**What is BDD Framework? VVVVVVVVIMMMMMPPPPPPPPPP**

- BDD stands for Behavior Driven Development framework

- BDD Framework is extension of TDD (Test Driven Development) framework.

- BDD Framework focus on scenario not in Test Cases

- BDD Framework focus on What to Test not on how to test.

- in BDD Framework we use plain English language to write the requirements

- we use gherkin keyword to write feature file with plain English language

**there are different tools available in BDD**

1) Cucumber

2) JBehave

3) SpecFlow

**BDD framework support different languages**

1) Java

2) Java script

3) Perl

4) Python

5) PHP

6) Groovy script

7) Ruby

**in my current company we use Cucumber with java platform**

**In cucumber we have 3 most important files**

1) Feature file

2) Step Definition

3) Test Runner

**1) Feature file**

- we create feature file with dot feature extension

- inside the Feature file first keyword must be the Feature followed by colon (:) then short description of functionality

- then next we can use Scenario followed by colon then short description of scenario or we can use Background keyword followed by colon and then short description of Backgrounds or we can use Scenario Outline keyword followed by colon symbol then short description of scenarios.

- Once we declare the Scenario then we write different Steps in Scenario by using different keyword

I) Given

2) When

3) Then

4) And

5) But

6) \*

- above all keyword is used to create the steps in feature files.

- Given is used to mention the preconditions

- When keyword is used to mention the actions or events

- Then keyword is used to mention expected condition or result

- And keyword is used to combine the two or more steps

- But keyword is used to declare the multiple conditions

-\* keyword is used to declare any above conditions,

\* keyword is also called as global keyword

- by using Examples keyword followed by colon (:) is used to mention the test data from Scenario Outline

**2) Step Definition file**

-Step Definition file means java class

- inside the Step Definition file we write selenium script code as the snippets

**3) Test Runner file**

- inside the Test Runner class we define location of feature by using features keyword and

- location of Step Definition file by using glue keyword or extraGlue keyword

Feature: Test Orange HRM application

Scenario: validate login functionality

Given user is on login page

When user enter valid username and valid password

Then user click login button

package StepDefinition;

import java.time.Duration;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class LoginSteps {

static WebDriver driver;

@Given("user is on login page")

public void user\_is\_on\_login\_page() {

// connect browser

driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

driver.manage().deleteAllCookies();

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

driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

}

@When("user enter valid username and valid password")

public void user\_enter\_valid\_username\_and\_valid\_password() {

driver.findElement(By.name("username")).sendKeys("Admin");

driver.findElement(By.name("password")).sendKeys("admin123");

}

@Then("user click login button")

public void user\_click\_login\_button() {

driver.findElement(By.xpath("//button[text()=' Login ']")).click();

}

}

package RunnerTest;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features={"src/test/java/Features"},

extraGlue={"StepDefinition"}

)

public class TestRunner {

}

**Gherkin keywords and their use**

1) Feature: it used to mention the main functionality name or main feature name

2) Scenario: it used to mention the Scenario name.

it used to mention what is exact requirement we have to test it.

3) Given it is used to mention the precondition for steps

4) When it is used to mention the actions or events

5) Then it is used to mention the expected outcome or result

6) And it is used to combine the 2 or more steps

7) But it is used to declare multiple conditions

8) \* it is used to declare any above conditions

and it is also called as global keyword

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

**Test Runner**

1) @RunWith(Cucumber.class) annotation : it tells to Junit, run the class as Cucumber Test class

2) @CucumberOptions() : it tells lots of things as like

i) features= "feature file location" where is feature file locations by using features keyword

ii) g;ue="StepDefinition" where is step definition file location by using glue or extraGlue keyword

iii) dryRun=true keyword it check the mapping between feature file steps vs implemented snippets inside the

step definition package.

iv) monochrome=true keyword is used to display the output console in proper readable format

v) tags=@tagName keyword, is used to specific scenario or specific scenario group.

vi)plugins ="reports" keyword is used to generate the different types of reports.

**how to check mapping between feature file step and implemented steps inside the Step Definition package? VVVVIMMMPPP**

**what is use of dryRun keyword in cucumber?**

- dryRun=true we use inside the @CucumberOptions annotation in Test Runner class.

dryRun=true keyword is used to check the mapping between feature steps and implemented snippets inside the Step definition package and once we use dryRun=true keyword it check mapping and if mapping is incorrect then it generate the snippet for feature file steps.

- once we use dryRun=true keyword it just check the mapping and it is not going to execute any scenario.

- but if we use dryRun=false keyword , so first it run the scenario from feature and if mapping is incorrect then it generate the snippets for that scenario.

Note:

---> as per the old version we use strict keyword to check mapping.

iv) monochrome=true keyword is used to display the output console in proper readable format

**What is use of plugin keyword in cucumber?** VVVVVIMMMMPPPPPP

plugin keyword is used to generate the different types of reports in cucumber

1) json file report

2) xml file report

3) html report

4) txt file report

plugin={

"json:Reports/abc.json",

"junit:Reports/abc.xml",

"html:Reports/abc.html",

"pretty:Reports/abc/txt"

}

**how to run specific scenario from Feature file? VVVVVIMMMMPPPPPP**

- first we define the tagname in feature file by using @ symbol then tagname.

- we can define tagname in scenario level as well as feature file.

- then we use tags keyword inside the @CucumberOptions annotation by passing @tagname.

and it will run the specific scenario.

tags="@LoginPage"

package TestRunner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features="src/test/java/Features",

glue="StepDefinition",

dryRun=false,

monochrome=true,

plugin= {

"html:Reports/abc.html",

"json:Reports/abc.json",

"junit:Reports/abc.xml",

"pretty:Reports/abc.txt"

},

tags="@LoginPage"

)

public class RunnerTest {

}

**how to skip specific scenario from feature file? VVVVVIMMMMPPPPPP**

- first we define the tagname in feature file by using @ symbol then tagname.

- we can define tagname in scenario level as well as feature file.

- then we use tags keyword inside the @CucumberOptions annotation by passing not @tagname.

and it will skip the specific scenario

e.g.

tags="not @HomePage"

package TestRunner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features="src/test/java/Features",

glue="StepDefinition",

dryRun=false,

monochrome=true,

plugin= {

"html:Reports/abc.html",

"json:Reports/abc.json",

"junit:Reports/abc.xml",

"pretty:Reports/abc.txt"

},

tags="not @HomePage"

)

public class RunnerTest {

}

**How to execute 2 scenarios from feature file?**

- first we define the tagname in feature file by using @ symbol then tagname.

- we can define tagname in scenario level as well as feature file.

- then we use tags keyword inside the @CucumberOptions annotation by passing @tagname or @tagname

and it will run the 2 scenario

e.g.

tags="@LoginPage or @PIMPage"

package TestRunner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features="src/test/java/Features",

glue="StepDefinition",

dryRun=false,

monochrome=true,

plugin= {

"html:Reports/abc.html",

"json:Reports/abc.json",

"junit:Reports/abc.xml",

"pretty:Reports/abc.txt"

},

tags="@LoginPage or @PIMPage"

)

public class RunnerTest {

}

Scenarios:

1) login functionality

2) home page functionality

3) PIM page functionality

@EndToEndTesting

Feature: Test the Orange HRM application

@LoginPage @RegressionTesting

Scenario: Validate login functionality

Given user is on login page

When user enter username and password

Then user click on login button

@HomePage @SanityTesting

Scenario: validate Home Page Functionality

When user is on home page and validate home page title

And user validate home page url

And user validate home page logo

@PIMPage @SystemTesting

Scenario: validate pim page functionality

Given user is on pim page and verify the url

When user click on add employee

And user enter first name, middle name and last name

And user click on save button

package StepDefinition;

import static org.junit.jupiter.api.Assertions.assertEquals;

import java.time.Duration;

import org.junit.jupiter.api.Assertions;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class OrangeHRMSteps {

static WebDriver driver;

@Given("user is on login page")

public void user\_is\_on\_login\_page() {

driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

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

driver.manage().deleteAllCookies();

driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

}

@When("user enter username and password")

public void user\_enter\_username\_and\_password() {

driver.findElement(By.name("username")).sendKeys("Admin");

driver.findElement(By.name("password")).sendKeys("admin123");

}

@Then("user click on login button")

public void user\_click\_on\_login\_button() {

driver.findElement(By.xpath("//button[text()=' Login ']")).click();

}

@When("user is on home page and validate home page title")

public void user\_is\_on\_home\_page\_and\_validate\_home\_page\_title() {

String actualTitle = driver.getTitle();

Assertions.assertEquals(actualTitle, "OrangeHRM");

}

@When("user validate home page url")

public void user\_validate\_home\_page\_url() {

boolean actualResult = driver.getCurrentUrl().contains("orange");

Assertions.assertEquals(actualResult, true);

}

@When("user validate home page logo")

public void user\_validate\_home\_page\_logo() {

boolean actualLogo = driver.findElement(By.xpath("//img[@alt='client brand banner']")).isDisplayed();

Assertions.assertEquals(actualLogo, true);

}

@Given("user is on pim page and verify the url")

public void user\_is\_on\_pim\_page\_and\_verify\_the\_url() {

driver.findElement(By.xpath("//span[text()='PIM']")).click();

boolean actualResult = driver.getCurrentUrl().contains("pim");

