beside any code line and Debug as - Java Application

Step by step debugging – F6

Step into particular method -= F5

Resume execution (quit Debugging) – F8

Variables and Methods – stringName

Creating driver object –

WebDriver driver = new ChromeDriver ();

ChromeDriver is a class which implements WebDriver interface.

Locators –

ID, class has numbers – means values change frequently (do not use in script)

Class name – has spaces - do not use in script (Selenium does not accept this value in locator)

Error – compound classes can’t be accepted.

Validate xpath before script runs – In console write below -

$x (“<xpath>”)

CSS - $(“<CSS>”)

Xpath Syntax –

//tagname[@attributes=’value’]

Suppose html is –

<input class="input r4 wide mb16 mt8 username" type="email" value="" name="username" id="username" aria-describedby="error" style="display: block;">

Xpath –

//input[@type=’email’]

**CSS-**

By.cssselector(“input[id=’email’]”)

No tag name – By.cssselector(“[id=’email’]”)

Or

Tagname#<value of ID attribute>

OR

Tagname.classname

Eg -

<input class="input r4 wide mb16 mt8 username" type="email" value="" name="username" id="username" aria-describedby="error" style="display:block;">

**Input#username** (is =’username’) Or SKIP TAG NAME - **#ID -> #USERNAME**

**Input. input r4 wide mb16 mt8 username**

**Regular expression of Xpath and CSS–**

In case, we have just one attribute in html of a field or attribute value is too long then we go for this.

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

tagname[attribute\*=’value’] --- **CSS**

**Exercise to practice –**

**Rahulshettyacademy.com**

**Parent child relationship xpaths –**

**Parent to child traversal –**

//div[@class='gb\_Xa gb\_Fg gb\_i gb\_Eg gb\_Jg gb\_Wf']/div/div[2]")).click()

Div [2] is second div under parent node.

Sibling traversal –

//div[@class='gb\_Xa gb\_Fg gb\_i gb\_Eg gb\_Jg gb\_Wf']/div/div[2]/following-sibling::ul

Xpath – can travel from child to parent and parent to child

CSS – cannot travel back from child to parent

//div[@class='gb\_Xa gb\_Fg gb\_i gb\_Eg gb\_Jg gb\_Wf']/div/div[2]/parent::ul

**In case, no code is available – no html –**

**//\*[text()=’ Selenium ‘]** make sure that spaces are also provided in the same way as in website.

CSS is light as compared to xpath.

Tricks for CSS

#id value

.classname

In case class name has space. For e.g. – “input first” then replace space by “.”

.input.first - This will work in Selenium if blanks are replaced by “.”

**DROPDOWN –**

On selecting a dropdown to find xpath. If we find a “Select” tag in html. This means it’s a STATIC Dropdown.

And

All options inside in it has <option> tag.

**import** org.openqa.selenium.support.ui.Select;

Select s = **new** Select(driver.findElement(By.*cssSelector*(".valid"))); // identify the xpath of dropdown element

s.selectByIndex(3);

s.selectByValue("AED");

s.selectByVisibleText("INR");

//various options to select values

While (true)

{

}

Dynamic Dropdown –

Scenario of flight booking site.

Flight from City and To City have same cities Options and have same xpaths. If we want to select a city from to dropdown. It’s not possible directly because selenium scans from top left corner. It will always locate xpath from From City Dropdown. To city will not be picked.

Solution – (CompleteXpath)[2]

One more Solution – In order to avoid indexing.

Write an xpath of Parent Child Relationship

By.xpath(“Parent Xpath<one single space>child xpath”)

**In DCAS, Private address and Mailing address have same dropdown for city names then we can use this.**

If we not do this, then ElementNotVisible Exception is thrown because script always checks in From Dropdown but we have provided some other value in that. After Selecting dropdown closes automatically but driver still searches and hence not visible.

If wrong Xpath – NoSuchElement Exception - Means unable to locate element

ElementNotInteractableException - Element is reachable by keyboard (basically expand the windows, in order to avoid this.)

InvalidElementStateException – Not visible in DOM

ElementClickInterceptedException - Indicates that a click could not be properly executed because the target element was obscured in some way.

StaleElementReferenceException: It occurs when we try to click on such element which is not present in the webpage.

AUTO SUGGESTIVE DROPDOWN –

Example – I type mum in dropdown of flight booking and the dropdown already start displaying results below of places which contains mum in their names. On clicking on enter from keyboard, the first option from auto suggestion of dropdown will be selected in dropdown.

In case, we want 2 option, then press auto down.

To wipe out already data present in text fields - .clear() method is used.

For keyboard events, Use Sendkeys(keys.<eventname>);

Eg. Xpath.sendkeys(keys.ENTER);

WEBELEMET – to store xpaths in selenium.

Eg- WebElement s = driver.findelement (By.id ……);

**Checkboxes –**

**3 Boolean methods in selenium – returns true or false.**

**IsSelected –** to determine if checkbox is selected or not.

**IsDisplayed**

**IsEnabled**

**//count the number of checkboxes**

1. Find out common locator for all checkboxes
2. Write <xpath> or <css>?
3. Driver.findelements – to find all the elements of type checkbox
4. .size method

InvalidSelectorException – When xpath provided with method “.IsSelected” is wrong.

**ASSERTIONS** –

These are present in TESTNG jar.

Download it and add in project.

Assert.assertFalse(condition here) 🡪 Condition should always return false

Similarly, Assert.assertTrue(condition here) 🡪 Condition should always return true

Asset.assetEquals(actual, expected) 🡪 It compares actual (output from script) vs expected.

Check if any WebElement is Enabled or Disabled in a Web Page-

.IsEnabled() 🡪 if enabled then True else False

But, it Fails when an element looks Disabled but on clicking on it becomes enabled.

Solution –

Check which attribute is getting changed on disabling and enabling of element.

Xpath.getAttribute(“<name of attribute which changes>”).contains(“<value that changes>”)

Example-

driver.findElement(By.*id*("ctl00$mainContent$view\_date2")).click(); // element which is disabled

**if**(driver.findElement(By.xpath("//div[@id='Div1'")).getAttribute("style").contains("1") // after clicking becomes enabled and style tag attribute value changes to 1 after click from 0.5

{

Assert.assertTrue(**true**);

}

**else**

{

Assert.assertFalse(**false**);

**Alerts** – Pop Up windows – Cannot grab xpath because these are not in html but instead JavaScript alerts. **Alert class**

**To Handle Web Based Pop Ups by below means -**

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

OR

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

Grab text from alert –

driver.switchTo().alert().getText()

**To Handle Window Based pop ups – Use Robot Classes.**

**Arrays –**

String[] arr

String arr[] ={“dd”, “tyu”};

Arrays take less memory than Array list. So it’s better to use array in code and at run time convert array into array list.

String [] items = {“platinum”, “gold”, “diamond”} ; // array

List<String> itemsneeded = Arrays.*asList* (items); //arraylist

//itemsneeded is array list name and items is array name

**WAITS – to avoid Sync issues –**

**Types –**

Implicit wait

Explicit Wait

Thread.sleep

Fluent Wait

* Implicit Wait – It is defined globally.

Eg – I want my full script to wait globally for specific number of seconds before throwing exception on page load.

Syntax –

driver.manage().timeouts().implicitlywait(3,Timeunits.SECONDS);

driver – webdriverobject and 3 is seconds

Advantages –

1. It is defined globally. If I have 100 steps which involves multiple waits.

Script always wait globally on every page load before driver object is killed.

1. It listens continuously to DOM of web page.

Eg – we have defined implicit wait of 5 seconds. But, page is loaded within 2 seconds. So implicit wait objects is listening to DOM. If it loads in 2 seconds then this will not wait for 5 seconds. After 2 seconds, it will click on element.it will not wait for 5 seconds.

Disadvantages –

1. Suppose, we have to search hotel in a city and time taken for fetching hotel results is 5 secs. When I search hotels in a country it takes 15 seconds. If we specify the implicit wait of 15 seconds and assume, due to some performance issue in app it’s taking 12 seconds to load results for cities instead of 5 sec. Our script will pass because it will wait for 15 seconds but business req is of 5 seconds. So, we will not be able to test the scenario of 5 seconds.
2. Due to some technical issue in application, results are not loaded then driver will wait for 15 secs to throw exception. Longer time.

* Explicit Wait -

Is applied to specific webelement. It does not listen to the DOM like Implicit wait.

It can be achieved in Selenium in 2 ways – One with webdriver object and other as Fluent Wait.

Syntax –

WebDriverWait wait = new WebDriverWait(driver, 5);

// webdriverwait is class defined in selenium for explicit wait

wait.until(ExpectedConditions.visibilityofElementLocated(By.cssSelector(“.btn”));

//instead of visibilityOfEle… we have many methods in java to work on.

ExpectedConditions contains a list of rules for expectations implemented by default:

visibilityOfElementLocated(By locator)

visibilityOf(WebElement element)

textToBePresentInElement(By locator, String text)

titleContains(String title)

presenceOfElementLocated(By locator)

presenceOfAllElementsLocatedBy(By locator)

invisibilityOfElementLocated(By locator)

invisibilityOfElementWithText(By locator, String text)

elementToBeClickable(By locator)

stalenessOf(WebElement element)

alertIsPresent()

* Thread.Sleep – is part of java not Selenium. It will completely halt the execution for that specified time. It will not interact with DOM.

Thread.sleep(3000); - means execution is halted for 3 seconds and if page loads within 2 seconds. It will not resume execution. It will only resume after 3 seconds.

Static Methods - Are the methods which does not require object reference to call them in main class i.e. if I have made a method static then, no need to call them thru object, can be directly called by their name in main method.

IMP:

Scope: Local variables are visible only in the method or block they are declared whereas instance variables can been seen by all methods in the class. Place where they are declared: Local variables are declared inside a method or a block whereas instance variables inside a class but outside a method

* Fluent Wait – It listens to DOM repeatedly but at regular intervals whereas explicit wait listens to DOM continuously.

There is some polling time allocated with this wait. Suppose, 10 seconds is time given and 2 sec is polling time. After every 2 seconds, fluent wait listens to DOM.

Usage –

Suppose, I have to verify confirmation messages on a payment page of a website. Messages are displaying in a particular order…

Firstly, your payment is being processed.

Secondly, your payment is successful.

Third, order is placed successfully.

I have to check if second message is displayed on webpage. But, all three messages have same attributes (i.e. same locators).

If we use explicit wait, then it will only capture first message n pass. But, in case of fluent wait, we give some polling time. After fixed polling time, it will check and if my polling time is 4 seconds and second message displays on 8th sec. I can validate that easily with fluent wait.

Keyboard events can be handled by –

ACTIONS class.

ACTION Class– It handles single user gesture performed on web application by mouse or keyboard.

* Action action = actions.build();

action.perform - not build is require here. We can directly perform.

ACTIONS CLASS – It handles various user gestures performed on web application by mouse or keyboard.

* Context Click 🡪 right click on webelement

Actions actions = new Actions(driver);

WebElement elementLocator = driver.findElement(By.id("ID"));

actions.contextClick(elementLocator).perform();

* Double click on element

Actions actions = new Actions(driver);

WebElement elementLocator = driver.findElement(By.id("ID"));

actions.doubleClick(elementLocator).perform();

* Drag and Drop the element

Syntax –

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

/Element which needs to drag.

WebElement From=driver.findElement(By.xpath("//\*[@id='credit2']/a"));

//Element on which need to drop.

WebElement To=driver.findElement(By.xpath("//\*[@id='bank']/li"));

a.dragAndDrop(From,To).build().perform()

build() -> Ready to execute and its use when we have multiple operation.

Perform() -> exceute

* Handles Mouse Hover events.

Actions a = **new** Actions(driver); // actions class in selenium under interactions.actions package.

a.moveToElement(driver.findElement(By.*xpath*(“”))).build().perform(); //always build the action first and then perform.

For single action – no need of build. Directly perform.

* Composite Actions – Multiple Actions concatenated on a single webelemnt –

E.g. -

a.moveToElement(driver.findElemet(By.xpath()) -> xpath of text box).click().keyDown(Keys.***SHIFT***).sendKeys("hello").doubleClick().build().perform();

click on webelement then text provided in send keys should display in website in caps because of SHIFT down key then the text should be selected as double click is performed.

1. Move to element
2. Click on SHIFT from down to write in CAPS keyDown(Keys.***SHIFT)***
3. Send the text
4. Do double click to select entire text

SCENARIOS –

1.[How to right click and open the application in new window.](http://www.ufthelp.com/2014/11/close-vs-quit-in-selenium.html)  
  
Code:-

Actions oAction=new Actions(driver);  
 oAction.contextClick(//div[contains(text(),'Shop Online on Amazon - Get the best deals on Amazon')]).perform();

oAction.sendKeys("w").perform();

2. How to fetch tool tip of a control in Selenium .  
  
Code:-

Actions oAction=new Actions(driver);  
 oAction.clickAndHold(driver.findElement(By.xpath())).perform();

3. Enter text in caps in textbox ?

WebElement element = driver.findElement(By.*cssSelector*(".classname"));

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

actions.moveToElement(element).click().keyDown(element, Keys.***SHIFT***).sendKeys(element , "selenium");

Action enterIncaps = actions.build();

enterIncaps.perform();

4. Drag and Drop in Selenium ?

WebElement draggable = selenium.findElement(By.xpath("//\*[@id='draggable']"));  
 WebElement droppable = selenium.findElement(By.xpath("//\*[@id='droppable']"));  
 Actions oAction = **new** Actions(selenium);  
 *//Performing Drag and Drop operation*   
 oAction.dragAndDrop(draggable, droppable).perform();

5.How to [refresh a application window](http://www.ufthelp.com/2014/11/Methods-Browser-Refresh-Selenium.html) using Action class? (Ctrl+F5)  
  
Code:-

Actions oAction=new Actions(driver);  
 oActions.keyDown(Keys.CONTROL).sendKeys(Keys.F5).perform();

6.

