//div[@class=’login\_logo’]

//div[@id=’inventory\_container’]

If we want to find child element, we need to use findElements by and use the variable of parent eleement to find the child element. 1:21

When there are multiple element. We use bracket around the xpath impession and put [1].

Here in this case we have done everything in one page. Later we organize to page object model. And we cant use an element twice and we need to find a way that we don’t use it twice. For that we create a base page for functionality. Ok in base page, we create a mothod called getDriver and use it, call it and fuck it.

The method has to return something and that is not a voif method. If any method returns something that is not a void method. It has to be a return type. If the variable in string the return type would be string. If it was a int then return type is int. We don’t want to make it an object that’s why we created the method static.

This is the turning point that I have missed in all along. Only static variable can go in to a static method.

public class Base {

WebDriver driver;

public WebDriver getDriver() {//usually this is void method.

//since this is using the web element, this should be a web driver instead of void and there is something to return,driver.

driver = new ChromeDriver();//previously this was web driver and since we are using this everywhere, we have to use this in class level.

driver.get("https://saucedemo.com/v1/");

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

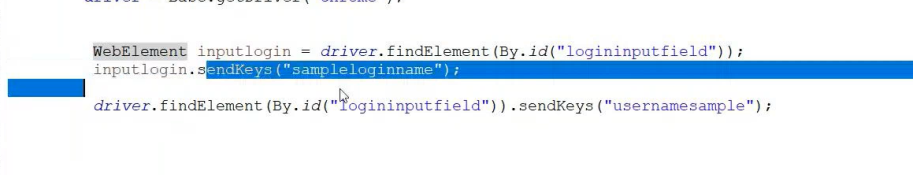
driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(3));

return driver;}}

this way I have to call the driver to the other class. In all pages, we need the main method. If I want t pass an parameter to the get driver method for example browser then I can pass argument to the main method s to be used where ever I use the get drive.

*driver*=Base.*getDriver*();

every tesing has property file. And the basic information is kept there. When there are page object model testing, all of the classed will have main class except for the base class. 20.51.same way and 2 different types



Normally we do: old way

WebElement title = *driver*.findElement(By.*xpath*("//div[@class='login\_logo']"));

WebElement username = *driver*.findElement(By.*xpath*("//input[@id='user-name']"));

WebElement password = *driver*.findElement(By.*xpath*("//input[@id='password']"));

WebElement loginbutton = *driver*.findElement(By.*xpath*("//input[@id='login-button']"));

username.sendKeys("standard\_user");password.sendKeys("secret\_sauce");loginbutton.click();

the new way:

*@FindBy* (xpath = "//div[@class='login\_logo']")//this line and next line are a single line. No . or , or ; or://

WebElement usernamefield;

*@FindBy* (xpath = "//input[@id='user-name']")

WebElement passwordfield;

*@FindBy* (xpath = "//input[@id='login-button']")

WebElement loginbuttonfield;

In page object model we do this. We do the send keys or functionality void method.



We need to memorize: pagefactory

PageFactory.*initElements*(driver, this);

Hold ctrl + click open declaration.

public class PageFactory {

/\*\*

\* Instantiate an instance of the given class, and set a lazy proxy for each of the WebElement and

\* List&lt;WebElement&gt; fields that have been declared, assuming that the field name is also the

\* HTML element's "id" or "name". This means that for the class:

\*

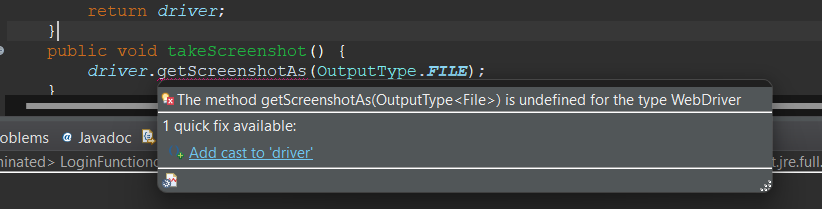
Any java/selenium web or Jre functions, we donot understand, hold ctrl and click.

Page factory initializes the elements using the driver of this class.

The page factory needs to be in the constructor. When the constructor has been constructed. Page factory initializes the elements with this driver of this class.

If we want to take a screenshoot we need a library. We need common io for this from maven repository.

Memorize screenshot



Memorize

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

// will come error 2 times and fix it class 12: 1:08:00.

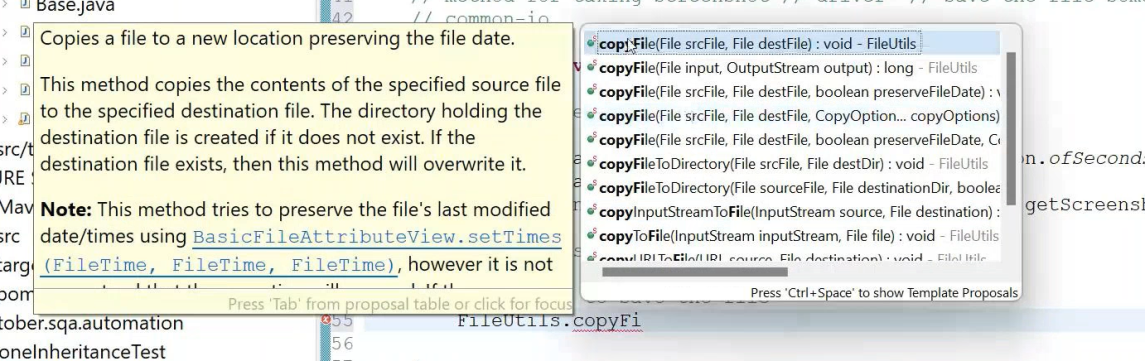
public void takeScreenshot() {

File fileScrennshotfile =((TakesScreenshot) *driver*).getScreenshotAs(OutputType.***FILE***);// will come error 2 times and fix it class 12: 1:08:00.

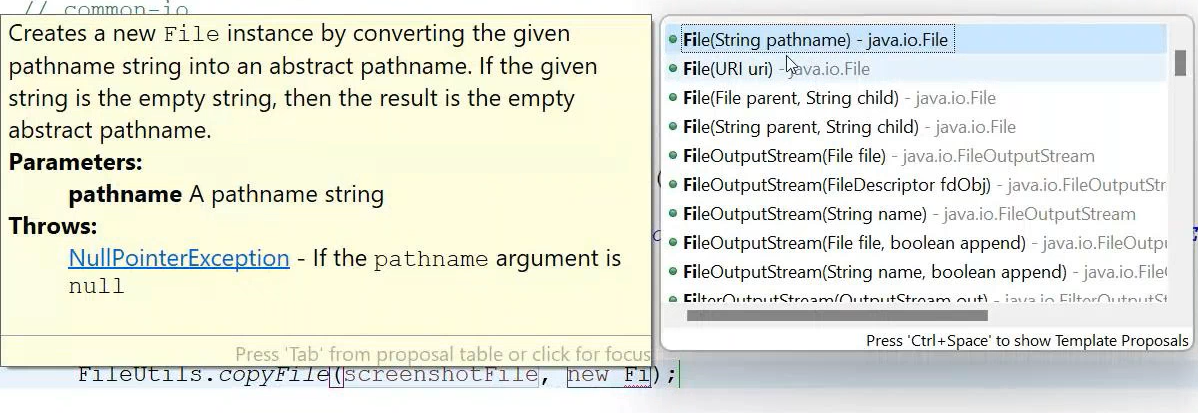
//this part is called type casting

*driver*.manage().window().fullscreen(); }

saving the file comes from file util class and select copy filees(file srcFile,file destFile):void FileUtils.



A new path name



We ususally put inside src.

public static void takeScreenshot() throws IOException {

*driver*.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(3));

*driver*.manage().window().fullscreen();

File fileScrennshotfile =((TakesScreenshot) *driver*).getScreenshotAs(OutputType.***FILE***);// will come error 2 times and fix it class 12: 1:08:00.

FileUtils.*copyFile*(fileScrennshotfile, new File("Screenshotname"+".png"));

//this part is called type casting

}

After we take the screen shot we refresh. And the file will appear.

FileUtils.*copyFile*(fileScrennshotfile, new File("Screenshotname"+".png"));

While refreshing it. We need to refresh the maven project and we shall see the screenshoot. 1:20:00s

We have to login first to test the product page so it is a pre requisite for testing the product page. So here we need to create a reference of loginpage object to ensure the login functionlaity. 1:36

Driver.quit is closing the driver. Driver.close is closing the window.

While inspecting we need to click individual button and get the xpath.

