Technical Assessment

Kavindu Dilshan
wgkdilshan@gmail.com
7-27-2022

Table of Contents

Preface

For this assignment, I will be using **Page Object Model** along with the **Data-Driven and Keyword Driven approach** to developing the Automation Framework from scratch to demonstrate my test design thinking as well as technical test automation skills.

Also, I have used the below technologies as well

- **Java**
- **TestNG**
- **Maven**
- **Extent Reports**
- **Log4j**
- **Apache poi**

1. Introduction

Purpose

This document defines the following **Eight** items

1. Prerequisites & Technologies Used
2. Pages & Objects of the Automation Framework
3. Keywords
4. Reusable Functions/methods of the automation framework
5. Test Suites & Test cases of the Test Automation project
6. Test Data
7. Configurations (TestNG Annotations)
8. How to Execute Test cases using **testng.xml** and generate the HTML report

Prerequisites

- Java should be installed and the java path also should be set (Java 1.8 used for this project)
- Automation Project is based on Maven and all the required artifacts can be easily updated by updating via maven

2. Pages & Objects of the Automation Framework

- All the pages will be defined **for each section** of the application following the pre-defined naming conversion (“PG_”)
- Also, there will be a Common Page (PG_Common) to maintain all the common object locators that will be shared throughout the Automation Framework
- Apart from the above-mentioned pages Login, related object locators will be maintained separately
- Similarly, all the other object locators will be maintained separately

Page Name	Automation Page Name
Common Page	PG_Common
Login Page	PG_Login
Home Page	PG_Home
Menu Page	PG_Menu

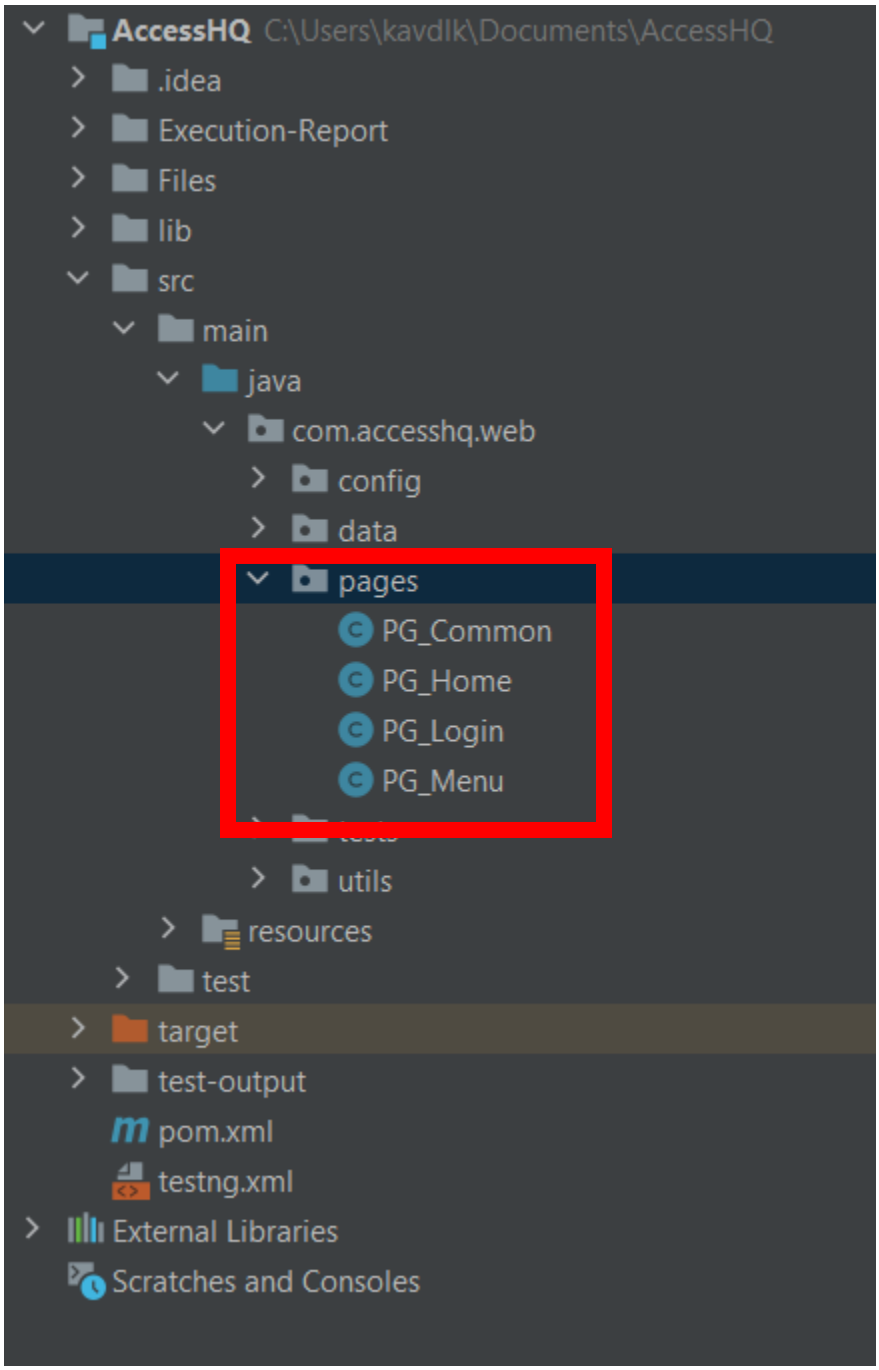
Naming Conventions used to declare Object locators:

Object Type	Naming Convention	Example
Page	PG_<Page Name>	PG_Login
Link	lnk_<Link Name>	lbl_FAQ
Button	btn_<Button Name>	btn_CommonByText
Check Box	chk_<Check Box Name>	chk_AcknowledgeAndConsent
Text Field	tf_<Text Field Name>	tf_UserName
Element	ele_<Element Name>	ele_SelectFiles
Label	Lbl_<Label Name>	lbl_CurrencyCode
Radio Button	rdo_<Radio Group Name>	rdo_CriminalLiabilityCheck
Drop down	dd_<dropdown name>	dd_Activity

- Pages are maintained as below in the Automation Project.

  Package: com.accesshq.web.pages

  Directory: AccessHQ\src\main\java\com\accesshq\web\pages



- Object locators for each page are stored based on the **Page Object Model**
- **Example:**
  Login Page (PG_Login)

PG_Login.java

```java
package com.accesshq.web.pages;

public class PG_Login {
    public static String lbl_Header() {
        return "//h3";
    }

    public static String tf_Username() {
        return "//label[text()='Username']/following-sibling::input";
    }

    public static String tf_Password() {
        return "//label[text()='Password']/following-sibling::input";
    }

    public static String btn_Signin() {
        return "//span[text()='Login']";
    }
}
```