String col=driver.findElement(xpath).getCssValue("color");

System.***out***.println(col); // rgba(60, 60, 60, 1)

String hex = Color.*fromString*(col).asHex();

System.***out***.println(hex); // #3c3c3c actual value

String bagCol= driver.findElement(xpath).getCssValue("background-color");

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

String hex1 = Color.*fromString*(bagCol).asHex();

System.***out***.println(hex1);

7. how to crop images using selenium webdriver

Syntax: act.moveToElement(WebElement toElement, int xOffset, int yOffset).click().build().perform();

act.moveToElement(xpath of webElement, 10, 25).click().build().perform();

8.

public class ScrollByPixel {

@Test

public void ByPixel() {

System.setProperty("webdriver.chrome.driver", "E://Selenium//Selenium\_Jars//chromedriver.exe");

WebDriver driver = new ChromeDriver();

JavascriptExecutor js = (JavascriptExecutor) driver; -> doing type casting of driver

JavascriptExecutor -> interface

// Launch the application

driver.get("http://demo.guru99.com/test/guru99home/");

//To maximize the window. This code may not work with Selenium 3 jars. If script fails you can remove the line below

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

// This will scroll down the page by 1000 pixel vertical

js.executeScript("window.scrollBy(0,1000)");

OR js.executeScript("window.scrollTo(0,document.body.scrollHeight);");

}

}

executeScript("window.scrollBy(x-pixels,y-pixels)");

Multiple WINDOW HANDLING –

Parent and Child window handling. If we are on a webpage and on clicking specific element new window (child window) is opened. Then, selenium will not work with child window. It will always stick to Parent window. We have to explicitly switch our control to child windows from parent window.

Webelement windowid = driver.getWindowHandle(); // It will store the address of parent window, to get windowhandle of current window.

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

All windowids are fetched and stored in Set. First id stored will be parent window. Second one will be immediate child and third will be next child and so on..

We can iterate from one window to another using Iterator.

.next() will fetch first window ID

Next .next() will fetch next window ID and after that switch to child window.

Set<String> windowids= driver.getWindowHandles(); // set will store all windowids

Iterator<String> itr= windowids.iterator(); // Iterate between parent and child window

While(itr.hasNext())

{

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

sop(driver.getTitle());

}

String parentid = itr.next();

String childid = itr.next();

//switch to child window

driver.switchTo().window(childid);

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

//switch to parent window

driver.switchTo().window(parentid);

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

Or

Set<String> windowids= driver.getWindowHandles(); // set will store all windowids

Iterator<String> itr= windowids.iterator(); // Iterate between parent and child window

while(itr.hasNext())

{

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

sop(driver.getTitle());

}

FRAMES – This is just hosted on webpage and prepared somewhere else. It has no html source on current page. We have to handle it separately because they are not part of webpage. The html that go inside a frame may not be from the same webpage.

<iframe> tag is identification that whether a container on a webpage is in frames or not.

We can switch to frames using index, weblement and id.

If we have to find it on the basis of id, then id has to be specifically defined by developer in html source with <iframe> tag.

driver.switchTo().frame(driver.findElement(By.*cssSelector*("iframe.demo-frame")));

driver.switchTo().defaultContent(); - to come out of frames to normal page.

driver.switchTo().parentFrame(); - immediate parent frame.

Mutiple frames –

driver.switchTo().frame(driver.findElement(By.*cssSelector*("iframe.demo-frame"))).size();

Methods Provided by Selenium for Handling iFrames

**Selenium provides the following built-in methods to switch back and forth from iframes.**

switchTo.frame(int frameNumber)

switchTo.frame(string frameName)

switchTo.frame(WebElement frameElement)

switchTo().defaultContent()

TABLES- identify the parent of table first then navigate to each and every row from parent xpath.

Then, identify the common css for each n every row. Then we will traverse by index to each row.

Identify parent table css-

WebElement table=driver.findlement(By.xpath(“<table unique xpath>”))

Identify rows in a table –

Int rowcount = table.findelements(By.xpath(“<row common xpath>”)).size();

Traverse to 3rd column in a row –

table.findelement(By.xpath(“<row commmon xpath><single space>div:nth-child(3)”))

or

table.findelement(By.cssSelector(“<row commmon css><single space>div:nth-child(3)”))

Disadvantage of Selenium –

In Autosuggest Dropdowns, Dropdown text is hidden and selenium cannot get the text of such dropdown list using .getText() method.

Solution – Javascript Executor

Javascript DOM API can extract hidden elements from html.

1. Initialize object of Javascript Executor

JavascriptExecutor js = (JavascriptExecutor) driver;

1. Get element in javascript and return

String script = "return document.getElementById(\"autocomplete\").value;";

1. Execute the script

String text = (String) js.executeScript (script);

SSL Certifications –

For secured website we get on page like – your connection is not secure.

Proceed Anyways

Like – I have for DCAS – sit, dev environments.

1. Add DesiredCapabilities class and write method to accept SSL or insure certs.

DesiredCapabilities ch = DesiredCapabilities().chrome();

//1st way

ch.acceptInsecureCerts(); // this is boolean method

//2nd way

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

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

1. Write chromeoptions class and merge desiredcapabilities object to that.

ChromeOptions is used to edit properties of browser.

//Merge this capabilities to browser

ChromeOptions co = **new** ChromeOptions();

co.merge(ch);

Now pass this chromeoptions object to driver..

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

Cookies Handling oe delete Browser session –

driver.manage().deleteAllCookies();

driver.manager().deleteCookieNamed(“<cookiename>”);

Take Screenshot –

Steps –

1. Cast driver object to TakesScreenshot (import takescreenshot package)
2. getScreenshotAs method is invoked only when driver object is casted to TakesScreenshot. As parameters to this method, we specify OutputType as file.
3. After taking screenshot, output to be stored as File.
4. Use FileUtils methods to copy the screenshot captured to specific location.

If FileUtils package is not present in eclipse, download jar from commons.apache.org

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

FileUtils.*copyFile*(src, **new** File("C:\\Users\\navdeepkaurravala\\screenshot.png"));

JAVA Imp questions –

* Identifiers in java – Are the name of Packages, Variables, methods, Classes, interfaces.
* Abstract Class in java –

1. Can have constructor how, since we cannot create instance of Abstract class –

Java, an instance of an abstract class cannot be created, we can have references of abstract class type though.

An abstract class can contain constructors in Java. And a constructor of abstract class is called when an instance of an inherited class is created. For example, the following is a valid Java program.

|  |
| --- |
| // An abstract class with constructor  abstract class Base {      Base() { System.out.println("Base Constructor Called"); }      abstract void fun();  }  class Derived extends Base {      Derived() { System.out.println("Derived Constructor Called"); }      void fun() { System.out.println("Derived fun() called"); }  }  class Main {      public static void main(String args[]) {         Derived d = new Derived();      }  } |

Output:

Base Constructor Called

Derived Constructor Called

1. In java, we can have abstract class without abstract method. – This allows us to create classes which can be inherited but cannot be instantiated.
2. Abstract class can have final methods – which cannot be overriddden.

SELENIUM GRID –

Allows you to run the test cases on different machines against different browsers.

We can run tests in parallel on multiple machines.

Grid decides on which node test script is to be run.

It has one HUB and several Nodes.

Hub is where all the test scripts are stored and nodes are where these are run parallely.

* Suppose, I have office windows laptop and I want to check whether my test scripts are running on iOS. Since, I do not have infrastructure to check whether the script fails / pass in my system. Then, I will make other iOS device as node and use it run the script.
* When I have more test cases to execute then, they can be run parallely on different nodes to save time.

Steps

1. In order to run our test cases remotely, we require **selenium Standalone Server jar** server to be present in our local machine (Hub).
2. Invoke the jar from CMD and register it as Hub (server) –>

Java –jar <jarname.jar> -role hub

1. Notice one ipaddress in CMD. This is the port where this server is listening.

url/grid/console in chrome.

1. Go to other machine (desktop) and Register it as Node.
2. Download selenium Standalone server jar in other machine and invoke jar.

Java –jar <jarname.jar> -role webdriver –hub <ipaddress given in hub cmd> -port 3344 // 3344 is just random port so that node can listen

// for node use webdriver

(Check java is properly configured)

But, the above registered node has no knowledge about where the gecko driver or chrome driver path is present in test cases. Since we have all this in Hub not in Node machine. For this download, these drivers in node machine and place in same folder where selenium standalone server jar is present.

Java –Dwebdriver.chrome.driver =”C://< path of chromedriver.exe>” –jar <jarname.jar> -role webdriver –hub <ipadress/grid/console> -port 6556

// -D is used to pass parameters in cmd. This tell cmd that, while executing these parameters has to be taken care of.

To run the test remotely we need to specify that on which platform and which browser our script should run.

For this, **DesiredCapabilities is used in Selenium.**

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

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

DesiredCapabilities dc = **new** DesiredCapabilities();

dc.setBrowserName ("chrome"); // accepts browser names in small letters by default

dc.setVersion("12.1");

dc.setPlatform(Platform.***WINDOWS***);

WebDriver driver = **new** RemoteWebDriver (**new** URL ("https://localhots/4444/wd/hub"),dc);

}

If my matching Node i.e. Windows version 12.1 is not found then error thrown is

Selenium WebDriver Exception – error forwarding the new session is not found.

NOTe – We can stop node server by hitting CTRL+C in cmd.

TestNG – It is a testing framework which gives control over test cases.

* Download Testng plugIn from testNG.org website
* Go to TestNG.org
* Click on Installation
* Follow the steps –
* Go to eclipse and click on Help -> install new software -> open a window
* Paste Url mentioned in testNG.org to eclipse in the window and enter
* TestNG will be loaded , check the checkbox -> Next -> accept licence -> finish
* Go to window tab in eclipse -> prefernces -> type testNG in search. It will tell you if installed successfully.

TestNg does not require main method. It itself acts as java complier.

TestNG needs all executable code in a method and after that define TestNG anntations on top of method. Run as TestNG Test not java application this time.

Syntax –

@Test // import the annotation package to support this in current code

Public void method1()

{

//executable code

}

* In one class, we can define multiple cases in a single class file.

To Create TestNG XML ->

Go to Eclipse -> Go to project Created -> Right Click -> TestNG-> Convert to TestNg-> Next-> Finish

Structure of XML-

Test Suite – Set of test cases (can have multiple test cases)

Test

Classes

Class names

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

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

<suite name=*"FirstSuite"*>

<test name=*"Test"*>

<classes>

<class name=*"TestNG\_Package.BasicTestNG1"*/> // packagename.classname

<class name =*"TestNG\_Package.CLass2"*/> //classes can contain n number of test cases

</classes>

</test> <!-- Test -->

<test name=*"Test"*>

<classes>

<class name=*"TestNG\_Package.Class3"*/>

<class name =*"TestNG\_Package.Class4"*/>

</classes>

<class name =TestNG\_

</suite> <!-- FirstSuite -->

Run this XML as testNG Suite.

* Exclude methods from execution.

Exclude particular test cases while execution.

<Suite name =’abc’>

<test name =’day1’>

<classes>

<class name = ‘package1.test1>

<methods>

<exclude name =’loan1/>

</methods>

</class>

</classes>

* Similarly include can also be used to include a specific test

<Suite name =’includeTest’>

<test name =’my test’>

<classes>

<class name =’aa’/>

<methods>

<include name = ‘bookmark’/>

</methods>

</classes>

</test>

</suite>

Exclude or include test cases with Regex (Regular expression) while running.

E.g. – I have a class where all methods related to API are written with naming convention as – APILoginIn, APIRegister, APILogout.

If I want to exclude all cases related to API then I can use Regular expression.

<class name =’package1.testNG1’>

<methods>

<exclude name =’API.\*’/>

</methods>

</class>

* If I want to all run test cases from a package. E.g Regression ones.

<Suite name =’test1’>

<test name =’boo’>

<Packages>

<package name =’package name’/>

</packages>

<test>

</suite>

ANNOTATIONS in TestNG –

* If we have to excute a particular test before all other test cases then,

@BeforeTest Annotation is used in TestNG. – this can be used in case data has to be cleared before running test cases or set up connection

It will execute before the execution of all Test method present in the particular test folder at once .

Scope of this annotation is at Test Folder label

@AfterTest Annotation is used when we want something to run at end. E.g – delete cookies code, close connections

It will execute after the execution of all Test method present in the particular test folder at once .

Scope of this annotation is at Test Folder label

There scope is only limited to one test folder.

<Classes>

<class =”b”/>

<class = “r”/> // contains before test

</classes>

Before test is executed at first from a set of classes defined in test.

@BeforeSuite and @After Suite –It will execute before executing of all the tests methods present in the suite. Scope is in entire xml.

