Wolfram Coding Assignment

Created By: Samuel Benison (Samuel.jeyarajvictor@mavs.uta.edu)

Date: 04/28/2015

Task Description:

1. Go to Wolfram Cloud (www.wolframcloud.com).

2. Click Wolfram Programming Cloud. Please create a new Wolfram ID, and you can subscribe to a Free plan.

3. After signing in successfully, you will be taken to what we call, the Homescreen.

4. From the Homescreen, users can create new notebooks, upload files, etc.

5. On the right-hand side of the application, you will see a red "New" button.

6. By clicking the down-arrow button, you can create different types of files (.nb, .html, .css, etc).

7. Create a .nb notebook.

8. In the new notebook, in the file header, click the file name field, "(unnamed)" -- you will see that the extension, ".nb" is automatically present.

Overview:

This project is primarily created to perform a set of automation tests into certain features of Wolfram Cloud using Selenium WebDriver, TestNG and ReportNG.

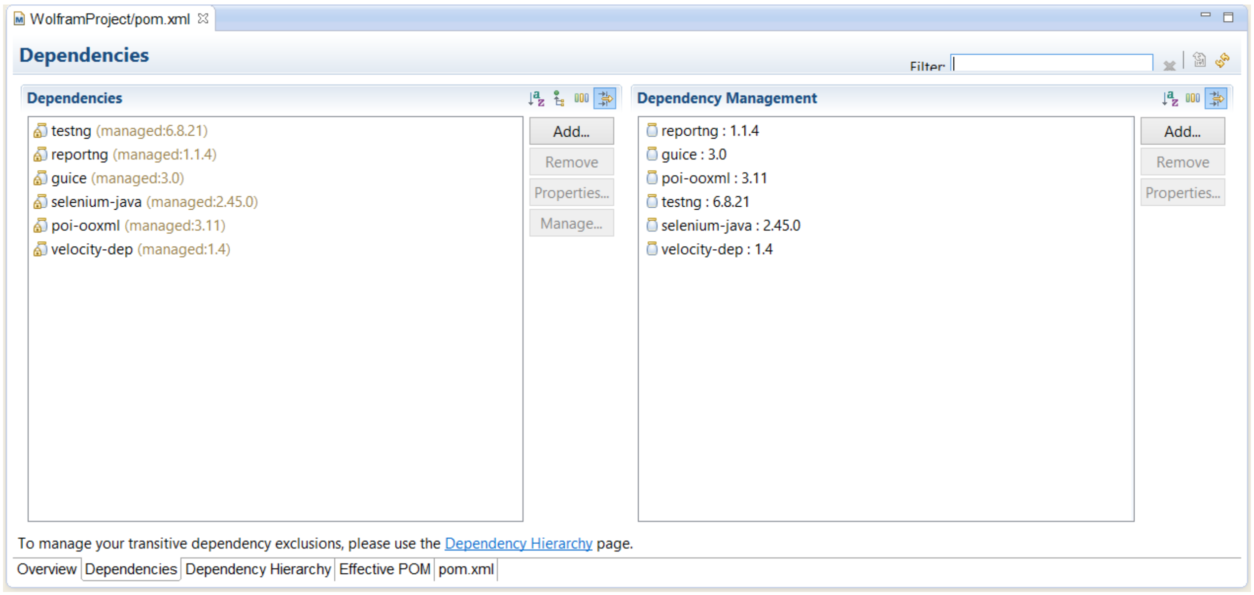
Development Specifications:

The following libraries and tools were used to create this project:

- Selenium WebDriver (Java)
- TestNG
- ReportNG
- Maven
- Apache POI
- Firebug (along with Firepath)

Why Maven?

Maven is a very useful Software Management Tool which helps me to automatically import the required libraries, and link it with the project. Since this project deals with a lot of libraries, it is very useful to keep the libraries intact.


WolframProject/pom.xml

Dependencies

Filter:

Dependencies

testng (managed:6.8.21)
reportng (managed:1.1.4)
guice (managed:3.0)
selenium-java (managed:2.45.0)
poi-ooxml (managed:3.11)
velocity-dep (managed:1.4)

Add... Remove Properties... Manage...

