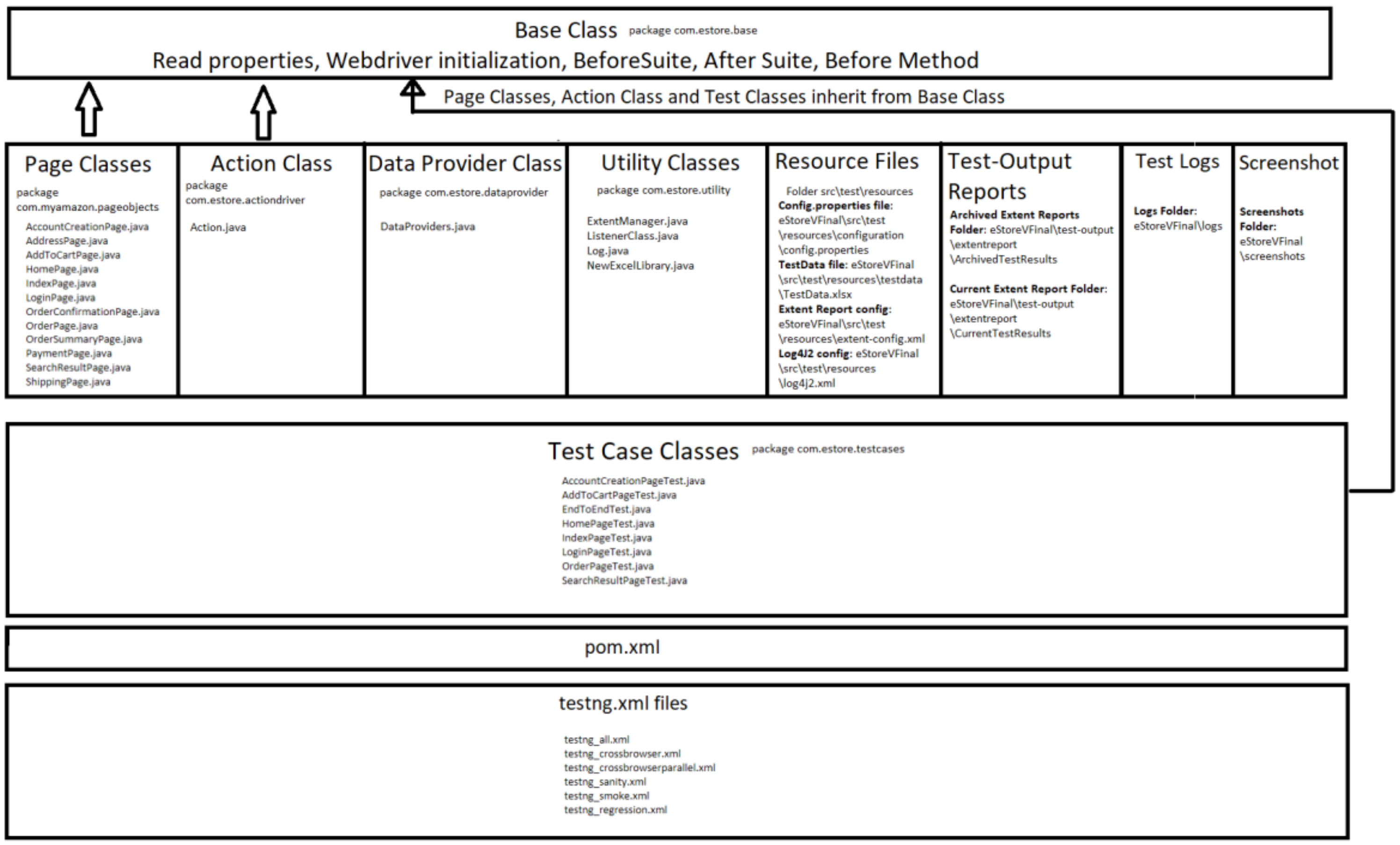
ReadMe

Thursday, June 9, 2022 9:43 PM

Project Structure:


Base Class package com.estore.base
Read properties, Webdriver initialization, BeforeSuite, After Suite, Before Method
Page Classes, Action Class and Test Classes inherit from Base Class
Page Classes
package com.myamazon.pageobjects
AccountCreationPage.java
AddressPage.java
AddToCartPage.java
HomePage.java
IndexPage.java
LoginPage.java
OrderConfirmationPage.java
OrderPage.java
OrderSummaryPage.java
PaymentPage.java
SearchResultPage.java
ShippingPage.java
Action Class
package com.estore.actiondriver
Action.java
Data Provider Class
package com.estore.dataprovider
DataProviders.java
Utility Classes
package com.estore.utility
ExtentManager.java
ListenerClass.java
Log.java
NewExcelLibrary.java
Resource Files
Folder src\test\resources
Config.properties file: eStoreVFinal\src\test\resources\configuration\config.properties
TestData file: eStoreVFinal\src\test\resources\testdata\TestData.xlsx
Extent Report config: eStoreVFinal\src\test\resources\extent-config.xml
Log4J2 config: eStoreVFinal\src\test\resources\log4j2.xml
Test-Output Reports
Archived Extent Reports Folder: eStoreVFinal\test-output\extentreport\ArchivedTestResults
Current Extent Report Folder: eStoreVFinal\test-output\extentreport\CurrentTestResults
Test Logs
Logs Folder: eStoreVFinal\logs
Screenshot
Screenshots Folder: eStoreVFinal\screenshots
Test Case Classes package com.estore.testcases
AccountCreationPageTest.java
AddToCartPageTest.java
EndToEndTest.java
HomePageTest.java
IndexPageTest.java
LoginPageTest.java
OrderPageTest.java
SearchResultPageTest.java
pom.xml
testng.xml files
testng_all.xml
testng_crossbrowser.xml
testng_crossbrowserparallel.xml
testng_sanity.xml
testng_smoke.xml
testng_regression.xml

Pre-requisites:

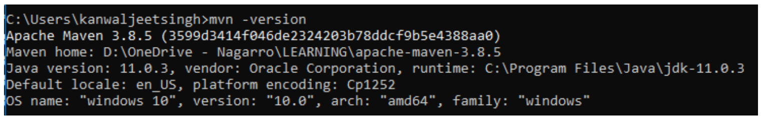
- A system running Windows 10 OR Windows 11
- A working Internet connection
- Access to an account with administrator privileges
- Access to the command prompt
- A copy of Java installed and ready to use, with the JAVA_HOME environment variable set up

Getting the system ready for running tests using CLI:

Step 1: Install Maven on Windows
Step 2: Verify Maven Installation

In the command prompt, use the following command to verify the installation by checking the current version of Maven:

mvn -version



```
C:\Users\kanwaljeetsingh>mvn -version
Apache Maven 3.8.5 (3599d3414f046de2324203b78ddcf9b5e4388aa0)
Maven home: D:\OneDrive - Nagarro\LEARNING\apache-maven-3.8.5
Java version: 11.0.3, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-11.0.3
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

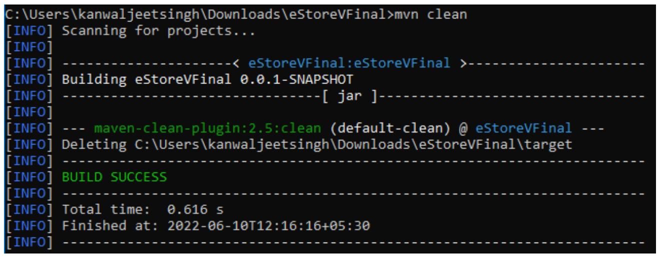
Running tests using CLI:

1. Copy the Project ZIP file on any directory on the system

2. Extract the zip file content on a location on hard drive
   We have extracted in C:\Users\kanwaljeetsingh\Downloads\eStoreVFinal

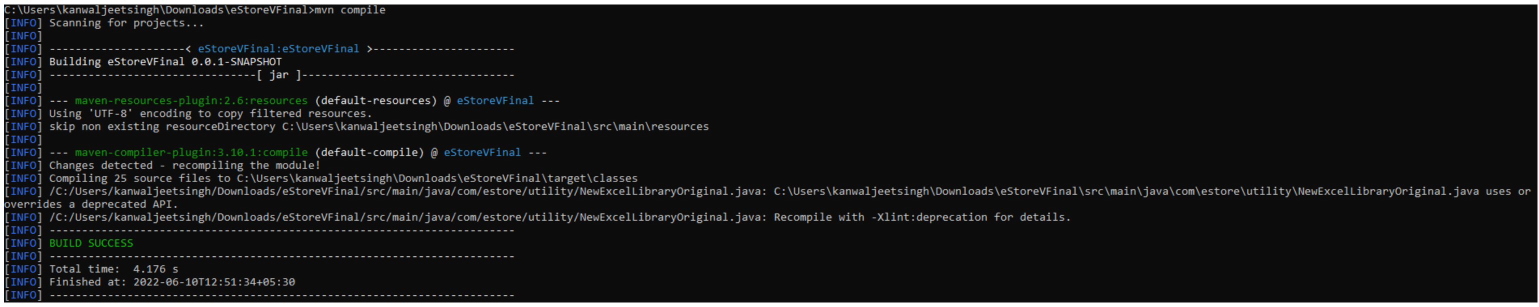
3. Use the address bar in the directory and type CMD
   Command Line Interface would open

4. Run the command **mvn clean** on CLI:



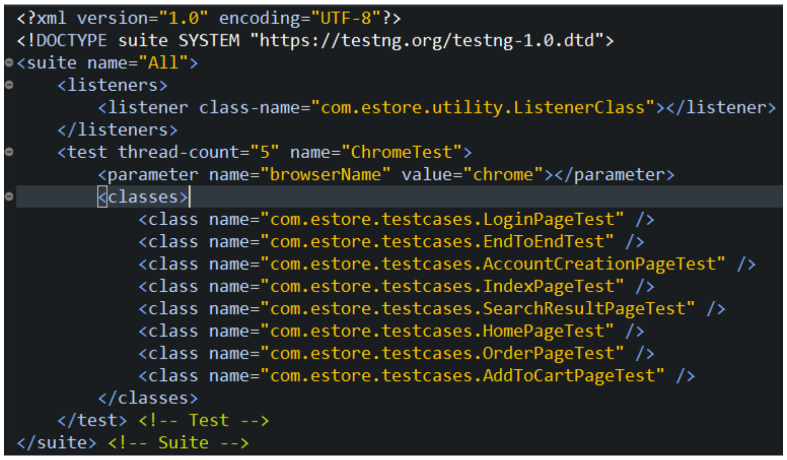
```
C:\Users\kanwaljeetsingh\Downloads\eStoreVFinal>mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------------< eStoreVFinal:eStoreVFinal >----------------------
[INFO] Building eStoreVFinal 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ eStoreVFinal ---
[INFO] Deleting C:\Users\kanwaljeetsingh\Downloads\eStoreVFinal\target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.616 s
[INFO] Finished at: 2022-06-10T12:16:16+05:30
[INFO] ------------------------------------------------------------------------
```

5. Run the command **mvn compile** on CLI:



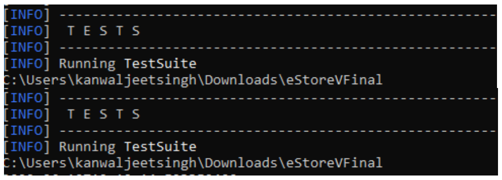
```
C:\Users\kanwaljeetsingh\Downloads\eStoreVFinal>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< eStoreVFinal:eStoreVFinal >---------------------
[INFO] Building eStoreVFinal 0.0.1-SNAPSHOT
[INFO] -------------------------------[ jar ]-------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ eStoreVFinal ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\kanwaljeetsingh\Downloads\eStoreVFinal\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.10.1:compile (default-compile) @ eStoreVFinal ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 25 source files to C:\Users\kanwaljeetsingh\Downloads\eStoreVFinal\target\classes
[INFO] /C:/Users/kanwaljeetsingh/Downloads/eStoreVFinal/src/main/java/com/estore/utility/NewExcelLibraryOriginal.java: C:\Users\kanwaljeetsingh\Downloads\eStoreVFinal\src\main\java\com\estore\utility\NewExcelLibraryOriginal.java uses or
overrides a deprecated API.
[INFO] /C:/Users/kanwaljeetsingh/Downloads/eStoreVFinal/src/main/java/com/estore/utility/NewExcelLibraryOriginal.java: Recompile with -Xlint:deprecation for details.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.176 s
[INFO] Finished at: 2022-06-10T12:51:34+05:30
[INFO] ------------------------------------------------------------------------
```

6. Run the command **mvn test** on CLI
   This would start the test run using the testng_all.xml file in the project directory
   This file is set to execute all test cases available in the project:

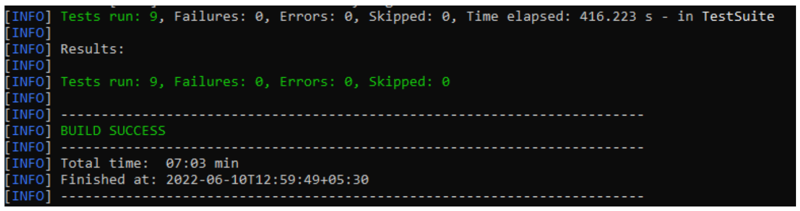


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="All">
    <listeners>
        <listener class-name="com.estore.utility.ListenerClass"></listener>
    </listeners>
    <test thread-count="5" name="ChromeTest">
        <parameter name="browserName" value="chrome"></parameter>
        <classes>
            <class name="com.estore.testcases.LoginPageTest" />
            <class name="com.estore.testcases.EndToEndTest" />
            <class name="com.estore.testcases.AccountCreationPageTest" />
            <class name="com.estore.testcases.IndexPageTest" />
            <class name="com.estore.testcases.SearchResultPageTest" />
            <class name="com.estore.testcases.HomePageTest" />
            <class name="com.estore.testcases.OrderPageTest" />
            <class name="com.estore.testcases.AddToCartPageTest" />
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="All">
	<listeners>
		<listener class-name="com.estore.utility.ListenerClass"></listener>
	</listeners>
	<test thread-count="5" name="ChromeTest">
		<parameter name="browserName" value="chrome"></parameter>
		<classes>
			<class name="com.estore.testcases.LoginPageTest" />
			<class name="com.estore.testcases.EndToEndTest" />
			<class name="com.estore.testcases.AccountCreationPageTest" />
			<class name="com.estore.testcases.IndexPageTest" />
			<class name="com.estore.testcases.SearchResultPageTest" />
			<class name="com.estore.testcases.HomePageTest" />
			<class name="com.estore.testcases.OrderPageTest" />
			<class name="com.estore.testcases.AddToCartPageTest" />
		</classes>
	</test> <!-- Test -->
</suite> <!-- Suite -->
```



