Selenium class notes

1. If we are calling the static method, we do no need to create an object of the class. But if we are calling the static method from another class then we would need to call it by class name and not by object name.

Refer testing1 method in Test2 example.

1. Constructor is the same as class name. You can have n number of constructors, but the parameters must be different from previous constructor. Constructor has no return type (e.g. void) and it can be public only.
2. X path has 2 types
3. Relative / attribute

* Syntax: //tagname[@attributeName = ’attributeValue’]

E.g. //input[@id=’email’]

* // represents anything which covers up and down part while processing the tree.

E.g. .//input[@id=’email’]//a Will find all a on tree of id = email.

* Tag name can be ignored by \*.

E.g. //\*[@id=’email’]

* Multiple attributes can be used as well, and both needs to be satisfied.

E.g. //input[@id=’email’ and @name=’email’]

* Multiple attributes can be used as well and one of the conditions needs to be satisfied.

E.g. //input[@id=’email’ or @name=’email’]

* Multiple node founds and needs one value. E.g. (//input[@id=’email’])[2]. Where 2nd match of the matching condition.
* Need to find and element when the values are keep changing E.g. (//div[contains(@id,’js\_’)])[2]. Where id contains js\_ and has 2nd matching from the page. We can use starts-with instead of contains.
* Mix of both E.g. //\*[@id=’email’]/div[2]/input & .//table[@role='presentation']/tbody/tr[2]/td[2]/input

1. Absolute / position -> Describes the position of the value

* E.g. html/body/div[1]/div[3]/div/div/div/div/div[2]/form/table/tbody/tr[2]/td[1]/input

1. While using class name, note class name cannot contains space. But it can be used in xpath

E.G. fire.findElement(By.*className*("\_5633 \_5634 \_53ij")).getText();

1. When we use .findelement(), element must be on the page. While we use List<WebElement> then it will search for the element and result based on the what found. Source: ElementPresentTest.java
2. We can use the Fiddler for the performance testing.
3. There are 3 ways to find frame
4. *driver*.switchTo().frame(0); // This is the frame index or ith frame on the page
5. *driver*.switchTo().frame("Name / ID"); // Only works with Name or id attribute on iframe tag
6. *driver*.switchTo().frame(*driver*.findElement(By.*className*("demo-frame"))); // This applies with iframe class

*driver*.switchTo().parentFrame(); // To move to parent iframe

1. 2 types of waits
2. Implicit Wait

Implicit wait will be used when we are waiting for some Web Element to be available on the page. Following example will wait until the element is available.

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

1. Explicit Wait

Explicit wait will be used when an element is present on the page but value will change while we are processing the test. In below example the element on xpath is blank currently but will change to given string while we are testing.

E.g. WebDriverWait wait = **new** WebDriverWait(driver, 30);

wait.until(ExpectedConditions.*textToBePresentInElementLocated*(By.*xpath*(".//div[@class='dEOOab RxsGPe']"), "Wrong password. Try again or click Forgot password to reset it."));

Note: The time given on wait is max time, if element is found or matches to string, following code will be executed and wait will be terminated.

1. Use following class for specific tasks
2. Select class – For drop down selection
3. Action class – Drag and Drop test / Hover test
4. Alert Class – Handle alerts
5. JavascriptExecutor – scrolling down
6. Iterator class – Iterate the value. Used for multiple windows
7. CSS Selector Syntax

Tagname[attributeName=’attributeValue’]

e.g. input[id=’email’]

e.g. input[id\*=’nam’] // ID includes nam. Id=’name’

e.g. input[id^=’emai’] // ID starts with email. ID=’email’

e.g. input[id$=’mail’] // ID ends with mail. ID=’email’

e.g. input[id=’email’][name=’email’] // Input has name and ID as email

e.g. input[id=’email’]>div>input // Where ID is email and finding the value in div/input

