Cucumber Framework

-Cucumber is a Behavior Driven Development (BDD) automation testing tool which is used to write automation testing steps for each behavior/functionality using gherkin langauge.

-It offers a way to write tests that anybody can understand, regardless of their technical knowledge.

The steps to create the cucumber project:

1)Add the cucumber eclipse plug in for eclipse IDE.

2)Create the Maven project

3)Add the depedencies in pom.xml

```
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-core -->
    <dependency>
      <groupId>io.cucumber</groupId>
      <artifactId>cucumber-core</artifactId>
      <version>4.4.0</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-html -->
    <dependency>
      <groupId>io.cucumber</groupId>
      <artifactId>cucumber-html</artifactId>
      <version>0.2.7</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/net.sourceforge.cobertura/cobertura -->
    <dependency>
      <groupId>net.sourceforge.cobertura</groupId>
      <artifactId>cobertura</artifactId>
      <version>2.1.1</version>
      <scope>test</scope>
    </dependency>

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

```
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>4.4.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>4.4.0</version>
    <scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-jvm-deps -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-jvm-deps</artifactId>
    <version>1.0.6</version>
    <scope>provided</scope>
</dependency>


<!-- https://mvnrepository.com/artifact/net.masterthought/cucumber-reporting -->
<dependency>
    <groupId>net.masterthought</groupId>
    <artifactId>cucumber-reporting</artifactId>
    <version>4.7.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.hamcrest/hamcrest-core -->
<dependency>
    <groupId>org.hamcrest</groupId>
    <artifactId>hamcrest-core</artifactId>
```

```
    <version>2.1</version>
    <scope>test</scope>
</dependency>


<!-- https://mvnrepository.com/artifact/io.cucumber/gherkin -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>gherkin</artifactId>
    <version>5.1.0</version>
</dependency>


<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
</dependency>

<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>

<dependency>
                        <groupId>com.sun</groupId>
                        <artifactId>tools</artifactId>
                        <version>1.8</version>
                        <scope>system</scope>
```

<systemPath>C:\Program Files\Java\jdk1.8.0_321\lib\tools.jar</systemPath>

</dependency>

<!-- https://mvnrepository.com/artifact/log4j/log4j -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

</dependencies>

4) Create the folder structure:

-pageObjects

-steps

-testRun

-Features

-drivers

4)Create the feature files under Features folder

Login.feature

Feature: Login

@sanity

Scenario: Successful Login with Valid Credentials

Given User Launch Chrome browser

When User opens URL "http://admin-demo.nopcommerce.com/login"

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

And Click on Login

Then Page Title should be "Dashboard / nopCommerce administration"

And close browser

@regression

```
Scenario Outline: Login Data Driven
        Given User Launch Chrome browser
        When User opens URL "http://admin-demo.nopcommerce.com/login"
        And User enters Email as "<email>" and Password as "<password>"
        And Click on Login
        Then Page Title should be "Dashboard / nopCommerce administration"
        When User click on Log out link
        Then Page Title should be "Your store. Login"
        And close browser

        Examples:
                | email | password |
                | admin@yourstore.com          |          admin |
                | admin1@yourstore.com | admin123   |
```

5) Create the step definition file:

-We can auto generate the step definition or create it manually using the cucuber annotations.

```
public class Stepdef
{
        @Before
        public void setup() throws IOException
        {

        System.setProperty("webdriver.gecko.driver",configProp.getProperty("firefoxpath"));
                        driver = new FirefoxDriver();
        }

        //Login steps.....................
        @Given("User Launch Chrome browser")
        public void user_Launch_Chrome_browser() {
                logger.info("************* Launching Browser *****************");
```

```
        lp=new LoginPage(driver);
}

@When("User opens URL {string}")
public void user_opens_URL(String url) {
logger.info("************* Opening URL  *****************");
driver.get(url);
 driver.manage().window().maximize();
}

@When("User enters Email as {string} and Password as {string}")
public void user_enters_Email_as_and_Password_as(String email, String password) {
        logger.info("************* Prvding user and password *****************");
        lp.setUserName(email);
        lp.setPassword(password);
}

@When("Click on Login")
public void click_on_Login() {
        logger.info("************* click on login *****************");
  lp.clickLogin();
}

@Then("Page Title should be {string}")
public void page_Title_should_be(String exptitle) throws InterruptedException {

        if(driver.getPageSource().contains("Login was unsuccessful"))
        {
                logger.info("************* Login failed *****************");
                driver.close();
                Assert.assertTrue(false);
        }
        else
```

```
                {
                        logger.info("************* Login Passed *****************");
                        Assert.assertEquals(exptitle, driver.getTitle());
                }
                Thread.sleep(3000);

        }


        @Then("close browser")
        public void close_browser() {
                logger.info("************** cloding browser *****************");
          driver.quit();
        }


}
```

6)Create the TestRunner class:

-The test runner class will start the running the step definitions.

```
@RunWith(Cucumber.class)
@CucumberOptions(
                features=".//Features//",
                glue="stepDefinitions",
                monochrome=true,
                tags= {"@sanity"},
                plugin= {"pretty","html:test-output"}
                )
public class TestRunner {
}
```

7) Run the cucumber test

-Right click on TestRunner.java ->Run as Junit