Assertions.assertTrue(actualResult);

}

@When("user click on add employee")

public void user\_click\_on\_add\_employee() {

driver.findElement(By.xpath("//a[text()='Add Employee']")).click();

}

@When("user enter first name, middle name and last name")

public void user\_enter\_first\_name\_middle\_name\_and\_last\_name() {

driver.findElement(By.name("firstName")).sendKeys("Rahul");

driver.findElement(By.name("middleName")).sendKeys("abc");

driver.findElement(By.name("lastName")).sendKeys("Shah");

}

@When("user click on save button")

public void user\_click\_on\_save\_button() {

driver.findElement(By.xpath("//button[text()=' Save ']")).click();

}

}

package TestRunner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features="src/test/java/Features",

glue="StepDefinition",

dryRun=false,

monochrome=true,

plugin= {

"html:Reports/abc.html",

"json:Reports/abc.json",

"junit:Reports/abc.xml",

"pretty:Reports/abc.txt"

},

tags="@EndToEndTesting"

)

public class RunnerTest {

}

**Gherkin Keyword in feature files VVVVVVIMMPPPPPPPPP**

1) Feature:

2) Scenario: -------------------------------------->Example:

3) Given

4) When

5) Then

6) And

7) But

8) \*

9) Scenario Outline: -------------------------------------->Scenario Template:

10) Examples: -------------------------------------->Scenarios:

11) Background:

**Test Runner class keyword VVVVVVIMMPPPPPPPPP**

1) @RunWith() annotation

2) @CucumberOptions() annotation

3) features keyword

4) glue or extraGlue keyword

5) dryRun keyword

6) monochrome keyword

7) plugin

8) tags keyword

**How to comments feature file steps? VVVVVVIMMPPPPPPPPP**

we use # symbol to comments the feature files.

**What is Parameterization in cucumber? VVVVVVIMMPPPPPPPPP**

**or**

**How to achieve Parameterization in cucumber? VVVVVVIMMPPPPPPPPP**

- if we pass String test data within the double quote in feature file steps then automatically cucumber will consider as String parameter in snippets

- if we pass integer test data in feature file steps then automatically cucumber will consider as integer parameter in snippets

-and cucumber will generate snippets with parameters or arguments.

- and then we can use parameter or arguments within the scripts.

Feature: Test Facebook application Functionality

Scenario: Validate Register Page Functionality

Given user open a "edge" browser

And user enter first name as "Dipali" and last name as "Patil"

And user select date as 25 and month as "Jun" and year 2010

package StepDefinition;

import java.time.Duration;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.openqa.selenium.edge.EdgeDriver;

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

import io.cucumber.java.en.Given;

public class RegisterPageTestStep {

static WebDriver driver;

@Given("user open a {string} browser")

public void user\_open\_a\_browser(String browsername) {

if (browsername.equalsIgnoreCase("chrome")) {

driver = new ChromeDriver();

} else if (browsername.equalsIgnoreCase("edge")) {

driver = new EdgeDriver();

} else if (browsername.equalsIgnoreCase("incognito")) {

ChromeOptions opt = new ChromeOptions();

opt.addArguments("--incognito");

driver = new ChromeDriver(opt);

} else {

System.out.println("Please Enter browser name");

}

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

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

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

}

@Given("user enter first name as {string} and last name as {string}")

public void user\_enter\_first\_name\_as\_and\_last\_name\_as(String FirstName, String LastName) {

driver.findElement(By.name("firstname")).sendKeys(FirstName);

driver.findElement(By.name("lastname")).sendKeys(LastName);

}

@Given("user select date as {int} and month as {string} and year {int}")

public void user\_select\_date\_as\_and\_month\_as\_and\_year(Integer Date, String Month, Integer Year) {

WebElement wbDate = driver.findElement(By.name("birthday\_day"));

Select selDate = new Select(wbDate);

// convert Integer into String

String date = Date.toString();

selDate.selectByVisibleText(date);

WebElement wbMonth = driver.findElement(By.name("birthday\_month"));

Select selMonth = new Select(wbMonth);

selMonth.selectByVisibleText(Month);

WebElement wbYear = driver.findElement(By.name("birthday\_year"));

Select selYear = new Select(wbYear);

String year = Year.toString();

selYear.selectByVisibleText(year);

}

}

**Scenario Example**

**Scenario Outline Scenario Template**

**Examples**  Scenarios

**diff between Scenario and Scenario Outline keyword? VVVVVIMMMPPPPPPPP**

- we use Scenario and Scenario Outline keyword in feature file.

- Scenario keyword is used to mention what is exact requirements or plain requirements.

- and Scenario Outline keyword is used to achieve the data driven testing in cucumber and we maintain the set of Test Data in Examples keyword by using

pipe symbol.

**diff between Scenario Outline and Scenario Template keyword? VVVVVIMMMPPPPPPPP**

- we use Scenario Outline and Scenario Template keyword in feature file.

- Scenario Outline and Scenario Template keyword use for same purpose to achieve the data driven testing in cucumber.

**diff between Examples and Scenarios keyword? VVVVVIMMMPPPPPPPP**

- we use Examples and Scenarios keyword in feature file.

- Examples and Scenarios keyword use for same purpose to maintain the set of test data for Scenario Outline or Scenario Template.

**diff between Scenario and Scenarios keyword? VVVVVIMMMPPPPPPPP**

- we use Scenario and Scenarios keyword in feature file.

- Scenario keyword is used to mention the what is exact requirement or plain requirement.

- and Scenarios keyword is used to maintain the set of test data for Scenario Outline or Scenario Template.

**Diff between Example and Examples keyword? VVVVVIMMMPPPPPPPP**

- we use Example and Examples keyword in feature file.

- Example keyword is similar as Scenario keyword and it is used to mention what is exact requirements or plain requirements.

- Examples keyword is similar as Scenarios keyword and it used to maintain the set of test data for Scenario Outline or Scenario Template.

**How to achieve data driven testing in cucumber? VVVVVIMMMPPPPPPPP**

What is use of Scenario Outline keyword and Examples keyword in feature file?

what is data table in feature file?

- there are 2 ways we can achieve the data driven testing in cucumber

1) by using Scenario Outline: keyword

2) by using DataTable keyword

**1) by using Scenario Outline: keyword**

- Scenario Outline: keyword is used to achieve the data driven testing in cucumber.

- instead of using Scenario keyword we use Scenario Outline keyword and we maintain the test data in Examples keyword by using Pipe symbol.

- once we use Scenario Outline keyword it will run the same scenario (all steps) multiple times as per the test data.

Feature: Test Facebook application Functionality

Scenario Outline: Validate Register Page Functionality

Given user open a "<browsername>" browser

And user enter first name as "<firstname>" and last name as "<lastname>"

And user select date as "<date>" and month as "<month>" and year "<year>"

Examples:

| browsername | firstname | lastname | date | month | year |

| chrome | Dipali | patil | 21 | Feb | 2010 |

| edge | Rahul | Patil | 12 | Dec | 2005 |

| chrome | Kavita | page | 24 | Mar | 2009 |

| incognito | Kriti | Patil | 18 | Jun | 2019 |

| edge | Komal | page | 11 | Aug | 2003 |

package StepDefinition;

import java.time.Duration;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.openqa.selenium.edge.EdgeDriver;

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

import io.cucumber.java.en.Given;

public class RegisterPageTestStep {

static WebDriver driver;

@Given("user open a {string} browser")

public void user\_open\_a\_browser(String browsername) {

if (browsername.equalsIgnoreCase("chrome")) {

driver = new ChromeDriver();

} else if (browsername.equalsIgnoreCase("edge")) {

driver = new EdgeDriver();

} else if (browsername.equalsIgnoreCase("incognito")) {

ChromeOptions opt = new ChromeOptions();

opt.addArguments("--incognito");

driver = new ChromeDriver(opt);

} else {

System.out.println("Please Enter browser name");

}

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

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

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

}

@Given("user enter first name as {string} and last name as {string}")

public void user\_enter\_first\_name\_as\_and\_last\_name\_as(String FirstName, String LastName) {

driver.findElement(By.name("firstname")).sendKeys(FirstName);

driver.findElement(By.name("lastname")).sendKeys(LastName);

}

@Given("user select date as {string} and month as {string} and year {string}")

public void user\_select\_date\_as\_and\_month\_as\_and\_year(String Date, String Month, String Year) {

WebElement wbDate = driver.findElement(By.name("birthday\_day"));

Select selDate = new Select(wbDate);

selDate.selectByVisibleText(Date);

WebElement wbMonth = driver.findElement(By.name("birthday\_month"));

Select selMonth = new Select(wbMonth);

selMonth.selectByVisibleText(Month);

WebElement wbYear = driver.findElement(By.name("birthday\_year"));

Select selYear = new Select(wbYear);

selYear.selectByVisibleText(Year);

}

}

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

Feature: Test Guru99 Application

Scenario Outline: Validate Register Page Functionality

Given user is on register page

When user enter contact information as "<firstname>" and "<lastname>" and "<phone>" and "<email>"

And use enter Mailing information as "<address>", "<city>", "<state>", "<postalCode>" and "<country>"

And user enter user information as "<username>" and "<password>" and "<confirmPassword>"

