**README**

IMPORTANT

**Video Demo is in**

<https://www.youtube.com/watch?v=Yv1ZAdFZ-fQ>

A) Setup

=========

Pre-requisite software

- Java 1.8

- Eclipse IDE

- TestNG

- Maven

Use Eclipse to import this folder as Project

B) Dependencies

================

Refer to pom.xml for dependencies.. If you clone directly and use pom.xml from the git, you **do not need** to install any additional dependencies

C) User Test data

=================

User test data is located in \UserData

There are currently a few excel xlsx files

UserData.xlsx

Tempate\_UserData.xlsx

Each excel file contains a single test data. You can create as many test data by having different excel files.

Under Configuration.java getUserDataFileName ()

Change the user file name in this method if you are using a different file

/\*

\* CHANGE HERE

\*/

**private** String getUserDataFileName (){

**return** getUserDataPath()+"UserData.xlsx"; // CHANGE THIS IF YOU WANT TO READ DIFFERENT USER DATA !

}

D) Other changes that you need make

==============================

Chrome Driver / Firefox path

In Configuration.java , you will need to state the path for the chrome or firefox driver



E) Test Document

==============

Folder:

\Documents

Test Plan in Word format

2 Test Cases (001 / 002) in Excel format

F) User Data

==========

Folder :

UserData

2 Excel format files

Template\_UserData

UserData

You can change the data in UserData and the scripts will pick it up.

G) Prevent Browser from closing after running script

==========================================

In Scenario1.java and Scenario2.java

@AfterTest

**public** **void** afterTest() {

config.quitDriver();//You can comment out to prevent browser from closing

}

You can comment the config.quitDriver() (under @AfterTest method) if you do not want the browser to close after running the script

**What is new in this version 2**

Rewrote the script into objects and documentation (comments in each modules)

I have broken the script in 5 structures.

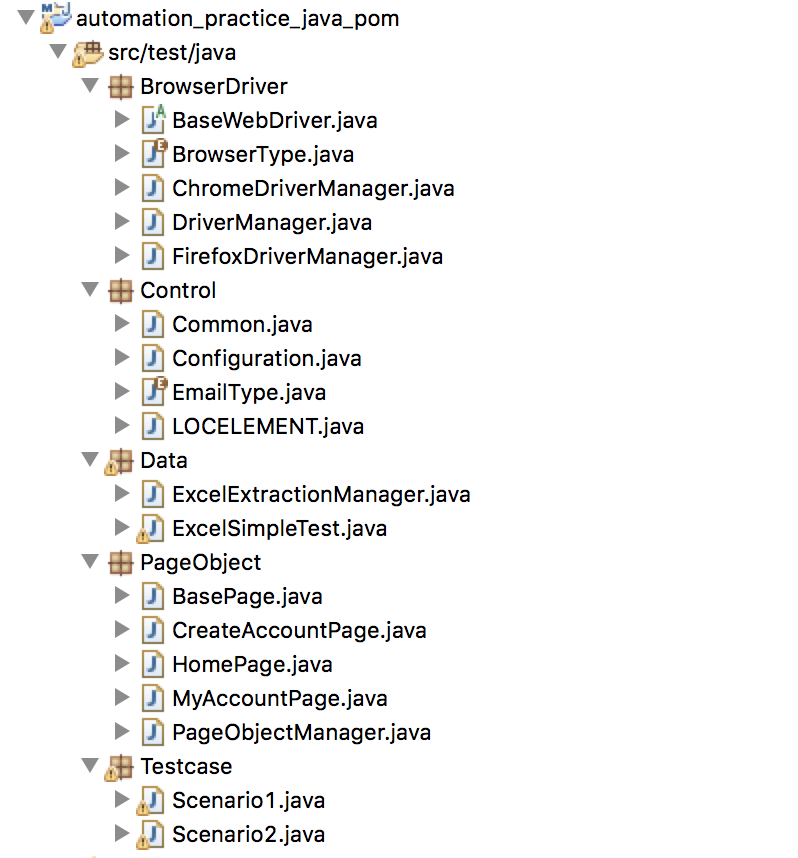
**BrowserDriver** – Browser Manager that will return the Web Drivers of selected browser

