According to page object model, every page should have separate page class.

Selenium, playwright, cypress etc codes won’t go into page step def, it will go into respective page class. In page class, we have the locators, ui actions, ui elements. Step def will call the page class. Features will call the step def.

Testng preferred because scenarios within feature file can run in parallel which is not possible in junit. Junit version 5 and above may support but not sure.

Runner files will be present in the same package as step def.

A screenshot of a computer

Description automatically generated

# Let’s use intellij-

A screenshot of a computer

Description automatically generated

Select this maven archetype-

Give name here also, else it shows untitled.

A screenshot of a computer

Description automatically generated

Give name for project and select build system as maven-

A screenshot of a computer

Description automatically generated

You can click advance settings. Then give group id and artifact id if you need.

A screenshot of a computer

Description automatically generated

We changed group id to the application name which is called as opencart.

Default skeleton created-

A screenshot of a computer

Description automatically generated

Delete the App and App test as seen above.

Now goto pom to add dependencies-

A screen shot of a computer code

Description automatically generated

Dependencies added for cucumber with java, cucumber with testng, testng, selenium java-

A screenshot of a computer program

Description automatically generated

Once added in right hand side click maven. Reload the project. This is refresh in eclipse.

Then it will show you to install gherkin and cucumber java tool, tell yes and install and restart ide.

A screenshot of a computer

Description automatically generated

# Note for searching inside maven repo-

Search inside mvn repo using artifact name. see few pictures above this, we can see selenium has artifact name as selenium java.

A screenshot of a computer

Description automatically generated

Note-

All file names should have camel case convention. Example loginPage.feature.

# To get the auto suggestions for given when then etc-

File -> settings -> search for cucumber -> install it.

A screenshot of a computer

Description automatically generated

Install gherkin also.

A screenshot of a computer

Description automatically generated

Cucumber for groovy also-

A screenshot of a computer

Description automatically generated

Install cucumber + -

A screenshot of a computer

Description automatically generated

Ocd-



# See the feature name and its description-

A screen shot of a computer

Description automatically generated

Scenario is normal scenario.

Scenario outline is used for parameterisation.

# Run feature file and see in left side we get all test failed and in right hand side in console the error messages-

A screenshot of a computer program

Description automatically generated

# See the locator for the error message-

There are three classes and we can use any one which will return unique element.

A screenshot of a computer

Description automatically generated

Press “control+click” on respective line in feature file and you go to the respective step def.

A screenshot of a computer

Description automatically generated

Good feature in intellij-

We can run all lines or specific scenario also.

A screen shot of a message

Description automatically generated

Note-

Assertions should be written inside step def only.

Page classes should not have any assertions.

Page classes should have page actions or page methods only, selenium methods, browser specific methods etc.

“@then” used generally used for assertion.

# Testng xml file-

This is to give parallel configuration like how many threads to be used.

A screenshot of a computer

Description automatically generated

Verbose in testng xml file, will tell how much detailed logset we need to generate.

Threadcount in testng xml will tell how many threads we need to use in parallel approximately. We have given around 5 threads to run 5 scenarios.

To run in parallel mode we need to also add maven sure fire plugin in pom file.

Give path of testng xml file inside the surefire plugin, so maven can pick it up easily when we run using maven or Jenkins.

A screen shot of a computer program

Description automatically generated

Give path of runner file inside testng xml file.

A screenshot of a computer

Description automatically generated

# Flow of run through maven is –

Pom file -> testng xml -> test runner -> test runner collects all scenarios.

Now we can run the testng.xml file or directly run from maven.

# How to open terminal in intellij itself-

A screenshot of a computer program

Description automatically generated

Tried pwd command-

A screenshot of a computer

Description automatically generated

Try the below command-



# How to run specific scenario in cucumber-

Add tag in feature file.

A screen shot of a computer

Description automatically generated

In runner file add the tags.

A screen shot of a computer

Description automatically generated

Run it as it works.