And user click on submit button

Examples:

| firstname | lastname | phone | email | address | city | state | postalCode | country | username | password | confirmPassword |

| Anjali | Gurav | 909090 | Anjali@gmail.com | Pune | Pune | MH | 221122 | INDIA | anjali@gmail.com | Anjali@123 | Anjali@123 |

| Neha | Gupta | 808080 | Neha@gmail.com | Mumbai | Mumbai | MH | 123456 | ERITREA | Neha@gmail.com | Neha@123 | Neha@123 |

Examples:

| firstname | lastname | phone | email | address | city | state | postalCode | country | username | password | confirmPassword |

| Rahul | Durke | 7070790 | Rahul@gmail.com | Delhi | Delhi | DL | 123456 | CANADA | Rahul@gmail.com | Rahul@123 | Rahul@123 |

| Manoj | Shah | 12445555 | Manoj@gmail.com | Delhi | Delhi | DL | 132424 | AMERICAN SAMOA | Manoj@gmail.com | Manoj@123 | Manoj@123 |

Examples:

| firstname | lastname | phone | email | address | city | state | postalCode | country | username | password | confirmPassword |

| Amar | Salve | 909090 | Amar@gmail.com | Dubai | Delhi | DL | 910101 | MALDIVES | Amar@gmail.com | Amar@123 | Amar@123 |

| Nikhil | Patil | 909090 | Nikhil@gmail.com | Dubai | Delhi | DL | 910101 | INDIA | Nikhil@gmail.com | Nikhil@123 | Nikhil@123 |

Examples:

| firstname | lastname | phone | email | address | city | state | postalCode | country | username | password | confirmPassword |

| Punam | Mahajan | 909090 | Amar@gmail.com | Dubai | Delhi | DL | 910101 | AMALDIVES | Punam@gmail.com | Punam@123 | Punam@123 |

Feature: Test Guru99 Application

# Scenario Outline == Scenario Template

Scenario Template: Validate Register Page Functionality

Given user is on register page

When user enter contact information as "<firstname>" and "<lastname>" and "<phone>" and "<email>"

And use enter Mailing information as "<address>", "<city>", "<state>", "<postalCode>" and "<country>"

And user enter user information as "<username>" and "<password>" and "<confirmPassword>"

And user click on submit button

# Examples == Scenarios :

Scenarios:

| firstname | lastname | phone | email | address | city | state | postalCode | country | username | password | confirmPassword |

| Anjali | Gurav | 909090 | Anjali@gmail.com | Pune | Pune | MH | 221122 | INDIA | anjali@gmail.com | Anjali@123 | Anjali@123 |

| Neha | Gupta | 808080 | Neha@gmail.com | Mumbai | Mumbai | MH | 123456 | ERITREA | Neha@gmail.com | Neha@123 | Neha@123 |

Examples:

| firstname | lastname | phone | email | address | city | state | postalCode | country | username | password | confirmPassword |

| Rahul | Durke | 7070790 | Rahul@gmail.com | Delhi | Delhi | DL | 123456 | CANADA | Rahul@gmail.com | Rahul@123 | Rahul@123 |

| Manoj | Shah | 12445555 | Manoj@gmail.com | Delhi | Delhi | DL | 132424 | AMERICAN SAMOA | Manoj@gmail.com | Manoj@123 | Manoj@123 |

Examples:

| firstname | lastname | phone | email | address | city | state | postalCode | country | username | password | confirmPassword |

| Amar | Salve | 909090 | Amar@gmail.com | Dubai | Delhi | DL | 910101 | MALDIVES | Amar@gmail.com | Amar@123 | Amar@123 |

| Nikhil | Patil | 909090 | Nikhil@gmail.com | Dubai | Delhi | DL | 910101 | INDIA | Nikhil@gmail.com | Nikhil@123 | Nikhil@123 |

Examples:

| firstname | lastname | phone | email | address | city | state | postalCode | country | username | password | confirmPassword |

| Punam | Mahajan | 909090 | Amar@gmail.com | Dubai | Delhi | DL | 910101 | AMALDIVES | Punam@gmail.com | Punam@123 | Punam@123 |

package Features;

import java.time.Duration;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

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

import io.cucumber.java.en.Given;

import io.cucumber.java.en.When;

public class RegisterPageTestStep {

static WebDriver driver;

@Given("user is on register page")

public void user\_is\_on\_register\_page() {

driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

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

driver.get("https://demo.guru99.com/test/newtours/register.php");

}

@When("user enter contact information as {string} and {string} and {string} and {string}")

public void user\_enter\_contact\_information\_as\_and\_and\_and(String fname, String lname, String phone, String email) {

driver.findElement(By.name("firstName")).sendKeys(fname);

driver.findElement(By.name("lastName")).sendKeys(lname);

driver.findElement(By.name("phone")).sendKeys(phone);

driver.findElement(By.name("userName")).sendKeys(email);

}

@When("use enter Mailing information as {string}, {string}, {string}, {string} and {string}")

public void use\_enter\_mailing\_information\_as\_and(String address, String city, String state, String pincode,

String country) {

driver.findElement(By.name("address1")).sendKeys(address);

driver.findElement(By.name("city")).sendKeys(city);

driver.findElement(By.name("state")).sendKeys(state);

driver.findElement(By.name("postalCode")).sendKeys(pincode);

WebElement wbCountry = driver.findElement(By.name("country"));

Select selCountry = new Select(wbCountry);

selCountry.selectByVisibleText(country);

}

@When("user enter user information as {string} and {string} and {string}")

public void user\_enter\_user\_information\_as\_and\_and(String uname, String pass, String cpass) {

driver.findElement(By.name("email")).sendKeys(uname);

driver.findElement(By.name("password")).sendKeys(pass);

driver.findElement(By.name("confirmPassword")).sendKeys(cpass);

}

@When("user click on submit button")

public void user\_click\_on\_submit\_button() throws InterruptedException {

Thread.sleep(5000);

driver.findElement(By.name("submit")).click();

Thread.sleep(5000);

driver.quit();

}

}

**how to achieve data driven testing using Cucumber.**

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

there are 2 ways we can achieve the data driven testing in cucumber

**1) by using Scenario Outline keyword**

**2) by using DataTable class.**

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

**2) by using DataTable class.**

- we can achieve the data driven testing in cucumber using DataTable class.

- we declare the test data in steps level and we generate the snippets, in snippets it will provide the DataTable class arguments or parameter.

- and these Test data accessible only to above steps only.

- and if we mention the multiple test data then it will run above steps multiple times.

-there are multiple ways we can achieve the Data driven testing by using DataTable class

I) by using asLists() method from DataTable class.

2) by using cells() method from DataTable class.

3) by using asMaps() method from DataTable class.

- in my current framework we use asLists() method from DataTable class.

**I) by using asLists() method from DataTable class.**

- return type of asLists() method is List<List<String>>

List<List<String>> list = dataTable.asLists();

String a = list.get(0).get(0);

String b = list.get(0).get(1);

e.g.

Feature: Test orange HRM application

Scenario: Validate the login functionality

Given user is login page

When user enter valid username and valid password

| Admin | admin123 |

Then user click on login button

@When("user enter valid username and valid password")

public void user\_enter\_valid\_username\_and\_valid\_password(DataTable dataTable)

{

List<List<String>> list = dataTable.asLists();

String uname = list.get(0).get(0);

String pass= list.get(0).get(1);

}

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

**2) by using cells() method from DataTable class.**

- return type of cells() method is List<List<String>>

List<List<String>> list = dataTable.cells();

String a = list.get(0).get(0);

String b = list.get(0).get(1);

e.g.

Feature: Test orange HRM application

Scenario: Validate the login functionality

Given user is login page

When user enter valid username and valid password

| Admin | admin123 |

Then user click on login button

@When("user enter valid username and valid password")

public void user\_enter\_valid\_username\_and\_valid\_password(DataTable dataTable)

{

List<List<String>> list = dataTable.cells();

String uname = list.get(0).get(0);

String pass= list.get(0).get(1);

}

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

**3) by using asMaps() method from DataTable class.**

- return type of asMaps() method is List<Map<String,String>>

- if we using as Maps() method then we have to use columns name compulsory

List<Map<String,String>> listmap = dataTable.asMaps();

String Uname = listmap.get(0).get("username");

String Pass = listmap.get(0).get("password");

e.g.