How to find xpath while anchor link //a[contains(@href,'cart.html')]

q. rabbi needs to find how to find xpath with containts class 12 1:44:00

class 12 1:51:

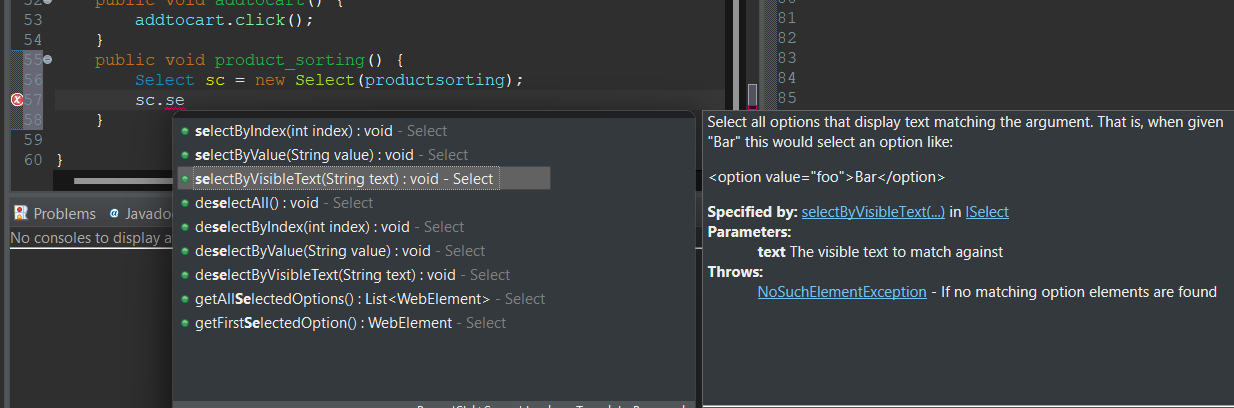
in sorting function, we need to use @findby Class

*@FindBy*(className = "product\_sort\_container")

WebElement productsorting;

Selenium has a built in function to select class. In select function, we can select by value or by visible text. This is interview question. There is a built in function in selenium like scanner function in java. We have to call the function and use reference variable. Among the value and visible text we use the visible text as the values are often changed by the developer.

Can we automate select class? Ans: yes we can use the selenium select function and automate the select class.



public void product\_sorting() throws InterruptedException {

Select sc = new Select(productsorting);

sc.selectByVisibleText("Name (A to Z)");Thread.*sleep*(2000);

sc.selectByVisibleText("Name (Z to A)");Thread.*sleep*(2000);

sc.selectByVisibleText("Price (low to high)");Thread.*sleep*(2000);

sc.selectByVisibleText("Price (high to low)");

}

Q. why did rabbi create lp new

ublic static LoginObjectclass *lp*;

ublic static void loginfunction () {

*lp*= new LoginObjectclass(*driver*);*lp*.enterusername("standard\_user");*lp*.enterpassword("secret\_sauce");*lp*.clicklogin();

}

Testng



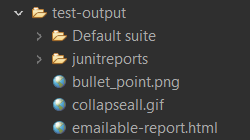
As like the java programs we give annotation of testNG@test where default run and debug arrises. Assert fail() when I intectionally fail one test. All tests will run only the assertfail will fail

*@Test*

void forgetpassword() {

System.***out***.println("I am forget password");Assert.*fail*();//i am intentionally failing the test

There is emailable report in the testoutput folder.



We can set priority with the priority settings in the @test annotation

*@Test*(priority = 4)

void successfullogout() {System.***out***.println("I am successful log out"); }

After priority we can give, and provide value. This is called key and value pair. [] is the sign of array and {} this is the value of array.

Priority Setting:

One test we can make depending on the other method or functions to pass od complete:

*@Test* (priority = 5,dependsOnMethods = {"successfullogout"})

void unsuccessfullogout() {System.***out***.println("I am unsuccesful logout");

I need to understand the report on testNG.

Negative priority:We can provide negative value in priority and the negative value will pass first then the positive value.

Enabled true/false; if the @test(enabled is true) then only the test will run and false will not run.

*@Test*(priority = 3,enabled = false)

void unsuccessfullogin() {

System.***out***.println("I am unsuccessful login");Assert.*fail*();

}

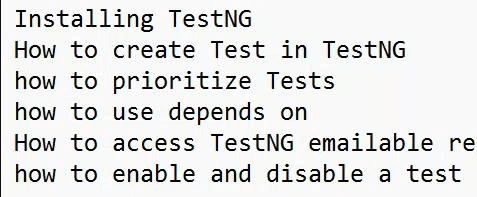
*@Test* (priority = -1,enabled = true)

void forgetpassword() {

System.***out***.println("I am forget password");Assert.*fail*();//i am intentionally failing the test

Where it is applicable, when I don’t want to comment out a test case then I can put enable false.

What we have learned so far



Any syso should be done in method

Normal array:

int [] numbers = {10,20,30,40,50};

Dynamic or matrix array:

int[] [] xy = {{1,2,3},{4,5,6},{7,8,9}};

int [] numbers = {10,20,30,40,50};

int[] [] xy = {{1,2,3},{4,5,6},{7,8,9}};

void print() {

System.***out***.println(xy[2][1]);

This is how we write multidimnetional array:

int [] numbers = {10,20,30,40,50};

int[] [] xy = {{1,2,3},{4,5,6},{7,8,9}};

*@Test*

void print() {

System.***out***.println(xy[2][1]);

}

Console 8

Data provider:

There are 2 mothods in data provider. One will provide the data and other will use the data. Data provider returns a multi dimentional array.

How the data provider is formed: it has a return type so should not be a void method.

public Object [][]dataprovider() {

Object[][] userdata = {{"userOne","usepasswordOne"},{"userTwo","userpasswordTwo"}};

return userdata;

}

This is the data receiver

*@Test*(dataProvider = "Dataprovider")//data is here now

void datauserfromprovider() {//receive data from object [][]}

*@Test*(dataProvider = "credentials")//data is here now

public void datauserfromprovider(String username,String password) {//receive data from object [][]

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

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

Receiver is above

Provider is below

*@DataProvider*(name = "credentials")

public Object [][]Dataprovider() {

Object[][] userdata = {{"userOne","usepasswordOne"},{"userTwo","userpasswordTwo"}};

return userdata;

}

How we can implement those in real life

*@DataProvider*(name = "Saucedata")

public Object [][]Saucedemo() {

Object[][] userdata = {{"standard\_user","secret\_sauce"},{"problem\_user","secret\_sauce"},{"locked\_out\_user","secret\_sauce"}};

return userdata;

}

Saucedemo

*@Test*(dataProvider = "Saucedata")

public void saucedemologin(String username,String password) {//receive data from object [][]

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

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

Console

standard\_user

secret\_sauce

problem\_user

secret\_sauce

locked\_out\_user

secret\_sauce

PASSED: TestNgTestRunnerClasses.TestNgLoginFunctiontest.saucedemologin("locked\_out\_user", "secret\_sauce")

PASSED: TestNgTestRunnerClasses.TestNgLoginFunctiontest.saucedemologin("standard\_user", "secret\_sauce")

PASSED: TestNgTestRunnerClasses.TestNgLoginFunctiontest.saucedemologin("problem\_user", "secret\_sauce")

Text xml: this is a powerfull feaure of testng. It can group data, manipulate our test, manipulate parameters, pass parameters, we can select which group to run.

How to create xml file. Right click on the class, go down to testng, select convert to testngxml file.

After this we need to go to the xml file and type the 2 string value of the parameters. We need 2 parameter files. One for browser and one for url.

If there is no class name we have to manually print class name. in name we need package name and class name which we can get from package properties.

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

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

<suite name="Suite">

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

<parameter name="browsername" value="Chrome"></parameter>

<parameter name="url" value="https://saucedemo.com/v1/"></parameter>

<classes>

<class name="TestNgTestRunnerClasses.ParameterPassingTNG"/>

</classes>

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

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

After creating the file the file will not run from the class level. It will only run from xml file and we need to right click and select run testng suite.

From console

[RemoteTestNG] detected TestNG version 7.9.0

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

SLF4J: Defaulting to no-operation (NOP) logger implementation

SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

Chrome

https://saucedemo.com/v1/

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

Suite

Total tests run: 1, Passes: 1, Failures: 0, Skips: 0

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

We need to pass the parameter from the xml file.

The mistake I made that I was runnning the test from the class instead of running it from xml file. Instead of passing the parameter from method names, we are passing the parameters from the xml file.

We pass the parameter to testng only from xml file?

Assertion: assertion is a short form of if else condition. Need to know all assert functions and cases we need to use the assertion. We have not created any xml file so we can run the program from class@test run. This is replacement of if else condition. Assertion is reflected in the report and this is very importane feature of test ng.

*@Test*

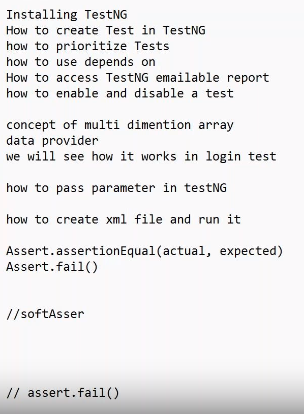
public void titletest() {

String actualtext ="Products";

String expectedtest = "Products";

Assert.*assertEquals*(actualtext, expectedtest);

We learned



Things we learn from annotation in test ng



-isDiplayed and conditions like if or else if is replaced by assert.true

Just sysout a condition of boolean value does not give us validation just will pass the test. Asset true will validate:

public class SoftAssertTest {

*@Test*

public void saucedemohomepagelogin(){

WebDriver driver = new ChromeDriver();

driver.get("https://www.saucedemo.com/v1/");driver.manage().window().maximize();

WebElement logo= driver.findElement(By.*xpath*("//div[@class='login\_logo']"));logo.isDisplayed();//this gives boolean value. we can replace with Assert.true

//System.out.println(logo.isDisplayed());//instead of this we do the following

Assert.*assertTrue*(logo.isDisplayed());//if this is true only then the test will pass

In test ng we validate any condition with assert.true function. In abov case if we put assert false, the test will fail. Assert false is for checking the opposite condition. This is used for netive condition chacking or negative testing.