e.g. #email // Where ID is email

e.g. .mbs.\_3ma.\_6n.\_6s.\_6v // . indicates class Note: has . for all class

1. Diff between List Vs. Set

List has index, set does not

List allows duplicate value, set will not allow

Use:- allWindows.get(1) for list

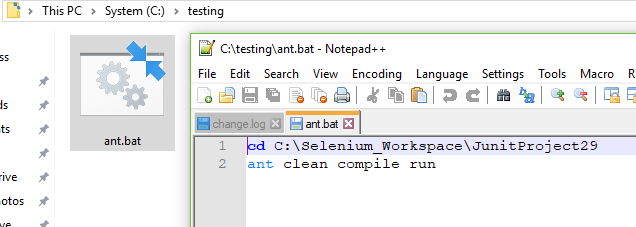
Use:- allWindows

1. To run all command, it can be mentioned one after another with the space in between.

Step 1: Copy project directory path

Step 2: ant clean compile run

Step 3: create a txt file with commands as above and save as to ant.bat



12. Suite runner class in junit is class file while in testing its xml

TestNG class notes

1. TestNG annotations

1. BeforeMethod

2. BeforeClass

3. BeforeTest

4. BeforeSuite

5. AfterMethod

6. AfterClass

7. AfterTest

8. AfterSuite

9. DataProvider

1. Following annotations will be used for test NG
2. @BeforeTest – This code will run before we start testing the environment. E.G. open a browser
3. @AfterTest – This code will run after our test is finished. E.G. Closing the browser
4. @BeforeMethod – This code will run before all method. E.G. run in grid
5. @AfterMethod – This code will run after all method. E.G. run on grid
6. @Test – This are the test code. It will run alphabetically. Parameters are as following:
7. (priority = <int>) – If priority is given, it will follow the seq. Lowest number is high priority
8. (dependsOnMethods = {“<MethodName>”,”<MethodName>”}) – assign dependencies
9. @FindBy :- This is used to find the values. Ref. Project:- BasicFramework

Note: No need to use main method while using TestNG.

1. Assert class is used for checking values and making conditions

E.g. String expectedTitle = “XXXXXX”;

String actualTitle = “YYYYYY”;

Assert.assertEquals(actualTitle, expectedTitle);

Note: it will check the strings.

E.g. It will return Boolean value.

Assert.assertTrue(2>1, “Error Message”);

1. Soft assert is used to skip the failure of the test case

E.g. SoftAssert sAssert = new SoftAssert();

E.g. SAssert.assertAll(); 🡪 It will collect the errors

6. Priority and depends on can be used together to run the test case affectively. Priority defines the order of execution while dependsOnMethod defines the dependency. i.e. main functionality should be performed only after login method.

7. Junit does not create a report by itself – we use ant while TestNG creates a report by itself – Its basic report

8. Reports. Custom, Xslt, Extent, Allure.

Junit class notes

1. Junit annotation
2. Rule – To run batch test
3. Test – Test case
4. After class – Will run at the end of all test cases & before and after methods. It must be static.
5. Before class - Will run at the beginning of execution. It must be static.
6. After – will run after all test cases
7. Before – Will run before all test cases
8. MethodOrder – to specify the order of the methods
9. Assertion
10. Assert equals

E.g. Assert.*assertEquals*(expecedErrmsg, actualErrmsg);

2. Assert true

E.g. Assert.*assertTrue*("Verifying 2>3", 2>3);

ANT

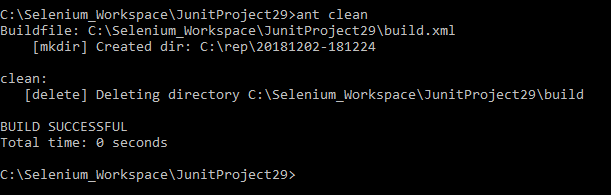
1. ANT is a build tool. Used to

* Build a project
* To generate report
* Run code from CMD

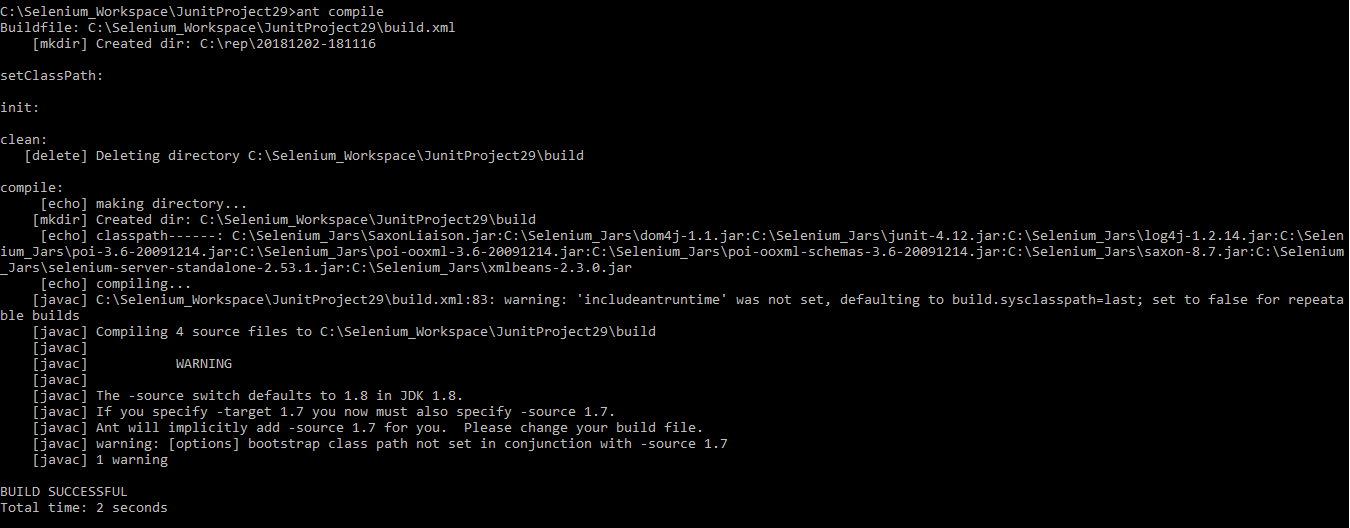
1. We would need a build.xml. We keep it in a project directory. Also, environment variable as ANT\_HOME and also add path.