A screenshot of a computer

Description automatically generated

# Curly brackets not mandatory for those fields in runner file, when there is only one value-

A screen shot of a computer

Description automatically generated

Run the first set of code using maven-

|  |
| --- |
| Pom –  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <groupId>org.openCart</groupId>  <artifactId>CucumberNewVersionWithKaran</artifactId>  <version>1.0-SNAPSHOT</version>  <packaging>jar</packaging>   <name>untitled</name>  <url>http://maven.apache.org</url>   <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <maven.compiler.source>11</maven.compiler.source>  <maven.compiler.target>11</maven.compiler.target>  </properties>   <dependencies>  <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->  <dependency>  <groupId>io.cucumber</groupId>  <artifactId>cucumber-java</artifactId>  <version>7.13.0</version>  </dependency>  <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->  <dependency>  <groupId>io.cucumber</groupId>  <artifactId>cucumber-testng</artifactId>  <version>7.13.0</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.testng/testng -->  <dependency>  <groupId>org.testng</groupId>  <artifactId>testng</artifactId>  <version>7.8.0</version>  <scope>test</scope>  </dependency>  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.10.0</version>  </dependency>  </dependencies>   <build>  <plugins> <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-surefire-plugin</artifactId>  <version>2.20</version>  <configuration>  <suiteXmlFiles>  <suiteXmlFile>src/test/resources/testng.xml</suiteXmlFile>  </suiteXmlFiles>  </configuration> </plugin>  </plugins>  </build>  </project> |
| Login feature-  Feature: login functionality for open cart application   As a user of opencart  I want to be able to login to my account  So that I can access my account features and manage my orders   Background:  Given I am on open cart login page   Scenario: login success with correct credentials  Given i have entered valid user name and valid password  When i click on login button  Then i should be logged in successfully   Scenario Outline: unsuccessful login with invalid or empty credentails  Given i have entered invalid "<userName>" and "<password>"  When i click on login button  Then i should error message as mentioned in "<errorMessage>"   Examples:  | **userName** | **password** | **errorMessage** |  | **invalid@gmail.com** | **invalidPassword** | **Warning: No match for E-Mail Address and/or Password.** |  | **abcde** | **validPassword** | **Warning: No match for E-Mail Address and/or Password.** |  | **valid@gmail.com** | **abcdef** | **Warning: No match for E-Mail Address and/or Password.** |    Scenario: navigating to forgot password page  When i click on **"Forgotten Password"** link  Then i should be redirected to password reset page |
| Login page object-  package org.openCart.pages;  import org.openqa.selenium.By; import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement;  */\*\*  \* here we write page locators, page actions  \*/* public class LoginPage {   private WebDriver driver;   //by locators  private By emailInputLocator=By.*name*("email");  private By passwordInputLocator=By.*name*("password");  private By loginButtonLocator=By.*xpath*("//input[@type='submit']");  private By forgotPasswordLinkLocator=By.*linkText*("Forgotten Password");  private By logoutLinkLocator=By.*linkText*("Logout");   //constructor  public LoginPage(WebDriver driver){  this.driver=driver;  }   //page methods or page actions  //every page method public in nature  //page objects classic example of encapsulation. private by locators accessed and used by public methods.   public void enterEmail(String emailField){  WebElement emailInput= driver.findElement(emailInputLocator);  emailInput.sendKeys(emailField);  }   public void enterPassword(String passwordField){  WebElement passwordInput= driver.findElement(passwordInputLocator);  passwordInput.sendKeys(passwordField);  }   public void clickLoginButton(){  WebElement loginButton= driver.findElement(loginButtonLocator);  loginButton.click();  }   public boolean checkForgotPasswordLink(){  return driver.findElement(forgotPasswordLinkLocator).isDisplayed();  }   public boolean checkLogoutLink(){  return driver.findElement(logoutLinkLocator).isDisplayed();  }   public void clickForgotPasswordLink(){  WebElement forgotPasswordLink= driver.findElement(forgotPasswordLinkLocator);  forgotPasswordLink.click();  }   public void login(String userName, String password){  enterEmail(userName);  enterPassword(password);  clickLoginButton();  }   public String getForgotPasswordPageURL(){  String forgotPasswordPageURL= driver.getCurrentUrl();  return forgotPasswordPageURL;  }  } |