After login we need to get the product page url with the following code

driver.getCurrentUrl();

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

How do we validate the expected and current url

String expectedUrl= "https://www.saucedemo.com/v1/inventory.html";

String currentUrl = "https://www.saucedemo.com/v1/inventory.html";

Assert.*assertEquals*(expectedUrl, currentUrl, "passed the test");

Soft assert since soft assert is a class we need to create on object of it. After doing soft assert we have to include softassert all at the end by convention.

SoftAssert sa = new SoftAssert();//SA is a class and we need to create an object of it

sa.assertTrue(logo.isDisplayed());

sa.assertAll();//by convention we have to do assert all.

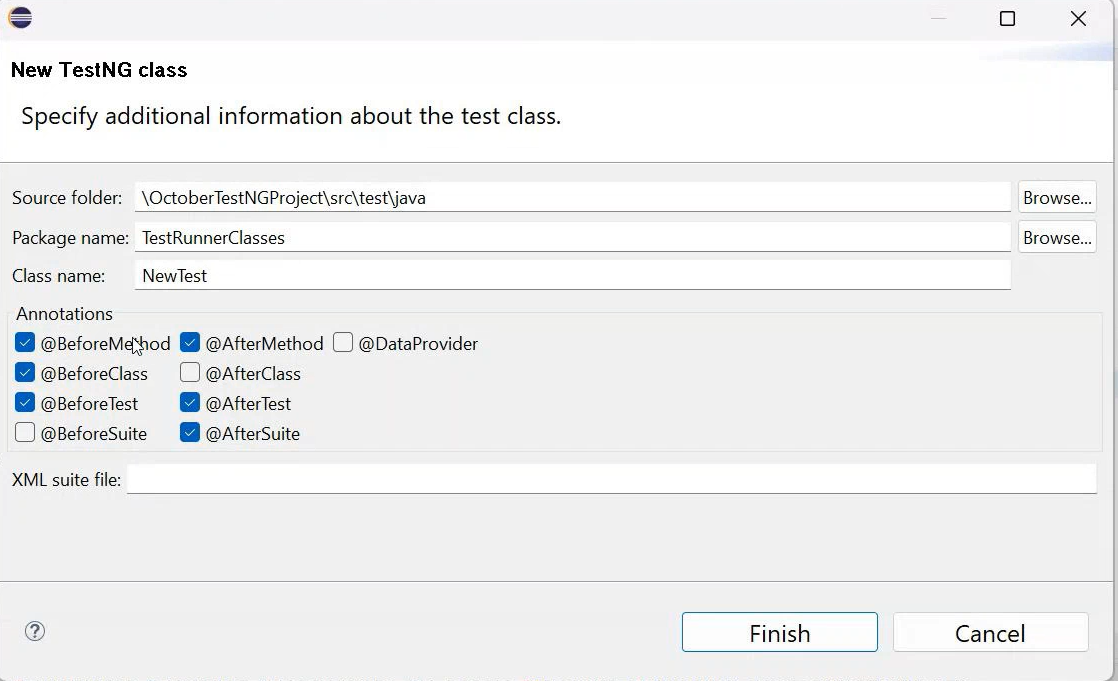
All hard asserts and soft asserts needs to be pacticed and mastered.

Annotation:

Normal java = package -> class-> method

Test ng= suite(group of class) -> test->-class->functions/steps->

Basic annotations of testng



Right click on the package select testng and select testng class and this will appear.

If we provide before or after annotaion we don’t provide @test annotation.

public class AnnotationTest {

*@Test* public void logintest() {

System.***out***.println(" I am init the driver "); }

*@BeforeMethod*

public void driverReady() {

System.***out***.println(" I am driver ready "); }

*@AfterMethod*

public void tearDown() {

System.***out***.println("I am log out the method and log out ");}}

If we put another method, the before and after will run after each method. Total run 4 times where it ran 2 times in the test.

Testing serial: very important.

Before test

Beforeclass

Test test

After class

After test

public class beforeClassAfterclass {

*@Test*

public void logintestt() {

System.***out***.println(" I am successfull login ");

}

*@Test*

public void logouttestt() {

System.***out***.println(" I am unsuccessful login");

}

*@BeforeClass*

public void driverReadyy() {

System.***out***.println(" I am driver ready ");

}

*@AfterClass*

public void tearDownn() {

System.***out***.println(" I am log out the method and log out ");

}

*@BeforeTest*

public void abc() {

System.***out***.println(" I am before test ");

}

*@AfterTest*

public void xyz() {

System.***out***.println(" I am after test ");

}

}

Console:

I am before test

I am driver ready

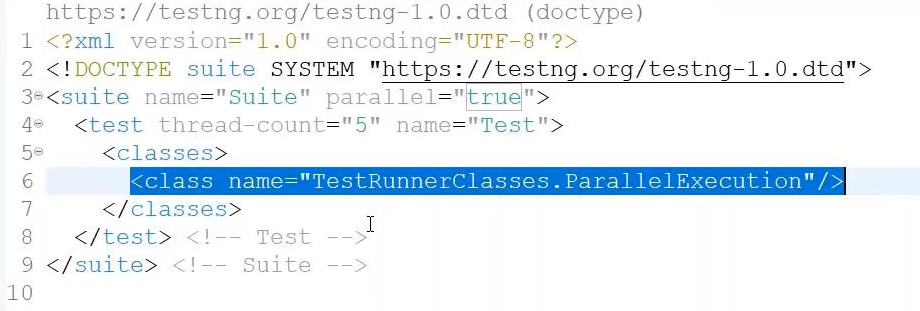
I am successfull login

I am unsuccessful login

I am log out the method and log out

I am after test

Parallal execution needs to be done from xml file.



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

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

<suite name="Suite"parallel="true">

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

<classes>

<class name="TestNgTestRunnerClasses.parallaltesting"/>

</classes>

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

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

Class console

This is line one from test one

This is line two from test one

This is line one from test two

This is line two from test two

If we call the reference veriable like

Productpageobject pp = new Productpageobject();

Then we declare the variable at global level like this Productpageobject pp;

Every time we use one method and we check it.

Look in the google: how to upload a file in selenium.

Q. when we can use css selector for xpath writing

Suite in test ng: how to run suite. We need a smoke test for example and we need to run only 2 not all. Then we can group through suite.

By adding groups function to the test I want to add regression or smoke test. We run groups not from the class, from xml.

Learn: null pointer exception for testNG suite and groups. If there is any pre requisite for testing, this has to come under group or we can put @beforeclass(alwaysrun=true).

Xml code for groups under suite: for regression and smoke testing.

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

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

<suite name="Suite">

<groups>

<run>

<include name="regression"></include>

</run>

</groups>

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

<classes>

<class name="TestNGtestingPageObject.CopyProductPageTestNgRunned"/>

</classes>

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

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

If we bring any data from any page we have to create a global variable of the page at first and call new driver at the before class test run.

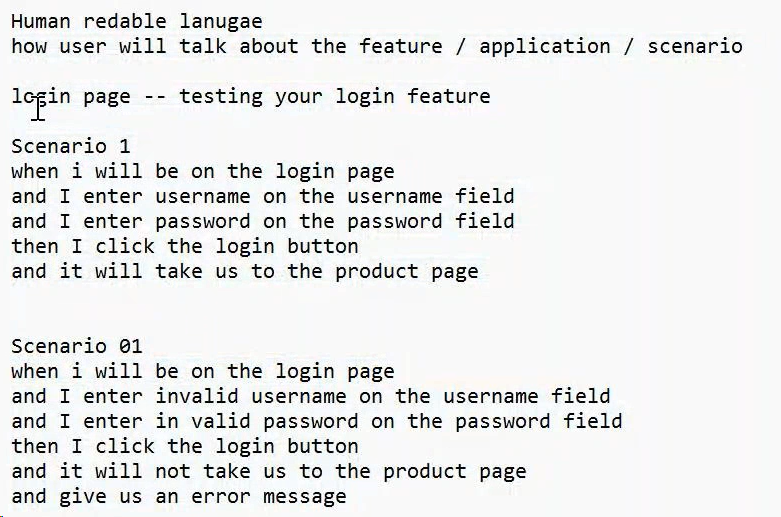
Cucumber:

Normally we do the test driven development and now we do behavior driven language BDD.

Gherkin keywords: the word and keywords we are going to use is gherkin keywords or gherkin language in cucumber.

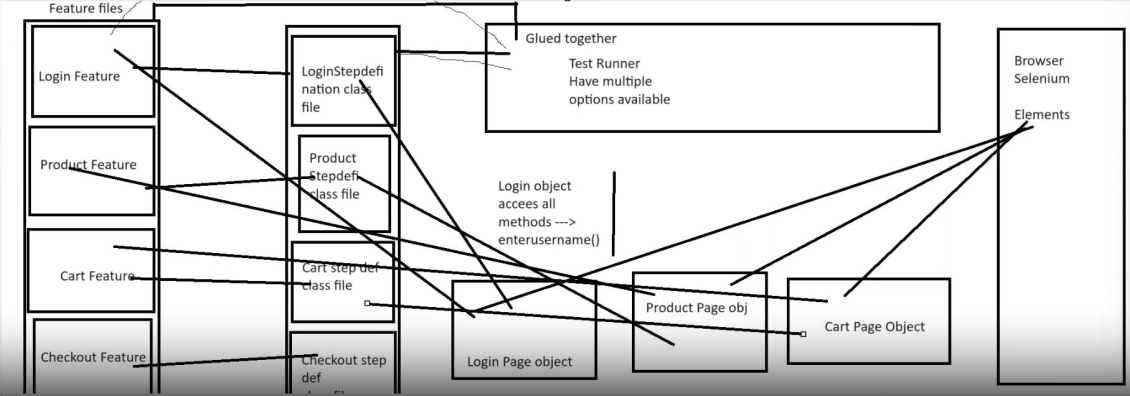
In TDD: TDD has the test cases developed first and then the development happen. In BDD the development happen first then the test cases and test scenario happens. In BDD framework we use the user story to test our features. BDD is humanly readable language. We describe our features in plain human readable language.