1. Commands

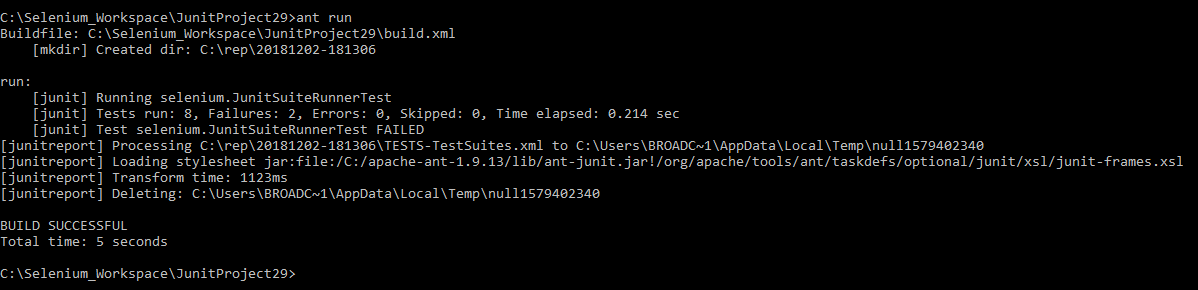
Ant clean



Ant compile



Ant run



XSLT Report

Steps to Generate Xslt report

1. Download the build.xml. Copy it to project

2. Move testng.xml to project level

3. Create new package xslt

4. Move testing-resuls.xsl to xslt package

5. Change the Jars path to folder where below jars are avaialble

1. saxon-8.7

2. SaxonLiaison

6. Check the target and make sure the xml file name on name="run"

// Run following commands on project directory

7. ant clean compile run

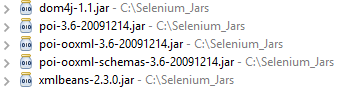
8. ant makexsltreports

Maven

1. Set up
   1. Group id = package name
2. Page object Model
   1. Page files – resides in src/main/java
   2. Test files – resides in src/test/java
3. Difference between ant and Maven
   1. POM.xml is the file just as build.xml
   2. Ant and maven both generate the report
   3. Both runs by command
   4. Maven has a structure ant does not
   5. Maven has a project category as maven and ant is a simple java project
   6. Maven downloads dependencies or Jars by itself while ant does not
4. Generate XSLT report (ref. http://www.testinginterviewquestion.com/2014/08/testng-xslt-reporting-using-maven-in.html)
   1. Copy plugin
   2. If there are 2 plugins with different versions. It replaces copied plugin version to newer version from another same plugin.
   3. Change the file as mentioned on the website. You will need to add many more plugin reporting etc.

XLS Reading

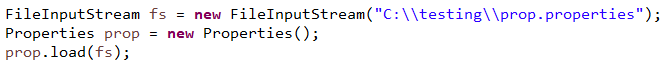
1. Download the following Jars



1. Column starts from 0 while row starts from 1
2. Read Xls\_Reader.java for the methods
3. We read Apache POI jars

Property File Reading

1. To read property file, we use default class Properties.



1. Property file is a key value pair; hence it cannot contain the duplicate keys.
2. If the file has 2 keys, it will overwrite the keys. It will consider the last repeating value and delete all other matching pair.
3. To read the value. E.g. prop.getProperty("browser");
4. To set the value. E.g. prop.setProperty("lastName", "Dave");
5. Once we set the value, it will store in Buffer. We have to commit the changes.

