**// till day=disable method**

**Strick = true – to shows build failure which are disable method**

**Plugin**

**Testing,xml – runner name keep into testing.xml- it shows test-output**

**======**

**Build tag -maven compiler, surefire plugin – it is used to run Command file.**

**Steps to follow:**

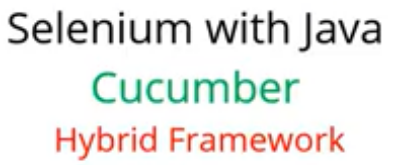
1) Launch the **Eclipse** IDE and from Help menu, click “**Install** New Software”. 2) You will see a dialog window, click “**Add**” button. 3) Type name as you wish, let's take “**Cucumber**” and type “http://**cucumber**.github.com/**cucumber**-**eclipse**/update-site” as location. Click OK.

**For Adding Dependencies**

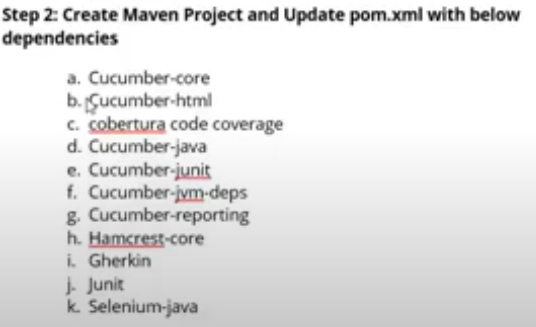
**Search cucumber Skeleton -> go to pom.xml section then add dependencies and plugin**

**Steps to Integrate Eclipse-Cucumber-Natural plugin:**

1. Open **Eclipse**, Click Help -> **Eclipse** Market Place.
2. Type '**cucumber**' in 'Find' and click on Install.
3. Select '**Cucumber** Editor' and click continue.
4. Follow through the prompts.
5. Once all the Steps are complete, **eclipse** will restart.

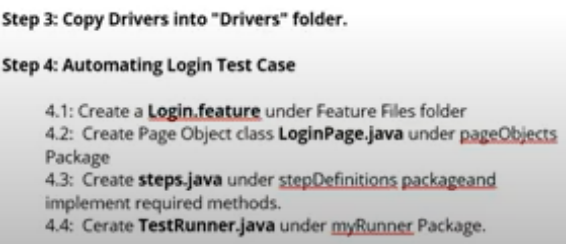








1. in page object pages- page elements.
2. Page objects, step Definition, utilities-package-src/test/java
3. Target folder- report will be generated inside the target folder





**//a[@href='login.php']- Xpath for Mercury -SignOn**

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

Feature : Login

Scenario: Successful Login with Valid Credentials

Given User Lanuch Chrome browser

When User open URL "https://admin-demo.nopcommerce.com/login"

And User enters Email as "admin@yourstore.com" and Password as "admin"

And Click on Login

Then Page Title should be "Dashboard / nopCommerce administration"

When User click on Log out link

Then Page Title should be "your store.Login"

And Close browser

**I can pass some parameter through my Scenario to the Step definition file -as @Parameters**

**So I am going to pass as parameters-**

1. URL <https://admin-demo.nopcommerce.com/login>
2. User Name and password-[admin@yourstore.com](mailto:admin@yourstore.com) and Password as "admin"
3. Title as a parameter- "Dashboard / nopCommerce administration"
4. **And This Title as parameter -**"your store.Login"

**Example**

**Mercury Tours/ http://demo.guru99.com/test/newtours/**

**Page object/Elements Model-** organizing the page elements-

We need to create 1)identification method 2) Operational Method

dr.findElement(By.name(“”).sendkeys(); This statement contains two diff Actions.

1. Identification of the element 2) Operational /Action[what kind of operation we need do on that element]

**So we need two classes- 1) one class for Identification method and**

1. **Another class for Operational method**

So I have a software with five pages- like

1. Login page -3 elements
2. Home pages- 10
3. Customer-20
4. Transactions-30

So for every page we need to create a class and that class contain only Identification method not operational method.

WebEelement username=dr.findElement(By.name(“”).

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

We can use online white board for writing

Test/Main class – here we will do operation [sendkeys();]=========

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

Page Factory –

A simple and easier implementation of page object model in selenium

Selenium’s inbuilt and optimized page object model concept

As POM,has separation of objects and tests.

Uses method initElements to initialize web elements

On calling initElements method all objects on that page gets initiated.

