A yellow and white text

Description automatically generated

Getting from excel is not recommended for bdd framework. Actual purpose of bdd is without excel and stuff we can use the examples and data tables etc.

Gitlab code for Naveen – cucumber pom series.

[GitHub - naveenanimation20/LatestCucumber6WithPOM: Latest Cucumber6 BDD framework with POM(Page Object Model)](https://github.com/naveenanimation20/LatestCucumber6WithPOM)

You can download the Cucumber Eclipse Plugin:

[Cucumber-Eclipse > Update Site](https://cucumber.github.io/cucumber-eclipse/update-site/)

Cucumber Java Skeleton Repo::

[GitHub - cucumber/cucumber-java-skeleton: This is the simplest possible setup for Cucumber-JVM using Java.](https://github.com/cucumber/cucumber-java-skeleton)

Latest eclipse plugin url:

[GitHub - cucumber/cucumber-eclipse-update-site-snapshot: Cucumber Eclipse Update Site Snapshots](https://github.com/cucumber/cucumber-eclipse-update-site-snapshot)

Full cucumber code repo-

[GitHub - naveenanimation20/Cucumber6LatestFeatures: Cucumber version 6 Latest Features](https://github.com/naveenanimation20/Cucumber6LatestFeatures)

Full cucumber code repo-

[GitHub - naveenanimation20/Cucumber6LatestFeatures: Cucumber version 6 Latest Features](https://github.com/naveenanimation20/Cucumber6LatestFeatures)

Full cucumber code repo-

[GitHub - naveenanimation20/Cucumber6LatestFeatures: Cucumber version 6 Latest Features](https://github.com/naveenanimation20/Cucumber6LatestFeatures)

Full code repo-

[GitHub - naveenanimation20/LatestCucumber6WithPOM: Latest Cucumber6 BDD framework with POM(Page Object Model)](https://github.com/naveenanimation20/LatestCucumber6WithPOM)

Full code repo-

[GitHub - naveenanimation20/LatestCucumber6WithPOM: Latest Cucumber6 BDD framework with POM(Page Object Model)](https://github.com/naveenanimation20/LatestCucumber6WithPOM)

# Excel sheet prepared –

A screenshot of a computer

Description automatically generated

This is the page which we will automate-

A screenshot of a computer

Description automatically generated

# Excel reader class-

You can use anywhere. Its already built with all methods of excel. When time permits go through the methods.

This has been created by Naveen for working with excel.

A screenshot of a computer

Description automatically generated

# We need to use apache poi api to work with excel-

Google search for it.

A screenshot of a computer

Description automatically generated

Click on the mvn link.

Select the latest version and paste in dependency.

A screenshot of a computer

Description automatically generated

# Add in pom file-

A computer code with text

Description automatically generated

Another dependency we need to add-

Same version as poi should be added for this dependency.

A screenshot of a web page

Description automatically generated

Select latest one and paste in pom.

A screenshot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

# How to write the css selector for the success message-

A screenshot of a contact us

Description automatically generated

Div#center\_column p – <tagName#id> <childTag>

Tomorrow if you add another sheet for new page or application or scenario .. just add the sheet names as below-

A screenshot of a computer

Description automatically generated

Always close the excel first and then run the cases.