Feature: Test orange HRM application

Scenario: Validate the login functionality

Given user is login page

When user enter valid username and valid password

| username | password |

| Admin | admin123 |

Then user click on login button

@When("user enter valid username and valid password")

public void user\_enter\_valid\_username\_and\_valid\_password(DataTable dataTable)

{

List<Map<String,String> listMap = dataTable.asMaps();

String Uname=listMap.get(0).get("username");

String Pass = listMap.get(0).get("password");

}

Feature: Test the Orange HRM Application

Scenario: Validate the login Functionality

Given user is on login Page

When user enter valid username and valid password

| Admin | admin123 |

Then user click on login button

package StepDefinition;

import java.time.Duration;

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import io.cucumber.datatable.DataTable;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class LoginPageSteps {

static WebDriver driver;

@Given("user is on login Page")

public void user\_is\_on\_login\_page() {

driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

}

@When("user enter valid username and valid password")

public void user\_enter\_valid\_username\_and\_valid\_password(DataTable dataTable) {

List<List<String>> list = dataTable.asLists();

String UName = list.get(0).get(0);

System.out.println(UName);

String Pass = list.get(0).get(1);

System.out.println(Pass);

driver.findElement(By.name("username")).sendKeys(UName);

driver.findElement(By.name("password")).sendKeys(Pass);

}

@Then("user click on login button")

public void user\_click\_on\_login\_button() {

driver.findElement(By.xpath("//button[text()= ' Login ']")).click();

}

}

package StepDefinition;

import java.time.Duration;

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import io.cucumber.datatable.DataTable;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class LoginPageSteps {

static WebDriver driver;

@Given("user is on login Page")

public void user\_is\_on\_login\_page() {

driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

}

@When("user enter valid username and valid password")

public void user\_enter\_valid\_username\_and\_valid\_password(DataTable dataTable) {

List<List<String>> list = dataTable.cells();

String UName = list.get(0).get(0);

System.out.println(UName);

String Pass = list.get(0).get(1);

System.out.println(Pass);

driver.findElement(By.name("username")).sendKeys(UName);

driver.findElement(By.name("password")).sendKeys(Pass);

}

@Then("user click on login button")

public void user\_click\_on\_login\_button() {

driver.findElement(By.xpath("//button[text()= ' Login ']")).click();

}

}

Feature: Test the Orange HRM Application

Scenario: Validate the login Functionality

Given user is on login Page

When user enter valid username and valid password

| username | password |

| Admin | admin123 |

Then user click on login button

package StepDefinition;

import java.time.Duration;

import java.util.List;

import java.util.Map;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import io.cucumber.datatable.DataTable;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class LoginPageSteps {

static WebDriver driver;

@Given("user is on login Page")

public void user\_is\_on\_login\_page() {

driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

}

@When("user enter valid username and valid password")

public void user\_enter\_valid\_username\_and\_valid\_password(DataTable dataTable) {

List<Map<String, String>> listmap = dataTable.asMaps();

String UName = listmap.get(0).get("username");

String Pass = listmap.get(0).get("password");

driver.findElement(By.name("username")).sendKeys(UName);

driver.findElement(By.name("password")).sendKeys(Pass);

}

@Then("user click on login button")

public void user\_click\_on\_login\_button() {

driver.findElement(By.xpath("//button[text()= ' Login ']")).click();

}

}

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

Feature: Test Cogmento CRM Application

Scenario: Validate login page functionality

Given user open url in "chrome" browser

When user enter valid username and password

| prafulp1010@gmail.com | Pr@ful0812 |

Then user click on login button

Scenario: validate home page functionality

When user is on home page and validate title

And user validate home page url

Scenario: validate contact page functionality

When user click on contact link

And user click on create button

Then user enter first name , last name and select status

| Amruta | Patil | On Hold |

And user click on save button

Scenario: validate companies page functionality

When user click on companies link

And user click on create button

Then user enter name , website , address and email

| name | website | address | email |

| amdocs | www.amdocs.com | pune | contact@amdocs.com |

And user click on save button

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

@Given("user open a url in {string} browser")

public void user\_open\_a\_url\_in\_browser(String string)

{

}

@When("user enter valid username and password")

public void user\_enter\_valid\_username\_and\_password(Datatable dataTable)

{

}

Feature: Test Cogmento CRM Application Functionality

Scenario: validate login page functionality

Given user open a url in "headless" browser

When user enter valid username and password

| prafulp1010@gmail.com | Pr@ful0812 |

Then user click on login button

Scenario: validate home page Functionality

When user is on home page and validate title

And user validate url

Scenario: validate contact page functionality

When user click on contact link

And user click on create button

Then user enter firstname , last name and select status

| Amruta | Patil | inactive |

And user click on save button

Scenario: validate company page functionality

When user click on company link

And user click on company page create button

Then user enter name, website, address and email

| name | website | address | email |

| HSBC | www.hsbc.com | Pune | contact@hsbc.com |

And user click on save button

package StepDefinitions;

import java.time.Duration;

import java.util.List;

import java.util.Map;

import org.junit.jupiter.api.Assertions;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.openqa.selenium.edge.EdgeDriver;

import io.cucumber.datatable.DataTable;

import io.cucumber.java.AfterStep;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class EndToEndPageTestSteps {

static WebDriver driver;

@Given("user open a url in {string} browser")

public void user\_open\_a\_url\_in\_browser(String browsername) {

if (browsername.equalsIgnoreCase("chrome")) {

driver = new ChromeDriver();

} else if (browsername.equalsIgnoreCase("edge")) {

driver = new EdgeDriver();

} else if (browsername.equalsIgnoreCase("incognito")) {

driver = new ChromeDriver(new ChromeOptions().addArguments("--incognito"));

} else if (browsername.equalsIgnoreCase("headless")) {

ChromeOptions opt = new ChromeOptions();

opt.addArguments("--headless");

driver = new ChromeDriver(opt);

} else {

System.err.println("Please Enter valid browsername");

}

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

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

driver.get("https://ui.cogmento.com/");

}

@When("user enter valid username and password")

public void user\_enter\_valid\_username\_and\_password(DataTable dataTable) {

List<List<String>> list = dataTable.asLists();

String Uname = list.get(0).get(0);

String Pass = list.get(0).get(1);

driver.findElement(By.name("email")).sendKeys(Uname);

driver.findElement(By.name("password")).sendKeys(Pass);

}

@Then("user click on login button")

public void user\_click\_on\_login\_button() {

driver.findElement(By.xpath("//div[text()='Login']")).click();

}

@When("user is on home page and validate title")

public void user\_is\_on\_home\_page\_and\_validate\_title() {

String actualTitle = driver.getTitle();

Assertions.assertEquals(actualTitle, "Cogmento CRM");

}

@When("user validate url")

public void user\_validate\_url() {

String actualUrl = driver.getCurrentUrl();

Assertions.assertTrue(actualUrl.contains("cogmento"));

}

@When("user click on contact link")

public void user\_click\_on\_contact\_link() {

driver.findElement(By.xpath("//a[@href='/contacts']")).click();

}

@When("user click on create button")

public void user\_click\_on\_create\_button() {

driver.findElement(By.xpath("//a[@href='/contacts/new']")).click();

}

@Then("user enter firstname , last name and select status")

public void user\_enter\_firstname\_last\_name\_and\_select\_status(DataTable dataTable) {

List<List<String>> list = dataTable.cells();

String Fname = list.get(0).get(0);

String LName = list.get(0).get(1);

String status = list.get(0).get(2);

driver.findElement(By.name("first\_name")).sendKeys(Fname);

driver.findElement(By.name("last\_name")).sendKeys(LName);

driver.findElement(By.name("status")).click();

List<WebElement> statusList = driver.findElements(By.xpath("//div[@name='status']/child::span"));

for (WebElement value : statusList) {

String statusValue = value.getText();

if (statusValue.equalsIgnoreCase(status)) {

value.click();

break;

}

}

}

@Then("user click on save button")

public void user\_click\_on\_save\_button() {

driver.findElement(By.xpath("//button[text()='Save']")).click();

}

@When("user click on company link")

public void user\_click\_on\_company\_link() {

driver.findElement(By.xpath("//a[@href='/companies']")).click();

}

@When("user click on company page create button")

public void user\_click\_on\_company\_page\_create\_button() {

driver.findElement(By.xpath("//a[@href='/companies/new']")).click();

}

@Then("user enter name, website, address and email")

public void user\_enter\_name\_website\_address\_and\_email(DataTable dataTable) {

List<Map<String, String>> listMap = dataTable.asMaps();

String name = listMap.get(0).get("name");

String website = listMap.get(0).get("website");

String address = listMap.get(0).get("address");

String email = listMap.get(0).get("email");

driver.findElement(By.name("name")).sendKeys(name);

driver.findElement(By.name("url")).sendKeys(website);

driver.findElement(By.name("address")).sendKeys(address);

driver.findElement(By.name("value")).sendKeys(email);

}

// Hooks Annotations

@AfterStep

public void tearDown() throws InterruptedException {

Thread.sleep(3000);

}

}

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

Feature: Test Cogmento CRM Application Functionality

Scenario: validate login page functionality

Given user open a url in "headless" browser

When user enter valid username and password

| prafulp1010@gmail.com | Pr@ful0812 |

Then user click on login button

Scenario: validate home page Functionality

When user is on home page and validate title

And user validate url

Scenario Outline: validate contact page functionality

When user click on contact link

And user click on create button

Then user enter "<firstname>" , "<lastName>" and select "<status>"

And user click on save button

Examples:

| firstname | lastName | status |

| | | On Hold |

| | |inactive |

| | | active |

| | |terminate |

Scenario: validate company page functionality

When user click on company link

And user click on company page create button

Then user enter name, website, address and email

| name | website | address | email |

| HSBC | www.hsbc.com | Pune | contact@hsbc.com |

And user click on save button

Scenario: validate Deals page functionality

When user click on deals link

And user click on deals page create button

Then user enter title and user select date as 15 and month and year as "March 2025" and time as "08.45"

And user click on save button

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

**What is use of Background keyword in cucumber?**  VVVVVIMMPPPPPPPPP

- background is used to mention the precondition for each and every scenario in feature file.