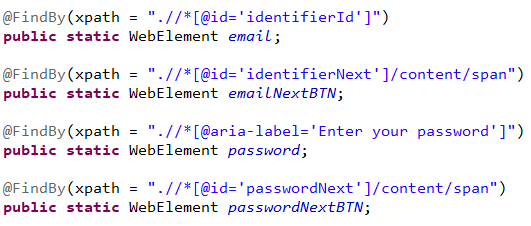
E.g. FileOutputStream nf = **new** FileOutputStream("C:\\testing\\prop.properties");

prop.store(nf, "File Edited");

Page Object Model Framework

1. When we use page object framework, we use following lines and annotation @FindBy to find the values on the page.

E.g. PageFactory.*initElements*(driver, **this**); // We must mention this line right after we initiate the browser



1. #Interview. We used page object model using Java, Selenium, test NG and Maven. In our project we have separate page file and test files for each of our functionality. In test File we have all test using TestNG annotation. In page file, we maintain the elements and calls the test cases. We also keep utilities class or helper class all in one place. We use property file for test data. We use Maven to download the dependencies and generate the report.
2. BDD :- Behavior Driven Framework / Cucumber

Agile Methodology

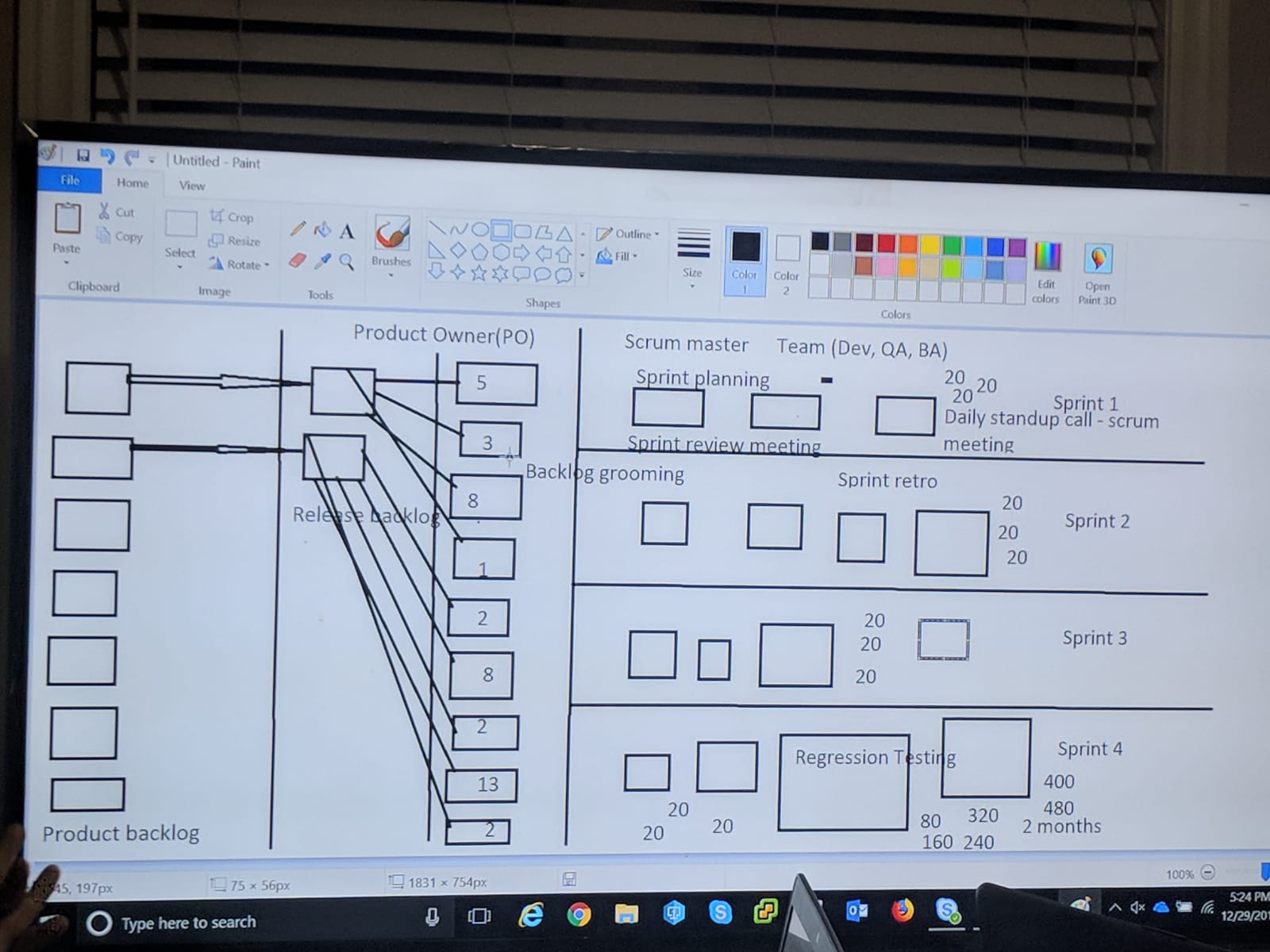
Src: <https://www.youtube.com/watch?v=UTkEDgyKDAY> + after video

* 1. backlog grooming - All user stories are given number either points based on complexity, hours, etc. They are rated and based on sprint, they are estimated for release. they are done every day. Rating are done based on Fibonacci series.