For example we have 2 scenario with right user name, password and wring username and password. We have to test the both scenario. We had methods in testng and we have methods in cucumber. Any step we dicsuss in cucumber becomes a method. Like any step on the test steps becomes a method . if we test in cucumber we have to keep the test data if the dev donot provide us any test data. For every situation we crate methods and the methods are called step definition.

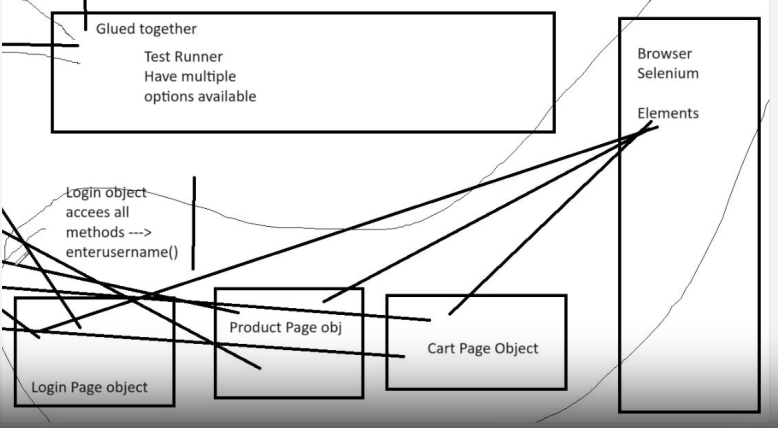


In BDD we have 3 components:

1. Feature file: single or multiple scenario. We have to decide which feature to test. We need to decide what method to use. Like annotation. The steps we work on testing the features becomes the step definition. They have multiple scenario given, when, then, and .
2. Step definition: for which feature which method is responsible.
3. Test runner: Junit. This test runner basically run the test. Test runner basically glues the step definition and features. It glues the step definition with the feature file and run the test.

Every function in the testing is a feature. Here in cucumber, test annotations (testng) are replaced by step definition. How the cucumber framework will look like:

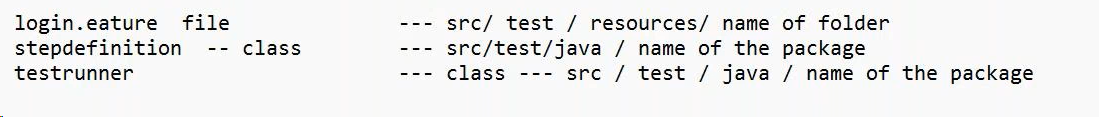
Selenium Page object model will be the same for testng and cucumber. Only difference is user methods. Page object model only deals with the selenium methods.

this part is the base part and doesnot matter if it is testng or cucumber. This part only dealing with the browser. The other part of the framework is managing the test.

Feature is an extension like txt or png. We use it as .feature. not Feature or features. Feature file will be located in to a folder.

Step definition will be a class file and location will be src/test/java.

Test runner will be a class file and location will be src/test/java.



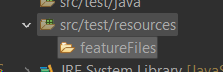
We can keep groupID and artifact ID same in maven project.

We create groupID in reverse the way we write the website address. For example, google.com will be com.google.

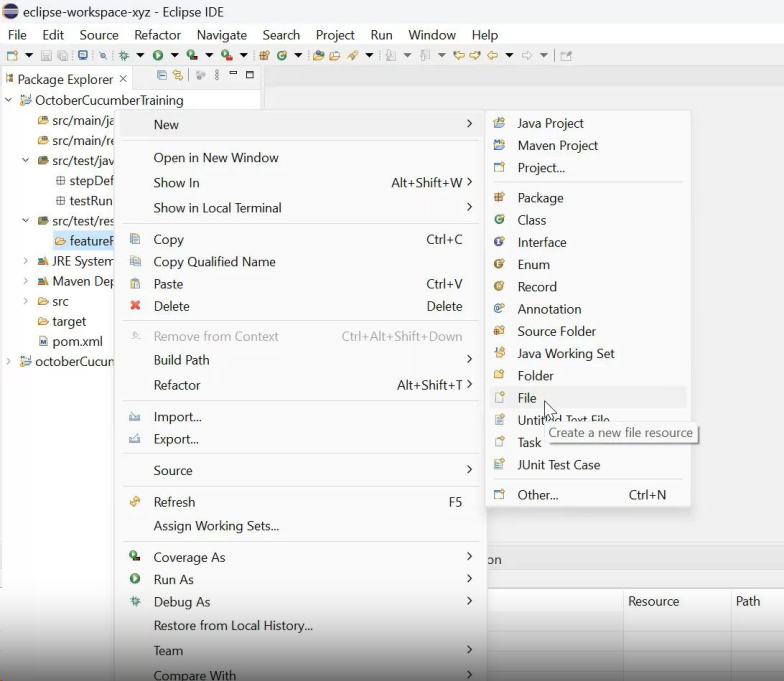
Cucumber has to be installed in 4 ways.

1. IDE level: from help menu eclipse market.
2. Project level: JVM java> maven repository
3. Project level: Junit4> maven repository
4. Project level: jvm core> maven repository
5. <dependency>
6. <groupId>io.cucumber</groupId>
7. <artifactId>cucumber-java</artifactId>
8. <version>7.15.0</version>
9. </dependency>
10. <dependency>
11. <groupId>io.cucumber</groupId>
12. <artifactId>cucumber-junit</artifactId>
13. <version>7.15.0</version>
14. <scope>test</scope>
15. </dependency>
16. <dependency>
17. <groupId>io.cucumber</groupId>
18. <artifactId>cucumber-core</artifactId>
19. <version>7.15.0</version>
20. </dependency>

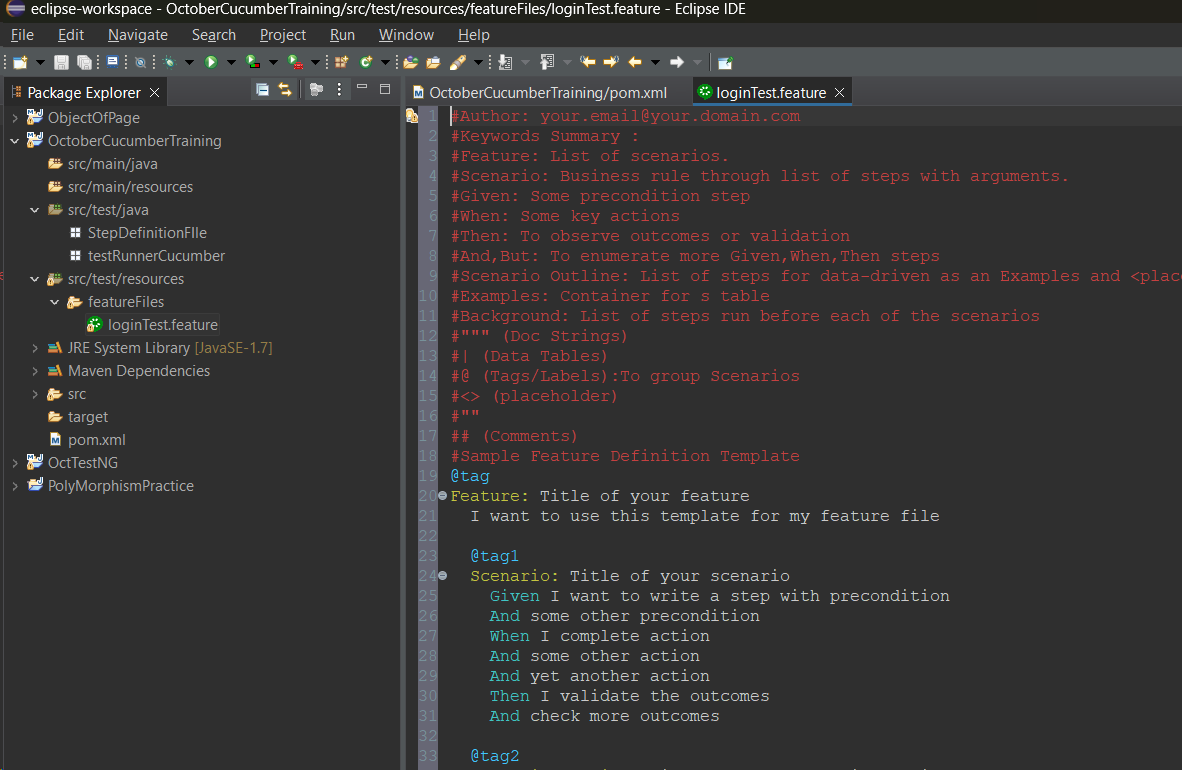
**Before making a project: we need to create feature file under src/test/resources. Right click and create new folder. Here the folder name is featureFiles.**

****

**First we need to create 2 packages to in src/test/java: testrunner and stepdefinition. Then we hav eto create a file in the file we created in the resources.**

****

**After the file I create, the following window appears:**

****

**File name: login.feature(here .feature in the extension). In the feature file we comment out by using #. The compiler will not read this file the # upfront.**

#Author: your.email@your.domain.com//for my email address when I work.