| Login page steps-  package org.openCart.stepDefs;  import io.cucumber.java.After; import io.cucumber.java.Before; import io.cucumber.java.en.Given; import io.cucumber.java.en.Then; import io.cucumber.java.en.When; import org.openCart.pages.LoginPage; import org.openqa.selenium.By; import org.openqa.selenium.WebDriver; import org.openqa.selenium.chrome.ChromeDriver; import org.testng.Assert;  */\*\*  \* here we will write step def code with help of cucumber annotations  \* we write selenium/java code also.  \* this will also call the page methods from the respective page class.  \*/* public class LoginPageStepDef {   private WebDriver driver;  private LoginPage loginPage;   //before and after annotations are known as hooks in cucumber.   @Before  public void setUp(){  driver=new ChromeDriver();  }   @After  public void tearDown(){  if(driver!=null){  driver.quit();  }  }   @Given("I am on open cart login page")  public void i\_am\_on\_open\_cart\_login\_page() {  driver.get("https://naveenautomationlabs.com/opencart/index.php?route=account/login");  loginPage=new LoginPage(driver); // we create object of this login page class so we can access methods from the  //page class.  }   @Given("i have entered valid user name and valid password")  public void i\_have\_entered\_valid\_user\_name\_and\_valid\_password() {  loginPage.enterEmail("cucumbertesting1960@gmail.com");  loginPage.enterPassword("Malaravi@123");  }   @When("i click on login button")  public void i\_click\_on\_login\_button() {  loginPage.clickLoginButton();  }  @Then("i should be logged in successfully")  public void i\_should\_be\_logged\_in\_successfully() {  Assert.*assertEquals*(loginPage.checkLogoutLink(), true);  }   */\*\*  \* when a method being called returns boolean, then better to check the validation with assert statement  \* in the calling class.  \** ***@param*** *userName  \** ***@param*** *password  \*/* @Given("i have entered invalid {string} and {string}")  public void i\_have\_entered\_invalid\_and(String userName, String password) {  loginPage.enterEmail(userName);  loginPage.enterPassword(password);  }   @Then("i should error message as mentioned in {string}")  public void i\_should\_error\_message\_as\_mentioned\_in(String errorMessage) {  //Assert that error on page is equal to error we are expecting  Assert.*assertEquals*(driver.findElement(By.*cssSelector*(".alert-danger")).isDisplayed(), true);  }   @When("i click on {string} link")  public void i\_click\_on\_link(String string) {  loginPage.clickForgotPasswordLink();  }  @Then("i should be redirected to password reset page")  public void i\_should\_be\_redirected\_to\_password\_reset\_page() {  //System.out.println(loginPage.getForgotPasswordPageURL());  Assert.*assertTrue*(loginPage.getForgotPasswordPageURL().contains("account/forgotten"));  //one trick for remembering whether to use assert equals or assert true is check return type.  //the getforgotpasswordpageurl returns string so we check if the string is present or not.  } } |
| Test runner-  package runner;  import io.cucumber.testng.AbstractTestNGCucumberTests; import io.cucumber.testng.CucumberOptions; import org.testng.annotations.DataProvider;  @CucumberOptions(features = "src/test/resources/features", glue="org.openCart.stepDefs", plugin={"pretty", "html:target/cucumber-reports", "json:target/cucumber.json"}, // we are generating json and html reports. monochrome = true, publish = true)  //monochrome will give good output format in console. //publish will help to publish in cloud.  //to run test cases in parallel mode extend the class abstracttestngcucumbertests. public class TestRunner extends AbstractTestNGCucumberTests {   //this will help in running all scenarios in parallel mode.  //name of the method should exactly match. if we scenario instead of scenarios then override throws error  //"method does not override from its superclass"  //this will collect all scenarios and then run them in parallel.  @Override  @DataProvider(parallel = true)  public Object[][] scenarios(){  return super.scenarios();  } } |
| Testng xml –  <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd"> <suite name="open cart parallel suite" verbose="2" parallel="classes" thread-count="5">  <test name="login page test">  <classes>  <class name="runner.TestRunner"/>  </classes>  </test> </suite> |

Output-

A screenshot of a computer

Description automatically generated

This is the cucumber cloud report which we are talking-

A screen shot of a computer

Description automatically generated

See how report looks when opened-

A screenshot of a computer

Description automatically generated

We can edit the feature files on cloud also and write scenarios if required-

A screenshot of a computer

Description automatically generated

Let’s see if json report and html report generated in target folder-

A screenshot of a computer

Description automatically generated

Right click on report.

Then copy path reference.

A screenshot of a computer

Description automatically generated

Copy absolute path.

A screenshot of a computer

Description automatically generated

Entire path will be copied.

Paste in browser.

Json report in json format-

A screenshot of a computer

Description automatically generated

Html report in html format-

A screenshot of a computer

Description automatically generated

But this is in html format code.

We want in nice html.

So go back to intellij.

Refactor and rename the report.

A screenshot of a computer

Description automatically generated

Rename the report and add “.html”

A screenshot of a computer

Description automatically generated

Done.

Reopen the report in browser and voila.

A screenshot of a computer

Description automatically generated

# Project structure-

# Ocd-

A computer screen shot of a blue screen

Description automatically generated

A screenshot of a computer

Description automatically generated

# Codes from this chapter –

Already pasted up.