Feature: Test Cogmento CRM Application Functionality

Background:

Given user open a url in "chrome" browser

When user enter valid username and password

| prafulp1010@gmail.com | Pr@ful0812 |

Then user click on login button

Scenario: validate home page Functionality

When user is on home page and validate title

And user validate url

Scenario: validate contact page functionality

When user click on contact link

And user click on create button

Then user enter firstname , last name and select status

| Amruta | Patil | inactive |

And user click on save button

Scenario: validate company page functionality

When user click on company link

And user click on company page create button

Then user enter name, website, address and email

| name | website | address | email |

| HSBC | www.hsbc.com | Pune | contact@hsbc.com |

And user click on save button

**What is use *Background keyword in cucumber?***

*- we use Backgound in feature* file.

- and it is used to mention pre condition for each and every scenario in feature file.

What is Hooks in cucumber?

- in cucumber Hooks annotation are used to mention the pre condition as well as post condition.

- we declare these Hooks annotation inside the Step Definition file.

- there are different types of Hooks in cucumber

1) Pre condition hooks

2) Post Condition hooks

1) Pre condition hooks

1) @BeforeAll

2) @Before

3) @BeforeStep

2) Post Condition hooks

1) @AfterAll

2) @After

3) @AfterStep

**1) @BeforeStep**

- it is precondition for each and every in feature.

- we declare @BeforeStep pre condition annotation in step definition files.

syntax:

@BeforeStep

public void setup()

{

//statement

}

**2) @AfterStep**

- it is post condition for each and every step in feature file.

- we declare @AfterStep post condition annotation in Step Definition file

syntax:

@AfterStep

public void teardown()

{

//statement

}

Feature: Test Sauce Demo application

Scenario: Validate Scenario1 all Functionality

Given Scenario1 and Step1

When Scenario1 and Step2

Then Scenario1 and Step3

package StepDefinition;

import io.cucumber.java.AfterStep;

import io.cucumber.java.BeforeStep;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class SauceDemoSteps {

@Given("Scenario1 and Step1")

public void scenario1\_and\_step1() {

System.out.println("Scenario1 and Step1");

}

@When("Scenario1 and Step2")

public void scenario1\_and\_step2() {

System.out.println("Scenario1 and Step2");

}

@Then("Scenario1 and Step3")

public void scenario1\_and\_step3() {

System.out.println("Scenario1 and Step3");

}

@BeforeStep

public void setUp()

{

System.out.println("Before Step Annotation");

}

@AfterStep

public void tearDown()

{

System.out.println("After Step Annotation");

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

}

}

package TestRunner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features="src/test/java/Features",

glue="StepDefinition"

)

public class RunnerTest {

}

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

**3) @Before pre condition Hooks annotation**

- it is precondition for each and every scenario in feature file

- we use @Before Hooks pre condition annotation inside the Step Definition file

syntax:

@Before

public void setup()

{

//statement

}

**4) @After post condition Hooks annotation**

- it is post condition for each and every scenario in feature file

- we declare @After hooks post condition annotation inside the Step Definition file

syntax:

@After

public void tearDown()

{

//statement

}

Feature: Test Sauce Demo application

Scenario: Validate Scenario\_One all Functionality

Given Scenario1 and Step1

When Scenario1 and Step2

Then Scenario1 and Step3

Scenario: Validate Scenario\_Two all Functionality

Given Scenario2 and Step1

When Scenario2 and Step2

Scenario: Validate Scenario\_Three all Functionality

Given Scenario3 and Step1

When Scenario3 and Step2

package StepDefinition;

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;

public class SauceDemoSteps {

@Given("Scenario1 and Step1")

public void scenario1\_and\_step1() {

System.out.println("Scenario1 and Step1");

}

@When("Scenario1 and Step2")

public void scenario1\_and\_step2() {

System.out.println("Scenario1 and Step2");

}

@Then("Scenario1 and Step3")

public void scenario1\_and\_step3() {

System.out.println("Scenario1 and Step3");

}

@Given("Scenario2 and Step1")

public void scenario2\_and\_step1() {

System.out.println("Scenario2 and Step1");

}

@When("Scenario2 and Step2")

public void scenario2\_and\_step2() {

System.out.println("Scenario2 and Step2");

}

@Given("Scenario3 and Step1")

public void scenario3\_and\_step1() {

System.out.println("Scenario3 and Step1");

}

@When("Scenario3 and Step2")

public void scenario3\_and\_step2() {

System.out.println("Scenario3 and Step2");

}

@Before

public void setUp() {

System.out.println("Before Hooks Annotation");

}

@After

public void tearDown() {

System.out.println("After Hooks annotation");

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

}

}

package TestRunner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features="src/test/java/Features",

glue="StepDefinition",

dryRun=false

)

public class RunnerTest {

}

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

**Note:**

\*\*\*\*\*\*\* these below hooks annotation present inside Cucumber above 6 version only

if u want to use these annotation we have to use static methods in hooks annotation level.

**5) @BeforeAll pre condition hooks annotation**

- it is pre condition for all the scenario in feature file

- we declare @BeforeAll pre condition hooks annotation inside the Step Definition file

syntax:

@BeforeAll

public static void setup()

{

//statement

}

6) @AfterAll post condition hooks annotation

- it is post condition for all scenarios in feature file

- we declare @AfterAll post condition hooks annotation inside the Step Definition file

syntax:

@AfterAll

public static void teardown()

{

//statement

}

Feature: Test Sauce Demo application

Scenario: Validate Scenario\_One all Functionality

Given Scenario1 and Step1

When Scenario1 and Step2

Then Scenario1 and Step3

Scenario: Validate Scenario\_Two all Functionality

Given Scenario2 and Step1

When Scenario2 and Step2

Scenario: Validate Scenario\_Three all Functionality

Given Scenario3 and Step1

When Scenario3 and Step2

package StepDefinition;

import io.cucumber.java.AfterAll;

import io.cucumber.java.BeforeAll;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class SauceDemoSteps {

@Given("Scenario1 and Step1")

public void scenario1\_and\_step1() {

System.out.println("Scenario1 and Step1");

}

@When("Scenario1 and Step2")

public void scenario1\_and\_step2() {

System.out.println("Scenario1 and Step2");

}

@Then("Scenario1 and Step3")

public void scenario1\_and\_step3() {

System.out.println("Scenario1 and Step3");

}

@Given("Scenario2 and Step1")

public void scenario2\_and\_step1() {

System.out.println("Scenario2 and Step1");

}

@When("Scenario2 and Step2")

public void scenario2\_and\_step2() {

System.out.println("Scenario2 and Step2");

}

@Given("Scenario3 and Step1")

public void scenario3\_and\_step1() {

System.out.println("Scenario3 and Step1");

}

@When("Scenario3 and Step2")

public void scenario3\_and\_step2() {

System.out.println("Scenario3 and Step2");

}

@BeforeAll

public static void setup() {

System.out.println("Before All Pre condition annotation");

}

@AfterAll

public static void tearDown() {

System.out.println("After All Pre condition annotation");

}

}

package TestRunner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features="src/test/java/Features",

glue="StepDefinition",

dryRun=false

)

public class RunnerTest {

}

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

**@BeforeAll**

- pre condition for all the scenario in feature file

**@AfterAll**

- post condition for all the scenario in feature file

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

**@Before**

- pre condition for each and every scenario in feature file

**@After**

- post condition for each and every scenario in feature file

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

**@BeforeStep**

- pre condition for each and every step in feature file

**@AfterStep**

- post condition for each and every step in feature file

package StepDefinition;

import io.cucumber.java.After;

import io.cucumber.java.AfterAll;

import io.cucumber.java.AfterStep;

import io.cucumber.java.Before;

import io.cucumber.java.BeforeAll;

import io.cucumber.java.BeforeStep;

public class Hooks {

@BeforeAll

public static void setup() {

System.out.println("Before All");

}

@Before

public void setup1() {

System.out.println("Before");

}

@BeforeStep

public void setUp2() {

System.out.println("Before Step");

}

@AfterStep

public void tearDown2() {

System.out.println("After Step");

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

}

@After

public void tearDown1() {

System.out.println("After");

}

@AfterAll

public static void tearDown() {

System.out.println("After All");

}

}

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

Note:

can we 2 times same pre condition

yes we can use and it execute pre condition as per the alphabetical order

can we use 2 times same post condition annotation?

yes we can use and it execute post condition as per the reverse alphabetical order

- if u want to maintain pre condition as well as post condition sequence then we use order attribute.

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

if we mention background keyword as well as Hooks annotation in cucumber project, then what is execution flow? **VVVVIMMMPPPPPPPPPPPPPPP**

- first priority goes to Hooks pre condition annotations then Backgound keyword.

- we declare hooks pre condition in Step definition file and

we declare Background keyword in feature file.

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

**diff between Background keyword and Hooks annotations?**  VVVVIMMMPPPPPPPPPPPPPPP

- we declare background keyword in feature file

- we declare hooks annotation in step definition file

- by using Background keyword we can mention only pre conditions

- by using Hooks annotations we can use pre condition as well as post conditions.

- at the time of executing the project, cucumber gives priority for hooks and then Background keyword.