#Keywords Summary :

#Feature: List of scenarios.

#Scenario: Business rule through list of steps with arguments.

#Given: Some precondition step //when you click the button

#When: Some key actions //then should log in.

#Then: To observe outcomes or validation

#And,But: To enumerate more Given,When,Then steps

#Scenario Outline: List of steps for data-driven as an Examples and <placeholder>

#Examples: Container for s table

#Background: List of steps run before each of the scenarios

#""" (Doc Strings)

#| (Data Tables)

#@ (Tags/Labels):To group Scenarios

#<> (placeholder)

#""

## (Comments)

#Sample Feature Definition Template

And is used as an extension of when and then.

Scenerio out line is an out line with multiple data sets.

Right click and select pretty format: the class page of the file will be formatted with pretty format.

When I word in cucumber, its on me that how I use the wordings. I should try to improve the quality of the work than other testers like young people and give high quality output in cucumber.

Feature: Sauce demo login functionality test

Scenario: Login with valid credentials

Given User is on the SauceDemo login page //and the login page with all the fields are visible.

When I provide the username

And I provide the password

And I click the login button

Then I should be able to login and see the product page

And I will find the 6 products in the product page

Every part in scenario has to be defined in step. That’s why it is called step definition. The steps are there as methods.

Feature: Sauce demo login functionality test

Scenario: Login with valid credentials

Given User is on the SauceDemo login page //and the login page with all the fields are visible.//driver will find the login page

When I provide the username//driver will provide the user name

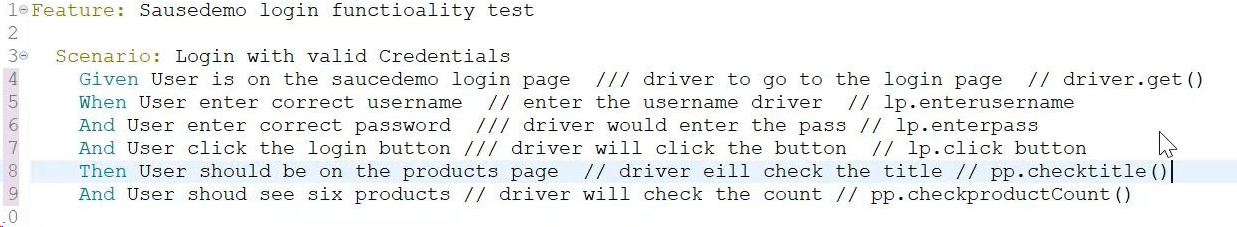
And I provide the password //driver will provide the password

And I click the login button //driver will click the login button

Then I should be able to login and see the product page // driver will see the product page title

And I will find the 6 products in the product page //driver will count the 6 products.

The objects we have previously used are using here



Feature: Sauce demo login functionality test

Scenario: Login with valid credentials

Given User is on the SauceDemo login page //and the login page with all the fields are visible.//driver will find the login page

When I provide the username//driver will provide the user name

And I provide the password //driver will provide the password

And I click the login button //driver will click the login button

Then I should be able to login and see the product page // driver will see the product page title

And I will find the 6 products in the product page //driver will count the 6 products.

We have to run this from right click as cucumber features this will create the cucumber file. Cucumber will create all the methods for me

@Given("User is on the SauceDemo login page")

public void user\_is\_on\_the\_sauce\_demo\_login\_page() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

@When("I provide the username")

public void i\_provide\_the\_username() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

@When("I provide the password")

public void i\_provide\_the\_password() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

@When("I click the login button")

public void i\_click\_the\_login\_button() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

@Then("I should be able to login and see the product page")

public void i\_should\_be\_able\_to\_login\_and\_see\_the\_product\_page() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

@Then("I will find the {int} products in the product page")

public void i\_will\_find\_the\_products\_in\_the\_product\_page(Integer int1) {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

Then we go to the step definition package, create a class for step definition and paste the file we have created in the console.

package StepDefinitionFIle;

public class LoginStepDefinition {

*@*Given("User is on the SauceDemo login page")

public void user\_is\_on\_the\_sauce\_demo\_login\_page() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

*@*When("I provide the username")

public void i\_provide\_the\_username() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

*@*When("I provide the password")

public void i\_provide\_the\_password() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

*@*When("I click the login button")

public void i\_click\_the\_login\_button() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

*@*Then("I should be able to login and see the product page")

public void i\_should\_be\_able\_to\_login\_and\_see\_the\_product\_page() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

*@*Then("I will find the {int} products in the product page")

public void i\_will\_find\_the\_products\_in\_the\_product\_page(Integer int1) {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

}

@given is a decorator.

We have to delete the last 2 lines:

*@*When("I provide the username")

public void i\_provide\_the\_username() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}

At first it will give error to import;

import io.cucumber.java.en.Given;

public class LoginStepDefinition {

*@Given*("User is on the SauceDemo login page")

public void user\_is\_on\_the\_sauce\_demo\_login\_page() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

when we put \* instead of give, all error will go.

import io.cucumber.java.en.\*;

after all this, at the project level, we need to right click on the project an select convert to cucumber project. Then in class file all the methods will be cucumber.

Feature: Sauce demo login functionality test

Scenario: Login with valid credentials

Given User is on the SauceDemo login page

When I provide the username

And I provide the password

And I click the login button

Then I should be able to login and see the product page

And I will find the six products in the product page

There will be green cucumber sign in the class lines. When we press ctrl + click it will take us to the method on the file. Now they are glued with methods.

We can use driver at the class level so that all the methods can use.

Testng assertion was created inspired by junit.

There are cucumbertestng dependency. If we want to use testng assertions in cucumber, we need to use that dependency.

In previous, we write everything in the class. But in test runner, we will run everything before the class. We are using an operator @runwith.

*@RunWith*(Cucumber.class)

We have to select the option with io.cucumber.junit and then we select class.

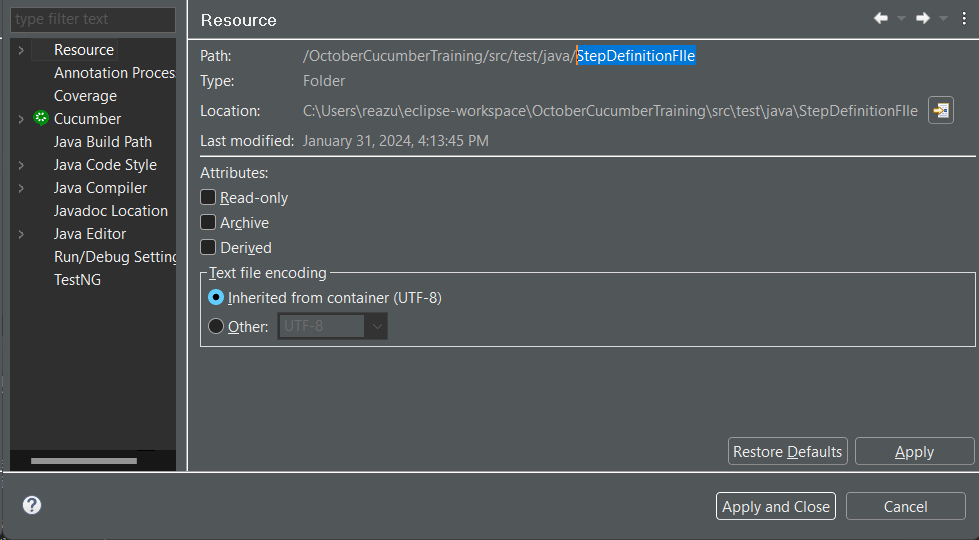


*@CucumberOptions*()//we pass arguments here

We pass arguments in the bracket.

Any function inside class is called method. Functions outside class are called functions.

We have to glue the feature file with the package name properties file name: here step definition.



Have to give , between feature and glue. Right click and run as junit test 1.

package testRunnerCucumber;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

*@RunWith*(Cucumber.class)

*@CucumberOptions*(

features = {"src/test/resources/featureFiles/loginTest.feature"},

glue = {"StepDefinitionFIle"}

)

public class testRunnerClass {}

we have to give pluggin if we want to see report:

cucumber functionality comes when we use , operator.