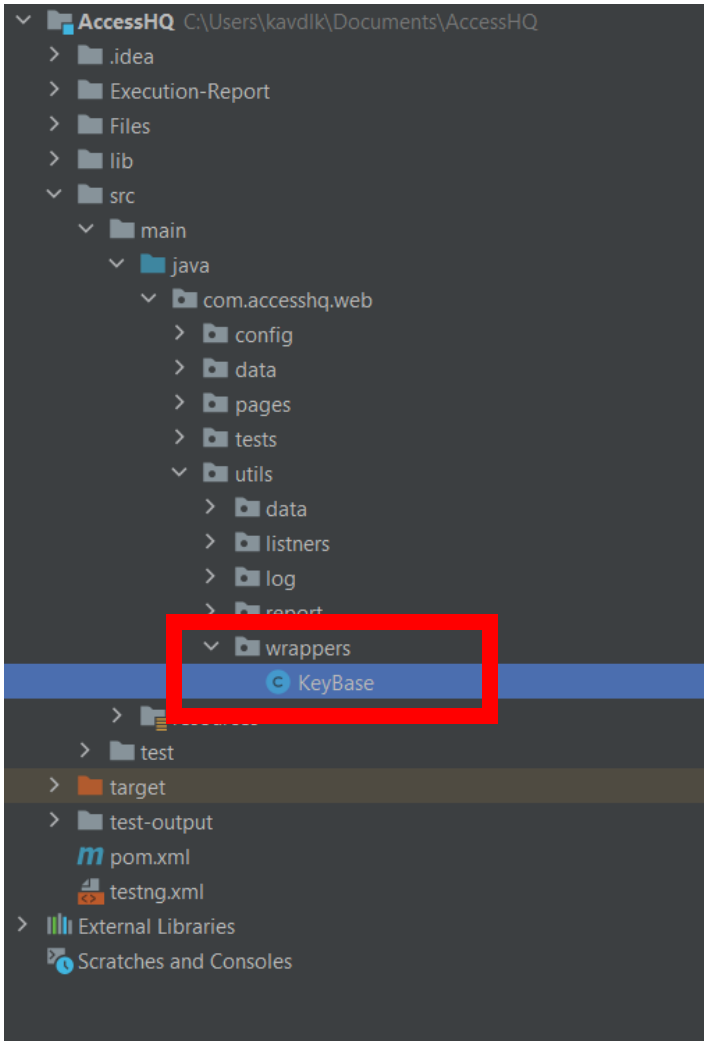
3. Keywords

- To perform browser actions such as **Clicking** on a certain element or to **Type** on a text field, separate keywords have been developed
- This helped the Automation Framework to minimize the scripting effort as well as handle exceptions easily
- Also, the methods we implement using these keywords can be easily readable

- **KeyBase** Class maintains all the Keywords in the Automation Project.

  Package: com. accesshq.web.wrappers

  Directory: AccessHQ\src\main\java\com\accesshq\web\wrappers



- **Example:**

  Click Command

```
public static void click(final String objectLocator) {
     //WebElement element = driver.findElement(By.xpath(objectLocator));
     WebDriverWait wait = new WebDriverWait(driver,  timeOutInSeconds: 20);
     By locator = By.xpath(objectLocator);

     WebElement element = wait.until(ExpectedConditions.presenceOfElementLocated(locator));

     try {
         pause( pauseTime: 3000);
         element.click();
     } catch (StaleElementReferenceException staleElementException) {
         element = driver.findElement(By.xpath(objectLocator));
     } catch (Exception ex) {
         ex.printStackTrace();
     }
     ExtentTestManager.getTest().log(Status.PASS,  details: "Clicked on Locator: "+objectLocator);
 }
```

- **Example:**

  Type Command

```
public static void type(final String objectLocator, final String text) {
    WebDriverWait wait = new WebDriverWait(driver,  timeOutInSeconds: 20);
    By locator = By.xpath(objectLocator);
    WebElement element = wait.until(ExpectedConditions.presenceOfElementLocated(locator));

    try {
        element.clear();
        element.sendKeys(text);
    } catch (StaleElementReferenceException staleElementException) {
        element = driver.findElement(By.xpath(objectLocator));
    } catch (Exception ex) {
       ex.printStackTrace();
    }

    ExtentTestManager.getTest().log(Status.PASS,  details: "Typed Text: "+text +" on Locator: "+objectLocator);
}
```

4. Reusable Libraries/methods of the automation framework

- All the reusable methods are categorized according to the main form sections and login functionalities
- Parameters will be used to send data to the method and can be reused in multiple numbers test cases by using different data for the same method
- When defining reusable methods relevant category will be appended to the method name
    - Example:

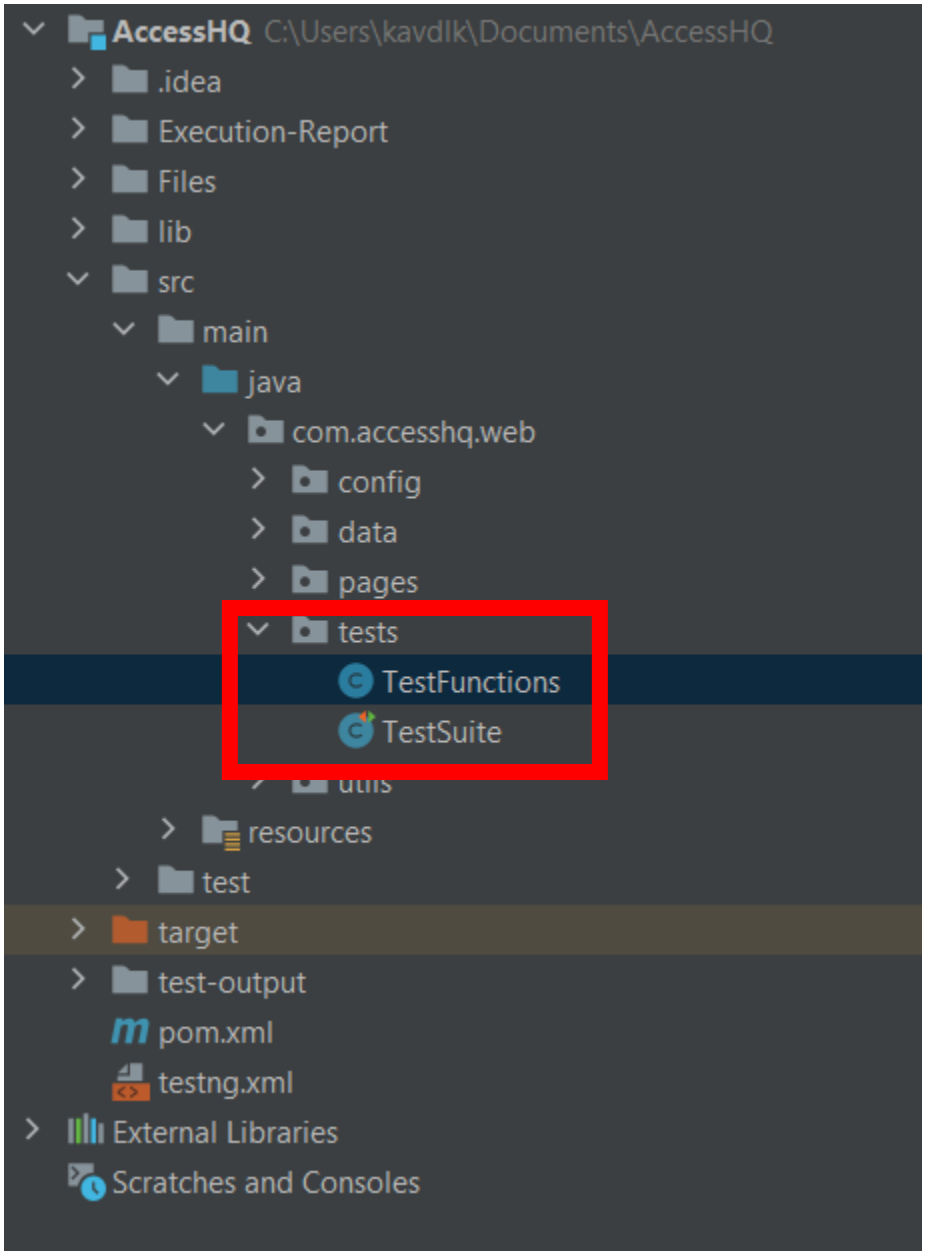
      bc_Login

      “bc_” – naming convention

- **TestFunctions** Class maintains all the Keywords in the Automation Project.