# Run the code for the excel-

|  |
| --- |
| Contact us feature page –  Feature: contact us feature  Scenario Outline: contact us scenario with different sets of data  Given user navigates to contact us page  When user fills the form from given sheetname "<sheetName>" and rownumber <rowNumber>  And user clicks on send button  Then it shows successful message "Your message has been successfully sent to our team."  Examples:  | sheetName | rowNumber |  | contactus | 0 |  | contactus | 1 | |
| Contact us page step def-  package parallel;  import java.io.IOException;  import java.util.List;  import java.util.Map;  import org.apache.poi.openxml4j.exceptions.InvalidFormatException;  import org.testng.Assert;  import com.pages.ContactUsPage;  import com.qa.factory.DriverFactory;  import com.qa.util.ExcelReader;  import io.cucumber.java.en.Given;  import io.cucumber.java.en.Then;  import io.cucumber.java.en.When;  public class ContactUsPageSteps {    private ContactUsPage contactUsPage=new ContactUsPage(DriverFactory.getDriver());    @Given("user navigates to contact us page")  public void user\_navigates\_to\_contact\_us\_page() {  DriverFactory.getDriver().get("http://www.automationpractice.pl/index.php?controller=contact");  }  @When("user fills the form from given sheetname {string} and rownumber {int}")  public void user\_fills\_the\_form\_from\_given\_sheetname\_and\_rownumber(String sheetName, Integer rowNumber)  throws InvalidFormatException, IOException {  //create object of excel reader to read the methods  ExcelReader reader=new ExcelReader();  //later on add this to config file and read the path of excel.  List<Map<String, String>> testData= reader.getData  ("E:\\Naveen Java Training\\Cucumber\\CucumberPomSeriesByKaranJulySevenA\\automation.xlsx", sheetName);  /\*\*  \* capture all the row data  \*/  String heading=testData.get(rowNumber).get("subjectHeading");  String email=testData.get(rowNumber).get("email");  String orderReference=testData.get(rowNumber).get("orderReference");  String message=testData.get(rowNumber).get("message");  //now fill the contact us form  contactUsPage.fillContactUsForm(heading, email, orderReference, message);  }  @When("user clicks on send button")  public void user\_clicks\_on\_send\_button() {  contactUsPage.clickSend();  }  @Then("it shows successful message {string}")  public void it\_shows\_successful\_message(String expectedSuccessMessage) {  String actualSuccessMessage=contactUsPage.getSuccessMessage();  Assert.assertEquals(actualSuccessMessage, expectedSuccessMessage);  }  } |
| Contact us page class-  package com.pages;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.support.ui.Select;  public class ContactUsPage {  private WebDriver driver;    private By subjectHeading=By.id("id\_contact");  private By email=By.id("email");  private By orderRef=By.name("id\_order");  private By messageText=By.id("message");  private By submitButton=By.id("submitMessage");  private By successMessage=By.cssSelector("div#center\_column p");    public ContactUsPage(WebDriver driver) {  this.driver=driver;  }    public String getContactUsPageTitle() {  return driver.getTitle();  }    //order ref and heading are select classes, so we use select class and create select class object and use  //the select methods.    public void fillContactUsForm(String heading, String emailId, String orderReference, String message) {  Select select=new Select(driver.findElement(subjectHeading));  select.selectByVisibleText(heading);  driver.findElement(email).sendKeys(emailId);  driver.findElement(orderRef).sendKeys(orderReference);  driver.findElement(messageText).sendKeys(message);  }    public void clickSend() {  driver.findElement(submitButton).click();  }    public String getSuccessMessage() {  return driver.findElement(successMessage).getText();  }  } |
| Runner-  package parallel;  import org.testng.annotations.DataProvider;  import io.cucumber.junit.CucumberOptions;  import io.cucumber.testng.AbstractTestNGCucumberTests;  @CucumberOptions(features = {"src/test/resources/parallel/ContactUs.feature"},  glue = { "parallel" },  monochrome=true,  plugin = { "pretty",  "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:",  "timeline:test-output-thread/" }  )  public class ParalelRunner extends AbstractTestNGCucumberTests {  /\*\*  \* we need to override a method from abstracttestngcucumbertests class  \* parallel=true means run in parallel, else it will run one by one.  \*  \* @return 2d array containing all scenarios.  \*/  @Override  @DataProvider(parallel = true)  public Object[][] scenarios() {  return super.scenarios();  }  } |

Cucumber report-

A screenshot of a computer

Description automatically generated

Sometimes all the feature files from the pack run even though we specify the file name. how to solve it.

Open run configuration. Edit and add wrong path in glue like src/test/java/parallel/contactuspagesteps.java and then run. It will throw error. Then remove and add only classpath: then run and it works. This also works sometimes and other times it doesn’t work. apply and close.

A screenshot of a computer

Description automatically generated

Then we got the correct output.

A screenshot of a computer

Description automatically generated

Note-

Excel sheet code is written in such a way that row number starts from zero when we are writing our code to read from excel.

Added more rows in excel-

A screenshot of a computer

Description automatically generated

We need to specify this in feature file also.

We have to specify the sheet name and the row numbers.

A screenshot of a computer

Description automatically generated

We get error as file cannot be processed if we keep the excel open when running our tests.

Run parallel runner and it should work fine.