@BeforeSuite – we can store all the environment related data in test case with this annotation.

@BeforeMethod and @AfterMethods – can be used to exceute code before each and every Test method which is defined in a class.

Scope is limited to a class.

@BeforeClass and AfterClass – scope is limited to a class. @BeforeClass It will execute before executing of all the Test methods in a class.

https://stackoverflow.com/questions/30587454/difference-between-beforeclass-and-beforetest-in-testng#:~:text=As%20we%20know%20from%20official,tag%20is%20run. -Reference

**Methods are executed in a class in an alphabet order**

**Scenarios – If I have 100 test cases in a project and I want to run 10 test cases. How can we achieve that?**

**Solution – Groups tag**

**In a project, it is used to run specific number of smoke test cases when new build comes.**

**Syntax –**

In java class –

@Test(groups = {“Smoke”})

Public void method1()

{

{

@Test(groups = {“Smoke”})

Public void method2()

{

{

**TestNG XML –**

<suite name =”’>

<test name =”Smoke Testing”>

<groups>

<run>

<include name =”Smoke”/> //same defined in class (group tag name in a class is Smoke)

</run>

</groups>

<classes>

<class name = “packagename.Class1”/>

<class name = “packagename.Class2”/>

</classes>

**In case you want to exclude Smoke Test cases and Run others –**

<groups>

<run>

<exclude name = “Smoke”/>

</run>

</groups>

**In case, we want to execute One method before other because of dependency between Methods –**

**In java class**

**@Test(dependsOnMethods = {“Method name on which it depends” , “another method name on which it depends”})**

**//a test case can depend on one or more methods.**

**This is testng helper attribute**

**SKIP a Test case during execeution because of some bug which is already reported –**

**@Test(enabled = false) // enabled is testng helper attribute**

**TimeOut helper attribute – it can be provided if a test cases is taking too long to load.**

**@Test(timeOut = 4000) //miliseconds (this is 40 seconds here)**

**It will wait for 40 sec before throwing error.**

**Parameterization in TestNG – Can be achieved in 2 ways.**

* **@parameters – be defined in XML**
* **@DataProvider**

1. **Parameters** - Drive common variables from central place to our test cases. Examples – website URL will be same for all test cases. So, we will not hardcode this in our code, instead place at a centralized place so that all tests can access. Here, centralized place is **XML**.

We can declare these at either Suite level (applicable to all test cases) or at test folder level (applicable to specific test cases).

<suite name=*"Suite"*>

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

<parameter name =*"username"* value =*"navdeep"*/>

<test thread-count=*"5"* name=*"Test"*>

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

In testNg class –

@Parameters({"URL", “*username*”})

@Test

**public** **void** WebsiteUrl(String Urlname, String Uname)

{

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

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

}

This parameters are applicable to just one method, same process to be followed if we have to mention for others.

1. DATA PROVIDERS in TestNg – we are not writing anything in XML related to this.

These are used if I have to pass different set of data for a specific test case.

E.g- I have to test my application with different combinations of Username and Password to verify whether only specific type of users have access to application. Suppose we have to take screenshot at every failed step.

We cannot write this at each and every step. But, we can achieve this by testng listeners.

On test failure, script is routed to a block where screenshots code is written.

@Test(dataProvider = "getData")

**public** **void** Test1( String uname, String pwd) {

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

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

}

@DataProvider

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

// Declare multi-dimensional object array with rows as set of data i.e. number

// of times data has to be passed

// and column as how many parameters in one set

Object[][] data = **new** Object[3][2]; // 3 times test will run with 2 data values

// first set values

data[0][0] = "firstUsername";

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

// 2nd set values

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

data[1][1] = "secondpwd";

// 3rd set values

data[2][0] = "thirdusername";

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

// return this data so that it can be passed to method for execution

**return** data;

LISTENERS INTERFACE in TestNG – it listens to execution results.

ITestListener**– is the name of the interface which implements Testng listeners.**

Package name is - **import** org.testng.ITestListener;

**public** **class** ListenersTest **implements** ItestListener

To load unimplemented methods in TestNG -> right click on class where ItestListener is implemented -> go to source -> Override/Implement Methods -> check all the methods

required.

METHODS in ItestListener Interface-

onTestStart

onTestSuccess

onTestFailure

onTestSkipped

onTestFailedButWithinSuccessPercentage

onTestFailedWithTimeout

onStart

onFinish

XML-

<suite name=*"Suite"*>

<listeners>

<listener class-name =*"TestNG\_Package.ListenersTest"*/> // where listener is implemented

</listeners>

<test thread-count=*"5"* name=*"Test"*>

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

//details of the failed test case will be captured in result object

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

ITestListener.**super**.onTestFailure(result);

System.***out***.println("I failed test case" + result.getName()); // gets name of test case failed

//to get name of test case failed

//ItestResult class describes the result and with help of object of the this class we can fetch test case name

}

**RUN tests in parallel – parallel keyword in Xml at suite level, thread-count indicates how many test cases will run in parallel.**

In Suite level, we have test cases hence we have written parallel = “tests”

<suite name=*"FirstSuite"* parallel =*"tests"* thread-count =*"3"*>

We can run even classes also in parallel. For this, define parallel at test level.

<test name=*"Test"* parallel =*"classes"* thread-count =*"3"*>

REPORTS in Testng –Refresh project -> **test-output** folder is created.

Test-output -> index.html -> right click -> open in browser or properties -> copy location path and paste in browser.

In built reporting feature in testng

GLOBAL VARIABLES -

Suppose we have URL, username and passwords credentials which are common for all test cases. SO, instead of hard coding in test cases, we maintain these separately.

If, some change in testing URL or credentials occur in future then we have to change these in one place not in all the test cases.

This can be achieved thru Properties Class in java (selenium does have specific API to handle this.)

Parametrization is for specific set of test cases if they some parameters to be passed. But, if the parameters are generic for all test cases then Properties

.

Create a file in java project with extension as .properties. Then only this will be identified by properties class which is in java.util package.

.properties file has –

Key = value

Example –

URL ="https://www.nykaa.com/"

username = "LoginTest"

Class code –

@Test

**public** **void** Perform() **throws** IOException {

Properties prop = **new** Properties();// create object of properties class

FileInputStream fis = **new** FileInputStream("C:\\Nav\\Selenium Practise\\TestNG\_Practise\\src\\TestNG\_Package\\DataDriven.properties");

//identify where properties file is located with help of FileInputStream

prop.load(fis); // load the file to read

System.***out***.println(prop.getProperty("URL")); // get property from file

**What is the difference between Pom.xml file and Trang.xml files difference ?**

pom.xml is the [configuration file for Maven projects](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html). One of its goals is to provide assistance in the compilation and building of a project when using Maven. You can think of it as an ant build.xml file or a makefile Make file if you're not familiar to Maven (actually, it can provide a lot more functionality)

web.xml is the Java EE web application deployment descriptor, where you specify for instance servlets, servlet mappings and other aspects of a webapp.

**JAVA**

**Class -**

A class is a user defined blueprint or prototype from which objects are created.  It represents the set of properties or methods that are common to all objects of one type.

**Object -**

It is a basic unit of Object Oriented Programming and represents the real life entities.  A typical Java program creates many objects, which as you know, interact by invoking methods.

**State – Attributes of an object.**

**Behavior- methods … response**

**Identity- unique name to an object**

String class in Java –

It has lots of methods to access that we have to create object of this class.

There are 2 ways to do so –

* String s = “My name is Nav”
* String s = new String (“My name is Nav”)

If I have String a = “oh God” and String b =”oh God”, both have same content then b will point to reference of a object. No new copy is created with new value.

Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool.

But, if I declare String s = new String (“oh God”) and another one with String s1 = new String (“oh God”). Then, in this case new memory allocation will be made to store s1. This will not check whether the content present in string object already exists in heap or not.

In such case, [JVM](https://www.javatpoint.com/jvm-java-virtual-machine) will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variables will refer to the object in a heap (non-pool).

Why Strings are Immutables in java?

Immutable means – cannot be altered or changed. (Constants).

Suppose I have two strings as below –

String str1 = “navdeep”;

String str2 = “navdeep”;

In String pool memory only one space is allocated because content is same in both strings. Both str1 and str2 will point to same reference in string pool.

If, I write str1.concat(“kaur”) 🡪 this will still hold navdeep . It will not make it navdeep kaur. Because, str2 is also pointing to this. It has dependency.

In order, to achieve the mutation in String, we have two classes StringBuffer and StringBuilder. These classes are mutable (they can be altered).

String Builder – it is not thread safe. It is non-synchronized. Means that it allows multiple threads to act on a single object at same time. It is faster as compared to StringBuffer because it does not have thread safe mechanism.

String Buffer – It is thread safe which means that StringBuffer will give control to one object at a time to work on specific String variable when there are multiple objects trying to access string at same time. This avoids false results.

Comparison in Strings – there are 2 methods to compare strings –

* .equals
* ==

.equals compares the content stored in two strings but

== compares the reference of strings.

* Both s1 and s2 refers to different objects.

|  |
| --- |
| public class Test {      public static void main(String[] args)      {          String s1 = new String("HELLO");          String s2 = new String("HELLO");          System.out.println(s1 == s2);          System.out.println(s1.equals(s2));      }  } |

**Output:**

false

true

Inheritance – Reduces Redundancy of code. We can inherit the common properties.

If class to be extended in other package, then import the package as below-

Import.<package where class is present>.<Class name to be extended/inherited>

Multilevel Inheritance is possible

Interface - It has only methods with declaration (no body) – no implementation is present. Only classes can implement these interface methods.

If a class implements interface then it has to implement all the unimplemented methods defined in an interface.

By default, all methods in interface are public (access outside class as well as package).

We can access these by object of class.

All methods of an Interface do not contain implementation (method bodies) as of all versions below Java 8. Starting with Java 8, **default and static methods may have implementation in the interface definition.** ... Interfaces cannot be instantiated, but rather are implemented.

Abstraction – is the process of hiding implementation details from user, only functionality will be provided to user.

Abstract Methods – Are the methods without any body with only signatures in a class and the class which extends (inherits) this class with Abstract methods has to write implementation for these methods.

Differences between Interface and Abstract Class –

* Interface has all abstract methods but Abstract class should have at least one abstract method.
* Methods without body are called “Concrete Methods” and with body are “non-concrete”
* Interfaces can have only Concrete methods but Abstract class can have both “Concrete” and “Non-concrete” methods.
* Abstract classes achieves Partial Abstraction whereas Interfaces achieve 100% Abstraction.
* If a class has just one method Abstract then that class will be treated as Abstract Class.
* Cannot create object of Abstract classes.
* Methods in abstract class cannot be private. They can either be default, public or protected. Because, these methods will be used by child classes. If private, then child classes will not be able to access. But, interfaces can only have public. Protected is also not allowed.
* Variables in Interface must be public, static and final but in abstract variables can have all access modifiers except private.

Polymorphism - This OOPs concept lets programmers use the same word to mean different things in different contexts. Examples are – Method Overloading and Method Overridding. It allows us to create single action in different ways.

Run Time Ploymorphism –

Method Overridding – Function Name, Parameters Counts, and Data Type of parameters is same.

When an overridden method is called through a reference of parent class, then type of the object determines which method is to be executed. Thus, this determination is made at run time.  
Since both the classes, child class and parent class have the same method print (). Which version of the method (child class or parent class) will be called is determined at runtime by JVM.

ParentClass pc = new Childclass();

class Parent {

    void Print()

    {

        System.out.println("parent class");

    }

}

class subclass1 extends Parent {

    void Print()

    {

        System.out.println("subclass1");

    }

Static Polymorphism -

Method overloading and Operator overloading is example of this.

Method Overloading: This allows us to have more than one method having the same name, if the parameters of methods are different in number, sequence and data types of parameters.

Example –

static int Multiply(int a, int b)

    {

        return a \* b;

    }

    // Method with the same name but 2 double parameter

    static double Multiply(double a, double b)

    {

        return a \* b;

    }

Arrays – It is a container which stores multiple values of same data type.

* int arr[] = new int[6]; // static array , we have declared memory here

arr[0] =1;

arr[1] = 2

:

:

arr[5] = 4

* int arr[] = {1,4,1,6,5} // dynamic memory allocation.

Encapsulation – This is used for code safety. Fields within a class private, then providing access to them via public methods. In this, data of a class is hidden from other class and can be accesed only thru public getter and setter methods.

**public** **class** EncapsulationJava {

**private** String name;

**private** **int** ids;

**public** **int** getids() {

**return** ids;

}

**public** String getname() {

**return** name;

}

**public** **void** setname(String name) {

**this**.name = name;

}

**public** **void** setids(**int** ids) {

**this**.ids = ids;

}

Interview –

* Find minimum number in multidimensional array / matrix.

**int** arr[][] = { { 6, 5, 9 }, { 4, 2, 1 } }; // 2 rows and 3 column array/matrix

**int** min = arr[0][0]; // here we assume that my first number in matrix is minimum hence we capture it as min

**for** (**int** i = 0; i < arr.length; i++) { // rows

**for** (**int** j = 0; j < arr.length; j++) { // columns

**if**(arr[i][j]<min)

{

min = arr[i][j];

}

}

}

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

}

}

* Find maximum number in an array-

**int** arr[][] = {{2,6,7},{8,3,9},{7,18,2}};

**int** max = arr[0][0];

**for** (**int** i=0; i<3; i++)

{

**for** (**int** j=0; j<3; j++)

{

**if** (arr[i][j]> max)

{

max = arr[i][j];

}

}

}

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

}

* Find minimum number in an array and then find maximum number in same column where we get minimum number.