-we can not use Background keyword 2 times in same feature file.,

- we can use Hooks annotation multiple times in Step Definition file

Feature: Test Sauce Demo application

Background:

Given backgroundkeywordstep1

Given backgroundkeywordstep2

Scenario: Validate Scenario\_One all Functionality

Given Scenario1 and Step1

When Scenario1 and Step2

Then Scenario1 and Step3

Scenario: Validate Scenario\_Two all Functionality

Given Scenario2 and Step1

When Scenario2 and Step2

Scenario: Validate Scenario\_Three all Functionality

Given Scenario3 and Step1

When Scenario3 and Step2

package StepDefinition;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class SauceDemoSteps {

@Given("Scenario1 and Step1")

public void scenario1\_and\_step1() {

System.out.println("Scenario1 and Step1");

}

@When("Scenario1 and Step2")

public void scenario1\_and\_step2() {

System.out.println("Scenario1 and Step2");

}

@Then("Scenario1 and Step3")

public void scenario1\_and\_step3() {

System.out.println("Scenario1 and Step3");

}

@Given("Scenario2 and Step1")

public void scenario2\_and\_step1() {

System.out.println("Scenario2 and Step1");

}

@When("Scenario2 and Step2")

public void scenario2\_and\_step2() {

System.out.println("Scenario2 and Step2");

}

@Given("Scenario3 and Step1")

public void scenario3\_and\_step1() {

System.out.println("Scenario3 and Step1");

}

@When("Scenario3 and Step2")

public void scenario3\_and\_step2() {

System.out.println("Scenario3 and Step2");

}

@Given("backgroundkeywordstep1")

public void backgroundkeywordstep1() {

System.out.println("background keyword step1");

}

@Given("backgroundkeywordstep2")

public void backgroundkeywordstep2() {

System.out.println("background keyword step 2");

}

}

package TestRunner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features="src/test/java/Features",

glue="StepDefinition",

dryRun=false

)

public class RunnerTest {

}

package StepDefinition;

import io.cucumber.java.After;

import io.cucumber.java.Before;

public class Hooks {

@Before

public void setUp() {

System.out.println("Before Hooks");

}

@After

public void tearDown() {

System.out.println("After Hooks");

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

}

}

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

**How to configure Cucumber with TestNG? VVVVVIMMMPPPPP**

- we have to add cucumber-testng maven dependency as well as testng maven dependency in pom xml file.

- and we have to remove the @RunWith() annotation and

we have to extends the AbstractTestNGCucumberTests class in Test Runner class.

package TestRunner;

import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

@CucumberOptions(

features="src/test/java/Features",

glue="StepDefinition",

dryRun=false

)

public class RunnerTest extends AbstractTestNGCucumberTests{

}

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>7.14.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.4.0</version>

<scope>test</scope>

</dependency>

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

What the different annotation present in cucumber VVVVVVIMMMPPPPPPPPP

1) @RunWith()

2) @CucumerOptions()

3) @BeforeAll

4) @AfterAll

5) @Before

6) @After

7) @BeforeStep

8) @AfterStep

9) @Given

10) @When

11) @Then

What are the different keyword present in feature file VVVVVVIMMMPPPPPPPPP

1) Feature:

2) Scenario: or Example:

3) Scenario Outline: or Scenario Template:

4) Examples: or Scenarios:

5) Background:

6) Given

7) When

8) Then

9) And

10) But

11) \*

What are the different keyword present in Test Runner class VVVVVVIMMMPPPPPPPPP

1) @RunWith()

2) @CucumberOptions()

3) features

4) glue or extraGlue

5) dryRun

6) monochrome

7) plugin

8) tags

<!--

https://mvnrepository.com/artifact/tech.grasshopper/extentreports-cucumber7-adapter -->

<dependency>

<groupId>tech.grasshopper</groupId>

<artifactId>extentreports-cucumber7-adapter</artifactId>

<version>1.7.0</version>

</dependency>

extent.properties inside the src/test/resources package

extent.reporter.spark.start=true

extent.reporter.spark.out=reports/abc.html

extent.reporter.pdf.start=true

extent.reporter.pdf.out=reports/abc.pdf

package TestRunner;

import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

@CucumberOptions(

features="src/test/java/Features",

glue="StepDefinition",

dryRun=false,

plugin= {"com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"}

)

public class RunnerTest extends AbstractTestNGCucumberTests{

}

<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>5.0.9</version>

</dependency>

<!--

https://mvnrepository.com/artifact/tech.grasshopper/extentreports-cucumber7-adapter -->

<dependency>

<groupId>tech.grasshopper</groupId>

<artifactId>extentreports-cucumber7-adapter</artifactId>

<version>1.7.0</version>

</dependency>

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

**Configure Extent Reports with Cucumber**

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

- Extent Adapter Reports is standard reporting library to generate the reports in cucumber BBD Framework.

- By using Extent Adapter Reports we can generate HTML reports as well as we can attach the screenshot in reports.

- To generate the Extent Reports we add extent-adapter dependency inside the pom.xml file as per the cucumber version.

- and then we create extent.properties file inside the src/test/resources package and we do configuration in properties file and we mention reports plugin name inside the test runner class.

**1) Create extent.properties file inside the src/test/resources**

step 1: if we want to start generating extent reports use extent.reporter.spark.start key and value as true

syntax:

extent.reporter.spark.start=true

step 2: if we want to store generated reports in destination location then use extent.reporter.spark.out as key and value as location

syntax;

extent.reporter.spark.out=ExtenReports/abc.html

**2) Inside the Test runner class and @CucumberOptions annotations we use plugin keyword and** inside the plugin we pass value as com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:

@RunWith(Cucumber.class)

@CucumberOptions

(

features="src/test/java/Features",

glue="StepDefinitions",

plugin="com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"

)

public class RunnerTest

{

}

extent.properties file in src/test/resources

1. store extent reports in base folder

basefolder.name=ExtentReports/

2. crate current date and time folder inside the ExtentReports folder

basefolder.datetimepatter = ddMMyyyy\_HHmmss

3. start generating Extent HTML reports

extent.reporter.spark.start=true

4. stop generating reports

extent.reporter.spark.out=/xyz.html

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

Extent PDF Reports

5. start generating pdf reports

extent.reporter.pdf.start=true

6. store generated pdf reports in destination location

extent.reporter.pdf.out=/abc.pdf

7. Mention project name in reports by using systeminfo.AppName= application name

systeminfo.AppName=Orange HRM Application

8. mention build name or release name inside the reports

systeminfo.build= Release 20.0

9. mention the operating system name

systeminfo.os=Window 11 OS

10. Mention Tester name

systeminfo.User=Aboli Shinde

11. create screenshot folder

screenshot.dir=/Screenshot/

12. store all screenshot from path

screenshot.rel.path=../Screenshot/

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

Take screenshot as per the scenario wise and attach screenshot to reports

- to take the takes the screenshot we use hooks annotations

@After

public static void teardown(Scenario scenario)

{

if(scenario.isFailed())

{

TakesScreenshot ts = (TakesScreenshot)driver;

byte [] b = ts.getScreenshotAs(OutputType.BYTES);

scenario.attach(b, "image/png","Screenshot name for failed Login Page Scenario")

}

else

{

TakesScreenshot ts = (TakesScreenshot)driver;

byte [] b = ts.getScreenshotAs(OutputType.BYTES);

scenario.attach(b, "image/png","Screenshot name for Passed Login Page Scenario");

}

}

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

basefolder.name=ExtentReports/

basefolder.datetimepattern=ddMMyyyy\_HHmmss

extent.reporter.spark.start=true

extent.reporter.spark.out=/extenteports.html

extent.reporter.pdf.start=true

extent.reporter.pdf.out=/extentreports.pdf

systeminfo.AppName=Orange HRM Application

systeminfo.Build=Release 20.0

systeminfo.OS=Window 11 64 bit

systeminfo.user=Aboli Shinde

screenshot.dir=Screenshot/

screenshot.rel.path=../Screenshot/

package StepDefinitions;

import java.io.File;

import java.text.SimpleDateFormat;

import java.time.Duration;

import java.util.Date;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.By;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import io.cucumber.java.AfterStep;

import io.cucumber.java.Scenario;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class PIMPageStepsTest {

static WebDriver driver;

@Given("user is on login Page")

public void user\_is\_on\_login\_page() {

driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

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

driver.manage().deleteAllCookies();

driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

}

@When("user enter valid username and password")

public void user\_enter\_valid\_username\_and\_password() {

driver.findElement(By.name("username")).sendKeys("Admin");

driver.findElement(By.name("password")).sendKeys("admin123");

}

@Then("user click on login button")

public void user\_click\_on\_login\_button() {

driver.findElement(By.xpath("//button[text()=' Login ']")).click();

}

@AfterStep

public static void tearDown(Scenario scenario) throws Exception {

Thread.sleep(4000);

if (scenario.isFailed()) {

TakesScreenshot ts = (TakesScreenshot) driver;

byte[] b = ts.getScreenshotAs(OutputType.BYTES);

scenario.attach(b, "image/png", "Login Scenario Failed Name");

File src = ts.getScreenshotAs(OutputType.FILE);

String date = new SimpleDateFormat("ddMMyyyy\_HHmmss").format(new Date());

FileUtils.copyFile(src, new File(System.getProperty("user.dir") + "//screenshot//" + date + ".png"));

} else {

TakesScreenshot ts = (TakesScreenshot) driver;

byte[] b = ts.getScreenshotAs(OutputType.BYTES);

scenario.attach(b, "image/png", "Login Scenario Passed Name");

File src = ts.getScreenshotAs(OutputType.FILE);

String date = new SimpleDateFormat("ddMMyyyy\_HHmmss").format(new Date());

FileUtils.copyFile(src, new File(System.getProperty("user.dir") + "//screenshot//" + date + ".png"));

}

}

}

**How to run parallel feature in cucumber with Junit?**  VVVVVVIMMPPPPPPPPPPPPP

- we can run feature file parallel by using maven surefire plugin or failsafe plugin.