A screenshot of a computer

Description automatically generated

Spark pdf report-

A screenshot of a computer

Description automatically generated

But the report shows sheet name and row number not the data inside it-

A close-up of a screen

Description automatically generated

Same issue with index spark html report-

A screenshot of a computer

Description automatically generated

Open cmd and go to the project location.

Run using the below command-



Forgot password failing and rest passed-

A screenshot of a computer

Description automatically generated

# Project structure-

A screenshot of a computer

Description automatically generated

# Codes from this chapter-

|  |
| --- |
| Pom-  <?xml version="1.0" encoding="UTF-8"?>  <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>CucumberPOMSeriesByKaranJulySeven</groupId>  <artifactId>CucumberPOMSeriesByKaranJulySevenA</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>CucumberPOMSeries</name>  <!-- FIXME change it to the project's website -->  <url>http://www.example.com</url>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <java.version>11</java.version>  <junit.version>4.13.1</junit.version>  <cucumber.version>6.9.0</cucumber.version>  <maven.compiler.version>3.8.1</maven.compiler.version>  <maven.surefire.version>2.22.2</maven.surefire.version>  </properties>  <dependencies>  <dependency>  <groupId>io.cucumber</groupId>  <artifactId>cucumber-java</artifactId>  <version>${cucumber.version}</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>io.cucumber</groupId>  <artifactId>cucumber-junit</artifactId>  <version>${cucumber.version}</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>${junit.version}</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>3.141.59</version>  </dependency>  <dependency>  <groupId>io.github.bonigarcia</groupId>  <artifactId>webdrivermanager</artifactId>  <version>4.2.2</version>  </dependency>  <!--  https://mvnrepository.com/artifact/tech.grasshopper/extentreports-cucumber6-adapter -->  <dependency>  <groupId>tech.grasshopper</groupId>  <artifactId>extentreports-cucumber6-adapter</artifactId>  <version>2.17.0</version>  </dependency>  <dependency>  <groupId>org.testng</groupId>  <artifactId>testng</artifactId>  <version>6.14.3</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>io.cucumber</groupId>  <artifactId>cucumber-testng</artifactId>  <version>${cucumber.version}</version>  <scope>test</scope>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->  <dependency>  <groupId>org.apache.poi</groupId>  <artifactId>poi</artifactId>  <version>4.1.2</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->  <dependency>  <groupId>org.apache.poi</groupId>  <artifactId>poi-ooxml</artifactId>  <version>4.1.2</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>${maven.compiler.version}</version>  <configuration>  <encoding>UTF-8</encoding>  <source>${java.version}</source>  <target>${java.version}</target>  </configuration>  </plugin>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-surefire-plugin</artifactId>  <version>${maven.surefire.version}</version>  <configuration>  <!-- to run with jenkins and to run using maven use the below includes -->  <includes>  <include>\*\*/ParalelRunner.java</include>  </includes>  <parallel>methods</parallel>  <!--<threadCount>4</threadCount>-->  <useUnlimitedThreads>true</useUnlimitedThreads>  </configuration>  </plugin>  <!-- <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-failsafe-plugin</artifactId>  <version>3.1.2</version>  <executions>  <execution>  <goals>  <goal>integration-test</goal>  </goals>  <configuration> -->  <!-- UNCOMMENT - To add any exclusions if required -->  <!-- <excludes> <exclude>\*\*/\*IT\*.java</exclude>  </excludes> -->  <!-- <includes> -->  <!--UNCOMMENT BELOW LINE - To execute feature files with a single runner -->  <!--give path of runner in the below step -->  <!-- <include>\*\*/MyTestRunner.java</include> -->  <!-- UNCOMMENT BELOW LINE - To execute feature  files with multiple runners -->  <!