Dependency Management

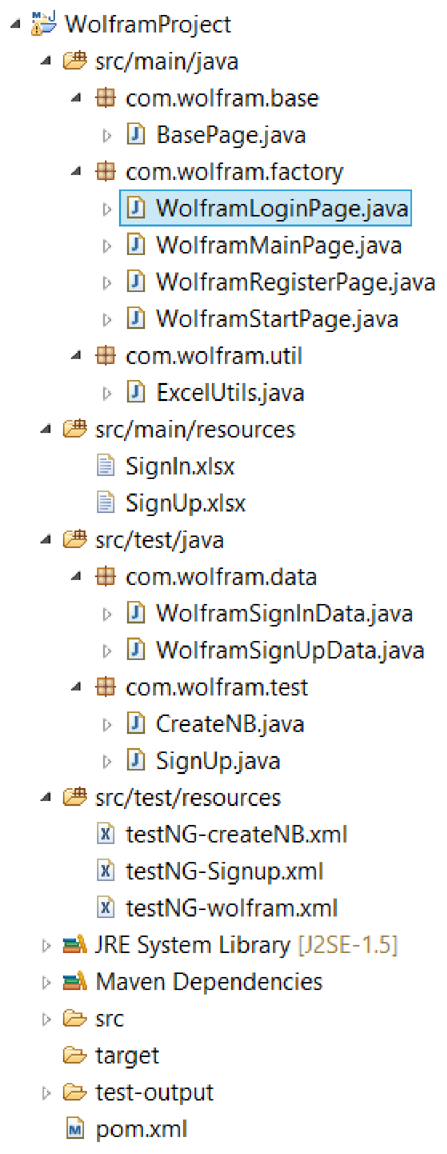
reportng : 1.1.4
guice : 3.0
poi-ooxml : 3.11
testng : 6.8.21
selenium-java : 2.45.0
velocity-dep : 1.4

Add... Remove Properties...

To manage your transitive dependency exclusions, please use the Dependency Hierarchy page.

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

Also, It helps me to maintain a specific structure (Artifact) to the project, which is useful for the design and implementation of the project. In this project, I have used the below structure,



```
WolframProject
  src/main/java
    com.wolfram.base
      BasePage.java
    com.wolfram.factory
      WolframLoginPage.java
      WolframMainPage.java
      WolframRegisterPage.java
      WolframStartPage.java
    com.wolfram.util
      ExcelUtils.java
  src/main/resources
    SignIn.xlsx
    SignUp.xlsx
  src/test/java
    com.wolfram.data
      WolframSignInData.java
      WolframSignUpData.java
    com.wolfram.test
      CreateNB.java
      SignUp.java
  src/test/resources
    testNG-createNB.xml
    testNG-Signup.xml
    testNG-wolfram.xml
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
  target
  test-output
  pom.xml
```

- Src/test/java
  - Contains all the TestNG Test Cases (Primary Location)
- Src/test/resources
  - Contains all the testNG xml files which can be used for Test Suites
- Src/main/java
  - Contains all the dependency files to the test cases such as PageFactory, BasePage, Utilities, etc.
- Src/main/resources
  - Contains all the data files i.e., Excel Sheets

Why Excel Sheet to handle data?

The primary reason is, because it is easy to send the project and can be executed by different people, Since there is no database setup and query execution required. Anyone can enter data in Excel Sheet as it is self-explanatory. Secondly, it is easy to classify 'DataSheet' and 'TestSheet', and use it in different parts of the program. Most of the data's from database can be exported to an xlsx/csv file, and it is easy to deal with for automation.

What is DataSheet?

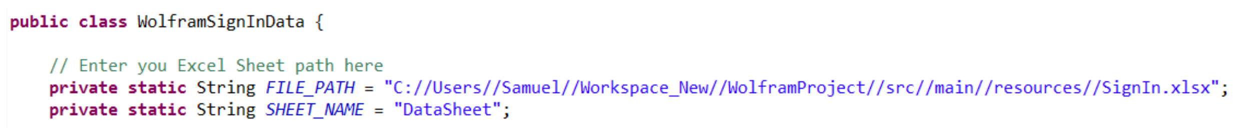
DataSheet is a Sheet in Excel which contains all the necessary data which needs to be given as input to the application while performing the automation testing. (i.e., Test Data)

What is TestSheet?