Mukhosto korte hobe to use pretty function.

package testRunnerCucumber;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

*@RunWith*(Cucumber.class)

*@CucumberOptions*(

features = {"src/test/resources/featureFiles/loginTest.feature"},

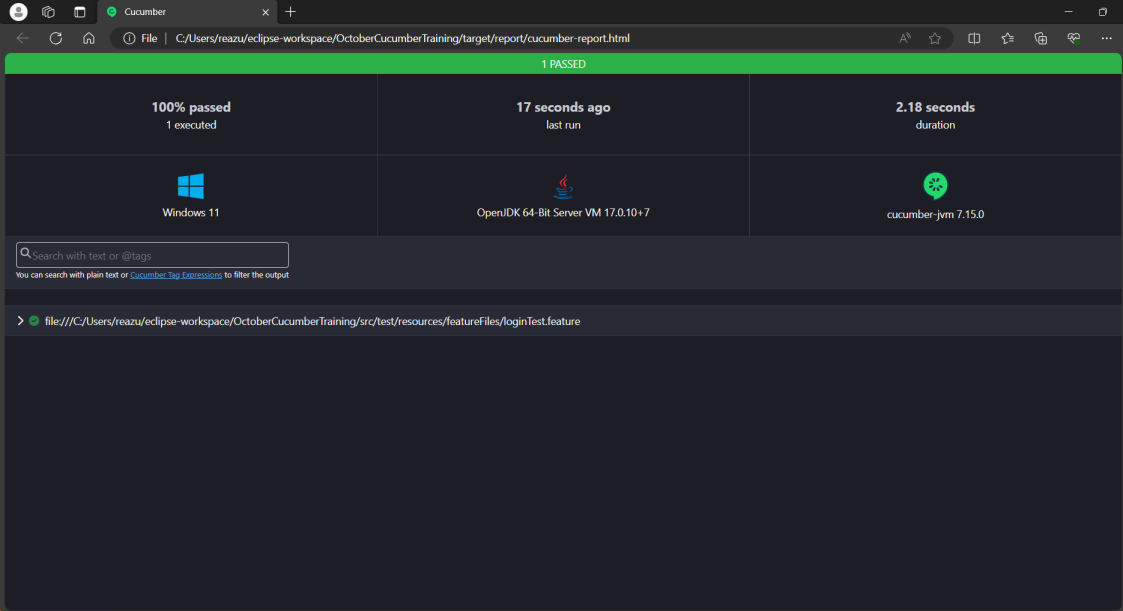
glue = {"StepDefinitionFIle"},

plugin = {"pretty", "html:target/report/cucumber-report.html"}

)

public class testRunnerClass {}

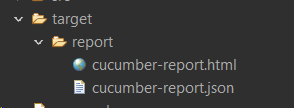
we refresh the folder and find the report in the target folder. Target folders keep the emailable reports.



If we want to do a json report, we can do like the report comes as follows: just add an extra line and replace json for html

plugin = {"pretty", "html:target/report/cucumber-report.html","json:target/report/cucumber-report.json"}

json



JSON: jave script object notation. In interview they can ask for both Json and html report. And we shall deliver both the report.

If I want to add another function to the step definition we have run the whole program:

And it will sort products

And click the small monitor beside the funnel button and we can find the method to be pasted in the class.

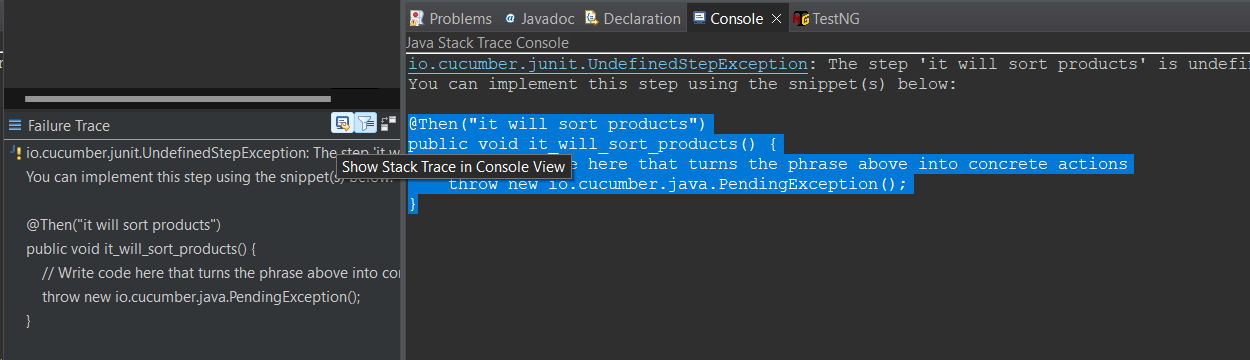
@Then("it will sort products")

public void it\_will\_sort\_products() {

// Write code here that turns the phrase above into concrete actions

throw new io.cucumber.java.PendingException();

}



By this process, we can run the code to find the step definition that is missing. we can do dry run = true to find where the step is missing. copy paste the step and run it

Every time before testing we can dry run to find what step definition is missing from the story: we can

*@RunWith*(Cucumber.class)

*@CucumberOptions*(

features = {"src/test/resources/featureFiles/loginTest.feature"},

glue = {"StepDefinitionFIle"},

plugin = {"pretty", "html:target/report/cucumber-report.html","json:target/report/cucumber-report.json"},

dryRun = true

we can find the code. Include in the steps and can delete the dry run.

We can also do the dry run false after we do the step definition inclusion.

The change the color. We use monochrome true function

dryRun = true,or false when code is corrected.

monochrome = true

read documents in <https://cucumber.io/> to understand cucumber. We can also use cucumber testng for annotation as cucumber does not have annotations.

The following are the cucumber testng annotations

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.9.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>7.15.0</version>

<scope>test</scope>

</dependency>

Source: <https://cucumber.io/docs/cucumber/checking-assertions/?sbsearch=TestNG>

We can use testng assertions for cucumber.

Like before and after method in testng, we can use hooks in cucumber

In Cucumber, hooks are blocks of code that can run at various points in the execution cycle. They are typically used for setup and teardown of the environment before and after each scenario. Hooks can be defined anywhere in the project or step definition layers using the methods @Before and @After.

Hooks can be useful when there are multiple scenarios and every scenario needs to be executed with the same setup and teardown steps. Hooks can also be used to perform background tasks that are not part of business functionality.

**There are three types of hooks in Cucumber:**

* Scenario Hooks: Run before and after each scenario
* Step Hooks: Can be written in a configuration class separately or in a step definition class
* Conditional Hooks: Can be defined anywhere in the project or step definition layers using the methods @Before and @After

@before @after only select cucumber io. Any method we create will not work if that is not public.

Hooks are help full for the testing framework so that the pre requisites are met. Like login page or checking functionality of credit card information.

In testng there is priority, cucumber has order to execute the functionality before or after.

*@Before*(order = 1)

public void runBeforeScenerio() {System.***out***.println(" this is before hook"); }

*@Before*(order = 2)

public void runBeforeScenerio2() {System.***out***.println(" this is before hook 2"); }

*@After*

public void runAfterScenerio() {System.***out***.println(" this is after hook"); }

Hard code: when we make a constant value on the script that is called hard code.

If we add or change anything in the feature file, we need to findout the revised code through dryrun the test runner. This dryrun will give us modified step definition.

Imp: if we donot do dry run false the browser will not open

When I provide the username "abc"

*@When*("I provide the username {string}")

public void i\_provide\_the\_username(String username) {

driver.findElement(By.*xpath*("//input[@id='user-name']")).sendKeys(username);

we can type the things and don’t need to change the

we cant write capital S in string in the quotation

*@When*("I provide the username {string}")

Scenerio outline: deals with data table.right click on the class / type sce then scenerios will come. The column firsr row is header and rest in information.

Everytime we paste a value in the cucumber page it should be pretty formatted.

Always change the text before pasting the previous scenario to scenario outline

We provide this in the string because this is string value

Scenario Outline:

Given User is in the SauceDemo login page

When I provide the wrong username <"username">

And I provide the right password <"password">

And I click the login button

Then User should get the <"status"> code

Examples:

| username | password | status |

| standard\_user | secret\_sauce | pass |

| locked\_out\_user | secret\_sauce | fail |

| problem\_user | secret\_sauce | fail |

| performance\_glitch\_user | secret\_sauce | fail |

Revised data: “” should be on the outside

Scenario Outline:

Given User is in the SauceDemo login page

When I provide the wrong username "<username>"

And I provide the right password "<password>"

And I click the login button

Then User should get the "<status>" code

In feature file we cannot comment out // or /\* we need to use #

#// Scenario Outline:

#Given User is in the SauceDemo login page

#When I provide the wrong username "<username>"

#And I provide the right password "<password>"

#And I click the login button

#Then User should get the "<status>" code

#

#Examples:

#| username | password | status |

#| standard\_user | secret\_sauce | pass |

#| locked\_out\_user | secret\_sauce | fail |

#| problem\_user | secret\_sauce | fail |

#| xyz | secret\_sauce | fail |\*/

Select all ctrl /

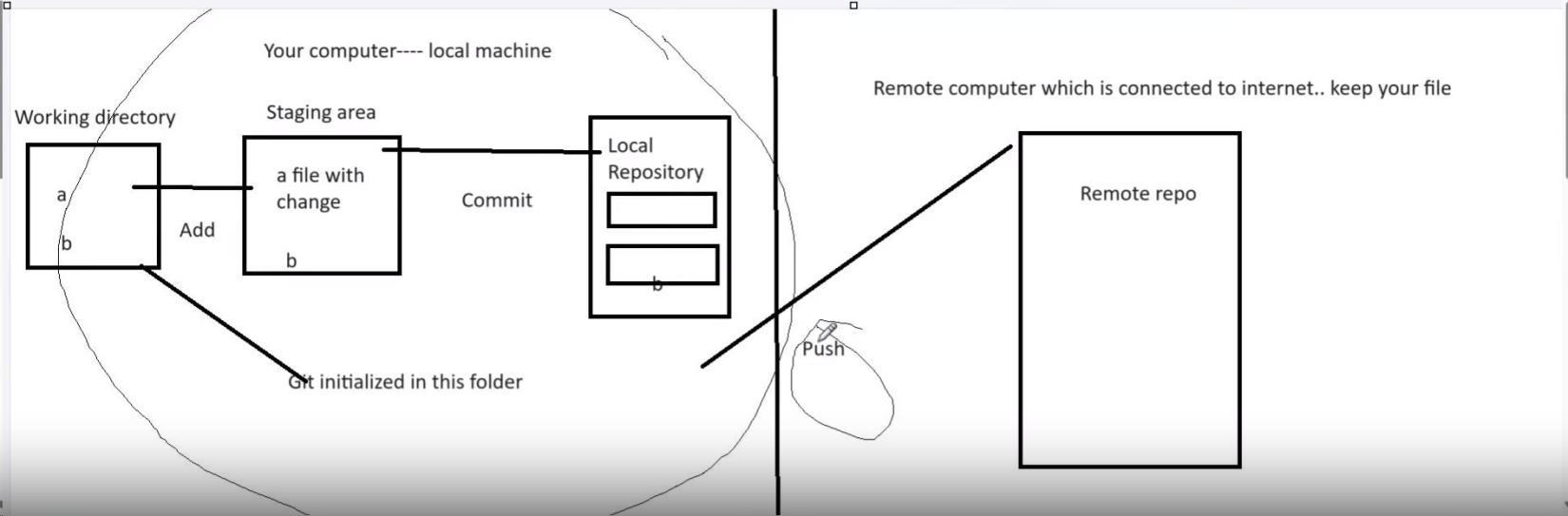
When we need to create run from feature file or when from junit.