* 1. Parking lot - On scrum meeting, any issues or challenges are discussed later. This is called Parking lot.
  2. Sprint review meeting - all member (QA, BA, Dev, Product owner) attend this review meeting to make sure all functionality is developed and tested according to PO requirements. Sometime, once its developed, it was presented to PO directly to save QA time. Also discussed the issues faced for achieving the target.
  3. Sprint Retro – The purpose of this is to document what can be done, what could be better and action items are noted.
  4. Definition of ready – It is a checklist of required things. Estimated properly, proper description, acceptance criteria, and must have mock up or wireframe. It must be marked ready to start working on dev, QA.
  5. Definition of done – It is a checklist of manual test cases, automation testing, unit testing, manual testing, code review. All checks are considered the done user story. If most of the cases are marked not done, it will be analyzed and changed the flow. Note: Automation can be done on n+1 where n is the current sprint.
  6. Regression testing – It is a testing consist of all test cases in each sprint. All test cases are tested before previous sprint. It is usually done before launching. Automation is also important because after each release the test are keep increasing. Automation must be performed.
  7. Triaged meeting – We (QA) mention the severity of the bug. Save0 has the most priority as well as Save1. It will be reviewed by the product owner has to be change the priority if they want to release with bug. Some companies have the policy for no release with high severity bugs. It is done every week. It is a decision whether to solve it and set the priority.

Agile notes

* Smoke testing must be done every day or every other day. Sanity testing must be done too before release. Find out risky area and test them as well.
* Maintenance and user story will be added with the points to make sure the application is maintained properly.
* Interview question: 5 Developer, 1 BA and 2 QA for team size. We were split into multiple teams.
* Release Frequency is 2 weeks. Automation testing: 60%, 40% manuals testing. For upgrading test cases, we have a goal to hire manual tester for coverage to 75% automation.

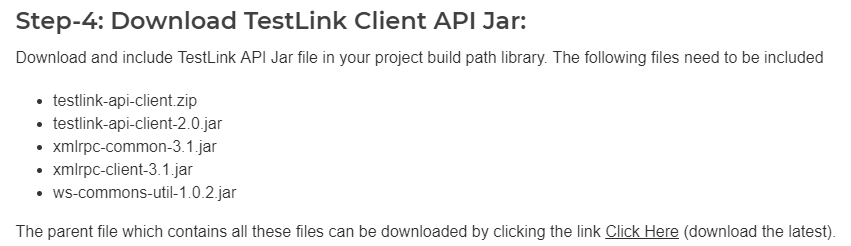


Test Link

Test link is used to create a manual and automated test cases. When we run the test manually or automated, we get the test result from Test link. We can integrate test link with our frame work to generate reports at one place.

Ref: <https://www.corestack.io/blog/step-by-step-procedure-for-integrating-selenium-with-testlink/>

We use below jar for the integration. It is integrated using the API key.



*Interview:* Explain the framework and say that We have used one to one mapping for test cases. We integrated our framework with test link. According to result, our test cases were updated.

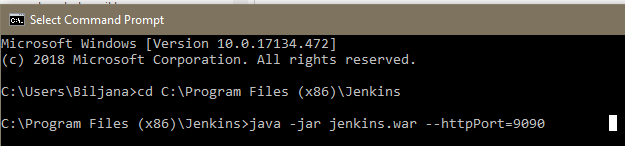
We can also update the result by TestLink listeners.

Jenkins - Its free download and practice much.

Jenkins is maintained by Dev-ops. They manage and troubleshoot. We create tests. We used to run tests using upstream and downstream projects. Pipeline is a new tool. It is written in groovy scripts. In job we select which code to run from Git, then select the frequency (upstream prj), and mention the command. We can set an email/emails to send test results.

Jenkins is a CICD (Continues Integration Continuous Development) tool.

To run Jenkins: Go to the path where it is installed.