This is an interesting type of data. TestSheet is nothing but a set of data corresponds to the data of DataSheet which defines the meaning of those data. For example, if we give a set of data in the DataSheet, how can we define the expected behavior of the test which are using those data? There is both Positive Test and Negative Test, and each will give different output. If the output is a negative result, and the intention of the test is to verify the negative flow, then the test case should pass instead of fail. So instead of writing two different workflows, TestSheet will help us to achieve all the flows of the testing with the help of a common code itself. All it requires is a simple 'True' or 'False' answer for a set of questions, and the program will automatically identify the intention of the particular testing (positive flow or negative flow).

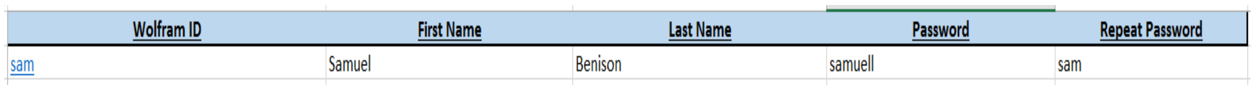
What should I do to run the Project?

1. Update the path of your excel sheet in WolframSignInData.java and WolframSignUpData.java files under the folder src/test/java.



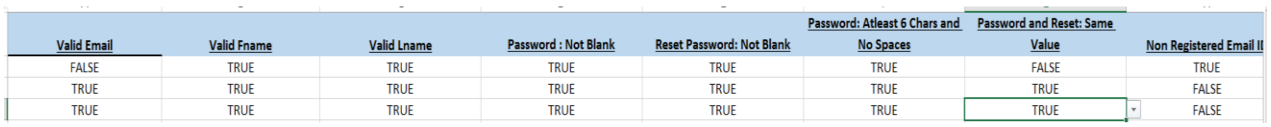
```
public class WolframSignInData {

    // Enter you Excel Sheet path here
    private static String FILE_PATH = "C://Users//Samuel//Workspace_New//WolframProject//src//main//resources//SignIn.xlsx";
    private static String SHEET_NAME = "DataSheet";
```

2. Check all the library files, and make sure it's synced with Maven or download and reference the library files.
3. Fill the data in the DataSheet and enter your input for the corresponsing TestSheet
   Example: DataSheet



Wolfram ID	First Name	Last Name	Password	Repeat Password
sam	Samuel	Benison	samuell	sam

Example: TestSheet

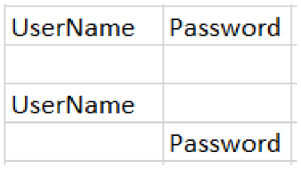


Valid Email	Valid Fname	Valid Lname	Password : Not Blank	Reset Password: Not Blank	Password: Atleast 6 Chars and No Spaces	Password and Reset: Same Value	Non Registered Email II
FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE

4. Fill as many rows you want in the excel sheet.
5. Run any one of the TestNG XML files and look for the results.

Important Note: (about Excel sheet)

When you are removing a row, which consists of a set data, make sure you select the entire row, right click on the row index and select delete. Because, the program will ignore null, but it will still consider empty string i.e., “”. So if you double click the cell and remove the content, then it is empty not null. The reason I made the program work that way is, because I want to test different scenarios which inserts empty string. Something like this,



UserName	Password
UserName	
	Password

Also, don’t forget to fill out the corresponding row of TestSheet for each row in DataSheet. If a TestSheet row is missing for a corresponding DataSheet tuple, the program will automatically take TRUE as the default answer for all columns, but it is highly recommended to enter TestSheet as well.

What is the Structural Design of the Project?

This project consists of 3 layers: Test Layer(Test Case), Factory Layer (Page Factory) and Object Layer (Base Page). My main agenda is to avoid redundant code as much as possible, using these design will certainly help that.

- Test Case:
  - This is the upper layer which consists of a set of Tests. This layer won’t have much details about the action performed in the automation, all it does is initiate a set of functions from factory method by passing parameters, and gather the result. Since this is very high level, people who doesn’t know much about the automation testing can easily create test case by passing arguments and calling methods.

