**[Hybrid Testing (Data + Keyword Driven) using Selenium](https://automationtestingsimplified.wordpress.com/2011/05/31/hybrid-testing-data-keyword-driven-using-selenium/)**

Today, we’ll see how to do hybrid testing using selenium. Let’s discuss what’s hybrid testing first. Hybrid testing is a combination of Data Driven testing along with keyword. Here we’ll use some keywords as the driving parameters in data driven testing the data sheet. The keywords will be defined by the user, and let’s call them user defined keywords.

Before getting into the framework, let’s discuss on what are the external files which will be required. The list of files that will be required in addition to selenium client driver and selenium server jar files are as below

1. TestNG: in order to data drive our test, we would require the latest version of testNG jar file which is now testng-5.14.1.jar and can be downloaded from <http://testng.org/doc/download.html>
2. JXL: in order to use Microsoft Excel files as data source we would need the jxl jar file which can be downloaded from <http://sourceforge.net/projects/jexcelapi/files/>
3. Latest Junit: this will be required to verify certain condition and can be downloaded from <https://github.com/KentBeck/junit/downloads>
4. TestNG plugin for Eclipse: This will be required to run the TestNG scripts in eclipse and the content of this plugin has to be placed inside the “dropin” folder inside the eclipse directory. The file can be downloaded from <http://testng.org/doc/download.html>

The first 3 file need to be added to the build path in the project in eclipse to use them. The fourth file needs to extracted in the dropin folder of the eclipse folder so that when you open the droping folder, a plugin folder is created inside it. This is all about configuration that is required for such testing.

Now let’s start with how to test using this framework. Let’s test the IBN Live website, where we’ll use the keyword ‘link’ to drive our test. So to do this, we need to place this keyword ‘link’ in the data sheet. This will decide the action when the ‘link’ keyword is passed to the program. The corresponding cell values for that particular row will be used for different operations on the same. Below is the data sheet that we are going to use.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| testData | **property** | **xPath** | **value** | **expectedResult** | **permission** | **name** |  |
|  | pagetitle |  |  | CNN-IBN, Live India News,Top Breaking News,World,India,Business,Sports, Entertainment & Health News |  |  |  |
|  | link | link=News |  |  | click | News |  |
|  | link | link=Politics |  |  | click | Politics |  |
|  | link | link=Movies |  |  | click | Movies |  |
|  |  |  |  |  |  |  | testData |

Here using selenium, we are going to test the page title of the landing page and use the link property to navigate to different pages. So first let me explain the data table. “testData” is the table marker which will tell the start and end point of the table. The column headers will be used as variables. Here property defines the property on which we are going to perform certain actions. xPath will be used to locate the properties, value will be used to enter some data or select any object, expected result will be used to compare the actual value to the expected value, permission will determine the course of action and name will be used to give meaningful name while generating result. Below is the sample code that will drive our Hybrid test.

import com.thoughtworks.selenium.\*;

import org.junit.AfterClass;

import org.openqa.selenium.server.SeleniumServer;

import org.testng.annotations.\*;

import java.io.File;

import jxl.\*;

public class DataDrivenTest extends SeleneseTestCase{

@BeforeClass

public void setUp() throws Exception {

setUp("http://ibnlive.in.com/", "\*iexplore");

selenium.open("/");

selenium.setTimeout("60000");

selenium.waitForPageToLoad("60000");

selenium.windowMaximize();

selenium.windowFocus();

}

@DataProvider(name = "DP")

public Object[][] createData() throws Exception{

Object[][] retObjArr=getTableArray("D:\\Work\\Data Driven Testing\\Data Driven Test\\Resources\\Data Sheet.xls",

"Data", "testData");

return(retObjArr);

}

@Test (dataProvider = "DP")

public void testElements(String property, String xPath, String value, String expectedResult, String permission,

String name) throws Exception {

if(property.equals("pagetitle")){

if(selenium.getTitle().equals(expectedResult)){

System.out.println("The title is displaying properly");

}

else

System.out.println("The title is not correct");

}

if(property.equals("link")){

if(selenium.isElementPresent(xPath))

System.out.println("The "+name+" is present");

else

System.out.println("The "+name+" is not present");

if(permission.equals("click"))

{

selenium.click(xPath);

selenium.waitForPageToLoad("60000");

}

}

}

@AfterClass

public void tearDown(){

selenium.close();

selenium.stop();

}

public String[][] getTableArray(String xlFilePath, String sheetName, String tableName) throws Exception{

String[][] tabArray=null;

Workbook workbook = Workbook.getWorkbook(new File(xlFilePath));

Sheet sheet = workbook.getSheet(sheetName);

int startRow,startCol, endRow, endCol,ci,cj;

Cell tableStart=sheet.findCell(tableName);

startRow=tableStart.getRow();

startCol=tableStart.getColumn();

Cell tableEnd= sheet.findCell(tableName, startCol+1,startRow+1, 100, 64000, false);

endRow=tableEnd.getRow();

endCol=tableEnd.getColumn();

System.out.println("startRow="+startRow="+endRow+", " +

"startCol="+startCol+", endCol="+endCol);

tabArray=new String[endRow-startRow-1][endCol-startCol-1];

ci=0;

for (int i=startRow+1;i<endRow;i++,ci++){

cj=0;

for (int j=startCol+1;j<endCol;j++,cj++){

tabArray[ci][cj]=sheet.getCell(j,i).getContents();

}

}

return(tabArray);

}

}//end of class

The createData method will create a virtual copy of the data sheet in memory. Inside it, we need to specify the data sheet path along with its name, the sheet name, the table marker that we discussed earlier. Here, the testElements method is driving the test depending on the input, it’s getting from the excel sheet. It can be customized to accommodate more types of properties like drop down, text box, text area, etc. according to user needs.