// code to fetch max number in a column where there is minimum number

// first identify min number

// identify column

// identify max number in that column

**int** arr[][] = { { 3, 1, 0 }, { 7, 6, 4 }, { 12, 2, 12 } };

**int** min = arr[0][0];

**int** mincol = 0;

**for** (**int** i = 0; i < 3; i++) {

**for** (**int** j = 0; j < 3; j++) {

**if** (arr[i][j] < min) {

min = arr[i][j]; // identify minimum number in matrix

mincol = j; // identify column where minimum number exists in matrix

}

}

}

**int** max = arr[0][mincol];

**int** k = 0;

**while** (k < 3) // identify row of that specific column , no need of 2 loop because we already

// know column

{

**if** (arr[k][mincol] > max) {

max = arr[k][mincol];

}

k++;

}

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

JAVA –

Output –

1 2 3 4

5 6 7

8 9

10

Code –

**int** k=1;

**for** (**int** i=0; i<4; i++) {

**for** (**int** j=0; j<4-i; j++) {

System.***out***.print(k);

System.***out***.print("\t");

k++;

}

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

}

------------------------------------------------------------------------------------

**public** **class** ReversePyramid {

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

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

**int** k=1;

**for** (**int** i=0; i<4; i++) {

**for** (**int** j=0; j<=i; j++) {

System.***out***.print(k);

System.***out***.print("\t");

k++;

}

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

}

}

OUTPUT –

1

2 3

4 5 6

7 8 9 10

--------------------------------------------------------------------------------------------------------------------------------------------------------

**package** CompleteJava;

**public** **class** SwapTwoNumbers {

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

**int** a = 78;

**int** b = 18;

a = a + b;

b = a - b;

a = a - b;

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

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

}

}

Pallindrome and reversal of string –

**package** CompleteJava;

**public** **class** StringReverse {

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

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

String str = "madam madam";

String newstr ="";

**for** (**int** i =str.length()-1 ; i>-1; i--)

{

newstr = newstr + str.charAt(i);

}

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

**if**(str.equalsIgnoreCase(newstr))

{

System.***out***.println("string is pallindrome");

}

**else**

{

System.***out***.println("Not a pallindrome");

}

}

}

Swap Numbers with Multiplication and Divide –

**public** **class** SwapTwoNumbersWithMultiply {

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

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

**int** a = 23;

**int** b = 43;

a = a \* b;

b = a / b;

a = a / b;

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

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

}

DATE Class in Java – it is inbuilt class present in package java.util.Date

But, if we want customized date formats. Then, we need to use –

SimpleDateFormat class present in java.text package.

Date d = **new** Date();

SimpleDateFormat sdf = **new** SimpleDateFormat("MM/dd/YYYY");

System.***out***.println(sdf.format(d));

MM/dd/YYYY – month should be in caps , days should be in small and year can be caps or small.

SimpleDateFormat sdf1 = **new** SimpleDateFormat("MM/dd/yyyy hh:mm:ss"); hh:mm:ss – small always.

DATEPICKER – to select date with

driver.get("https://www.spicejet.com/Default.aspx");

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

WebElement element = driver.findElement(By.*cssSelector*("button.ui-datepicker-trigger"));

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("arguments[0].setAttribute('value','"+"30-12-2017"+"');", element); // directly injecting the value of date in DOM .

//selenium cannot change the DOM but javascript can change

CALENDAR Class in Java –

It is present in package – java.util.calendar

It has lot of methods which date class does not have.

Calendar cal = Calendar.*getInstance*();

SimpleDateFormat sdf = **new** SimpleDateFormat("MM:dd:yyyy");

System.***out***.println(sdf.format(cal.getTime()));

System.***out***.println(cal.get(Calendar.***DAY\_OF\_MONTH***));

System.***out***.println(cal.get(Calendar.***DAY\_OF\_YEAR***));

System.***out***.println(cal.get(Calendar.***DATE***));

CONSTRUCTOR in Java - It is a block of code which is executed whenever an object of class is created. It is invoked when we create an object. It is same like method but unlike method it will never return value and its name should be same as class name.

If we do not declare constructor explicitly then the compiler will call implict (inbuilt) constructor.

Real-Time use – I will put all the variable declared in this constructor so that whenever I will create object then the block of code inside constructor will be executed. Hence, all the variables will be initialized.

2 types –

**Default constructor** – No parameters

**Parametrized** – with Parameters.

While Creating object if we do not specify any parameters then default constructor will be executed.

SCENARIO – if I have decalred default constuctor in my code and I pass parameters in the object of class. But I have not defined any parametrized constructor in class. Then, this will throw error because compiler will look for parametrized constructor now. If we have not defined any constructor then only implict is executed.

SUPER keyword in java – It will fetch the all the data from Parent class in process of inheritance.

Suppose, I have a Parent class and Child class with same variable name as

String name = “Navdeep” 🡪 this is in Parent class

String name = “Kaur” 🡪 this is in Child Class

Child class extends Parent Class

Child c = new Child ();

c.name (); //this will always call local variable in child class – local variables are given preference.

If I want Parent class method to be called then, Super.name

Same for Methods, e.g. super.methodname()

NOTE – if we want Constructor of Parent class to be called inside Child Class. Then mention like below –

Public Childclass()

{

Super(); // this will call constructor of Parent class.

Super() has to be first statement always in Child Constructor

<more code>

}

This Keyword in Java - This keyword ties to the current object. Object scope will be class level not the method level. This refers to current instance of a class.

Example,

Public class Demo(){

Int a =3;

Public void Method(){

Int a =5;

}

//object

Demo d = new Demo();

d.Method(); // this will always fetch a=5 not a=3.

If we want to fetch a=3 (at global level 🡪 class level)

Then we will use 🡪 this.a 🡪 this fetches class level variable value i.e. 3

INSTANCE variables – Variables defined in a class. They are tied up to the object of class. If object changes then these variable values will also change.

**public** **class** InstanceVariables {

// instance variables

String name;

**int** age;

String address;

//constructor

InstanceVariables(String name, **int** age, String address) {

**this**.name = name;

**this**.age = age;

**this**.address = address;

}

**public** **void** getName()

{

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

}

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

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

InstanceVariables iv = **new** InstanceVariables("Navdeep", 23, "Punjab");

InstanceVariables iv1 = **new** InstanceVariables("Nuppi", 29, "Punjab");

iv.getName();

iv1.getName();

}

Output – Navdeep

Nuppi

Static Variable – if variable is common and different objects are sharing this attribute. By static keyword we can make the copy of this variable shared among various variables. This saves memory because every time object is initialized the memory is allocated in heap. These are not dependent on instance. This belongs to class not instance. These are called class variables.

**public** **class** StaticVariables {

// instance variables

String name;

**int** age;

**static** String *address* = "Punjab"; // common among various variables

//constructor

StaticVariables(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

**public** **void** getName() {

System.***out***.println(name + " " + *address*);

}

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

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

StaticVariables sv = **new** StaticVariables("Navdeep", 23); // address is common among number of

// members. So make it static.

StaticVariables sv1 = **new** StaticVariables("Nuppi", 29);

sv.getName();

sv1.getName();

}

Local Variables – variables defined inside method and constructor scope.

Static Method – They can only have static variables. Static is independent of objects. To call a static method, we do not need objects. Just mention class name. All static methods can be directly accessed using –

Classname.methodname

Static Block - is used to define and initialize static variables.

static String city;

static int I;

static

{

City =”bangalore”;

I=0;

}

FINAL keyword– can be used with variables, classes and methods.

final int a =0; // value of a cannot be changed.

final class demo() { // final class cannot be extended (inherited)

}

final public method()

{

// final method cannot be overridden

}

PACKAGES – set of classes and interfaces.

Java.lang is a default package – inbuilt with java complier – has classes related to int , array etc.

Java.util – contains collections, interfaces.

If class A and Class b are in same package then they can access methods of each other thru objects.

Public class A{ public class B{

Public void abc () public void xyz(){

{ }

} main(){ a obj = new obj();

Obj.abc();}

To access abc in Class B – create object of class A in class B.

If class A and Class b are in different package then they can access methods of each other thru objects by importing package first.

Import.packagename.classA (classname)

Access Modifiers - public, private, protected and Default

If I do not specify any modifier with my method or variable, then java reads it as default. Can be accessed anywhere in package. This cannot be accessed outside the package even after importing packages.

Public – can be accessed across the packages after importing packages.

Private – cannot be accessed outside class, even not in same package.

Protected – can be accessed in sub class only ie. Within inherited classes in same package.

Exception Handling –

Try { //code that might fail

}

Catch(Exception e){ // exception message to be printed on catching the error

}

Exception is the parent class which catches all the exceptions – java. lang package

One try can be followed by multiple catch blocks.

Catch should be immediately followed by try block.

Finally Block – it will always be executed no matter if the script passes or fails. It will always be used with try catch blocks or only with try block. It will not be executed only when we forcefully stop our script.

finally {

}

JAVA COLLECTIONS FRAMEWORK – It is collection of interfaces and classes which stores and processes data efficiently. It contains tons of classes with useful methods which makes user work easy.

Package for Collections – java.util

Collection Types – List, Set and Map

* LIST –

It’s like array. We can store multiple values into list. It can contain duplicate values as well. We can add value and delete value at any index.

Below are the classes which implements List interface –

* ArrayList - it has dynamic size.
* ArrayList<String> a = new ArrayList<String>;

a.add(“abc”); // can add n number of values

* Whereas, Arrays have fixed length.

* In order to fetch elements present in arrayList – simply by a (ArrayList object)

But, for array we need traversal thru each n every element

* We can add new value at any index in Arraylist and other values will shift further.

Suppose, my Arryalist is “Don” and “Mag” – [Don Mag]

And I added another value at 0 index which is “lol”

My list will be – [lol Don Mag]

* Similarly, we can remove value at any index.
* a.contains(“specific String”) // boolean function – returns true if String present in ArrayList
* a.indexOf(“Navdeep”) // It will pull out index of Navdeep.
* a.isEmpty(); - returns true if arraylist is empty
* In ArrayList – a.size() but in array – a.length

ArrayList <String> al= **new** ArrayList <String>() ;

al.add("Navdeep");

al.add("Kaur");

al.add("Ravala");

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

al.add(1, "lol");

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

System.***out***.println(al.contains("Kaur"));

System.***out***.println(al.contains("pop"));

System.***out***.println(al.isEmpty());

al.remove(3);

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

al.remove("Kaur");

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

* Linked List
* Vector
* SET – Set interface does not accept duplicate values. There is no guarantee that elements are stored in sequential order while in list elements are stored in sequential order. We will not get any method related to index whereas in List we do have methods to insert or removed elements at specific index.
* Iterator Interface – If we want to iterate thru each and every object present in Set, then we will make use of this Iterator Interface.

Iterator< String> i= hs.iterator(); // hs is object of hashset

**while** (i.hasNext()) // validate till hashset has values

{

System.***out***.println(i.next()); // move to 0th index in first next, then in second next

}

}

Classes which implements SET interface are –

* Hash Set –

HashSet<String> hs = **new** HashSet<String>();

hs.add("Nav");

hs.add("deep");

hs.add("Kaur");

hs.add("ravala");

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

System.***out***.println(hs.remove("Kaur"));

System.***out***.println(hs.isEmpty());

System.***out***.println(hs.add("Nav"));//this will not be added to set-Duplicate

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

* Tree Set
* Linked Hash Set
* MAP – Map interface accepts values in key value pairs.
* Classes which implements SET interface are –
* HashMap- Functions present in HashMap are below –

1. hm.put(<key>,<value>) // put is used to add values to Hashmap. (hm is object of HasMap class)
2. hm.get(<key>) // fetch value present at particular key.
3. Hm.entrySet(); // Makes key value pairs as Set.

Now, our HashMap which stores key value pair is converted to Set.

To separate key and value , we use Map.Entry and cast hashset object to Map.Entry. If we will not Cast the hashset object to Map Entry then we wil get error because object is confused which value to fetch either key or value.

HashMap <Integer, String> hm = new HashMap <Integer, String>();

hm.put(0, "Nav");

hm.put(1, "kaur");

hm.put(2, "deep");

hm.put(3, "ravala");

System.***out***.println(hm.get(32));// prints null because no such index is present.

Set s = hm.entrySet();

Iterator i =s.iterator();

**while**(i.hasNext())

{

Map.Entry me = (Map.Entry) i.next(); // code will throw error if we do not cast this

System.***out***.println(me.getKey());

System.***out***.println(me.getValue());

}

Difference between HashMap and HashTable –

* HashMap is non-synchronized i.e. not thread safe. It means multiple programs can access HashMap at a time. Whereas, HashTable is thread safe. It allows only one program to work at a time.
* HashMap accepts “NULL” values. But, HashTable will not allow Null values.
* HashMap uses iterator to iterate thru values whereas HashTable uses Enumerator for this.
* HashTable <Integer, String> ht = **new** HashTable <Integer, String>();

// Rest all code same as HashMap

Practise –

Suppose I have Array with Duplicates values and my output should be array with unique values and how many time a value is repeated.

