What is Data Driven Testing?

Data-driven testing means using a single test to verify many different test cases by driving the test with input and expected values from an external data source instead of using the same hard-coded values each time the test runs.

If you look into our framework, you will find that till now we have been passing values in the code in the hard coded form. For e.g. *UserName & Password.* So far only one user name and password are being used and that is being declared in the constants class but just think of a scenario when there will be N number of combinations to test for a log-in functionality. And this is just not limited to username & password only, even in the open browser method we have been using just one browser. In the real world, there will be many more browsers to test and to pass the browser to test case is necessary and that can only be done with the use of *Data-driven technique.*

One more flaw is there in our framework, as there are two different input methods for username and password which are *'input_UserName'* & *'input_Password'*. Which is not at all right, as we move along with writing more test of the application, we will find N number of input fields and we will end up with hundreds of method of input text field.

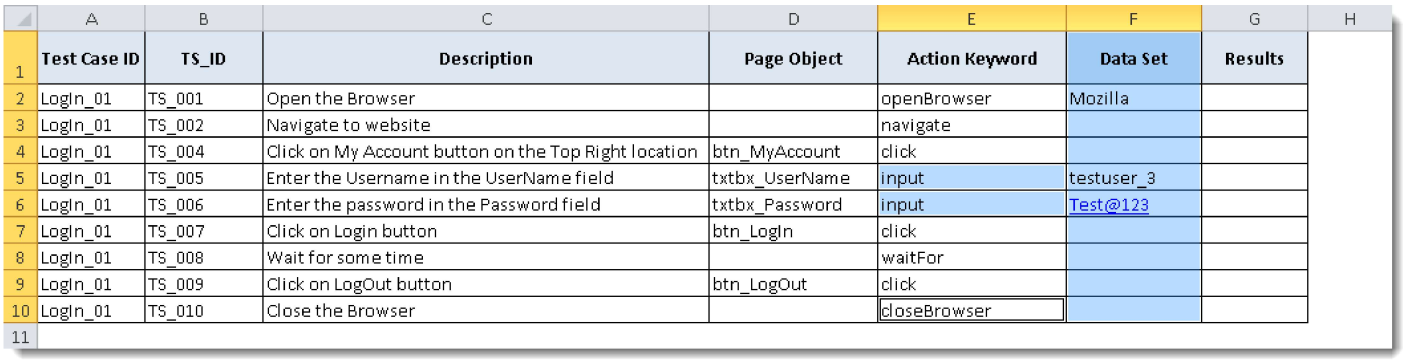
This situation can be overcome with the help of data-driven technique. Once we start with passing test data from the external source, we would not need different input text fields method. Only one method of input would be good enough for us, which will accept the test data.

As we have been driving everything from the *Data Engine* excel sheet, it is better to drive the set of test data from it as well.

Step 1: Set up Data Engine excel sheet

1. Create an extra column in the *test step* sheet and name it as *'Data Set'*. Insert this Data set column before the Results column.
2. Enter the *browser name, username,* and *password* in the *Data set* column for the relevant rows.
3. Replace the keyword *'input_Username'* with just *'input'* and do the same for *'input_Password'* as well.

Test Steps sheet:



	A	B	C	D	E	F	G	H
1	Test Case ID	TS_ID	Description	Page Object	Action Keyword	Data Set	Results	
2	LogIn_01	TS_001	Open the Browser		openBrowser	Mozilla		
3	LogIn_01	TS_002	Navigate to website		navigate			
4	LogIn_01	TS_004	Click on My Account button on the Top Right location	btn_MyAccount	click			
5	LogIn_01	TS_005	Enter the Username in the UserName field	txtbx_UserName	input	testuser_3		
6	LogIn_01	TS_006	Enter the password in the Password field	txtbx_Password	input	Test@123		
7	LogIn_01	TS_007	Click on Login button	btn_LogIn	click			
8	LogIn_01	TS_008	Wait for some time		waitFor			
9	LogIn_01	TS_009	Click on LogOut button	btn_LogOut	click			
10	LogIn_01	TS_010	Close the Browser		closeBrowser			
11								

Step 2: Modify Constants class

- Create a new *constant variable* for column *Data Set.*
- As we have shifted the result columns, do not forget to increment the number of columns for the Result constant variable.

```
public static final int Col DataSet =5 ;
public static final int Col_TestStepResult =6 ;
```