  Package: com.accesshq.web.tests

  Directory: AccessHQ\src\main\java\com\accesshq\web\tests



- **Example**:
  Login Functionality

  - URL, Username and Password are the parameters used for the Login method
  - Data for these parameters will be passed via excel sheet or property file
  - To implement the test steps **keywords** described earlier have been used

```
public static void bc_Login(String url, String Username, String Password) {
    writeToReport( comment: "Login scenario started");
    navigateToURL(url);
    click(PG_Login.btn_Signin());
    type(PG_Login.tf_Username(),Username);
    type(PG_Login.tf_Password(),Password);
    click(PG_Login.btn_Signin());
    writeToReport( comment: "Login scenario ended");
}
```

5. Test Suites & Test cases of the Test Automation project

- Test cases will be named with the predefined naming convention (“tc_”) followed by the test case number
    - Example:

      tc_01
- **tests** maintain all the TestSuites and Test cases in the Automation Project.

  Package: com. accesshq.web.tests

  Directory: AccessHQ\src\main\java\com\accesshq\web\tests

- Example:

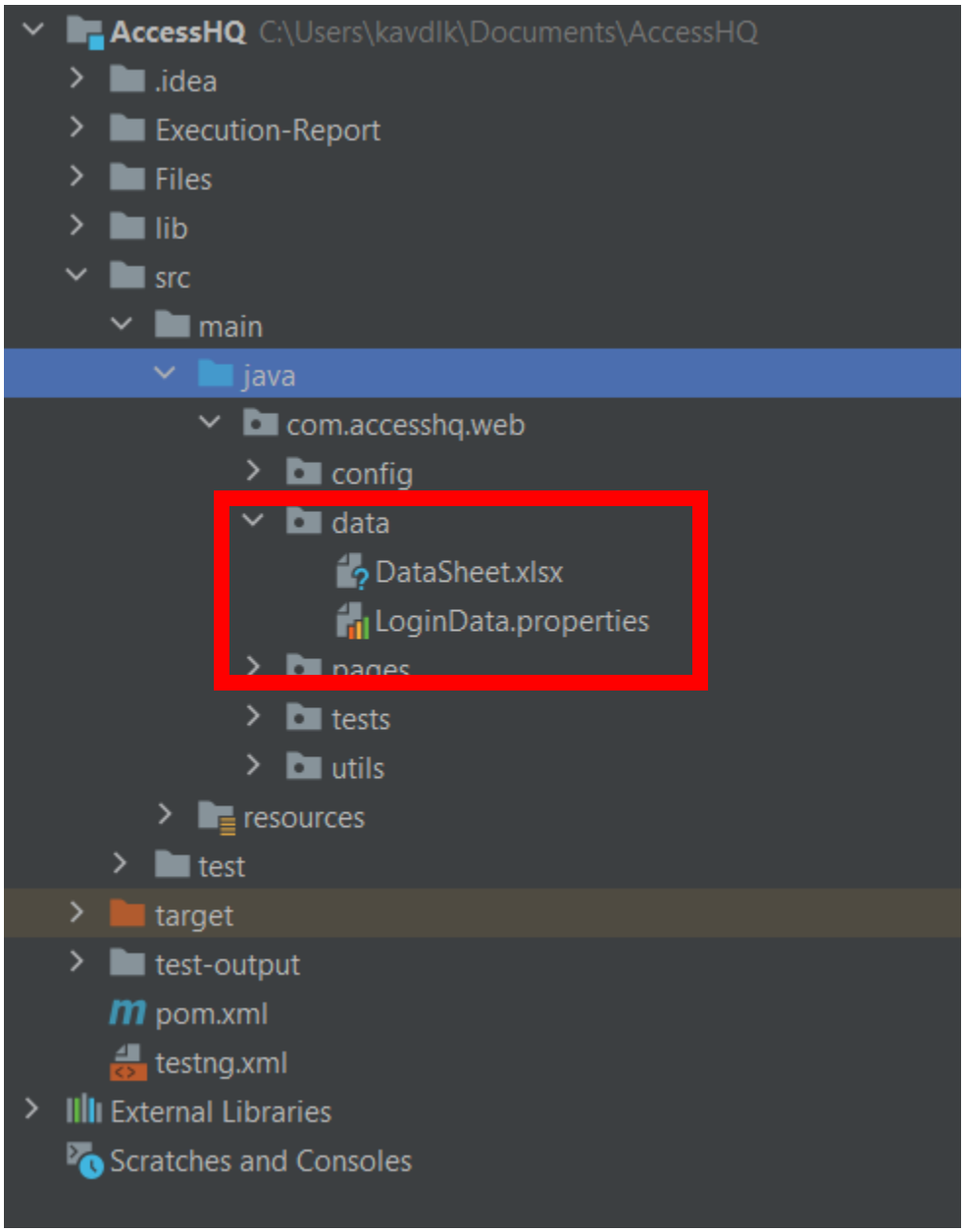
  tc_01

  - Test methods/ Functions developed will be called in the test cases
  - Using dataprovider relevant data will be passed

6. Test Data

- Each test case will have a Separate test data sheet in Excel to maintain test data that are unique to the test case
- There will be a property data file to manage the common data such as Login Details and Application URL

DataSheet.excel & LoginData.properties



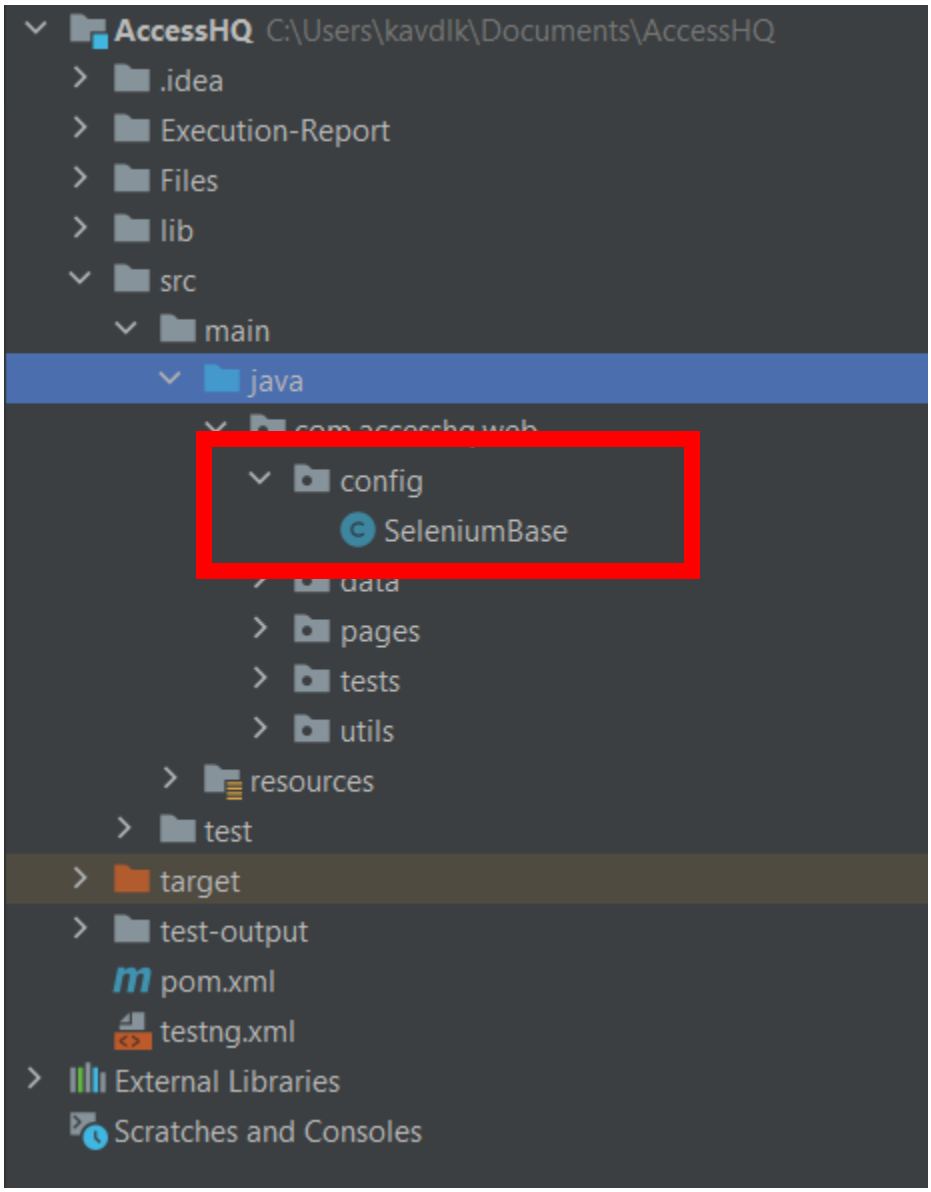
7. Configurations (TestNG Annotations)