Array – {3,3,5,6,7,7,5,5,6,9}

Repetition – 5 🡪 3, 6🡪 2, 3🡪 2, 7🡪 2 times, 9🡪 1 time

**int** arr[] = {2,3,2,2,3,3,5,4,4,5,7};

ArrayList <Integer> al = **new** ArrayList <Integer> (); // unique values to be stored in arraylist

**for** (**int** i =0; i<arr.length; i++)

{ **int** k =0;

**if** (!al.contains(arr[i]))

{

al.add(arr[i]);

k++;

**for** (**int** j =i+1; j<arr.length;j++) { // to traverse to next elements from current element

**if**(arr[i]==arr[j]) { // array at first index is equal to next index

k++;

}

}

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

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

}

}

}

OOPS Interview questions

1. What are the core concepts of OOPS?

OOPS core concepts are;

* 1. Abstraction
  2. Encapsulation
  3. Polymorphism
  4. Inheritance
  5. Composition
  6. Association
  7. Aggregation

1. What is Abstraction?

Abstraction is an OOPS concept to construct the structure of the real world objects. During this construction only the general states and behaviors are taken and more specific states and behaviors are left aside for the implementers.

1. What is Encapsulation?

Encapsulation is an OOPS concept to create and define the permissions and restrictions of an object and its member variables and methods. A very simple example to explain the concept is to make the member variables of a class private and providing public getter and setter methods. Java provides four types of access level modifiers: public, protected, no modifier and private.

1. What is the difference between Abstraction and Encapsulation?
   1. “Program to interfaces, not implementations” is the principle for Abstraction and “Encapsulate what varies” is the OO principle for Encapsulation.
   2. Abstraction provides a general structure of a class and leaves the details for the implementers. Encapsulation is to create and define the permissions and restrictions of an object and its member variables and methods.
   3. Abstraction is implemented in Java using interface and abstract class while Encapsulation is implemented using four types of access level modifiers: public, protected, no modifier and private.
2. What is Polymorphism?

Polymorphism is the occurrence of something in various forms. Java supports various forms of polymorphism like polymorphic reference variables, polymorphic method, polymorphic return types and polymorphic argument types.

1. What is Inheritance?

A subclass can inherit the states and behaviors of its super class is known as inheritance.

1. What is multiple inheritance?

A child class inheriting states and behaviors from multiple parent classes is known as multiple inheritance.

1. What is the diamond problem in inheritance?

In case of multiple inheritance, suppose class A has two subclasses B and C, and a class D has two super classes B and C.If a method present in A is overridden by both B and C but not by D then from which class D will inherit that method B or C? This problem is known as diamond problem.

1. Why Java does not support multiple inheritance?

Java was designed to be a simple language and multiple inheritance introduces complexities like diamond problem. Inheriting states or behaviors from two different type of classes is a case which in reality very rare and it can be achieved easily through an object association.

1. What is Static Binding and Dynamic Binding?

Static or early binding is resolved at compile time. Method overloading is an example of static binding.

Dynamic or late or virtual binding is resolved at run time. Method overriding is an example of dynamic binding.

# Can we execute a java program without a main method?

Yes, we can execute a java program without a main method by using a static block.

Static block in Java is a group of statements that gets executed only once when the class is loaded into the memory by Java ClassLoader, It is also known as a static initialization block. Static initialization block is going directly into the stack memory.

class StaticInitializationBlock{

   static{

      System.out.println("class without a main method");

      System.exit(0);

   }

}

In the above example, we can execute a java program without a main method (works until Java 1.6 version). Java 7 and newer versions don’t allow this because JVM checks the presence of the main method before initializing the class.

class without a main method.

# Can we overload Java main method?

Yes, we can overload the main method of Java. But JVM will only call the default main method only. See the example below.

## Example

public class Tester {

   public static void main(String args[]) {

     System.out.println("Default Main");

   }

   public static void main(String args) {

      System.out.println("Overloaded Main");

   }

}

# Can We declare main() method as Non-Static in java?

The public **static void main(String ar[])** method is the entry point of the execution in Java. When we run a .class file JVM searches for the main method and executes the contents of it line by line.

You can write the main method in your program without the static modifier, the program gets compiled without compilation errors.

But, at the time of execution JVM does not consider this new method (without static) as the entry point of the program.  It searches for the main method which is public, static, with return type void, and a String array as an argument.

public static int main(String[] args){

}

If such a method is not found, a run time error is generated.

## Example

In the following Java program in the class Sample, we have a main method which is public, returns nothing (void), and accepts a String array as an argument. But, not static.

import java.util.Scanner;

public class Sample{

   public void main(String[] args){

      System.out.println("This is a sample program");

   }

}

On executing, this program generates the following error −

Error: Main method is not static in class Sample, please define the main method

# Can we declare a main method as private in Java?

Yes, we can declare the main method as private in Java.

It compiles successfully without any errors but at the runtime, it says that the main method is not public.

## Example:

class PrivateMainMethod {

   private static void main(String args[]){

       System.out.println("Welcome to Tutorials Point");

    }

}

The above code is working successfully at compile time but it will throw an error at the runtime.

## Output:

Error: Main method not found in class PrivateMainMethod, please define the main

method as:

public static void main(String[] args)

or a JavaFX application class must extend javafx.application.Application

## Overriding main method

You cannot override static methods and since the public static void main() method is static we cannot override it.

1. What is a Class?

A class is the specification or template of an object.

1. What is an Object?

Object is instance of class.

Java interview questions

#### What is Runtime Polymorphism?

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time.

In this process, an overridden method is called through the reference variable of a super class.

#### What is the difference between abstraction and encapsulation?

Abstraction hides the implementation details whereas encapsulation wraps code and data into a single unit.

#### What is abstract class?

A class that is declared as abstract is known as abstract class. It needs to be extended and its method implemented. It cannot be instantiated.

#### Can there be any abstract method without abstract class?

No, if there is any abstract method in a class, that class must be abstract.

#### Can you use abstract and final both with a method?

No, because abstract method needs to be overridden whereas you can't override final method.

#### Is it possible to instantiate the abstract class?

No, abstract class can never be instantiated.

**What is interface?**

Interface is a blueprint of a class that have static constants and abstract methods.It can be used to achieve fully abstraction and multiple inheritance.

 Can you declare an interface method static?

No, because methods of an interface is abstract by default, and static and abstract keywords can't be used together.

#### Can an Interface be final?

No, because its implementation is provided by another class.

#### What is marker interface?

An interface that have no data member and method is known as a marker interface.For example Serializable, Cloneable etc.

#### What is difference between abstract class and interface?

|  |  |
| --- | --- |
| Abstract class | Interface |
| 1)An abstract class can have method body (non-abstract methods). | Interface have only abstract methods. |
| 2)An abstract class can have instance variables. | An interface cannot have instance variables. |
| 3)An abstract class can have constructor. | Interface cannot have constructor. |
| 4)An abstract class can have static methods. | Interface cannot have static methods. |
| 5)You can extends one abstract class. | You can implement multiple interfaces. |

#### Can we define private and protected modifiers for variables in interfaces?

No, they are implicitly public.

#### When can an object reference be cast to an interface reference?

An object reference can be cast to an interface reference when the object implements the referenced interface.

MAVEN

Is a project and build management tool for java framework (selenium as API).

Why Maven?

* Central Repository to get Dependencies

Mvnrepository.com is the official site to get jars of all projects. (Java, phython etc.)

Imp: When we build our project with Maven Configuration, we need not add any jar eclipse path. Because, Maven project will automatically connect to their offical site “Mvnrepository.com” and look for jar needed and place it in build path.

That’s why it is called project management tool.

HOW ? Maven identifies which jars to be added to project?

Solution – search Selenium java in “Mvnrepository.com” and Look for the version installed in your machine. Click on same version on site “Mvnrepository.com” and add 4 lines of code in xml. These lines of code are Address of specific jars to be added.

Fetch dependencies from Internal drive ->

### Using the Internal Repository

Using the internal repository is quite simple. Simply make a change to add a repositories element:

1. <project>
2. ...
3. <repositories>
4. <repository>
5. <id>my-internal-site</id>
6. <url>http://myserver/repo</url>
7. </repository>
8. </repositories>
9. ...
10. </project>

* Maintaining common structure in organization –

It suggests some templates for testing and development. With help of those we can maintain consistency in folder structure among various projects. We can inject our code into these templates.

**Go to Site -Maven in 5 minutes – This has template structure**

**--src folder 🡪 it has main n test folder**

**--test folder 🡪 will contain all the test scripts**

This structure is provided by maven.

* **Flexibility in Integrating with CI tools –**

If have to run 100 test cases in one night. Then we will do this with help of Continuous Integration tool – JENKINS. To to this, we need some build mgmt. to tools to provide this info to jenkins that 100 test cases to be run. Maven will act as build mgmt tool to integrate with jenkins.

* Plugins for test framework Execution.

**Installation -**

* Go to official sit - <https://maven.apache.org/download.cgi>

{Pre-requisite – java n Home variables}

* Click on “Binary zip archive” link

Maven Terms –

* Group ID – will identify the project uniquely across all the projects. Selenium project has unique group id.
* Artifact ID – is subproject of group id. Eg. Selenium is the project and java is subproject.

How to create Maven project in eclipse –

New -> Project -> Maven -> Maven Project -> Quickstart template -> Mention Group ID and artifact id (project name) and Package name (can be same as Group id) –> Project will be created with folder structure.

Src/main/java folder will contains utilities, page object file.

Src/main/test will have test scripts.

POM.xml – will contain project dependencies. To add these, navigate to official site – mvnrepository.com and Search selenium java version (same as in machine) and Testng. Copy the dependencies in pom xml under dependencies.

Sure Fire Plugin – This plugin is used to execute all the test cases in Maven. Copy the code for this from official site of maven acpache surefire plugin -> Usage and paste in pom.xml. Copy from build tag and paste in pom.xml

To run test case -> Commands in MAVEN-

All the commands will interact with Pom.xml. We run commands thru cmd. SO, make sure the path is where pom.xml is located.

* mvn clean – cleans project. Clears previous references.
* mvn complie – scans and check syntax validations in the code. When maven is integrated with Jenkins. Then, we do not have eclipse to tell us about syntax errors. That time mvn compile will inform us in case of any errors.
* mvn test – this will trigger test execution. It executes all the test cases present in test folder with suffix TEST

NOTE – if we do not run clean and compile commands then test command will automatically do compile.

Path - C:\Nav\Selenium Practise\MavenJavaLearning>mvn clean

When we run our test for the first time then maven will fetch all the dependencies and create one folder M2 and place in it. On running test next time, maven will search for all dependencies in local folder m2. This mvn test is run thru cmd, we need not to open the eclipse.

IMP – we need to suffix the class name with Test in order to make our class picked by maven to execute. Maven only identifies the classes with Test as suffix.

TestNG Xml trigger thru Maven –

Steps below -

* Mention testng dependencies in pom.xml file by copying code of testng dependencies from official site to pom.xml
* Using Suite xml files – these files can be created in normal way and then added to sure fire plugin configuration.

Inside Surefire configuration in pom.xml, add configuration for Suite XML.

(copy configuration from using testng tab in offical site 🡪 using suite xml files)

To run tests present in single class file – in Cmd 🡪 mvn –Dtest=<classname> test

PROFILING in Maven –

Suppose I have 2 testng xml files. One has all smoke test cases and other has all Regression test cases. If we want to switch often between these test suites then everytime it is not good to change pom.xml each and every time before running test cases.

Pom.xml –

<profiles>

<profile>

<id> smoke </id>

<build>

.// all code here

.

</build>

In Command Prompt – -P -> profiles and Regression is id i.e. profile name

mvn test –Pregression

### **Build Lifecycle Basics**

Maven is based around the central concept of a build lifecycle. What this means is that the process for building and distributing a particular artifact (project) is clearly defined.

For the person building a project, this means that it is only necessary to learn a small set of commands to build any Maven project, and the [POM](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html) will ensure they get the results they desired.

There are three built-in build lifecycles: default, clean and site. The default lifecycle handles your project deployment, the clean lifecycle handles project cleaning, while the site lifecycle handles the creation of your project's site documentation.

#### A Build Lifecycle is Made Up of Phases

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

For example, the default lifecycle comprises of the following phases (for a complete list of the lifecycle phases, refer to the [Lifecycle Reference](http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference)):

* validate - validate the project is correct and all necessary information is available
* compile - compile the source code of the project
* test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
* package - take the compiled code and package it in its distributable format, such as a JAR.
* verify - run any checks on results of integration tests to ensure quality criteria are met
* install - install the package into the local repository, for use as a dependency in other projects locally
* deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

These lifecycle phases (plus the other lifecycle phases not shown here) are executed sequentially to complete the default lifecycle. Given the lifecycle phases above, this means that when the default lifecycle is used, Maven will first validate the project, then will try to compile the sources, run those against the tests, package the binaries (e.g. jar), run integration tests against that package, verify the integration tests, install the verified package to the local repository, then deploy the installed package to a remote repository.

### Goals in Maven

Goal in maven is nothing but a particular task which leads to the compiling, building and managing of a project. A goal in maven can be associated to zero or more build phases. Only thing that matters is the order of the goals defined for a given project in pom.xml. Because, the order of execution is completely dependent on the order of the goals defined.