- declare browser name parameter in feature file.

- same browsername parameter inside the snippets.

- and write selenium script as per the scenarios

- then inside the pom.xml file

add build tag name

and inside the build tag name add plugins tag.

- inside the plugins tag add plugin tag.

- then inside the plugin tag add maven surefire plugin or fail safe plugin.

- then inside plugin tag add configuration plugin and then add includes tags and then include tag and pass the test Runner class name.

- then close include and includes tag.

- then add parallel tagname and pass methods as value and close parallel tag.

- and then add threadCount tagname and pass the thread count number.

pom.xml

<build>

<plugins>

<plugin>

<artifactid> maven surefire plugin artifactid </artifactid>

<groupid> maven surefire plugin groupid </groupid>

<version> maven surefire plugin version </version>

<configuration>

<includes>

<include> ../TestRunner.RunnerTest.java </include>

</includes>

<parallel> methods </parallel>

<threadCount> 4 </threadCount>

</configuration>

<plugin>

<plugins>

<build>

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

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

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

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.orangehrm</groupId>

<artifactId>OrangeHRMApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>OrangeHRMApplication</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<!--

1) Cucumber java dependency

2) cucumber core dependency

3) cucumber-junit dependency

4) Junit dependency / Junit Jupiter dependency

5) cucumber-testNG dependency

6) TestNG dependency

7) common io Dependency

8) apache POI Dependency

9) Ashot maven dependency

10) Extent reports dependency

11) Extent Cucumber Adapter dependency

12) maven surefire plugin

13) Selenium Maven Dependency

-->

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-core -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-core</artifactId>

<version>6.11.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>6.11.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>6.11.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/junit/junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</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>4.16.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M3</version>

<configuration>

<includes>

<include>../TestRunner.RunnerTest.java</include>

</includes>

<parallel>methods</parallel>

<threadCount>4</threadCount>

</configuration>

</plugin>

</plugins>

</build>

</project>

**How to run parallel feature in cucumber with TestNG?**  VVVVVVIMMPPPPPPPPPPPPP

- we can run parallel feature in cucumber with TestNG by using maven sure fire plugin or maven fail safe plugins.

- but we have to run feature file parallel, then we have create separate Runner classes for each and every feature files

- and then we create xml file for both the Test runner classes and we use parallel=methods in suite level annotations in testng.xml file.

<suite name="suite name" parallel="methods">

<test name="test case name" >

<classes>

<class name="TestRunner.RunnerTest.java"/>

<class name="TestRunner.RunTest.java"/>

</classes>

</test>

</suite>

- declare browser name parameter in feature file.

- use same browsername parameter inside the snippets.

- and write selenium script as per the scenarios

- then inside the pom.xml file

add build tag name

and inside the build tag name add plugins tag.

- inside the plugins tag add plugin tag.

- then inside the plugin tag add maven surefire plugin or fail safe plugin.

- then inside plugin tag add configuration plugin and inside the configuration tag add suiteXmlFiles tag and

then inside the suiteXmlFiles tag add suiteXml file tag by passing test.xml file name.

- then add parallel tagname and pass methods as keyword and close parallel tag.

- and then add threadCount tagname and pass the thread count number.

pom.xml

<build>

<plugins>

<plugin>

<artifactid> maven surefire plugin artifactid </artifactid>

<groupid> maven surefire plugin groupid </groupid>

<version> maven surefire plugin version </version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile> testng.xml </suiteXmlFile>

<suiteXmlFiles>

<parallel> methods </parallel>

<threadCount> 4 </threadCount>

</configuration>

<plugin>

<plugins>

<build>

<suite name="suite name" parallel="methods">

<test name="test case name" >

<classes>

<class name="TestRunner.RunnerTest.java"/>

<class name="TestRunner.RunTest.java"/>

</classes>

</test>

</suite>

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

<suite name="suite name" parallel="methods">

<test name="test case name 1" >

<classes>

<class name="TestRunner.RunnerTest.java"/>

</classes>

</test>

<test name="test case name 2" >

<classes>

<class name="TestRunner.RunTest.java"/>

</classes>

</test>

</suite>

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

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

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

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

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

<classes>

<class name="TestRunner.PIMPageRunnerTest"></class>

<class name="TestRunner.AdminPageRunnerTest"></class>

</classes>

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

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

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

**How to run scenario parallel in cucumber**  VVVVVVVIMMMPPPPPPPP

- we run scenario parallel in cucumber by overriding the scenarios() method from AbstractTestNGCucumberTests class,

- and we override these method in Test Runner class.

and after overriding we use @DataProvider annotation and we use parallel =true attribute in data provider annotation

package TestRunner;

import org.testng.annotations.DataProvider;

import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

@CucumberOptions(

features="src/test/java/Features",

glue="Steps"

)

public class RunnerTest extends AbstractTestNGCucumberTests{

@Override

@DataProvider(parallel = true)

public Object[][] scenarios() {

// TODO Auto-generated method stub

return super.scenarios();

}

}

**how to read data from Excel sheet in cucumber BDD Framework?**  **VVVVVVIMMMPPPPP**

- if we have large set of test data we can't store these test data in features files.

- we store these test data in Excel sheet and then we read it as per the requirements wise.

- Read the test data we use apache poi dependency.

e.g.

//step 1: read the file content

FileInputStream fis = FileInputStream("file path");

//step 2: load all excel sheets

XSSFWorkbook workbook =new XSSFWorkbook(fis);

//step 3: focus on specific sheet by using sheet index

XSSFSheet sheets = workbook.getSheetAt(0);

//step 4: capture total numbers of rows present inside the sheet

int rowcount = sheets.getLastRowNum();

//step 5: create object of List<Map<String, String> or ArrayList<Map<String, String> --> we store test data as per the row wise

List<Map<String, String> excelRowTestData = new ArrayList<Map<String, String>>();

//step 6: iterate all rows from excel sheet

for(int i=1; i<=rowcount; i++)

{

//step 7: count total columns in excel sheet

int cellCount = sheets.getRow(0).getLastCellNum();

//step 8: create object of LinkedHashMap<String, String> ---> we store the test data as per column wise

LinkedHashMap<String, String> excelColumnTestData = new LinkedHashMap<String, String>();

//step 9: iterate all the columns value

for(int j=0; j<cellCount; j++)

{

//step 10: capture column heading value

String columnHeadingName = sheets.getRow(0).getCell(j).getStringCellValue();

//step 11: capture column cell values

String cellValue = sheets.getRow(i).getCell(j).getStringCellValue();

//step 12: add values inside the LinkedHashMap by using put() method

excelColumnTestData.put(columnHeadingName, cellValue);

}

//step 13: add values inside the List<Map<String, String>

excelRowTestData.add(excelColumnTestData);

}

//capture the values from List<Map<String, String> object

String a = excelRowTestData.get(2).get("FirstName");

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

package Tutorial1;

import java.io.File;

import java.io.FileInputStream;

import java.util.ArrayList;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Demo1 {

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

// check Excel file is present or not

File f = new File("C:\\Users\\praf0\\OneDrive\\Desktop\\TestData.xlsx");

// step 2: read all file content

FileInputStream fis = new FileInputStream(f);

//step 3: load all Excel Sheets

XSSFWorkbook workbook =new XSSFWorkbook(fis);

//step4 : focus on specific sheet using sheet index or sheet name

XSSFSheet sheets = workbook.getSheet("PersonalInfo");

//step 5: count total rows in excel sheet

int rowCount = sheets.getLastRowNum();

//step 6: create object of List<Map<String, String>>

List<Map<String, String>> excelRowTestData = new ArrayList<Map<String,String>>();

//step 7: iterate all rows from Excel sheet

for(int i=1;i<=rowCount;i++)

{

//step 8: count total columns in excel sheet

int cellCount = sheets.getRow(0).getLastCellNum();

//step 9: create object of LinkedHashMap<String, String>>

LinkedHashMap<String,String> columnTestData = new LinkedHashMap<String,String>();

//step 10: iterate the all columns

for(int j=0;j<cellCount;j++)

{

//step 11: capture the column name or heading name

String collumnHeading = sheets.getRow(0).getCell(j).getStringCellValue();

//step 12: capture the column cell values

String cellValue = sheets.getRow(i).getCell(j).getStringCellValue();

//step 13: put values inside the LinkedHashMap object

columnTestData.put(collumnHeading, cellValue);

}

//step 14: add columns inside the List<Map<String, String>> object

excelRowTestData.add(columnTestData);

}

///capture the values from Excel sheet /List<Map<String,String>>

String a = excelRowTestData.get(3).get("LastName");

System.out.println(a);

}

}

Feature: Test Facebook Application

Scenario Outline: Validate the Register Page Functionality

Given user is on register page

When user enter firstname and lastname from "<ExcelSheetName>" and <rownumber>

Examples:

| ExcelSheetName | rownumber |

| PersonalInfo | 0 |

| PersonalInfo | 1 |

| PersonalInfo | 2 |

| PersonalInfo | 3 |

package Steps;

import java.time.Duration;

import java.util.List;

import java.util.Map;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import UitlityLayer.ExcelReader;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.When;

public class RegisterSteps {

static WebDriver driver;