**Testcase** – all the test case scenarios

**Control** – the common modules, configuration and strings

**Page Object** – Different pages object for the web. Currently in this homework there are only 3 pages – Home, Create Account and My Account. Test script will go through Page Object Manager to this individual page object

**Data** – Data models from user created data in excel and excel module manager



**The Flow**

Testcase script

Scenario 1 and 2

Selenium Framework

Web

Element

Strings

XPath

(LOCELEMENT)

Config data (user data / web driver)

Web Driver

Page Object Models

* Home
* CreateAccount
* MyAccount

User Data

Page Object Manager

Base Page Model (BasePage)

User data imprted

selenium and other extended properties

User data obj

Controls of project config

Web Driver

WebDriver manager (Chrome / FF)

Configuration

(Configuration)

Excel Extraction Manager (ExcelExtractionManager)

Result (Validate / Screen Capture)

Common selenium modules (Common)

**Explanation of flow**

In page object model, we just treat each page as object and call the keywords. This is easier to maintain and reusable. We will only need to focus on the keywords in the individual page objects in the test scripts.

The Configuration is the project settings and WebDriver itself. To configurate the project (like path / drivers / files) , you will need to change here. This is like the center module as the WebDriver (from Configuration) is passed around different classes.

The Configuration will get the Browser Driver (Chrome or Firefox) from the Driver Manager – DriverManager.

The Common encapsulate some frequently modified api (like Send / Click / ByXPath ). It is inherited by BasePage.

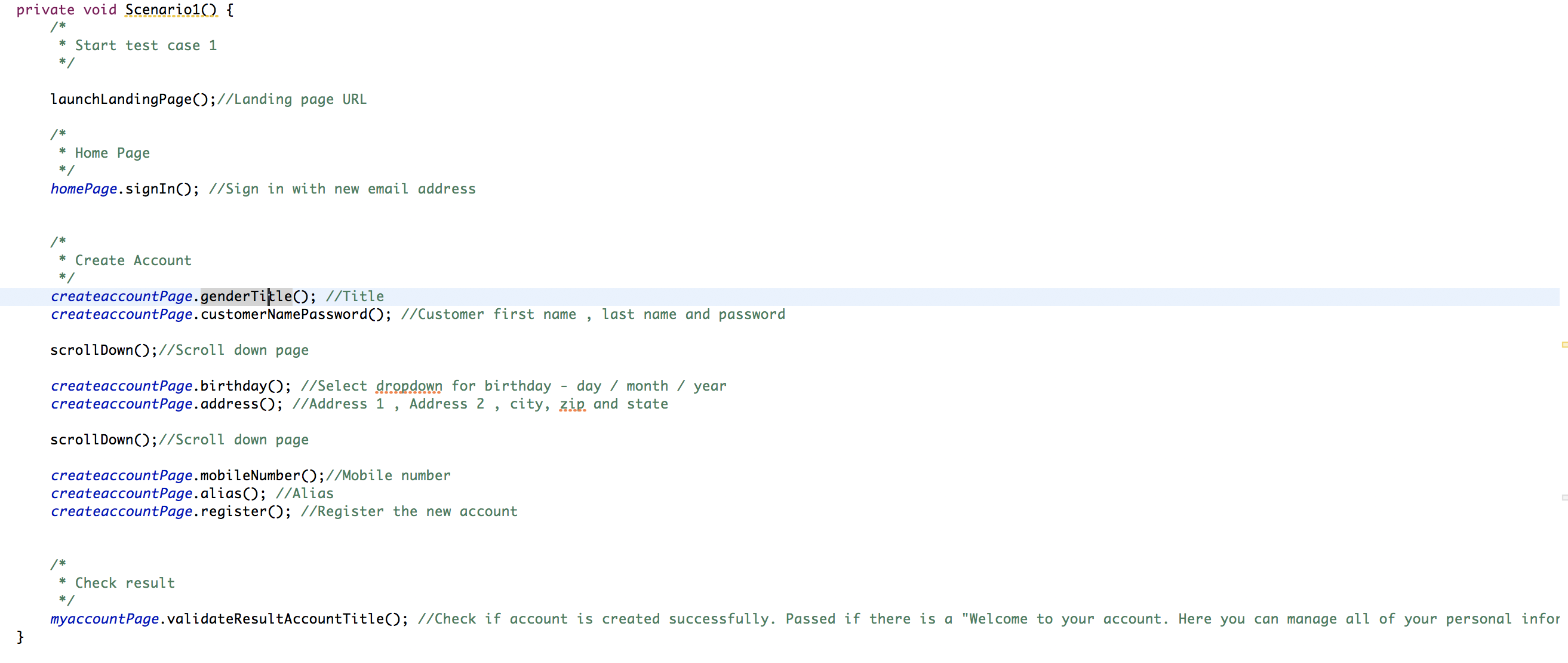
The BasePage will contain the validation and extend the Common class. It will also validate results for different page models.