-- <include>\*\*/\*Runner.java</include> -->  <!--</includes> -->  <!-- UNCOMMENT BELOW 3 LINES - To execute using  parallel or combination  option -->  <!-- we can write parallel with classes also.-->  <!-- <parallel>methods</parallel>  <threadCount>10</threadCount>  <perCoreThreadCount>true</perCoreThreadCount> -->  <!-- UNCOMMENT BELOW 3 LINES - To execute using  forking or combination option -->  <!-- <forkCount>2</forkCount>  <reuseForks>true</reuseForks>  <reportsDirectory>${project.build.directory}/failsafe-reports\_${surefire.forkNumber}</reportsDirectory>-->  <!--</configuration>  </execution>  </executions>  </plugin>-->  </plugins>  </build>  </project> |
| Excel reader which is standard code written by Naveen-  **package** com.qa.util;  **import** java.io.File;  **import** java.io.IOException;  **import** java.util.ArrayList;  **import** java.util.LinkedHashMap;  **import** java.util.List;  **import** java.util.Map;  **import** org.apache.poi.openxml4j.exceptions.InvalidFormatException;  **import** org.apache.poi.ss.usermodel.Cell;  **import** org.apache.poi.ss.usermodel.CellType;  **import** org.apache.poi.ss.usermodel.Row;  **import** org.apache.poi.ss.usermodel.Sheet;  **import** org.apache.poi.ss.usermodel.Workbook;  **import** org.apache.poi.ss.usermodel.WorkbookFactory;  **import** org.apache.poi.ss.util.NumberToTextConverter;  **public** **class** ExcelReader {  **public** List<Map<String, String>> getData(String excelFilePath, String sheetName)  **throws** InvalidFormatException, IOException {  Sheet sheet = getSheetByName(excelFilePath, sheetName);  **return** readSheet(sheet);  }  **public** List<Map<String, String>> getData(String excelFilePath, **int** sheetNumber)  **throws** InvalidFormatException, IOException {  Sheet sheet = getSheetByIndex(excelFilePath, sheetNumber);  **return** readSheet(sheet);  }  **private** Sheet getSheetByName(String excelFilePath, String sheetName) **throws** IOException, InvalidFormatException {  Sheet sheet = getWorkBook(excelFilePath).getSheet(sheetName);  **return** sheet;  }  **private** Sheet getSheetByIndex(String excelFilePath, **int** sheetNumber) **throws** IOException, InvalidFormatException {  Sheet sheet = getWorkBook(excelFilePath).getSheetAt(sheetNumber);  **return** sheet;  }  **private** Workbook getWorkBook(String excelFilePath) **throws** IOException, InvalidFormatException {  **return** WorkbookFactory.*create*(**new** File(excelFilePath));  }  **private** List<Map<String, String>> readSheet(Sheet sheet) {  Row row;  **int** totalRow = sheet.getPhysicalNumberOfRows();  List<Map<String, String>> excelRows = **new** ArrayList<Map<String, String>>();  **int** headerRowNumber = getHeaderRowNumber(sheet);  **if** (headerRowNumber != -1) {  **int** totalColumn = sheet.getRow(headerRowNumber).getLastCellNum();  **int** setCurrentRow = 1;  **for** (**int** currentRow = setCurrentRow; currentRow <= totalRow; currentRow++) {  row = getRow(sheet, sheet.getFirstRowNum() + currentRow);  LinkedHashMap<String, String> columnMapdata = **new** LinkedHashMap<String, String>();  **for** (**int** currentColumn = 0; currentColumn < totalColumn; currentColumn++) {  columnMapdata.putAll(getCellValue(sheet, row, currentColumn));  }  excelRows.add(columnMapdata);  }  }  **return** excelRows;  }  **private** **int** getHeaderRowNumber(Sheet sheet) {  Row row;  **int** totalRow = sheet.getLastRowNum();  **for** (**int** currentRow = 0; currentRow <= totalRow + 1; currentRow++) {  row = getRow(sheet, currentRow);  **if** (row != **null**) {  **int** totalColumn = row.getLastCellNum();  **for** (**int** currentColumn = 0; currentColumn < totalColumn; currentColumn++) {  Cell cell;  cell = row.getCell(currentColumn, Row.MissingCellPolicy.***CREATE\_NULL\_AS\_BLANK***);  **if** (cell.getCellType() == CellType.***STRING***) {  **return** row.getRowNum();  } **else** **if** (cell.getCellType() == CellType.***NUMERIC***) {  **return** row.getRowNum();  } **else** **if** (cell.getCellType() == CellType.***BOOLEAN***) {  **return** row.getRowNum();  } **else** **if** (cell.getCellType() == CellType.