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

Use annotation@FindBy to find WebElements

@FindBy can use id, name, css, xpath,tagName, linkText,partialLinkText etc.

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

Steps 1 -Create a class for each page

Create a package and class for Page Factory in src/test/java

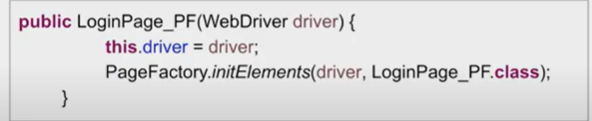
Like class name pageObject / pageFatory

Steps-2 -create locators of all objects to be used in that page using @FindBy

Steps-3 Create methods or actions to be performed on the objects

Steps -4 -Create constructor to get driver instance and

initialize Elements using method initElements



[here LoginPage\_PF.class] 🡪 this is class name]

PageFactory.*initElements*(dr, LoginPage.**class** );-🡪 whenever this will be called all these elements will be initialized on this web page.

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

whenever we create any object of this page of this class -like this

@FindBy(name="username")

WebElement userName ;

this constructor will always be called and we can use this property to

take advantage to initialize or to pass the webdriver.

So In the constructor we are getting the driver from the class which will be calling this class or creating the object for this class.

So whenever an object of this class is created this will run and it will initialize all the web elements

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

Step -5) Update Test Scripts to refer methods from PageFactory class.

**In the TestBase Class under the page object model--Naveen**

In the TestBase Class [Create a Separate Package under src/main/java and then a class – name TestBase.java]

Step-1🡪Declare a Global WebDriver and A Class variable Properties

Step-2🡪 Create a constructor to initialize prop variable with try catch block like this

**public** **class** TestBase1 {

WebDriver dr;

Properties prop; // Class variable properties

TestBase1(){ //**Constructor to read the properties**

**try** {

prop= **new** Properties();

FileInputStream ip;

ip= **new** FileInputStream("C:\\Users\\rumab\\eclipse-workspace\\CucumberPractices\\Config.properties");

prop.load(ip);

}**catch**(FileNotFoundException e) {

e.printStackTrace();

}**catch**(IOException e) {

e.printStackTrace();

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

Step-3 Create one Initialization method- To read the property in the Config.properties file like browser. And it will give String variable.

Conditions-

If browserName dot equals Chrome then I'll define the system property for Chrome . Now the Global variable will be initialized.

// Initialization method

**public** **void** Initialization() {

String browserName=prop.getProperty("browser");

//put one condition

**if**(browserName.equals("chrome")) { //

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

"C:\\Users\\rumab\\eclipse-workspace\\CucumberPractices\\Driver\\chromedriver.exe");

dr= **new** ChromeDriver();

// Now the Global variable will be initialized

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

// Initialization method

**public** **void** Initialization() {

String browserName=prop.getProperty("browser");

//put one condition

**if**(browserName.equals("chrome")) { //

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

"C:\\Users\\rumab\\eclipse-workspace\\CucumberPractices\\Driver\\chromedriver.exe");

dr= **new** ChromeDriver();

} **else** **if** (browserName.equals("firefox")) {

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

"C:\\Users\\rumab\\eclipse-workspace\\CucumberPractices\\Driver\\geckodriver.exe");

dr= **new** FirefoxDriver();

}**else** **if** (browserName.equals("IE")) {

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

"C:\\Users\\rumab\\eclipse-workspace\\CucumberPractices\\Driver\\IEDriverServer.exe");

dr= **new** InternetExplorerDriver();

}

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

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

**TestUtil Class -src/main/java-Package-class**

Let's you tomorrow your application is very slow.It's taking 30 seconds or 40 seconds for the page loading and Implicitly wait also you want to increase and you want to change.

So again you have to change inside the script so better we do one thing that we will create one util class over here inside the util package.

**public** **class** TestUtil {

**public** **static** **long** *Page\_load\_Timeout*=20;

**public** **static** **long** *Implicit\_Wait* =10;

// So inside the testbase remove the hardcode.So i will type this

//TestUtil.Page\_load\_Timeout instead of 20 and TestUtil.Implicit\_Wait instead of 10.

// it means class name [TestUtil] and Variable name

}

dr.manage().timeouts().pageLoadTimeout(TestUtil.*Page\_load\_Timeout*, TimeUnit.***SECONDS***);

dr.manage().timeouts().implicitlyWait(TestUtil.*Implicit\_Wait*, TimeUnit.***SECONDS***);

So let you tomorrow you want to change from 20 to 30 and 10 to 40 or 10 to 20 something like that if you want to change the value you can directly come TestUtil class.