In cucumber Assertion should be done in step definition. Not in the page object model.

IN case of new page, no testrunner is required but all the pages are required like login page list. Jus t copy and paste codes. Never manually type 2nd time the codes.

Q should we paste the testng cucumber dependencies in POM.xml for assertions?

Git and github



Working directory: where we keep our files and codes in the local computer.

Add: when we make changes or include anything in the staging area that is add.

Staging area: Temporary space where we keep the files.

Commit: when we finalize the data for a local repository.

Local repository: where we keep the finalized data.

Remote repository: Online or cloud space where we keep the files.

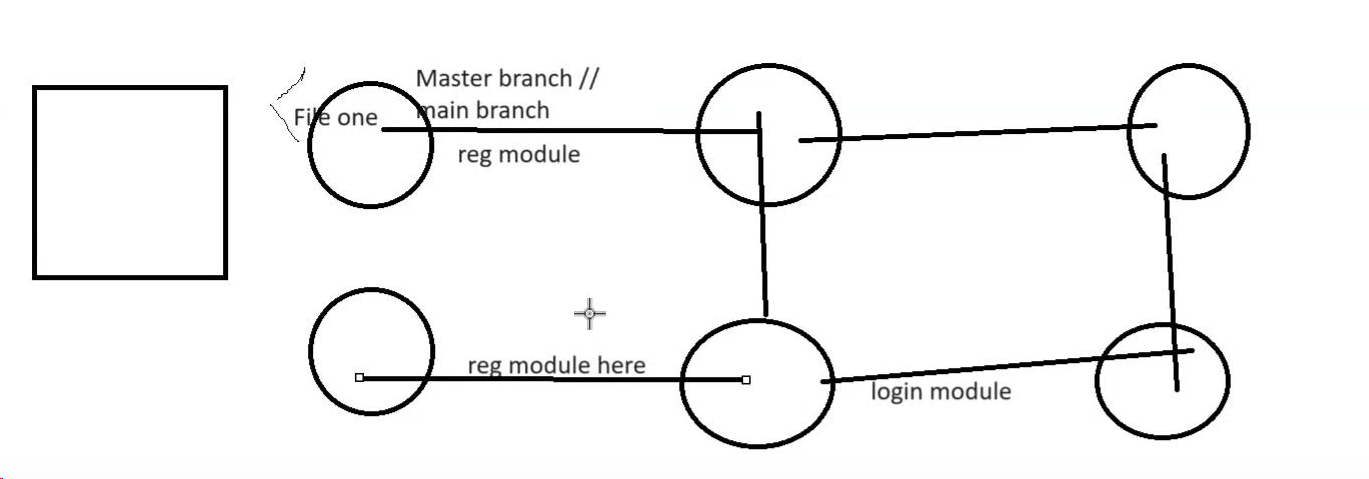
Push request: When we send the data to online repository

Pull request: when we ask someone to retrieve the data we have sent. We send pull request to the lead and he merge the code.

Clone: Stays in side remote git hub.

Branch: it is the clone of what we have in the working directory.

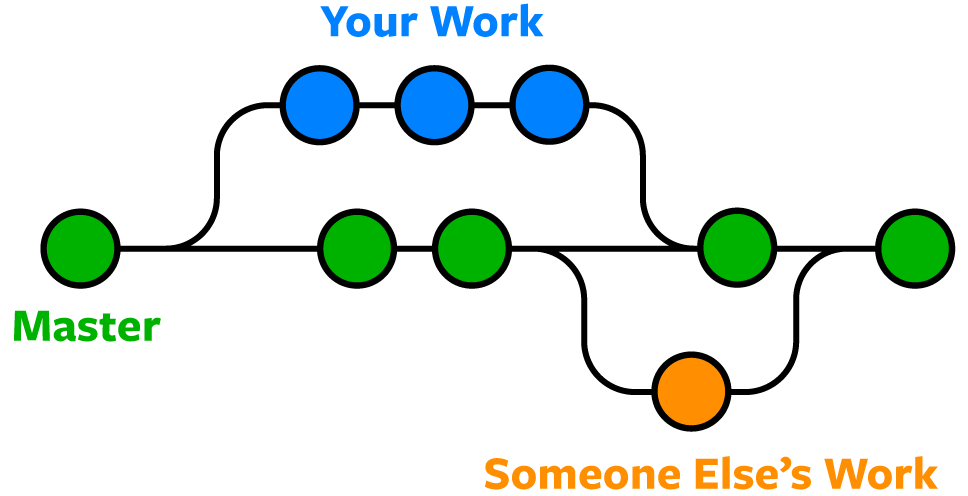
Remote/clone/pull/origin/push/initialize---command for this is git init



When we make the new codes we merge with the main branch. This way we keep on progressing with the codes and securely handle the data.

Git download:

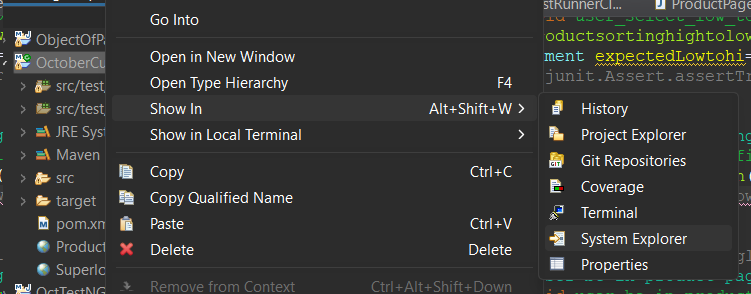
Git is a distributed version control system. Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development



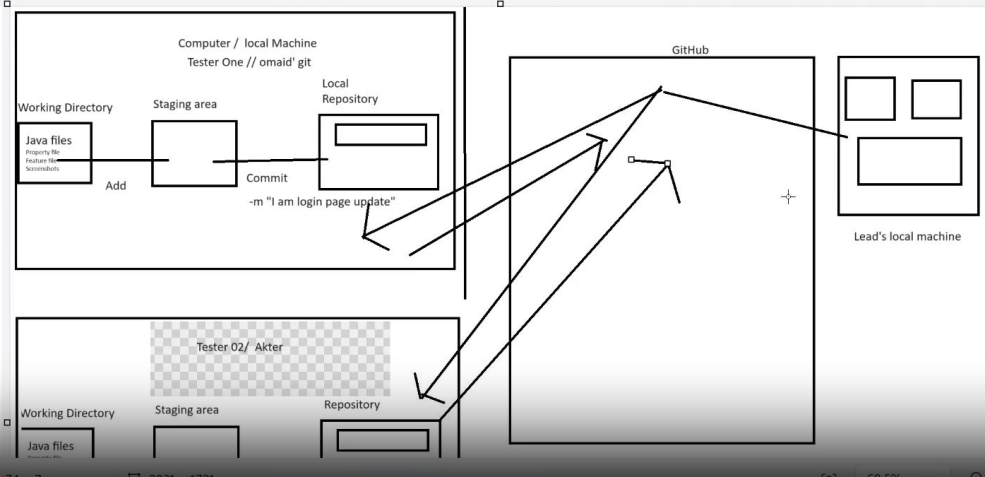
Git can do any file not only java file. Everyday what ever version we are making git keeps a track of it. Also collaborate the work among the programmers.

Git will monitor only the folder we tell it to monitor. The folder git monitors in called folder directory. Git can create a folder and monitor it or we can make a folder and tell git to monitor it.

Right click in the project select show in select system explorer. We have to go inside the folder and tell git to check all the files.