String ExcelSheetPath = "C:\\Users\\praf0\\OneDrive\\Desktop\\TestData.xlsx";

@Given("user is on register page")

public void user\_is\_on\_register\_page() {

driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

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

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

}

@When("user enter firstname and lastname from {string} and {int}")

public void user\_enter\_firstname\_and\_lastname\_from\_and(String ExcelSheetName, Integer rownumber)

throws InterruptedException {

ExcelReader excel = new ExcelReader();

List<Map<String, String>> testData = excel.getAllSheetTestData(ExcelSheetPath, ExcelSheetName);

String fname = testData.get(rownumber).get("FirstName");

String lname = testData.get(rownumber).get("LastName");

String uname = testData.get(rownumber).get("username");

String pass = testData.get(rownumber).get("password");

driver.findElement(By.name("firstname")).sendKeys(fname);

driver.findElement(By.name("lastname")).sendKeys(lname);

driver.findElement(By.name("reg\_email\_\_")).sendKeys(uname);

driver.findElement(By.name("reg\_passwd\_\_")).sendKeys(pass);

Thread.sleep(7000);

driver.quit();

}

}

package UitlityLayer;

import java.io.FileInputStream;

import java.util.ArrayList;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelReader {

static XSSFWorkbook workbook;

static XSSFSheet sheets;

public List<Map<String, String>> getAllSheetTestData(String excelSheetPath, String sheetName) {

try {

// read all file content

FileInputStream fis = new FileInputStream(excelSheetPath);

// load all excel sheet

workbook = new XSSFWorkbook(fis);

} catch (Exception e) {

e.printStackTrace();

}

// focus on specific sheet

XSSFSheet sheets = workbook.getSheet(sheetName);

// capture total rows in sheet

int rowCount = sheets.getLastRowNum();

// create object of List<Map<String, String>.

List<Map<String, String>> rowTestData = new ArrayList<Map<String, String>>();

// iterate all rows

for (int i = 1; i <= rowCount; i++) {

// count columns

int cellCount = sheets.getRow(0).getLastCellNum();

// create object of LinkedHashMap<String, String>>

LinkedHashMap<String, String> columnTestData = new LinkedHashMap<String, String>();

// iterate columns value

for (int j = 0; j < cellCount; j++) {

// capture the column heading or headers value

String columnHeading = sheets.getRow(0).getCell(j).getStringCellValue();

// capture ghe column cell value

String cellValue = sheets.getRow(i).getCell(j).getStringCellValue();

// put all value inside the LinkedHashMap object

columnTestData.put(columnHeading, cellValue);

}

// add value inside the List<Map<String, String> object

rowTestData.add(columnTestData);

}

return rowTestData;

}

}

package TestRunner;

import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

@CucumberOptions

(

features="src/test/java/Features",

glue="Steps")

public class RunnerTest extends AbstractTestNGCucumberTests{

}

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

**BDD Framework with Page Object Model Design Pattern**

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

package BaseLayer;

import java.time.Duration;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class BaseClass {

protected static WebDriver driver;

public static void initialization() {

// connect to actual browser

driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));

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

driver.manage().deleteAllCookies();

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

}

}

package PageLayer;

import java.util.List;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.FindBys;

import org.openqa.selenium.support.PageFactory;

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

import BaseLayer.BaseClass;

public class RegisterPage extends BaseClass {

// Create Object Repository using Page Object model with Page Factory

@FindBy(name = "firstname")

private WebElement firstname;

@FindBy(name = "lastname")

private WebElement lastname;

@FindBy(name = "reg\_email\_\_")

private WebElement username;

@FindBy(name = "reg\_passwd\_\_")

private WebElement password;

@FindBy(name = "birthday\_day")

private WebElement date;

@FindBy(name = "birthday\_month")

private WebElement month;

@FindBy(name = "birthday\_year")

private WebElement year;

@FindBys(@FindBy(xpath = "//div[text()='Gender']/following::label[@class='\_58mt']"))

private List<WebElement> listGender;

@FindBy(name = "websubmit")

private WebElement submit;

// initialize the object repository using PageFactory.initElements() method in

// constructor

public RegisterPage() {

PageFactory.initElements(driver, this);

}

// create associated method for each and every object repository without

// entering test data and as per the feature file step wise

public void enterFirstNameAndLastName(String FirstName, String LastName) {

firstname.sendKeys(FirstName);

lastname.sendKeys(LastName);

}

public void enterUsernameAndPassword(String Username, String Password) {

username.sendKeys(Username);

password.sendKeys(Password);

}

public void selectDateAndMonthAndYear(String Date, String Month, String Year) {

Select selDate = new Select(date);

selDate.selectByVisibleText(Date);

Select selMonth = new Select(month);

selMonth.selectByVisibleText(Month);

Select selYear = new Select(year);

selYear.selectByVisibleText(Year);

}

public void selectGender(String genderName) {

// iterate all values from list

for (WebElement abc : listGender) {

// capture the gender physical text or value

String value = abc.getText();

// check passing gender test data is equal to physical test data

if (value.equalsIgnoreCase(genderName)) {

// click on gender

abc.click();

break;

}

}

}

public void clickOnRegisterButton() {

submit.click();

}

}

Feature: Test Facebook Application

@RegisterPage

Scenario: validate the Register Page Functionality

Given user open register page url browser

When user enter first name and last name

And user enter username and password

And user select date, month and year

And user select gender

Then user click on register button

package Steps;

import BaseLayer.BaseClass;

import PageLayer.RegisterPage;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class RegisterPageSteps extends BaseClass {

private static RegisterPage registerPage;

@Given("user open register page url browser")

public void user\_open\_register\_page\_url\_browser() {

BaseClass.initialization();

}

@When("user enter first name and last name")

public void user\_enter\_first\_name\_and\_last\_name() {

registerPage = new RegisterPage();

registerPage.enterFirstNameAndLastName("Paven", "Shah");

}

@When("user enter username and password")

public void user\_enter\_username\_and\_password() {

registerPage.enterUsernameAndPassword("pavan@db.com", "Anjali@123");

}

@When("user select date, month and year")

public void user\_select\_date\_month\_and\_year() {

registerPage.selectDateAndMonthAndYear("15", "Jan", "2007");

}

@When("user select gender")

public void user\_select\_gender() {

registerPage.selectGender("MALE");

}

@Then("user click on register button")

public void user\_click\_on\_register\_button() {

// registerPage.clickOnRegisterButton();

}

}

package TestRunner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions

(

features="src/test/java/Features/TS\_001\_FacebookApplication\_RegisterPage.feature",

glue="Steps",

dryRun=false,

monochrome=true,

tags="@RegisterPage"

)

public class RunnerTest

{

}

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

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

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.facebook</groupId>

<artifactId>FacebookApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>FacebookApplication</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<!--

1) Cucumber java dependency

2) cucumber core dependency

3) cucumber-junit dependency

4) Junit dependency / Junit Jupiter dependency

5) cucumber-testNG dependency

6) TestNG dependency

7) common io Dependency

8) apache POI Dependency

9) Ashot maven dependency

10) Extent reports dependency

11) Extent Cucumber Adapter dependency

12) maven surefire plugin

13) Selenium Maven Dependency

-->

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>6.11.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>6.11.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-core -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-core</artifactId>

<version>6.11.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/junit/junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</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>4.16.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>maven-surefire-plugin</artifactId>

<version>3.3.0</version>

<configuration>

<includes>

<include>\*\*\RunnerTest.java</include>

</includes>

</configuration>

</plugin>

</plugins>

</build>

</project>

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

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

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.facebook</groupId>

<artifactId>FacebookApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>FacebookApplication</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<!--

1) Cucumber java dependency

2) cucumber core dependency

3) cucumber-junit dependency

4) Junit dependency / Junit Jupiter dependency

5) cucumber-testNG dependency

6) TestNG dependency

7) common io Dependency

8) apache POI Dependency

9) Ashot maven dependency

10) Extent reports dependency

11) Extent Cucumber Adapter dependency

12) maven surefire plugin

13) Selenium Maven Dependency

-->

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>6.11.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-core -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-core</artifactId>

<version>6.11.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>6.11.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>6.11.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.4.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/junit/junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</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>4.16.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>maven-surefire-plugin</artifactId>

<version>3.3.0</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</build>

</project>

<?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">

<classes>

<class name="TestRunner.RunnerTest"></class>

</classes>

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

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

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

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

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.facebook</groupId>

<artifactId>FacebookApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>FacebookApplication</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<!--

1) Cucumber java dependency

2) cucumber core dependency

3) cucumber-junit dependency

4) Junit dependency / Junit Jupiter dependency

5) cucumber-testNG dependency

6) TestNG dependency

7) common io Dependency

8) apache POI Dependency

9) Ashot maven dependency

10) Extent reports dependency

11) Extent Cucumber Adapter dependency

12) maven surefire plugin

13) Selenium Maven Dependency

-->

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>6.11.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-core -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-core</artifactId>

<version>6.11.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>6.11.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>6.11.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.4.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/junit/junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</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>4.16.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>maven-surefire-plugin</artifactId>

<version>3.3.0</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</build>

</project>

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