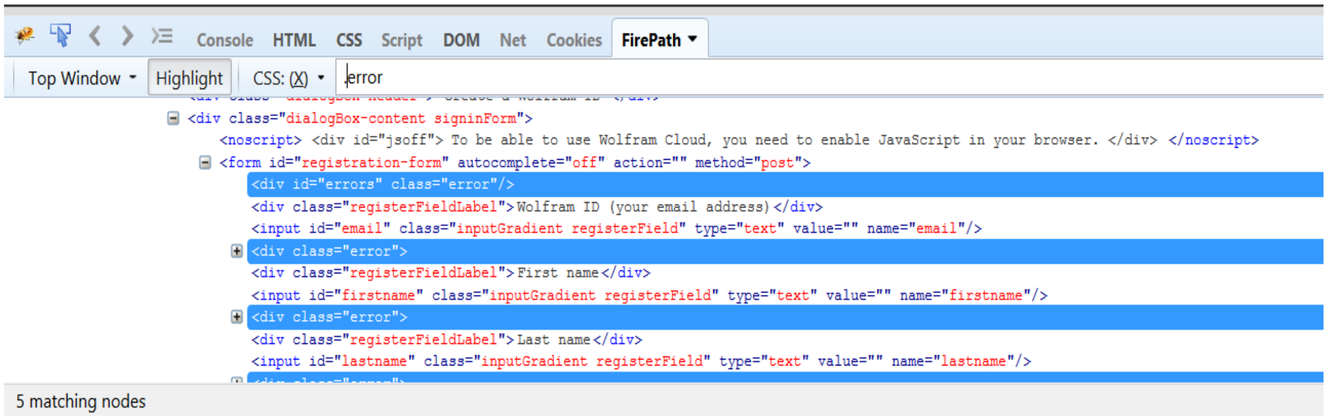
- Page Factory:

    - Page Factory is the middle layer, which performs the functions called by Test Layer. This is the heart of the application. It contains all the Web Elements and the actions which needs to be performed. Each Page will have separate Page Factory, and it interacts with both Test and Object Layer.

- Base Page:
    - Base Page is the lower layer, which cumulates all the common actions performed in the Factory Layer, and form a very generic function which can be called be the Factory Layer. It won't have any actions or elements stored in the memory, It receives the elements and instructions from the Factory Layer, and simply execute the task and return the result. Having Base Page will reduce the redundant code.

How will you find Elements?

I usually prefer to use CSSSelector, because it is easy to identify the elements accurately even when more than one elements have a common name. For example,



```
<div class="dialogBox-content signinForm">
  <noscript> <div id="jsoff"> To be able to use Wolfram Cloud, you need to enable JavaScript in your browser. </div> </noscript>
  <form id="registration-form" autocomplete="off" action="" method="post">
    <div id="errors" class="error"/>
    <div class="registerFieldLabel">Wolfram ID (your email address)</div>
    <input id="email" class="inputGradient registerField" type="text" value="" name="email"/>
    <div class="error">
    <div class="registerFieldLabel">First name</div>
    <input id="firstname" class="inputGradient registerField" type="text" value="" name="firstname"/>
    <div class="error">
    <div class="registerFieldLabel">Last name</div>
    <input id="lastname" class="inputGradient registerField" type="text" value="" name="lastname"/>
5 matching nodes
```

There are more than one div tag elements with class name 'error', how will I pinpoint a particular element? I need to consider the previous element i.e., div 'error' right next to an input with id 'email'. That will give more unique identification to the elements. For this purpose, CSSSelector will come in more handy rather than identifying by ID, ClassName, XPath, etc.