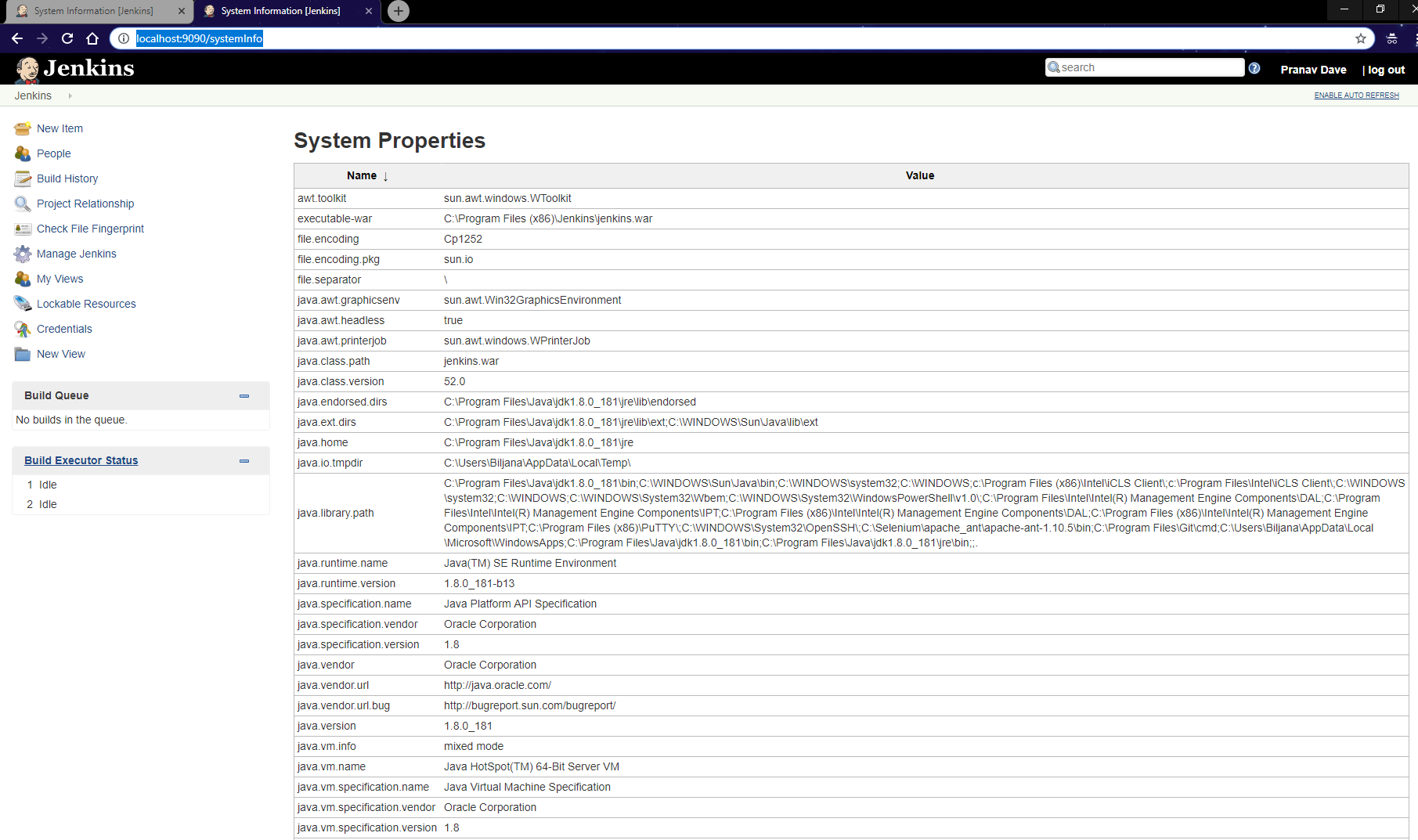


Jenkins runs on port 8080 using localhost by default but it can be customized as above.

Study: <https://youtu.be/89yWXXIOisk>

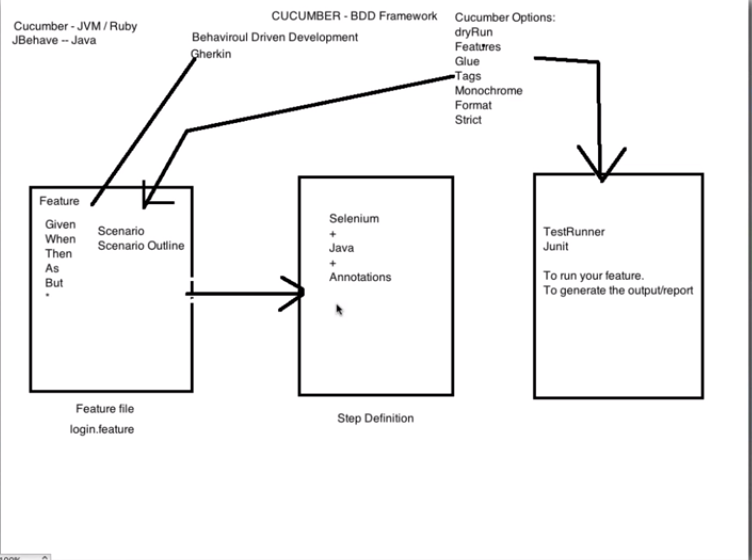
You can see all the system information on <http://localhost:9090/systemInfo> Note: Make sure the port number.