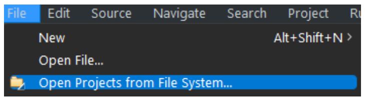
```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running TestSuite
C:\Users\kanwaljeetsingh\Downloads\eStoreVFinal
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running TestSuite
C:\Users\kanwaljeetsingh\Downloads\eStoreVFinal
```



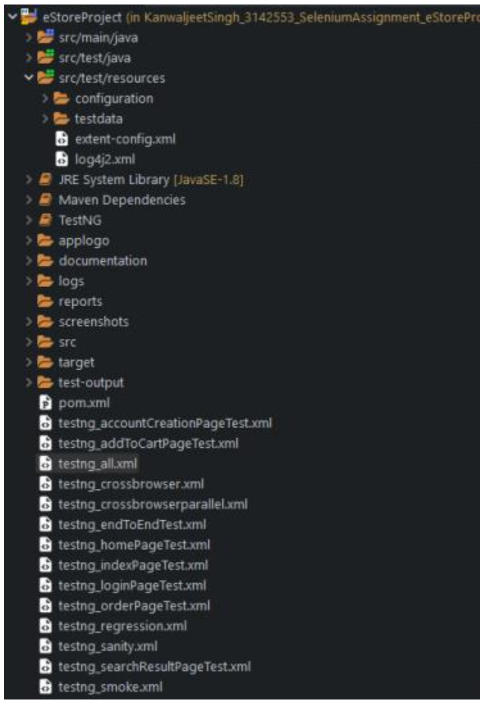
```
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 416.223 s - in TestSuite
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  07:03 min
[INFO] Finished at: 2022-06-10T12:59:49+05:30
[INFO] ------------------------------------------------------------------------
```

Running tests using TestNG.xml files:

1. Open the project in Eclipse using Open Projects from File System option under File menu:



2. Select the folder - eStoreProject from the location where the ZIP file is extracted
3. On opening, Project structure is shown as:



```
eStoreProject (in KanwaljeetSingh_3142553_SeleniumAssignment_eStorePr
  src/main/java
  src/test/java
  src/test/resources
    configuration
    testdata
    extent-config.xml
    log4j2.xml
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  TestNG
  applogo
  documentation
  logs
  reports
  screenshots
  src
  target
  test-output
  pom.xml
  testng_accountCreationPageTest.xml
  testng_addToCartPageTest.xml
  testng_all.xml
  testng_crossbrowser.xml
  testng_crossbrowserparallel.xml
  testng_endToEndTest.xml
  testng_homePageTest.xml
  testng_indexPageTest.xml
  testng_loginPageTest.xml
  testng_orderPageTest.xml
  testng_regression.xml
  testng_sanity.xml
  testng_searchResultPageTest.xml
  testng_smoke.xml
```

4. Project is divided into the following components:
   a. Project design phase:
      i. Base Class component -
         1) Package - com.estore.base
         2) Classes:
            a) BaseClass.java
      ii. Action Driver component -
         1) Package - com.estore.actiondriver
         2) Classes:
            a) Action.java
      iii. Page Classes component
         1) Package - com.estore.pageobjects
         2) Classes:
            a) AccountCreationPage.java
            b) AddressPage.java
            c) AddToCartPage.java
            d) HomePage.java
            e) LoginPage.java
            f) OrderConfirmationPage.java
            g) OrderPage.java
            h) orderSummaryPage.java
            i) PaymentPage.java
            j) SearchResultPage.java
            k) ShippingPage.java
      iv. Test Classes component
         1) AccountCreationPageTest.java
         2) AddToCartPageTest.java
         3) EndToEndPageTest.java
         4) HomePageTest.java
         5) IndexPageTest.java
         6) LoginPageTest.java
         7) OrderPageTest.java
         8) SearchResultPageTest.java
      v. Data Provider Class
         1) Package - com.estore.dataprovider
         2) Classes:
            a) DataProviders.java
      vi. Utility Classes component
         1) Package - com.estore.utility

2) Classes:
    a) ExtentManager.java
    b) ListenerClass.java
    c) Log.java
    d) NewExcelLibrary.java

b. Project execution phase:
   i. pom.xml
   ii. testng.xml files for test execution:
      1) Test case specific:
         a) testng_accountCreationPageTest.xml
         b) testng_addToCartPageTest.xml
         c) testng_endToEndTest.xml
         d) testng_homePageTest.xml
         e) testng_indexPageTest.xml
         f) testng_loginPageTest.xml
         g) testng_orderPageTest.xml
         h) testng_searchResultPageTest.xml
      2) Group specific:
         a) testng_smoke.xml
         b) testng_sanity.xml
         c) testng_regression.xml
      3) All test cases:
         a) testng_all.xml - **This file is also set to run using the mvn test command using CLI**
      4) Parallel execution:
         a) **testng_crossbrowserparallel - This file is set to execute *IndexPageTest* in browsers - Chrome, Firefox, Edge**
   iii. **logs** folder to store applogs in a file named appLog.log
   iv. **screenshots** folder to store screenshots in case of failures
   v. **test-output\extentreport** folder to store extent reports as:
      1) Current run report in CurrentTestResults folder
      2) Past run reports in ArchivedTestResults folder

Project Requirements and Implementation:

Following is the list of project requirements and how they have been implemented:

Site used: **http://automationpractice.com/**

- **Create at least 8 test cases across different pages**
  a. Test Cases have been created in the Test Classes component
     Following classes have been created, each containing one test case:
     i. AccountCreationPageTest.java
     ii. AddToCartPageTest.java
     iii. EndToEndPageTest.java
     iv. HomePageTest.java
     v. IndexPageTest.java
     vi. LoginPageTest.java
     vii. OrderPageTest.java
     viii. SearchResultPageTest.java
- **Create a Page Object Model Framework using TestNG to implement these test cases**
  a. Base Class, Action Class, Utility Classes and Page Classes have been created in the Base Class and Page Classes components
     PageFactory is used to initialize elements on pages
     It is ensured, the Base Class, Action Class, Utility Classes and Test Classes are kept in separate packages
     Following classes have been created:
     i. Base Class component -
        1) Package - com.estore.base
        2) Classes:
           a) BaseClass.java
     ii. Action Driver component -
        1) Package - com.estore.actiondriver
        2) Classes:
           a) Action.java
     iii. Utility Classes component
        1) Package - com.estore.utility
        2) Classes:
           a) ExtentManager.java
           b) ListenerClass.java
           c) Log.java
           d) NewExcelLibrary.java
     iv. Page Classes component
        1) Package - com.estore.pageobjects
        2) Classes:
           a) AccountCreationPage.java
           b) AddressPage.java
           c) AddToCartPage.java
           d) HomePage.java
           e) LoginPage.java
           f) OrderConfirmationPage.java
           g) OrderPage.java
           h) orderSummaryPage.java
           i) PaymentPage.java
           j) SearchResultPage.java
           k) ShippingPage.java
- **The framework should have following features**
- **Test Data should be read from properties file**