The above example can be solved by the query,

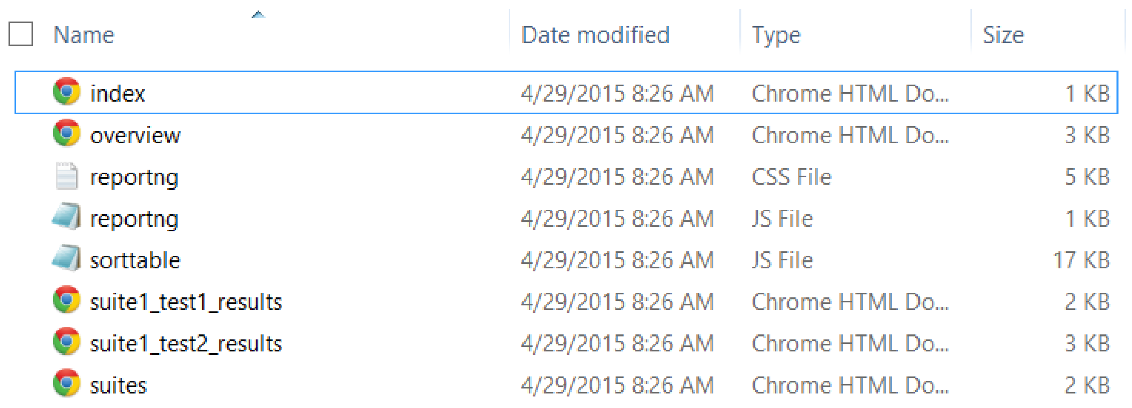
#email+div.error

Where '#' represents ID and '+' represents the concurrent element.

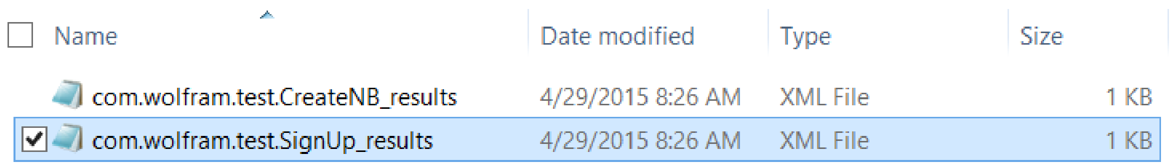
ReportNG:

I have used ReportNG to generate the test report, but I am not submitting the report as part of the submission since it has all sensitive information such as username, password, etc.

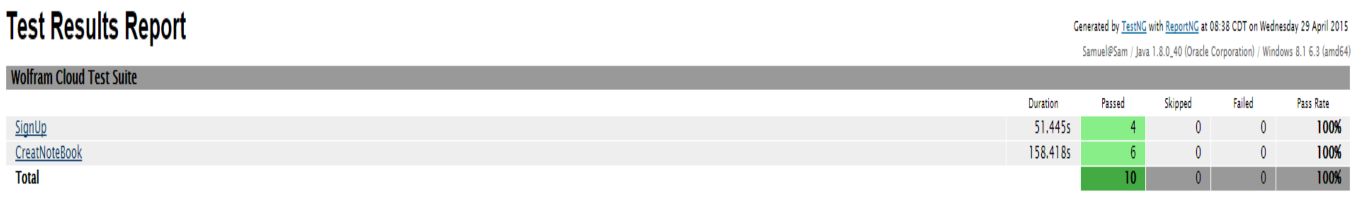
To show the proof that it is generating HTML, XML files, I have attached the below screenshots:



Name	Date modified	Type	Size
index	4/29/2015 8:26 AM	Chrome HTML Do...	1 KB
overview	4/29/2015 8:26 AM	Chrome HTML Do...	3 KB
reportng	4/29/2015 8:26 AM	CSS File	5 KB
reportng	4/29/2015 8:26 AM	JS File	1 KB
sorttable	4/29/2015 8:26 AM	JS File	17 KB
suite1_test1_results	4/29/2015 8:26 AM	Chrome HTML Do...	2 KB
suite1_test2_results	4/29/2015 8:26 AM	Chrome HTML Do...	3 KB
suites	4/29/2015 8:26 AM	Chrome HTML Do...	2 KB



Name	Date modified	Type	Size
com.wolfram.test.CreateNB_results	4/29/2015 8:26 AM	XML File	1 KB
✓ com.wolfram.test.SignUp_results	4/29/2015 8:26 AM	XML File	1 KB



Test Results Report

Generated by TestNG with ReportNG at 08:38 CDT on Wednesday 29 April 2015
Samuel@Sam / Java 1.8.0_40 (Oracle Corporation) / Windows 8.1 6.3 (amd64)

Wolfram Cloud Test Suite

	Duration	Passed	Skipped	Failed	Pass Rate
SignUp	51.445s	4	0	0	100%
CreatNoteBook	158.418s	6	0	0	100%
Total		10	0	0	100%