TestNG Annotations are used to control the prerequisite steps before @test and what happens after the @test ends.

- **SeleniumBase** maintains all the TestSuites and Test cases in the Automation Project.

  Package: com.accesshq.web.config

  Directory: AccessHQ\src\main\java\com\accesshq\web\config



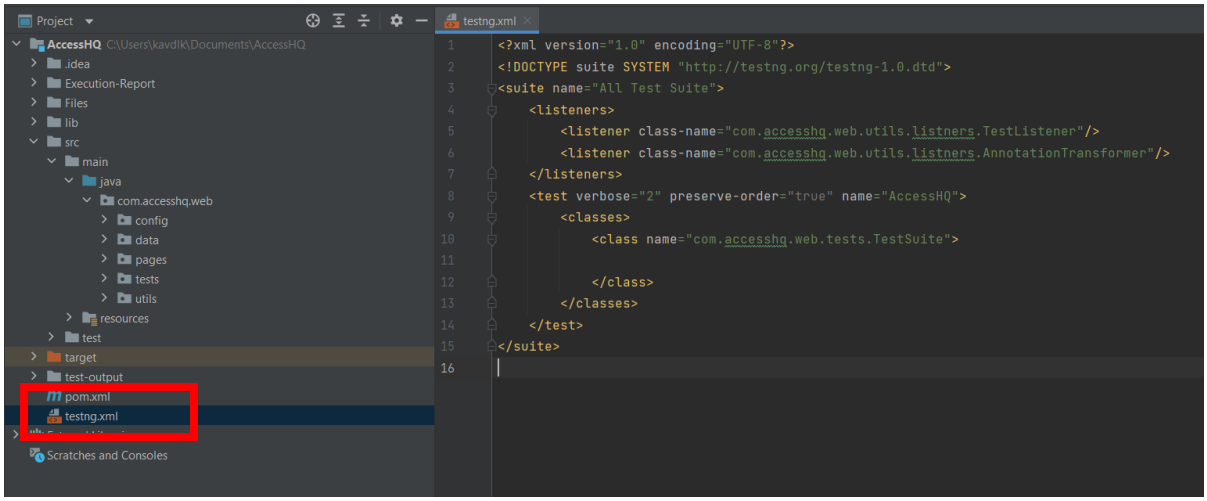
8. How to Execute Test cases using testng.xml and generate the HTML report

- **testing.xml** file is used to execute the test cases

  Package: com.accesshq.web.config

  Directory: AccessHQ\src\main\java\com\accesshq\web\config

testNG.xml



- To Execute Test cases simply **right click** on the testNG.xml file and click on **“Run”**

- Once the Execution is completed you can find the Execution Report in the below location