Individual pages (with BasePage as extension) are break into individual classes and properties. We only need to call the properties from the page rather than calling WebDriver. The BasePage will also contain properties for validation and screen capture beside inheriting Common properties. The Page Object Manager (PageObjectManager) will handle all the data between the test scenario and the individual page objects.

All xpath are kept in LOCELEMENT which is called only by the page objects.

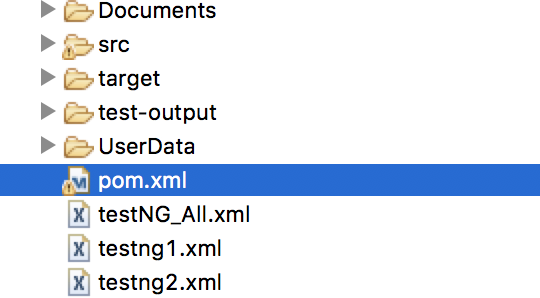
User data is using the Data Manager (ExcelExtractionManager). The data will be passed to individual page objects.

In scenario 1, we just need to call 3 page objects through the Page Object Manager (PageObjectManager) - home page (homePage) , the create account page (createaccountPage) and my account page (myaccountPage) and access all its keywords. We just need to maintain the page objects itself

Scenario 1 – much more readable than first version

**Execute the test**

There are few ways you can execute the 2 test cases – Scenario 1 and Scenario 2 by using either maven or TestNG plugin .



1. pom.xml – this will run the 2 test cases sequentially, in maven or at command

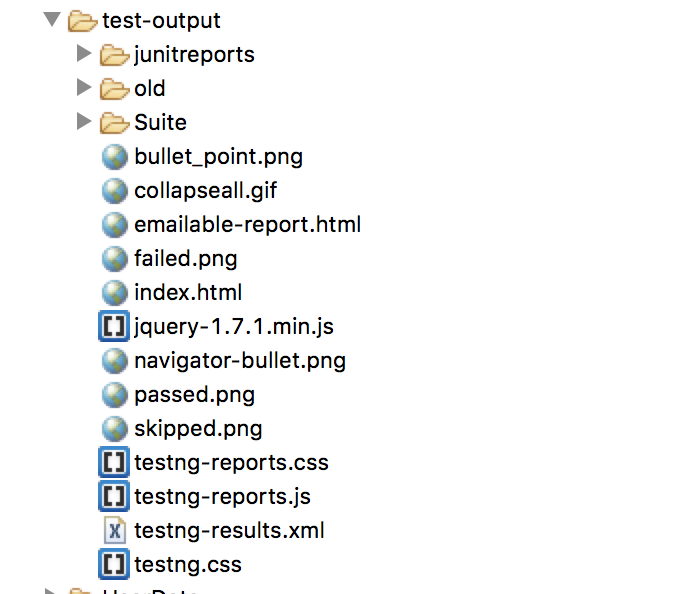
*mvn test*

1. testing\_All.xml this is similar as pom.xml – it will run 2 test cases
2. testng1.xml and testng2.xml – this will run individual test case