***ERROR***) {  **return** row.getRowNum();  }  }  }  }  **return** (-1);  }  **private** Row getRow(Sheet sheet, **int** rowNumber) {  **return** sheet.getRow(rowNumber);  }  **private** LinkedHashMap<String, String> getCellValue(Sheet sheet, Row row, **int** currentColumn) {  LinkedHashMap<String, String> columnMapdata = **new** LinkedHashMap<String, String>();  Cell cell;  **if** (row == **null**) {  **if** (sheet.getRow(sheet.getFirstRowNum()).getCell(currentColumn, Row.MissingCellPolicy.***CREATE\_NULL\_AS\_BLANK***)  .getCellType() != CellType.***BLANK***) {  String columnHeaderName = sheet.getRow(sheet.getFirstRowNum()).getCell(currentColumn)  .getStringCellValue();  columnMapdata.put(columnHeaderName, "");  }  } **else** {  cell = row.getCell(currentColumn, Row.MissingCellPolicy.***CREATE\_NULL\_AS\_BLANK***);  **if** (cell.getCellType() == CellType.***STRING***) {  **if** (sheet.getRow(sheet.getFirstRowNum())  .getCell(cell.getColumnIndex(), Row.MissingCellPolicy.***CREATE\_NULL\_AS\_BLANK***)  .getCellType() != CellType.***BLANK***) {  String columnHeaderName = sheet.getRow(sheet.getFirstRowNum()).getCell(cell.getColumnIndex())  .getStringCellValue();  columnMapdata.put(columnHeaderName, cell.getStringCellValue());  }  } **else** **if** (cell.getCellType() == CellType.***NUMERIC***) {  **if** (sheet.getRow(sheet.getFirstRowNum())  .getCell(cell.getColumnIndex(), Row.MissingCellPolicy.***CREATE\_NULL\_AS\_BLANK***)  .getCellType() != CellType.***BLANK***) {  String columnHeaderName = sheet.getRow(sheet.getFirstRowNum()).getCell(cell.getColumnIndex())  .getStringCellValue();  columnMapdata.put(columnHeaderName, NumberToTextConverter.*toText*(cell.getNumericCellValue()));  }  } **else** **if** (cell.getCellType() == CellType.***BLANK***) {  **if** (sheet.getRow(sheet.getFirstRowNum())  .getCell(cell.getColumnIndex(), Row.MissingCellPolicy.***CREATE\_NULL\_AS\_BLANK***)  .getCellType() != CellType.***BLANK***) {  String columnHeaderName = sheet.getRow(sheet.getFirstRowNum()).getCell(cell.getColumnIndex())  .getStringCellValue();  columnMapdata.put(columnHeaderName, "");  }  } **else** **if** (cell.getCellType() == CellType.***BOOLEAN***) {  **if** (sheet.getRow(sheet.getFirstRowNum())  .getCell(cell.getColumnIndex(), Row.MissingCellPolicy.***CREATE\_NULL\_AS\_BLANK***)  .getCellType() != CellType.***BLANK***) {  String columnHeaderName = sheet.getRow(sheet.getFirstRowNum()).getCell(cell.getColumnIndex())  .getStringCellValue();  columnMapdata.put(columnHeaderName, Boolean.*toString*(cell.getBooleanCellValue()));  }  } **else** **if** (cell.getCellType() == CellType.***ERROR***) {  **if** (sheet.getRow(sheet.getFirstRowNum())  .getCell(cell.getColumnIndex(), Row.MissingCellPolicy.***CREATE\_NULL\_AS\_BLANK***)  .getCellType() != CellType.***BLANK***) {  String columnHeaderName = sheet.getRow(sheet.getFirstRowNum()).getCell(cell.getColumnIndex())  .getStringCellValue();  columnMapdata.put(columnHeaderName, Byte.*toString*(cell.getErrorCellValue()));  }  }  }  **return** columnMapdata;  }  } |
| Feature file-  Feature: contact us feature  Scenario Outline: contact us scenario with different sets of data  Given user navigates to contact us page  When user fills the form from given sheetname "<sheetName>" and rownumber <rowNumber>  And user clicks on send button  Then it shows successful message "Your message has been successfully sent to our team."  Examples:  | sheetName | rowNumber |  | contactus | 0 |  | contactus | 1 |  | contactus | 2 |  | contactus | 3 |  | contactus | 4 |  | contactus | 5 | |