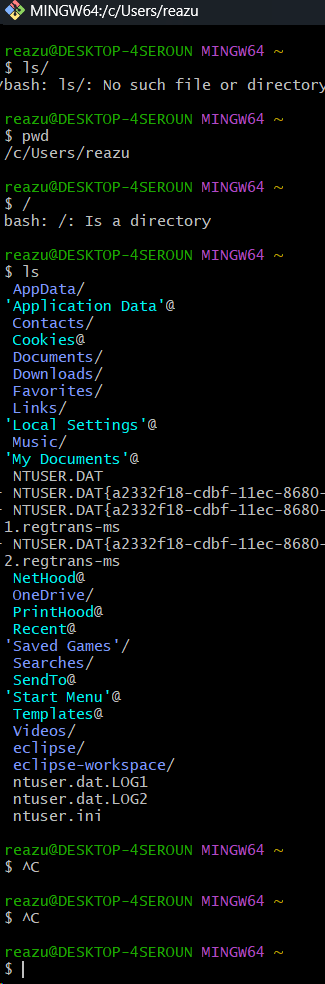
Git structure



Git Bash is a CLI(Command line interface):

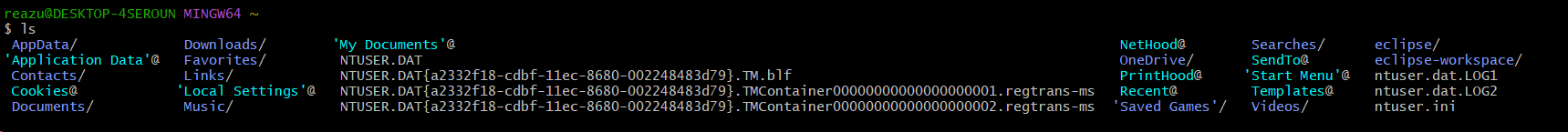
PwD= Present working directory.

Cd/=change directory . / is absolute path



Cd usr/

Ls



If we want to see the root folder then type ls

If we go to any folder and right click and select git bash here in the git bash then type pwd and enter:

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Documents

$ pwd

/c/Users/reazu/OneDrive/Documents

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Documents

$

When we decide that the folder needs to be monitored, we need to initialize git into that folder.

We need to tell git our information as people don’t know who we are?

We need to configure git before we operate:

reazu@DESKTOP-4SEROUN MINGW64 ~/eclipse-workspace/OctoberCucumberTraining

$ pwc

bash: pwc: command not found

reazu@DESKTOP-4SEROUN MINGW64 ~/eclipse-workspace/OctoberCucumberTraining

$ git config --global user.name "reaz"

reazu@DESKTOP-4SEROUN MINGW64 ~/eclipse-workspace/OctoberCucumberTraining

$ git config --global user.email reazun33@gmail.com

reazu@DESKTOP-4SEROUN MINGW64 ~/eclipse-workspace/OctoberCucumberTraining

reazu@DESKTOP-4SEROUN MINGW64 ~/eclipse-workspace/OctoberCucumberTraining

$ git config --global init.defaultBrnach main

reazu@DESKTOP-4SEROUN MINGW64 ~/eclipse-workspace/OctoberCucumberTraining

$ git config --list

diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

http.sslbackend=openssl

http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt

core.autocrlf=true

core.fscache=true

core.symlinks=false

core.editor=notepad

pull.rebase=false

credential.helper=manager

credential.https://dev.azure.com.usehttppath=true

init.defaultbranch=main

user.name=reaz

user.email=reazun33@gmail.com

init.defaultbrnach=main

*how to check git config: git conifg –list*

*what we need to set to configure git*

*name/email/main/*

*Commands:*

1. *bash: pwc: command not found*
2. *git config --global user.name "reaz"*
3. *git config --global user.email reazun33@gmail.com*
4. *git config --global init.defaultBrnach main*

*how to check: git config --list*

how to clear: clear+ enter

Lets create a new folder:

Now we have created a folder for git practice and opened git bash and typed “git init” then this is appeared:

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice

$ git init

Initialized empty Git repository in C:/Users/reazu/OneDrive/Desktop/Java git practice/.git/

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$

If we want to know the status of the git in the folder

Type git status

$ git status

On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$

After creating a file in the folder we get the git status

git status

On branch main

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

fileOne.java

nothing added to commit but untracked files present (use "git add" to track)

now we have to do the staging

git add .+enter

Untracked files:

(use "git add <file>..." to include in what will be committed)

fileOne.java

nothing added to commit but untracked files present (use "git add" to track)

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git add .

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: fileOne.java

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: fileOne.java

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: fileOne.java

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: fileOne.java

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: fileOne.java

Untracked files:

(use "git add <file>..." to include in what will be committed)

pom.xml

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$

If we add a file pom.xml from another folder

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git add pom.xml

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: fileOne.java

new file: pom.xml

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: fileOne.java

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$

***Here in the previous file the change was not added. If we press in the bash git add . then all the changes are added***

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git add .

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: fileOne.java

new file: pom.xml

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$

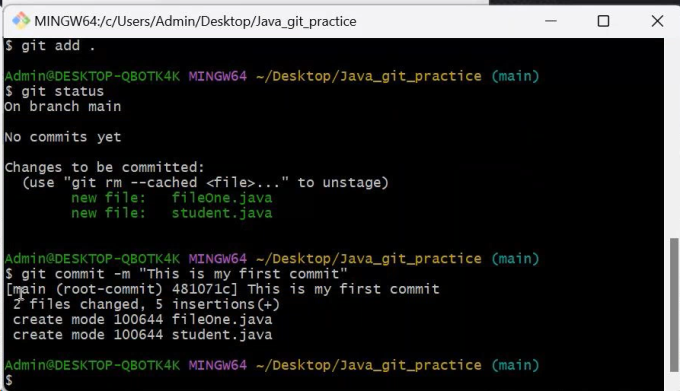
If we want to go back to the original condition of the file that was modified before,

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git restore fileOne.java

***Until we commit it , it is not going to keep the changes and we can always restore the data. Staging area is a temporary virtual place before we make up our mind to commit.***

***How to commit the staged files coz all the files are staged for my first commit***



reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: fileOne.java

new file: pom.xml

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git add .

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git commit -m " My first commit "

[main (root-commit) 4937c58] My first commit

2 files changed, 31 insertions(+)

create mode 100644 fileOne.java

create mode 100644 pom.xml

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$

This is the commit command: $ git commit -m " My first commit "

This has taken a snapshoot of my first commit.

After we send the commit this is gonna come when we check the status:

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git status

On branch main

nothing to commit, working tree clean

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$

If we create a folder in the git file we can get the status like the following

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git status

On branch main

Untracked files:

(use "git add <file>..." to include in what will be committed)

Properties/

nothing added to commit but untracked files present (use "git add" to track)

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git add .

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git status

On branch main

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: Properties/Propertise.docx

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

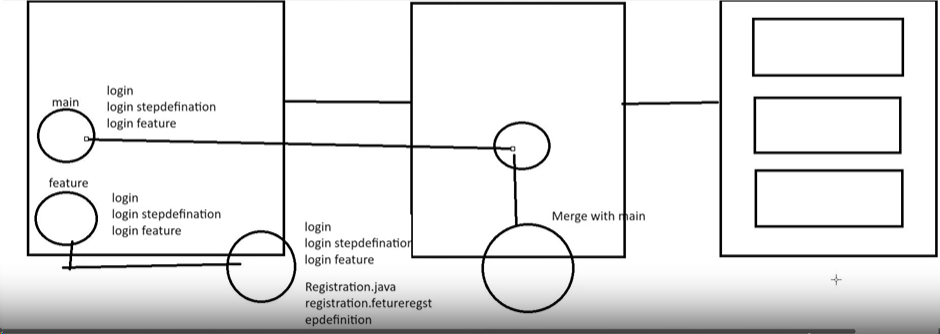
$

Branching: replicating my current work in to a different branch. We need to keep the original file intact so that my original work is not messed up. When all the files are working together we merge the files.

Main has the core file which has less data than the future files. When we merge with main, the old work an d the new work come in one place.

Branching concept:





How you create and give name of the branch:

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git branch

\* main

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git branch dayone

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git branch

dayone

\* main

If we want to switch the branch we need to checkout:

Now instead of main, dayone is active

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git checkout dayone

Switched to branch 'dayone'

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (dayone)

$ git branch

\* dayone

main

Q: when the file is cloned how can we find these 2 main and branch folder?

The moment we add a file in the folder it came to this:

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (dayone)

$ git status

On branch dayone

Untracked files:

(use "git add <file>..." to include in what will be committed)

Instruction to Java.docx

nothing added to commit but untracked files present (use "git add" to track)

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (dayone)

Magic: if we switch to main by checking out the folder is gone

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (dayone)

$ git status

On branch dayone

nothing to commit, working tree clean

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (dayone)

$ git checkout main

Switched to branch 'main'

Folder is the same but git is virtually managing the actions in the folder.

Checkout and switch the branch at the same time

Git checkout -b branchname(tarka).

We have to merge the main with dayone. We have to be in the main and command : git merge dayone

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (dayone)

$ git checkout main

Switched to branch 'main'

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git merge dayone

Updating 83b3707..f69d0e8

Fast-forward

Instruction to Java.docx | Bin 0 -> 9881 bytes

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 Instruction to Java.docx

we use –d flag to delete any branch

git branch –d nameofthebranch

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git branch -d dayone

Deleted branch dayone (was f69d0e8).

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git branch

\* main

How to create and switch a new branch same time

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git branch -d dayone

Deleted branch dayone (was f69d0e8).

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git branch

\* main

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (main)

$ git checkout -b daytwo

Switched to a new branch 'daytwo'

reazu@DESKTOP-4SEROUN MINGW64 ~/OneDrive/Desktop/Java git practice (daytwo)

$ git branch

\* daytwo

Main

If we use arrow key, we can see the previous commands.

1:28

**By git we can do the following tasks: for local repository**

**Create branch/Switch branch/restore branch/delete branch.**

Remote is git hub. Hiring company will give us the credentials to connect with the git hub. AS a tester we will have our git hub account and we put our work there.

Git hub account email.

[wellreaz@gmail.com](mailto:wellreaz@gmail.com) gmail password: derfaze123

github password: derfaze123

github username: rizlert