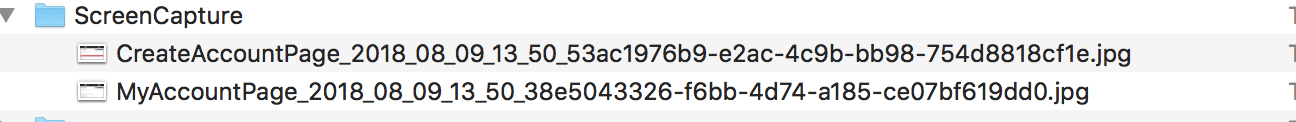
**Report**

Report can be found in your project folder

**\test-output\emailable-report.html**



Open the **emailable-report.html** file in your browser. You should get something like this for passed cases in next page:

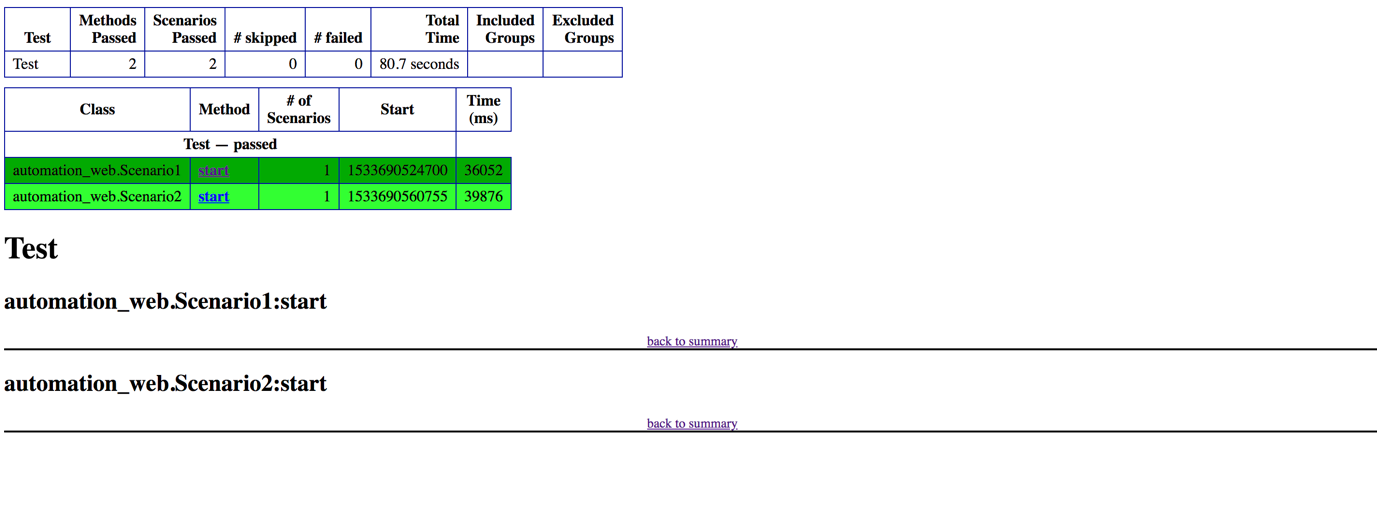


**Screen Capture Results**

Under the project folder \ScreenCapture

It will contain the screen capture images for the result in Create Account Page and My Account Page when you run the test scripts (1 & 2).

**emailable-report.html** – This is the summary results



One of the screen capture image result under \ScreenCapture