| Step def-  package parallel;  import java.io.IOException;  import java.util.List;  import java.util.Map;  import org.apache.poi.openxml4j.exceptions.InvalidFormatException;  import org.testng.Assert;  import com.pages.ContactUsPage;  import com.qa.factory.DriverFactory;  import com.qa.util.ExcelReader;  import io.cucumber.java.en.Given;  import io.cucumber.java.en.Then;  import io.cucumber.java.en.When;  public class ContactUsPageSteps {    private ContactUsPage contactUsPage=new ContactUsPage(DriverFactory.getDriver());    @Given("user navigates to contact us page")  public void user\_navigates\_to\_contact\_us\_page() {  DriverFactory.getDriver().get("http://www.automationpractice.pl/index.php?controller=contact");  }  @When("user fills the form from given sheetname {string} and rownumber {int}")  public void user\_fills\_the\_form\_from\_given\_sheetname\_and\_rownumber(String sheetName, Integer rowNumber)  throws InvalidFormatException, IOException {  //create object of excel reader to read the methods  ExcelReader reader=new ExcelReader();  //later on add this to config file and read the path of excel.  List<Map<String, String>> testData= reader.getData  ("E:\\Naveen Java Training\\Cucumber\\CucumberPomSeriesByKaranJulySevenA\\automation.xlsx", sheetName);  /\*\*  \* capture all the row data  \*/  String heading=testData.get(rowNumber).get("subjectHeading");  String email=testData.get(rowNumber).get("email");  String orderReference=testData.get(rowNumber).get("orderReference");  String message=testData.get(rowNumber).get("message");  //now fill the contact us form  contactUsPage.fillContactUsForm(heading, email, orderReference, message);  }  @When("user clicks on send button")  public void user\_clicks\_on\_send\_button() {  contactUsPage.clickSend();  }  @Then("it shows successful message {string}")  public void it\_shows\_successful\_message(String expectedSuccessMessage) {  String actualSuccessMessage=contactUsPage.getSuccessMessage();  Assert.assertEquals(actualSuccessMessage, expectedSuccessMessage);  }  } |
| Page class-  package com.pages;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.support.ui.Select;  public class ContactUsPage {  private WebDriver driver;    private By subjectHeading=By.id("id\_contact");  private By email=By.id("email");  private By orderRef=By.name("id\_order");  private By messageText=By.id("message");  private By submitButton=By.id("submitMessage");  private By successMessage=By.cssSelector("div#center\_column p");    public ContactUsPage(WebDriver driver) {  this.driver=driver;  }    public String getContactUsPageTitle() {  return driver.getTitle();  }    //order ref and heading are select classes, so we use select class and create select class object and use  //the select methods.    public void fillContactUsForm(String heading, String emailId, String orderReference, String message) {  Select select=new Select(driver.findElement(subjectHeading));  select.selectByVisibleText(heading);  driver.findElement(email).sendKeys(emailId);  driver.findElement(orderRef).sendKeys(orderReference);  driver.findElement(messageText).sendKeys(message);  }    public void clickSend() {  driver.findElement(submitButton).click();  }    public String getSuccessMessage() {  return driver.findElement(successMessage).getText();  }  } |
| Runner-  package parallel;  import org.testng.annotations.DataProvider;  import io.cucumber.junit.CucumberOptions;  import io.cucumber.testng.AbstractTestNGCucumberTests;  @CucumberOptions(features = {"src/test/resources/parallel/ContactUs.feature"},  glue = { "parallel" },  monochrome=true,  plugin = { "pretty",  "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:",  "timeline:test-output-thread/" }  )  public class ParalelRunner extends AbstractTestNGCucumberTests {  /\*\*  \* we need to override a method from abstracttestngcucumbertests class  \* parallel=true means run in parallel, else it will run one by one.  \*  \* @return 2d array containing all scenarios.  \*/  @Override  @DataProvider(parallel = true)  public Object[][] scenarios() {  return super.scenarios();  }  } |

Latest code available on GIT repo: [GitHub - naveenanimation20/LatestCucumber6WithPOM: Latest Cucumber6 BDD framework with POM(Page Object Model)](https://github.com/naveenanimation20/LatestCucumber6WithPOM)

# I removed all the feature files from parallel folder and placed in another folder and it is now running only contact us page-

A screenshot of a computer

Description automatically generated