In the last chapter of ***Parameterization in Cucumber***, we learned how to *parameterize* data. But with that trick, only limited functionality can be achieved of Data-Driven. As the test can be run multiple times. But by now that you know the anatomy of a *Data-Driven test*, here's a trick that simplifies the process of ***Data-Driven testing using Cucumber***. *Cucumber* inherently supports *Data-Driven testing* by the use of the ***Scenario Outline*** and ***Examples*** section. It is with these keywords that *Cucumber* allows for easy *Data-Driven testing* to be completed where no changes need to be made to the Java file. In this tutorial we learn, How to ***Implement a Scenario Outline in Data-Driven testing using Examples Keyword?***

Example keyword can only be used with the *Scenario Outline* Keyword.

> ***Scenario Outline*** *- This is used to run the same scenario for 2 or more different sets of test data. E.g. In our scenario, if you want to register another user you can data drive the same scenario twice*.
> ***Examples****– All scenario outlines have to be followed with the Examples section. This contains the data that has to be passed on to the scenario*.

Data-Driven Testing Using Examples Keyword

If you understood the concept of ***Parameterization in Cucumber***, you would find this one very easy. In this tutorial as well I am taking the same *LogIn* test scenario.

- Enter the ***Example Data*** just below the *LogIn* Scenario of the *Feature File.*

Examples:

| ***username*** | ***password*** |

| ***testuser_1*** | ***Test@153*** |

| ***testuser_2*** | ***Test@153*** |

Note*: The table must have a header row corresponding to the variables in the Scenario Outline steps.*

The Examples section is a table where each argument variable represents a column in the table, separated by "**|**". Each line below the header represents an individual run of the test case with the respective data. As a result, if there are 3 lines below the header in the Examples table, the script will run 3 times with its respective data.

. Need to update the Statement in the feature file, which tells Cucumber to enter username & Password.

And User enters *`<username>`* ***and*** *`<password>`*

Cucumber understands the above statement syntax and looks for the ***Examples*** Keyword in the test to read the Test Data.

The complete code will look like this:

```
Feature: Login Action

Scenario Outline: Successful Login with Valid Credentials
        Given User is on Home Page
        When User Navigate to LogIn Page
        And User enters "<username>" and "<password>"
        Then Message displayed Login Successfully
Examples:
    | username    | password |
    | testuser 1 | Test@153 |
    | testuser_2 | Test@153 |
```

. There are no changes in ***TestRunner*** class.

```
package cucumberTest;

import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
                features = "Feature"
                ,glue={"stepDefinition"}
                )

public class TestRunner {

}
```

. There are no changes in ***Test_Steps*** file from the previous chapter.

```
package stepDefinition;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;

public class Test Steps {
                public static WebDriver driver;
        @Given("^User is on Home Page$")
        public void user is on Home Page() throws Throwable {
                driver = new FirefoxDriver();
            driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
            driver.get("https://www.store.demoqa.com");
        }

        @When("^User Navigate to LogIn Page$")
        public void user Navigate to LogIn Page() throws Throwable {
                driver.findElement(By.xpath(".//*[@id='account']/a")).click();
        }

        @When("^User enters \"(.*)\" and \"(.*)\"$")
        public void user enters UserName and Password(String username, String password) t
                driver.findElement(By.id("log")).sendKeys(username);
            driver.findElement(By.id("pwd")).sendKeys(password);
            //driver.findElement(By.id("login")).click();
        }

        @Then("^Message displayed Login Successfully$")
        public void message displayed Login Successfully() throws Throwable {
                System.out.println("Login Successfully");
        }

        @When("^User LogOut from the Application$")
        public void user LogOut from the Application() throws Throwable {
                driver.findElement (By.xpath(".//*[@id='account_logout']/a")).click();
        }

        @Then("^Message displayed LogOut Successfully$")
        public void message displayed LogOut Successfully() throws Throwable {
                System.out.println("LogOut Successfully");
        }

}
```

. Run the test by Right Click on ***TestRunner class*** and Click ***Run As > JUnit Test*** Application.

This takes the parameterization one step further: now our scenario has “***variables***” and they get filled in by the values in each row. To be clear: by defining this, the scenario will run two times, passing in one row at a time. This makes it very easy to define a lot of examples, edge cases, and special outcomes. Instead of hardcoding the test data, variables are defined in the Examples section and used in the Scenario Outline section.

Note*: Please create your own username & password for the test, if you supply wrong* ***UserName & Password 3 times****, your IP will get blocked.*