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.testorg.testgroup</groupId>

<artifactId>project</artifactId>

<version>1.0</version>

<build>

<plugins>

<plugin>

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

<artifactId>maven-antrun-plugin</artifactId>

<version>1.1</version>

<executions>

<execution>

<id>id.pre-clean</id>

<phase>pre-clean</phase>

<goals>

<goal>run</goal>

</goals>

<configuration>

<tasks>

<echo>pre-clean phase</echo>

</tasks>

</configuration>

</execution>

<execution>

<id>id.clean</id>

<phase>clean</phase>

<goals>

<goal>run</goal>

</goals>

<configuration>

<tasks>

<echo>clean phase</echo>

</tasks>

</configuration>

</execution>

<execution>

<id>id.post-clean</id>

<phase>post-clean</phase>

<goals>

<goal>run</goal>

</goals>

<configuration>

<tasks>

<echo>post-clean phase</echo>

</tasks>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

JENKINS-it is contionuous Integration tool. Suppose, I have 10 test cases to be run as part of Sanity but all these take 5 hours. So, if these has to be run before office timing i.e. 9. Then, we not do this manually at 4AM, instead we will schedule a job in Jenkins for 4AM such that at 4, Jenkins hits maven and execute test cases.

Jenkins runs on its own server i.e. it has its own environment to run test cases, we just need to set Java and Maven versions to it. All members can set same versions in order to achieve consistency in results.

Suppose, everyone has different maven / java versions in their machine. On importing test project, they might get different results but with Jenkins, all can set same version to run and achieve consistency in results.

To run test cases in Jenkins, we need not to have java/maven setups in our system. It is Centralized system.

Process to Configure Jenkins in system –

* Download – go to official site – Generic Java Package (.war). Since we are using this locally, not hosting this on any server.
* Go to cmd -> reach the path where jenkins.war file (downloaded) is present. Then, to run the jenkins write in cmd ->

Java –jar jenkins.war –httpPort=9090 (if 8080 port is used by other process). Press Enter -> Jenkins is up and running.

* To run Jenkins, browser -> localhost:9090 -> enter -> Jenkins will open. Enter credentials – username and password – navdeep and Waheguru@91.
* In Jenkins -> Manage Jenkins -> Global Tool Configuration -> Add home path configuration of jdk and Maven.

Configuring Jenkins Setting and Workspace –

* In order to recognize our project by Jenkins either we can achieve this thru git hub or svn or we can place this project in Jenkins.
* Create a new job in Jenkins thru New Item link -> provide name and select freestyle project -> Ok -> Advanced (because we do not have git or svn (subversion)) -> Use custom workspace
* To achieve this, copy our maven project in .jenkins folder so that Jenkins can identify it. In Custom workspace –

Directory name - ${JENKINS\_HOME}/MavenJava and path where Maven project is present in.Jenkins folder in system.

Build Triggers – In this we specify when the test case triggers to be scheduled.

Build section-

Invoke top level Maven Targets -> Maven Version -> Goals: - Specify the command that Jenkins has to run like test –Pregression (regression profile). Although, our command in maven was mvn test but here Jenkins knows we are running maven only so need not to provide mvn explicitly. Save.

To run the test cases -> Go to job created -> Build now

Reports in Maven ->

Eclipse -> Refresh –> Target folder -> sure fire reports -> index.html – copy path and paste in browser

TestNg reports in Jenkins ->

* Install testng reporting plugin in Jenkins.
* Home -> Manage Jenkins -> Manage Plugins -> Install Testng Results plugin
* Testng Reports plugin interacts with testngresults.xml present in target folder of Maven Project.
* After installation -> Job (MavenJava) -> Post Build actions -> Select Publish TestNg Results -> It automatically fetches path of testng.xml file -> Save changes

Extent Reports ->

First we need to provide dependencies in Pom.xml from their official site.

2 class are involved in extent Reports –

* Extentreports – This is main class which drives reports execution. Attach object of extentsparkreporter to this class to fetch results.
* ExtentSparkReporter – This class expects path where extent report should be generated.This is basically responsible for creating report. This acts as helper class for extentreports class by providing extra configurations.
* System.getProperty(“user.dir”) ->this will provide the project path dynamically (project in which we want to store the extent reports) irrespective of whatever system we work in. :\Nav\Selenium Practise\ExtendReports\reports\index.html
* User.dir -> identifies the current project path in that reports folder will be created at run time and also index.html is created at run time.

@BeforeTest

**public** **void** configExtentReport() {

//two class are required for extend reports configuration

//ExtentReports and ExtendSparkReporter

String path = System.*getProperty*("user.dir")+"\\reports\\index.html"; // path of project folder in that reports subfolder with index.html file

ExtentSparkReporter esr = **new** ExtentSparkReporter(path);

esr.config().setReportName("My First Report"); // Change report name

// all the configurations are changed with use of config method

ExtentReports er = **new** ExtentReports(); // this will create report

er.attachReporter(esr); // attach esr object with configurations

er.setSystemInfo("Tester", "Navdeep Kaur");// set tester name

@Test

**public** **void** Start() {

er.createTest("Start"); // pass method name for which report to be generated

System.*setProperty*("webdriver.chrome.driver",

"C:\\Users\\navdeepkaurravala\\Documents\\Selenium\\Chrome Driver exe\\chromedriver.exe");

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

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

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

er.flush(); // mandatory after this only status is updated with pass or fail. If we don’t provide this then er will keep on listening.

After execution, refresh project and go to reports folder then copy the path from index.html in browser. We will be able to see extent reports.

Apache POI API – this POI API (library) connects excel to java test.

Needs 2 dependencies to be defined in pom.xml (copy dependencies from official site and paste in pom.xml) ->

1. Poi-ooxml
2. Poi

Steps to connect java to excel-

1. Identify workbook i.e. create object of XSSFWorkbook class.
2. Get access to specific sheet in a workbook because a workbook can contain n number of sheets.
3. Access all the rows in a sheet
4. From all the rows, get access to specific row.
5. Get access to all the cells and access the data from specific set.

NOTE – in POI API all methods are for rows. It does not contain any column specific methods.

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

ArrayList<String> arr = **new** ArrayList<String>();

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

// to identify the workbook path

FileInputStream fis = **new** FileInputStream("C:\\Users\\navdeepkaurravala\\Documents\\Selenium\\ExcelPOI.xlsx");

XSSFWorkbook workbook = **new** XSSFWorkbook(fis);

// get count of number of sheets present in a workbook

**int** count = workbook.getNumberOfSheets();

// Traverse thru all the sheets and fetch the desired sheet

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

// verify that the sheet present at particular index has name as Test Case, if

// yes go in below IF loop else No

**if** (workbook.getSheetName(i).equalsIgnoreCase("Test Case")) {

// Sheet is identified in above Step and now save that value in XSSFSheet object

XSSFSheet sheet = workbook.getSheetAt(i);

// Once sheet is identified -> identify rows in a sheet

// since sheet is a collection of rows, so iterate thru each first row and

// verify

// if Test Data column is present in first row.

Iterator<Row> itr = sheet.iterator(); // iterate thru row with help of iterator

Row firstrow = itr.next(); // itr.next fetches first row data i.e. headings

// Row contains n number of cells, now we have to fetch cell with value as 'Test

// Data'

Iterator<Cell> cellitr = firstrow.cellIterator();

// iterator till cells have value

**int** k = 0;

**int** column = 0;

**while** (cellitr.hasNext()) {

Cell cellval = cellitr.next(); // Traverse to all cells in a row with .next() method and store value

k++; // in a Cell variable

**if**(cellval.getStringCellValue().equalsIgnoreCase("Test Data"))// identify whether Test Data cell is

// present

{ // grab the column index of

// get column index of Test Data column - for that declare k n column variable

// whenever cell.hasnext - k is incremented

column = k;

}

// now our column is identified and index is stored in column , now identify

// where our test case is present in that column

}

//itr is declared earlier to capture rows, if more rows i.e. hasnext()

**while**(itr.hasNext()) {

// pull row in variable of type Row

Row row = itr.next();

//if row with column value of test data then fetch where our test case name is present

**if**(row.getCell(column).getStringCellValue().equalsIgnoreCase("<name of test>")) {

//Once a particular test row is present , fetch all data values in that row using cell iterator

Iterator <Cell> cv = row.cellIterator();

**if**(cv.hasNext()) {

Cell c = cv.next(); // traverse thru cells in a row

**if**(c.getCellType() == CellType.***STRING***)

{

arr.add(c.getStringCellValue());

}

**else** {

arr.add(NumberToTextConverter.*toText*(c.getNumericCellValue()));

}

}

}

}

}

}

APACHE LOG4j – It’s a logging framework commonly used in integration with selenium.

Why Log4j ?

* For every step we perform in test script, we need logs.
* These are useful, if we need only particular log data. Example – only failed test cases logs.
* Logs with Timestamps. These are useful when we need to debug our code at what time the error occurred.
* Scenario- Suppose code is deployed to Prod but some issue occurred. But, this was not present in testing environment. Then, as a proof testers can submit log report to validate that they have succssfully covered the module which failed in prod.

Log4j Setup –

* Go to official site of Apache -> download Apache 4j. -> Download Zip files.

Code-

**public** **class** Log4jDemo {

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

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

Logger log = LogManager.*getLogger*(Log4jDemo.**class**.getName()); // getlogger has

log.debug("Im debugging");

**if**(5>4)

{

log.info("Object is found");

log.error("Object is not found");

log.fatal("lallallal");

log.debug("Im debugging");

}

}

Output –

ERROR Pack1.Log4jDemo - Object is not found

FATAL Pack1.Log4jDemo – lallallal

log.info("Object is found"); and log.debug("Im debugging"); are not printed in output because configuration file is not present. Without configuration file, it only prints error or fatal messages.

\*\*\*\*\*\*\*When to Use log. Error, debug and info methods in Selenium test cases\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Use log. Error() to log when elements are not displayed in the page or if any validations fail

Use Log. Debug() When each Selenium action is performed like click, SendKeys, getText()

Use log.info() when operation is successfully completed ex: After loading page, or after any successful validations

Log.Fatal() - It’s just counterpart to log. Error().

Configuration file (.xml) in log4j –

* Where to Log?

I.e. Console output or separate file.

Appenders tag – has the info where to log the erors or messages.

For console - <Console> tag is present in Appenders.

* How to Log?

<Patterns> tag is used to customize the text printed before message. E.g. – Timestamp / ERROR: etc.

* What to Log?

Means- entire output or errors. This info to be given in <loggers> tag.

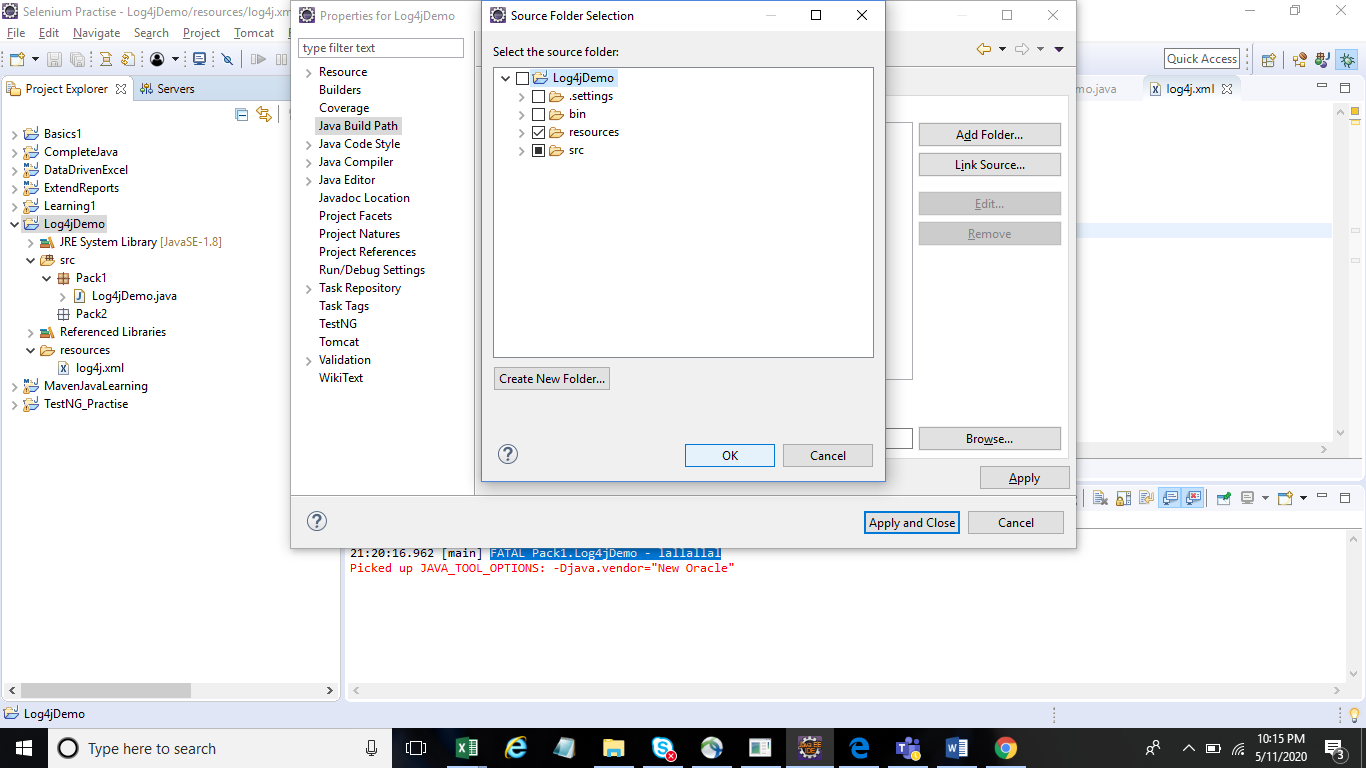
If we mention level = errors in root tag – then this will log only errors.