Step 3: Modify Action Keyword Class

- Change the method of *Action Keyword* class to accept data as an *argument.*
- As we have followed the concept of *reflection class* of java, we had to change each and every method of *Action Keyword* class to accept data as an *argument,* even if there is no data to be supplied for any action such as *click()* method. Click method just needs an object to click on, it does not need and test data but still, we need to change it to follow the rule of reflection, otherwise reflection won't work at all.
- Change the working code of the method *input,* replace the constants variable of username to data.
- Change the working code of the method *open browser.*

```
package config;

import java.util.concurrent.TimeUnit;
import static executionEngine.DriverScript.OR;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.ie.InternetExplorerDriver;
import executionEngine.DriverScript;
import utility.Log;

public class ActionKeywords {
        public static WebDriver driver;
//This block of code will decide which browser type to start
public static void openBrowser(String object,String data){
        Log.info("Opening Browser");
        try{
                //If value of the parameter is Mozilla, this will execute
                if(data.equals("Mozilla")){
                        driver=new FirefoxDriver();
                        Log.info("Mozilla browser started");
                        }
                else if(data.equals("IE")){
                        //You may need to change the code here to start IE Driver
                        driver=new InternetExplorerDriver();
                        Log.info("IE browser started");

                        }
                else if(data.equals("Chrome")){
                        driver=new ChromeDriver();
                        Log.info("Chrome browser started");
                        }

                int implicitWaitTime=(10);
                driver.manage().timeouts().implicitlyWait(implicitWaitTime, TimeU
        }catch (Exception e){
                Log.info("Not able to open the Browser --- " + e.getMessage());
                DriverScript.bResult = false;
        }
}

public static void navigate(String object, String data){
        try{
                Log.info("Navigating to URL");
                driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
                //Constant Variable is used in place of URL
                driver.get(Constants.URL);
        }catch(Exception e){
                Log.info("Not able to navigate --- " + e.getMessage());
                DriverScript.bResult = false;
                }
        }

public static void click(String object, String data){
        try{
                Log.info("Clicking on Webelement "+ object);
                driver.findElement(By.xpath(OR.getProperty(object))).click();
         }catch(Exception e){
                Log.error("Not able to click --- " + e.getMessage());
                DriverScript.bResult = false;
        }
        }
//Now this method accepts two value (Object name & Data)
public static void input(String object, String data){
        try{
                Log.info("Entering the text in " + object);
                driver.findElement(By.xpath(OR.getProperty(object))).sendKeys(dat
         }catch(Exception e){
                 Log.error("Not able to Enter UserName --- " + e.getMessage());
                 DriverScript.bResult = false;
                }
        }

public static void waitFor(String object, String data) throws Exception{
        public static void waitFor(String object, String data) throws Exception{
                try{
                        Log.info("Wait for 5 seconds");
                        Thread.sleep(5000);
                 }catch(Exception e){
                         Log.error("Not able to Wait --- " + e.getMessage());
                         DriverScript.bResult = false;
                }
                }

        public static void closeBrowser(String object, String data){
                try{
                        Log.info("Closing the browser");
                        driver.quit();
                 }catch(Exception e){

                         Log.error("Not able to Close the Browser --- " + e.getMessage())
                         DriverScript.bResult = false;
                }
                }

        }
```

Step 4: Modify Driver Engine script

- Create new variable for Test Data *'public static String sData;'*
- Add a statement to assign value to data variable from the Data Set column of sheet Test Step
- In *'execute_Action()'* block, pass *'sData'* variable to *invoke* method.

```
package executionEngine;

import java.io.FileInputStream;
import java.lang.reflect.Method;
import java.util.Properties;
import org.apache.log4j.xml.DOMConfigurator;
import config.ActionKeywords;
import config.Constants;
import utility.ExcelUtils;
import utility.Log;

public class DriverScript {

        public static Properties OR;
        public static ActionKeywords actionKeywords;
        public static String sActionKeyword;
        public static String sPageObject;
        public static Method method[];
        public static int iTestStep;
        public static int iTestLastStep;
        public static String sTestCaseID;
        public static String sRunMode;
        public static String sData;
        public static boolean bResult;
```

```
  public DriverScript() throws NoSuchMethodException, SecurityException{
      actionKeywords = new ActionKeywords();
      method = actionKeywords.getClass().getMethods();
  }

public static void main(String[] args) throws Exception {
  ExcelUtils.setExcelFile(Constants.Path_TestData);
  DOMConfigurator.configure("log4j.xml");
  String Path_OR = Constants.Path_OR;
      FileInputStream fs = new FileInputStream(Path_OR);
      OR= new Properties(System.getProperties());
      OR.load(fs);

      DriverScript startEngine = new DriverScript();
      startEngine.execute_TestCase();
}

private void execute_TestCase() throws Exception {
      int iTotalTestCases = ExcelUtils.getRowCount(Constants.Sheet_TestCases);
          for(int iTestcase=1;iTestcase<iTotalTestCases;iTestcase++){
              bResult = true
              sTestCaseID = ExcelUtils.getCellData(iTestcase, Constants
              sRunMode = ExcelUtils.getCellData(iTestcase, Constants.Co
              if (sRunMode.equals("Yes")){
                  iTestStep = ExcelUtils.getRowContains(sTestCaseID
                  iTestLastStep = ExcelUtils.getTestStepsCount(Cons
                  Log.startTestCase(sTestCaseID);
                  bResult=true;
                  for (;iTestStep<iTestLastStep;iTestStep++){
                  sActionKeyword = ExcelUtils.getCellData(iTestStep
                  sPageObject = ExcelUtils.getCellData(iTestStep, C
                      //Now we will use the data value and pass
                  sData = ExcelUtils.getCellData(iTestStep, Constan
                  execute_Actions();
                      if(bResult==false){
                          ExcelUtils.setCellData(Constants.
                          Log.endTestCase(sTestCaseID);
                          break;
                          }
                      }
                  if(bResult==true){
                  ExcelUtils.setCellData(Constants.KEYWORD_PASS,iTe
                  Log.endTestCase(sTestCaseID);
                      }

                  }

              }
      }

 private static void execute_Actions() throws Exception {

      for(int i=0;i<method.length;i++){

          if(method[i].getName().equals(sActionKeyword)){
              //This code will pass three parameters to every invoke me
              method[i].invoke(actionKeywords,sPageObject, sData);
              if(bResult==true){
                  ExcelUtils.setCellData(Constants.KEYWORD_PASS, iT
                  break;
              }else{
                                        }else{
                                                ExcelUtils.setCellData(Constants.KEYWORD_FAIL, iT
                                                ActionKeywords.closeBrowser("","");
                                                break;
                                                }
                                        }
                                }
                }

        }
```

Note: *For any explanation on un-commented code, please refer previous chapters.*

You can see now the whole Keyword Driven Framework is set for use. Now it is up to you to implement other enhancements in this framework like you can implement *HTML reporting* in it or you can even implement some sort of *automatic email reply* on any test fail or an automatic email sent to the project stakeholders with test execution results or any thing.