To install the CLI :- go to <http://localhost:9090/cli/> for instruction



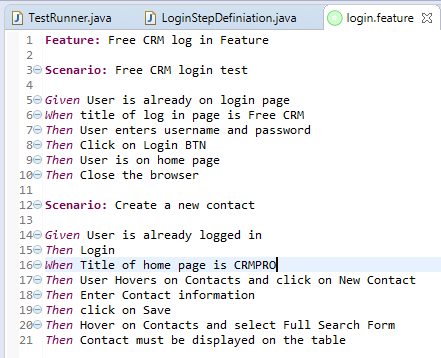
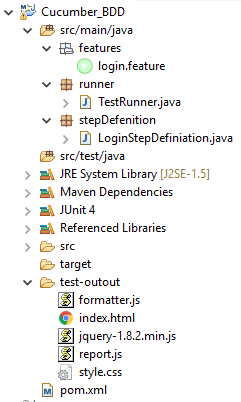
Cucumber

1. Key words in Cucumber BDD – Behavioral Data Driven framework.



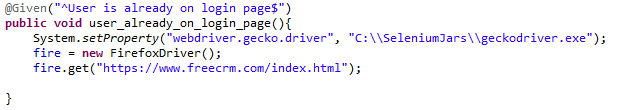
For Cucumber, need one .feature file where we mention all the steps. Then we need runner class. It is a simple Java file with CucumberOptions annotation. At the end we create one step definition file where we mention the test step.

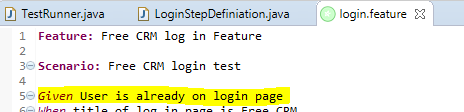
Project Directory Login.feature file



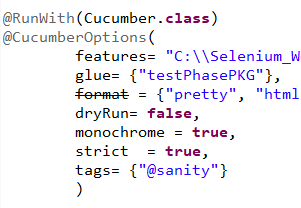
Runner class

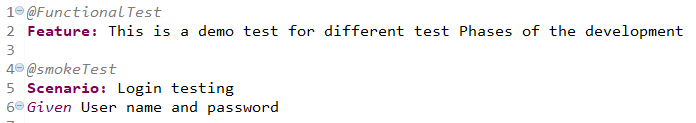
We have to make sure that the definition and the step on feature file is matching to each other. If there is a space, it can be replaced with underscore ( \_ ).





There are some options for runner class.

1. features:- It is a path of the feature file
2. glue:- It is a package name where step definition is written
3. format:- Output format. Also creates a basic report html/ Json/ xml
4. tags:- Tags are used to run different test cases. i.e. We can tag any test case to sanity/ smoke or any name. We also have to specify the tag we want to run in the runner class.

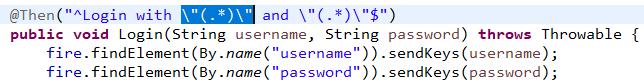
We have to make sure to provide the @FunctionalTest tag in the step definition file.

We can run conditional tests.

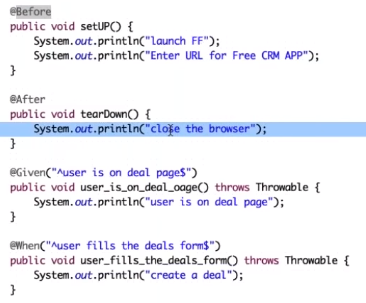
|  |  |  |
| --- | --- | --- |
| Condition | Operator | E.g. |
| OR | , | tags = {“@sanity, @smoke”} |
| AND | Separated – no operator | tags = {“@sanity”, “@smoke”} |
| Ignor | ~ | tags = {“~@sanity”, “@smoke”} |

1. monochrome:- It is used to make the console output readable. It is a Boolean value.
2. strict:- It is also a Boolean value. It will fail execution if there are any pending or undefined steps found.
3. dryRun:- It will check if the definition for all the steps are given or not. It is a Boolean value. It will not give an error, but we can check console to find out for missing definition. Test will only run if the value is set to false. Mapping between feature file and step definition file is proper or not.

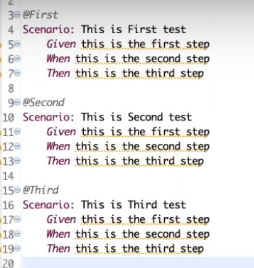
* To pass the data from the feature file (dynamic data), we can use regex \"(.\*)\" we have to pass the values and step definition must get those values in parameters.