If we say, level = trace – this will log all the things.

To create Configuration xml file –>

Create one folder ‘resources’ at project level -> create xml file -> GO to official site -> Configuration -> copy xml code and paste code in xml file.

In order to make this xml file identified by script -> Right click project (where test script is present) -> properties -> Java build path -> Source -> Add folder ->



Name this xml as – log4j2.xml – others names will not work

<Configuration status=*"WARN"*>

<Appenders>

<Console name=*"Console"* target=*"SYSTEM\_OUT"*> // prints all messages in console

<PatternLayout pattern=*"%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"*/>

</Console>

</Appenders>

<Loggers>

<Root level=*"error"*> // applies at root level i.e. to all classes in a project.

<AppenderRef ref=*"Console"*/>

</Root>

</Loggers>

</Configuration>

To apply error message printing to only one class –

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

<Configuration status=*"WARN"*>

<Appenders>

<Console name=*"Console"* target=*"SYSTEM\_OUT"*>

<PatternLayout pattern=*"%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"*/>

</Console>

</Appenders>

<!-- apply to a specific class -->

<Loggers>

<Logger name=*"Log4jDemo.Log4jDemo"* level=*"trace"* additivity=*"false"*>// package.class

<AppenderRef ref=*"Console"*/>

</Logger>

<!-- applies at project level -->

<Root level=*"trace"*>

<AppenderRef ref=*"Console"*/>

</Root>

</Loggers>

</Configuration>

additivity=*"false"*> - > this removes redundancy, because root level and class level will run together , if we do not include this tag then log data will come twice in concole.

To save Logs in aFile ->

Create a folder at project and add .logs file to it.

In XML-

<Properties>

<Property name=*"basePath"*>./logs</Property> // path of current project to log folder

</Properties>

<Appenders>

<RollingFile name=*"File"* fileName=*"${basePath}/prints.log"* //prints is file name filePattern=*"${basePath}/prints-%d{yyyy-MM-dd}.log"*>

<PatternLayout pattern=*"%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"*/>

//this line will automatically create new prints.log with date format when size =500 exceeds in current file.

<SizeBasedTriggeringPolicy size=*"500"* />

</RollingFile>

<Loggers>

<Logger name=*"Log4jDemo.Log4jDemo"* level=*"trace"* additivity=*"false"*>

<AppenderRef ref=*"File"*/> // instead of console

</Logger>

Page Object Pattern framework – There are lots of framework in market today like BDD, Data Driven, Hybrid but the latest one is Page Object Pattern Framework.

It has various advantages over other frameworks –

* Easy to maintain
* Easy readability of scripts/
* Reduce or eliminate duplicity.
* Re-use of code
* Relaiable

There is no change in steps for Mobile / web testing in using this pattern.

In this pattern, we will create java class for each page in a web application in which we will define all the objects (objects are xpaths, csslocators, id etc.)

Example –

1. For Home Page of application

Class name – Homepage which contains all the objects of Home page, no other page.

1. For login page of application

LoginClass will be created which will fetch all the objects of Login page.

Page Object Model Implementation –

1. I have to create page object classes for each pages of the web Application where write all object of webElement and write locators of WebElement:

* but information of webdriver object is present in Base class.
* In order to pass the info of driver object into pageObject classes, We will create parameterized constructor in the pageObject class and we will create object of this pageObject class in test case with parameter as webdriver object.
* On creating object, this parameterized constructor will be called automatically.
* And we will mention all our locators in separate file as below –

**public** **class** HomePageObjects {

WebDriver driver;

//create constructor

**public** HomePageObjects(WebDriver driver) {

**this**.driver = driver;

}

By email = By.*name*("emailMobile");

By submit = By.*xpath*("//button[@type='submit']");

**public** WebElement Email() {

**return** driver.findElement(email);

}

**public** WebElement submit() {

**return** driver.findElement(submit);

}

}

AND

**public** **class** NykaaMainPage {

By close = By.*cssSelector*(".close-btn");

WebDriver driver;

**public** NykaaMainPage(WebDriver driver) {

**this**.driver = driver;

}

**public** WebElement Close() {

**return** driver.findElement(close);

}

* Main Class where these are used

@Test

**public** **void** Login(){

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

System.*setProperty*("webdriver.chrome.driver",

"C:\\Users\\navdeepkaurravala\\Documents\\Selenium\\Chrome Driver exe\\chromedriver.exe");

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

driver.get("https://www.nykaa.com/auth/verify?ptype=auth");

HomePageObjects hpo = **new** HomePageObjects(driver);

hpo.Email().sendKeys("nannana");

hpo.submit().click();

NykaaMainPage mp = **new** NykaaMainPage(driver);

mp.Close().click();

PAGE OBJECT FACTORY – This is same as Page object Model but more simplified by using @FindBy Annotation to find elements instead of driver.findelement. @FindBy annotation is present in selenium.support package and initialize the Page factory object.

PageFactory.InitElements(driver, this);

Syntax –

@FindBy(xpath =”…”) // works same as driver.findelement

WebElement Loginbutton // Stores locator found in above line in username webelement

**public** **class** NykaaLogin {

WebDriver driver;

//create constructor

**public** NykaaLogin(WebDriver driver) {

**this**.driver = driver;

PageFactory.*initElements*(driver, **this**);

}

@FindBy(name = "emailMobile")

Private WebElement email;

@FindBy(xpath ="//button[@type='submit']")

WebElement submit;

/\*By email = By.name("emailMobile");

By submit = By.xpath("//button[@type='submit']");\*/

/\*public WebElement Email() {

return driver.findElement(email);

}

public WebElement submit() {

return driver.findElement(submit);

}\*/

**public** WebElement Email() {

**return** email;

}

**public** WebElement Submit() {

**return** submit;

}

}

---------------------SELENIUM FRAMEWORK DEVELOPMENT----------------------------------

* Create Maven Project and add Selenium and TestNg dependencies.
* Create Page Object Model in main Folder which contains Driver initialization method which is common to all test cases and return driver object so that it can be used among test cases.

NOTE – main folder will contain all utilities and properties files and test folder will contain executable test cases.

* Create properties file with value of browser as Chrome or Firefox or IE.
* <https://www.selenium.dev/downloads/> - to download browser exes.
* To create thread safe reporting when test cases will run in parallel. There are chances that 1 or more test cases fails. In order to prevent our extent reports object from overriding. We use Threadlocal class in java.

Headless Browser Execution- it means that while running test, we are unable to see browser opening. While browser being opened execution is carried.

ChromeOptions options = **new** ChromeOptions();

options.addArguments("--headless");

SAUCE LABS –It provides platform to run our test cases in cloud. If we lack particular infrastructure then we can get that on Cloud and run our test cases.

Connectivity between Sauce Labs and our test cases is developed thru “Desired Capabilities” class.

In order to identify what capabilities we need to give for a particular window version – Go to Platform Configurator link and select the desired versions and automatically capabilities code will be genrated. Just paste that in code.

<https://wiki.saucelabs.com/display/DOCS/Platform+Configurator#/>

To pass access key – Get from Setting of your Scauce Labs account.

**public** **class** SauceLabsTest {

**public** **static** **final** String ***USERNAME*** = "navdeep.kaur";

**public** **static** **final** String ***ACCESS\_KEY*** = "cff4915c-b8f9-4bdd-a59a-f1a9e85f8e4f";

**public** **static** **final** String ***URL*** = "http://" + ***USERNAME*** + ":" + ***ACCESS\_KEY*** + "@ondemand.saucelabs.com:80/wd/hub";

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

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

MutableCapabilities sauceOptions = **new** MutableCapabilities();

ChromeOptions browserOptions = **new** ChromeOptions();

browserOptions.setExperimentalOption("w3c", **true**);

browserOptions.setCapability("platformName", "Windows 10");

browserOptions.setCapability("browserVersion", "latest");

browserOptions.setCapability("sauce:options", sauceOptions);

WebDriver driver=**new** RemoteWebDriver(**new** URL(***URL***), browserOptions);

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

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

}

}

AUTO IT –

* It is used to handle Window Authentication Popups. These do not have any HTML written at backend. These are tied to browser.

driver.get("https://admin:admin@the-internet.herokuapp.com/");

// username and password then url

driver.findElement(By.*xpath*("//a[@href='/basic\_auth']"));

* It is also used to upload file from windows to web.
* Download AutoIT and run the setup.
* We need to develop the script to handle the file upload thru Scite editor present in AutoIt setup.
* Shift focus to file upload windows.
* Set text/path into filename edit box
* Click open to upload file.
* Shift focus to file upload windows ->
* Find path of the edit box in windows. Use autoIt Finder tool – Au3info – to record window component objects and drag focus on edit box.
* Write the code in Scite editor to identify the properties.
* Script Code -

ControlFocus(“title”, “text”, controlID) {ControlId is combination of Class and Instance).

Text – can be optional – just leave it Blank.

ControlSetText(“title”, “text”, controlID, “new text”)

//ControlSetText is similar to send keys in selenium.

//new text is the path of file which we want to upload

ControlClick(“title”, “text”, controlID) // this will click onm open button in file search window.

* Save script as .au3 extension.
* Go to path and right click on script file -> Compile Script -> .exe file is generated.
* Go to eclipse and write code to invoke this .exe file.
* Runtime.getRuntime().exec(“<path of .exe file>”)
* Download File and Verify if this downloaded –
* Java File package is used to locate the file which is downloaded.
* File f = new File(“C://Navdeepkaurravala//Downloads//Notice.pdf”);

if(f.exists()) // checkes if downloaded file exists

{

}

But, above code is system specific because it downloads to every system. Here it is navdeepkaurravala. Other system will have different path.

Make it generic with below code –

* Download the file at project level in eclipse by using Chromeoptions.

ChromeOptions options = new ChromeOptions();

String downloadPath=System.getProperty("user.dir"); // project level path

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

HashMap<String, Object> chromePrefs = new HashMap<String, Object>();

chromePrefs.put("profile.default\_content\_settings.popups", 0);

chromePrefs.put("download.default\_directory", downloadPath); // mandatory to set

ChromeOptions options=new ChromeOptions();

options.setExperimentalOption("prefs", chromePrefs);

WebDriver driver=new ChromeDriver(options);

driver.findElement(By.linkText("Download Now")).click();

Thread.sleep(5000);

File f=new File(downloadPath+"/converted.zip");

if(f.exists())

{

Assert.assertTrue(f.exists());

if(f.delete())

System.out.println("file deleted");

}}

Database Connection with Selenium –

* Download MSI Installer from <https://dev.mysql.com/downloads/installer/>.
* Finish the MYSQL setup and by default it will be stored in program files -> MySql.
* Go inside MySQL folder in Program Files -> MySql Workbench -> MySqlWorkbench.exe -> Open
* Click on LocalInstance MySql -> Window will open asking for credentials -> enter credentials
* By default 3 databases are created -> test, sakila and world
* Expand the database -> expand tables -> one can see how many tables are present.

Steps ->

* Create database first by following query -

create database Qadb;

* Create tables inside the database-
* use Qadb; #to make sql understand that we want tables to be created in Qadb database.

#create table with name Employeeinfo with columns name, id, location and age.

#varchar is datatype same as String.

* create table Employeeinfo (name varchar(20), id int, location varchar(30), age int);
* #to check the columns in a table

describe Employeeinfo;

* To Insert values in a table

Insert into Employeeinfo values (‘sam’, 123, ‘bangalore’, 29); #String in single quotes

To develop connection between eclipse and database –

* Download jar named – MySqlConnectorJava.jar and configure this in project thru Build Path -> external jars. This jar develops connection between MySql and java.
* Drivermanager.getConnection(url, user, password)

This user is username and password which was entered while setting up the database.

URL here is connection url with syntax as below–

“jdbc.mysql://”+host+”:”+port+”/databasename”;

* 1. Make a connection with DB
  2. Load JDBC driver using Class.forname()
  3. execute the query using executeQuery() method
  4. fetching the records and using those records in our test method.

Class Database

{

public static void main(String[] args) throws SQLException, ClassNotFoundException

{

String host="localhost";

String port= "3306";

Username=root;

Password=root

//Connection con=DriverManager.getConnection("jdbc:mysql://" + host + ":" + port +, "root", "root");

Connection con=DriverManager.getConnection(dbURL,host,+port+,+username+,+password)

Class.forname(“com.mysql.jdbc.Driver”);

Statement s=con.createStatement();

ResultSet rs=s.executeQuery("select \* from credentials where emp=12 ");

EmpID Name dept

1 2 sunil ECE

while(rs.next())

{

int empID=rs.getInt(1); //12

String name=rs.getString(2); //sunil

String dept=rs.getString(3); // ECE

}

//After executing quesries we are using those records in our test case.