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

**LoginPage- src/main/java-package-class**

The login page will be the child of this base. So use extends keyword.

All the page classes are the child of base class TestBase class.

Step-1

Define the PageFactory/ object repository

@FindBy(xpath="//a[@href='login.php']")

WebElement SignOn;

@FindBy(name="userName")

WebElement Username ;

@FindBy(name="password")

WebElement passWord ;

@FindBy(name="submit")

WebElement loginButton;

**public** **void** SigninButton() {

SignOn.click();;

}

Steps -2 How to initialize the elements with the help of page Factory?

/\*{PageFactory.initElements(dr, LoginPage.class }

whenever this will be called all these elements will be initialized on this web page

\*

\*/

//Initializing the Page object

**public** **void** clickOnSignIne(String submit) {

loginButton.click();

}

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

**Step 3- Create Action Method** – because Different features which are available on LoginPage

1)The first feature is that I want to **verify the title of the page**

**2)T**o verify image of this page

//Actions Methods

//This method should return title of the page

**public** String validateLoginPageTitle() {

**return** dr.getTitle();

}

//it returns true/false

**public** **boolean** validateMurcuryImage() {

**return** murcuryLogo.isDisplayed();

}

**public** HomePage login(String uName,String pwd) {

Username.sendKeys(uName);

passWord.sendKeys(pwd);

loginButton.click();

// it will return Home page. it will directly jump to the home page

//because after clicking on login button it's moving to home page

**return** **new** HomePage();

}

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

**LoginPageTest- Extends TestBase.java**

**Steps-1 Define TestNG Annotation with Method**

**Steps 2 –** Create setup() to call Initialization() into LoginPageTest from the TestBase class

Steps -3 – Create a Constructor of LoginPageTest class to call super keyword.

--super keyword first it will come inside this class[LoginPageTest.java], then it will come inside the Constructor to call superclass constructor[inside the TestBase’s Class Constructor].Because I want to initialize my properties.Because test base class constructor it compulsory to call. By using super keyword I have to call super class constructor[TestBase class].

**public** TestBase1(){ **// This is constructor of TestBase Class**

**try** {

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

FileInputStream ip;

ip= **new** FileInputStream("C:\\Users\\rumab\\eclipse-workspace\\CucumberPractices\\Config.properties");

*prop*.load(ip);

}**catch**(FileNotFoundException e) {

e.printStackTrace();

}**catch**(IOException e) {

e.printStackTrace();

}

}

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

And then it will come inside the setup() in LoginPageTest. Java class. And then it will call Initialization() from TestBase class[which is inside the TestBase class]

It will not give null pointer exception,because my properties are already defined and because we have already called test base class constructor.

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

Step-4 Create an object of LoginPage class.

login page variable I will define at the class level so that throughout my program I can use.

Then It will come over inside the setup() and it will initialize but Before initialization it will call the constructor of super class and it will do --- this things.

// Initialization method

**public** **void** initialization() {

String browserName=*prop*.getProperty("browser");

//put one condition

**if**(browserName.equals("chrome")) { //

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

"C:\\Users\\rumab\\eclipse-workspace\\CucumberPractices\\Driver\\chromedriver.exe");

*dr*= **new** ChromeDriver();

} **else** **if** (browserName.equals("firefox")) {

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

"C:\\Users\\rumab\\eclipse-workspace\\CucumberPractices\\Driver\\geckodriver.exe");

*dr*= **new** FirefoxDriver();

}**else** **if** (browserName.equals("IE")) {

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

"C:\\Users\\rumab\\eclipse-workspace\\CucumberPractices\\Driver\\IEDriverServer.exe");

*dr*= **new** InternetExplorerDriver();

}

*dr*.manage().window().maximize();

*dr*.manage().deleteAllCookies();

*dr*.manage().timeouts().pageLoadTimeout(TestUtil.*Page\_load\_Timeout*, TimeUnit.***SECONDS***);

*dr*.manage().timeouts().implicitlyWait(TestUtil.*Implicit\_Wait*, TimeUnit.***SECONDS***);

//Launch URL

*dr*.get(*prop*.getProperty("url"));

}

using super keyword.

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

Step -5 create teardown() method with @AfterMethod

Steps -6 – Create Test cases –

1. LoginPageTitleTest() and call the method from the LoginPage class and it will return String
2. left side title means ref variable and right side, what is the title of the loginpage.

So go to loginpage and right click and view page source. So login page title.