* We can reuse the task by assigning the same definition and the same description in feature file.
* To comment on feature file use #
* There are 2 hooks in the Cucumber. 1. After 2. Before. They work just like the Junit / TestNG before and after annotation. Make sure to import them from cucumber api while scripting. Hooks are defined in the step definition file.



* If you want to run any condition for the specific scenario, use tagged hooks. It can be defined like @Before(“@first”) where First is the scenario. It will be mentioned on the feature file before the Scenario keyword.

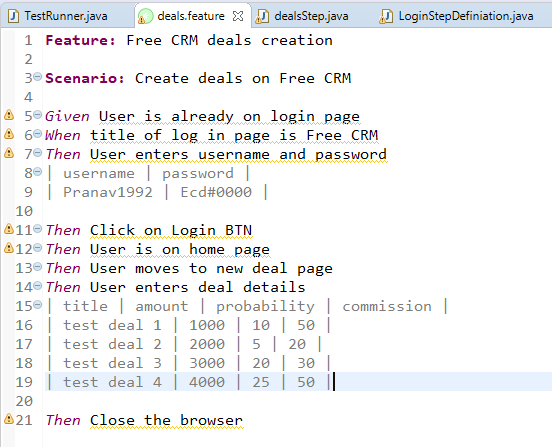


* We can also use the orders with the @Before and @After hooks. Just assign order=<int>

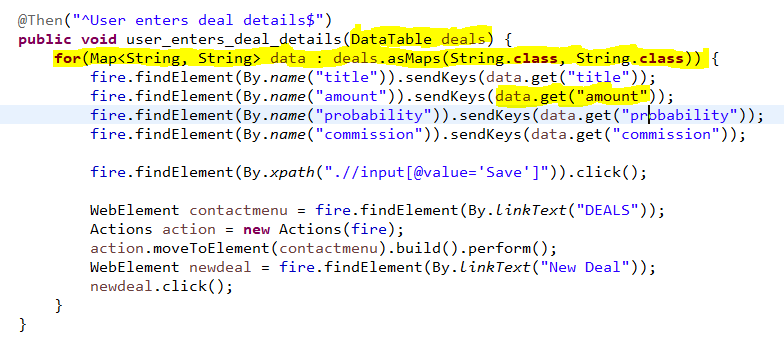
Data driven Example :- Cucumber

There will be three changes in the script as following

1. Create the feature file with the data as mentioned below



1. Create a DataTable class object in the method.
2. Crate and run For loop. Make sure to use .asMaps(KEY, VALUE) for reference.



\_\_\_\_\_\_\_\_ ONLINE COURSES TO LEARN \_\_\_\_\_\_\_\_

Visit - http://automationstepbystep.com/

------------ UI TESTING ------------

Selenium Beginners - https://bit.ly/2MGRS8K

Selenium Java Framework from Scratch - https://bit.ly/2N9xvR6

Selenium Python - https://bit.ly/2oyMp5x

Selenium Tips - https://bit.ly/2owxc50

Selenium Builder - https://bit.ly/2MKNtlq

Katalon Studio - https://bit.ly/2wARFdi

Robot Framework - https://bit.ly/2Px6Ue9

------------ API TESTING ------------

Web Services (API) - https://bit.ly/2MGafL7

SoapUI - https://bit.ly/2MGahmd

Postman - https://bit.ly/2wz8LrW

General - https://bit.ly/2PYdwmV

------------ MOBILE TESTING ------------

Mobile Playlist - https://bit.ly/2PxpeUv

------------ CI | CD | DEVOPS ------------

Jenkins Beginner - https://bit.ly/2MIn8EC

Jenkins Tips & Trick - https://bit.ly/2LRt6xC

Docker - https://bit.ly/2MInnzx

------------ VERSION CONTROL SYSTEM ------------

Git & GitHub - https://bit.ly/2Q1pagY

------------ PERFORMANCE TESTING ------------

JMeter Beginner - https://bit.ly/2oBbtIU

JMeter Intermediate - https://bit.ly/2oziNVB

JMeter Advanced - https://bit.ly/2Q22Y6a

JMeter Tips & Tricks - https://bit.ly/2NOfWD2

Performance Testing - https://bit.ly/2wEXbLS

------------ JAVA ------------

Java Beginners - https://bit.ly/2PVUcXs

Java Tips & Tricks - https://bit.ly/2CdcDnJ

------------ MAVEN ------------

Maven - <https://bit.ly/2NJdDRS>

------------Cucumber-----------

https://youtu.be/9uB7zfN6cAo

------------ OTHERS ------------

Redis- https://bit.ly/2N9jyCG

Misc - https://bit.ly/2Q2q5xQ

Tools & Tips - https://bit.ly/2oBfwoR

Interviews - https://bit.ly/2NIPPxk