Public void empDetails(int empID,String name,String dep)

{

driver.findElement(By.xpath(".//\*[@id=’empID’]")).sendKeys(empID));

driver.findElement(By.xpath(".//\*[@id='password']")).sendKeys(name);

driver.findElement(By.xpath(".//\*[@id='password']")).sendKeys(dept)’

}

* Alternative:

while(rs.next()) // 0 index will hasve no values, always give .nex() i.e. 1st index.

{

WebDriver driver= new FirefoxDriver();

driver.get("https://login.salesforce.com");

driver.findElement(By.xpath(".//\*[@id=’empID’]")).sendKeys(rs.getInt(1));

driver.findElement(By.xpath(".//\*[@id='password']")).sendKeys(rs.getString(2));

driver.findElement(By.xpath(".//\*[@id='password']")).sendKeys(rs.getString(3));

}

}

}

**GIT – Used to maintain and Control versions of code.**

**GitHub is the central repository to host code.**

To talk to Github from our local machine, we require Git Commands.

To create Git Commands, We require git software in our local machines.

GIT Commands –

1. Tell git who you are –

git config --global user.name “navdeepkaur”

git config --global user.email “navdeep.ravala18@gmail.com”

1. Create one folder with all project stuff to upload to git under navdeepkaurravala (usernmae) folder.

Go to CMD -> path -> c:/users/ navdeepkaurravala/git data

1. To make git remote server i.e. Git HUB understand which file to be read from our local, we write - git init. This command creates local workspace with name “.git” in defined path.
2. To push our files to Git hub we need to perform commit. Git searches for committed files in our local repo and will push only to hub.
3. Git has 2 levels of commit. First one is Stagging. When we commit our code it is moved to “Stagging” and commit only commits the code present in Stagging and git only pushes the code which is commit.
4. Command – git add \* will add the files present in .git to stagging. We can add specifc ones also by providing – git add <path of the specific file>
5. To commit changes to git - git commit –m “first commit”. to give message with commit using “-m”
6. To specifiy the path where the changes to be pushed – git remote add origin <github repo path>
7. To push code – git push origin master. Pushes code to master branch by default
8. To pull code from git hub for first time – git clone <https link here from git hub>
9. To pull the latest changes from github in already cloned project – git pull origin master
10. To get the status of latest changes in project – git status
11. git remote remove origin - to remove origin attached to git

BRANCHING in GIT -

Suppose, I have to do some major changes in project which can impact the already running scripts. So, I will first take the Branch of my project and will start making code changes to that. So that, already running scripts will not be impacted in case of failures. In the end, when everything is sorted then I will merge my Branch to Master branch.

To create New Branch in github -> git checkout –b develop {develop is branch name which is want to create in Git}

In order to switch to newly created command -> git checkout develop

To check which branch we are currently in > git branch

To delete branch -> git checkout –d develop

To push code to particular branch -> git push origin develop {develop is branch name}

In order to reset the latest committed changes -> git fetch origin

* Git reset –hard origin/master

Merge develop (created branch) into Master Branch ->

* Switch to Master Branch first -> git checkout master
* To check whether the code is up to date or not -> git pull origin master
* Merge branch into master -> git merge develop { here our develop branch will be merge to our active branch i.e. master one )

MERGE CONFLICTS -> When multiple people working on a same file in a branch and tries to merge into master then Merge conflicts occur and git shows in a file indicating what is there is master as >>>>master and what is there in current branch >>>>>Head

We can switch between branches thru Eclipse -> Right click on project -> Teams -> Switch to -> Branch names…

CUCUMBER –

* BDD – Behavior Driven Development.

It is used to avoid confusion in understanding the requirement from testing and development ends.

Test cases are defined in a Business Level by Ubiquitous Language.

Ubiquitous language is a semi-formal language that is shared by all members of software team.

Syntax of Ubiquitous language template –

In Order to (Achieve Something) -> Business Outcome

As a (Role)

I want to (Do this)

Example-

In Order to Pay Credit Card Bill

As a Net Banking customer

I want to enter the Amount and process payment

Write Test cases as per requirement mentioned in Feature file Template –

Given – All the pre-requisites

When – Actions to be performed

Then – Outcomes i.e. Validations

To work with Cucumber – Download Natural 0.7.6 from Eclipse market help in Help Option.

Cucumber supports Quickstart Maven Template. Test cases in cucumber are written in Gherkins language.

Add jars for cucumber - > Cucumber –java and Cucumber junit

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-java</artifactId>

<version>1.2.6</version>

<type>pom</type>

</dependency>

<!-- https://mvnrepository.com/artifact/info.cukes/cucumber-junit -->

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-junit</artifactId>

<version>1.2.5</version>

<scope>test</scope>

</dependency>

3 components/key termonologies in Cucumber –

* Feature file – it is a test case which contains info of what action to be performed
* Step Definition – has info of automation scripts i.e. actions
* Junit Test Runner – where is feature file and Step definition we need to pass these path to test runner.

Feature file – It has to be created as normal text file with extension as .feature. This file will contains Gherkin language data as following –

**Feature:** Login Application

**Scenario:** Home Page Login functionality

*Given* user is currently on landing page of application.

(Given contains Pre-requisites to be performed before test case starts execution)

*When* user clicks on submit button after entering credentials.

*Then* Home page of application is populated

*And* Correct application view is displayed

Step Definition – Create class naming – stepDefincation. It contains mapping of feature file to sript in following way-

**public** **class** stepDefination {

@Given("^user is currently on landing page of application$")

**public** **void** user\_is\_currently\_on\_landing\_page\_of\_application() {

System.***out***.println("Login Page");

}

@When("^user clicks on submit button after entering credentials$")

**public** **void** user\_clicks\_on\_submit\_button\_after\_entering\_credentials() {

System.***out***.println("User has clicked on submit button");

}

@Then("^Home page of application is populated$")

**public** **void** Home\_page\_of\_application\_is\_populated() {

System.***out***.println("Home page is displayed");

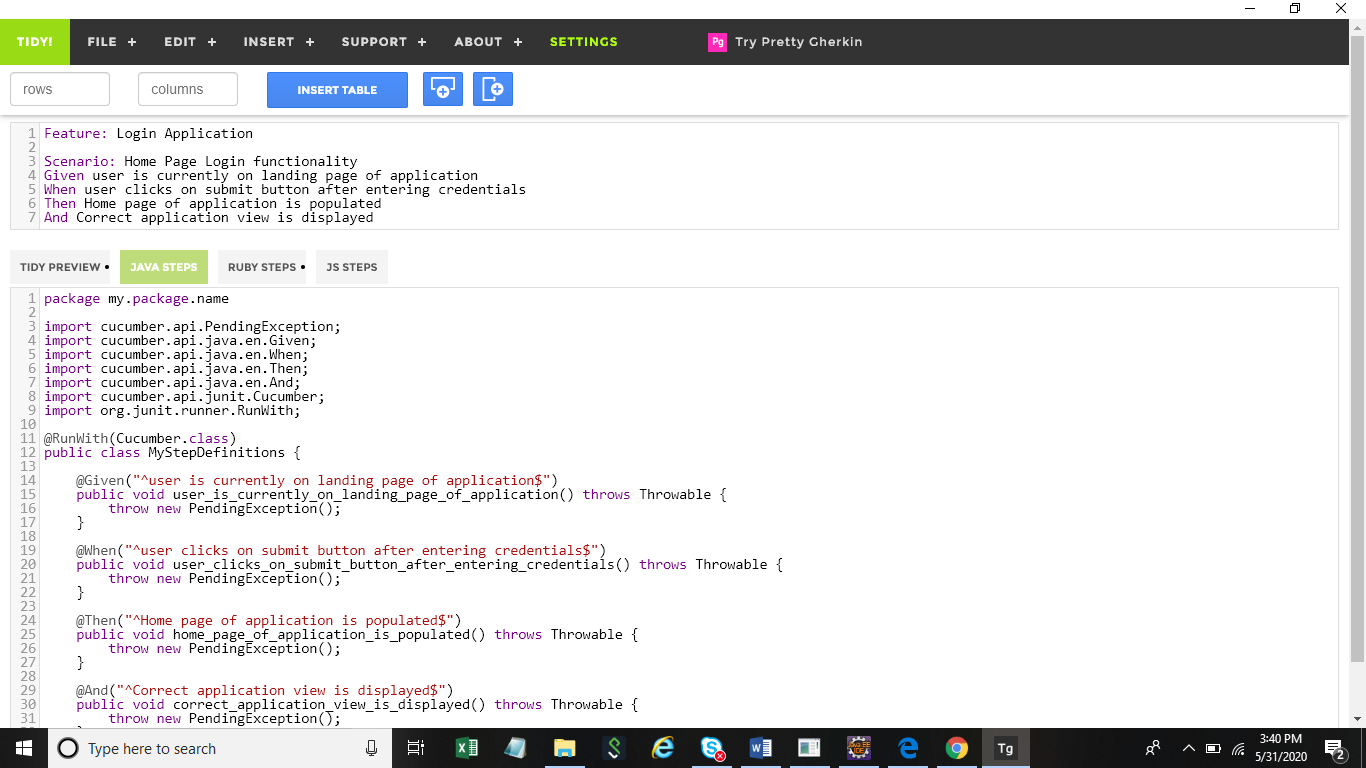
}

Tidy Gherkin -> it is app which automatically creates step defination class with all the mappings to feature file.

After installation, just enter – Chrome://apps which takes to all apps and you can observe Tidy Gherkin here.

Write feature file steps in Enter your Gherkin here..

And click on Java Steps-



TestRunner -> This will have all the code related to running the test.

@RunWith(Cucumber.**class**) // Running the test using junit

@CucumberOptions(

features = "src/test/java/Feature", // path of Package with feature file

glue = "stepDefination") // path of step defination package

**public** **class** TestRunner **extends** AbstractTestNGCucumberTests // running test in testing then RunWith should be commented

{

}

Regular Expression in Feature Files – When we have different data but same implementation i.e they are used when we need to run the same code with multiple parameters. Example, I want to log into my application with multiple users.

Code written inside “” (double quotes) will be treated as dynamic

Feature file code -

**Feature:** Login Application

**Scenario:** Home Page Login functionality

*Given* user is currently on landing page of application

*When* user clicks on submit button after entering credentials username "jo" and password "123"

*Then* Home page of application is populated

*And* Correct application view is displayed

**Scenario:** Home Page Login functionality

*Given* user is currently on landing page of application

*When* user clicks on submit button after entering credentials username "mo" and password "123"

*Then* Home page of application is populated

*And* Correct application view is displayed

StepDefination-

@When("^user clicks on submit button after entering credentials username \"([^\"]\*)\" and password \"([^\"]\*)\"$")

**public** **void** user\_clicks\_on\_submit\_button\_after\_entering\_credentials\_username\_something\_and\_password\_something(String strArg1, String strArg2) **throws** Throwable {

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

}

* d is used for digit
* + used to show one or more digit
* \d+ means at least one digit matches with the expected data
* . (dot) means any character
* .\* (dot star) means at least any character matches

Parameterization with Cucumber ->

In Feature file follow below syntax –

**Feature:** Login Into Application

**Scenario Outline:** OIM Portal Login functionality

*Given* Intialize the Browser

*And* Navigate to webpage with url "http://deviam02.dcas.dc.gov:14000/identity/"

*When* User enters Username as “<username>” and password as “<password>”

*Then* Verify that user is logged in successfully into portal.

**Examples:**

|username |password |

|abc |lala@12345 |

|bcd |baba@12345 |

Mention parameters under Examples and same names in when or Then in <> (square braces)

Integrate Cucumber with TestNG-

Cucumber Test Runner by default runs on JUNIT.

We need Cucucmber testng dependency to run cucumber with TestNg and inherit AbstractTestNGCucumberTests class in TestRunner class and comment the below line in code so that it is treated as TestNg not Junit –

//@RunWith(Cucumber.class)

<!-- https://mvnrepository.com/artifact/info.cukes/cucumber-testng -->

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-testng</artifactId>

<version>1.2.5</version>

</dependency>

In TestNG. XML add –

<classes>

<class name=*"CucumberOptions.TestRunner"*/>

</classes>

**What is the use of Background keyword in Cucumber?**

**Answer:** Background keyword is used to group multiple given statements into a single group. This is generally used when the same set of given statements are repeated in each scenario of the feature file.

JAVA Programs –

# Program: How to compare two sets and retain elements which are same on both sets?

import java.util.HashSet;

public class MyHashSetRetain {

    public static void main(String a[]){

        HashSet<String> hs = new HashSet<String>();

        //add elements to HashSet

        hs.add("first");

        hs.add("second");

        hs.add("third");

        hs.add("apple");

        hs.add("rat");

        System.out.println(hs);

        HashSet<String> subSet = new HashSet<String>();

        subSet.add("rat");

        subSet.add("second");

        subSet.add("first");

        hs.retainAll(subSet);

        System.out.println("HashSet content:");

        System.out.println(hs);

    }

}

# Program: How to copy all elements from HashSet to an array?

import java.util.HashSet;

public class MyHashSetToArray {

    public static void main(String a[]){

        HashSet<String> hs = new HashSet<String>();

        //add elements to HashSet

        hs.add("first");

        hs.add("second");

        hs.add("third");

        System.out.println("HashSet content: ");

        System.out.println(hs);

        String[] strArr = new String[hs.size()];

        hs.toArray(strArr);

        System.out.println("Copied array content:");

        for(String str:strArr){

            System.out.println(str);

        }

    }

}