  a. config.properties file is used to store the following data
```
email = test12312312312@gmail.com
password = ABC123
url = http://automationpractice.com/index.php
expectedUrl = http://automationpractice.com/index.php?controller=my-account
expectedMessage = Your order on My Store is complete.
implicitWait= 10
explicitWait = 10
pageLoadTimeOut=40
```

  b. config.properties file is loaded in the @BeforeSuite section in BaseClass.java:

```
@BeforeSuite(groups = { "Smoke", "Sanity", "Regression" })
public void beforeSuite() throws IOException {
    ExtentManager.startReport();
    prop = new Properties();
    FileInputStream fileInput =
            new FileInputStream(System.getProperty("user.dir") + "\\src\\test\\resources\\configuration\\config.properties");
    System.out.println(System.getProperty("user.dir"));
    prop.load(fileInput);
}
```

- **Use Maven as build execution tool**
  a. Project created is using the Maven project template and has the pom.xml file to download dependencies and plugins
  b. Pom.xml is located under root folder of the project
  c. It has the scope **compile** under the TestNG dependency:

```
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.4.0</version>
    <scope>compile</scope>
</dependency>
```

  d. Using the Maven menu, the project can be compiled and tests run
  This is also depicted in the **Running tests using CLI** part of this document

- **Implement proper waits.**
  a. Following waits and timeouts have been incorporated in the project:
     i. Explicit Wait
     ii. Implicit Wait
     iii. Page Load Timeout
  b. Implicit Wait is used in BaseClass.java when loading URL in browser:

```
// Implicit TimeOuts
getDriver().manage().timeouts().implicitlyWait(Integer.parseInt(prop.getProperty("implicitWait")),
        TimeUnit.SECONDS);
```

  c. PageLoadTimeout is used in BaseClass.java when loading URL in browser:

```
// PageLoad TimeOuts
getDriver().manage().timeouts().pageLoadTimeout(Integer.parseInt(prop.getProperty("pageLoadTimeOut")),
        TimeUnit.SECONDS);
```

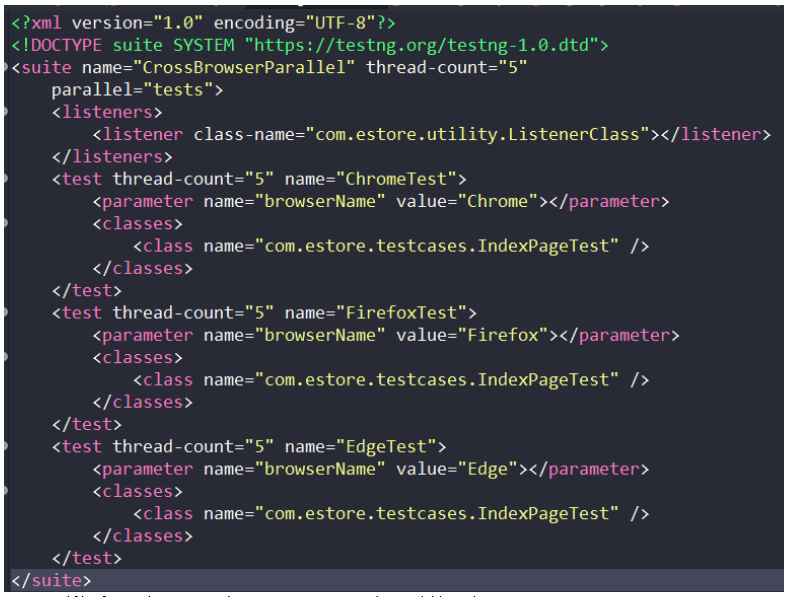
  d. Explicit Wait is used in the methods used in Action.java class in the click and isDisplayed methods:

```
WebDriverWait wait = new WebDriverWait(getDriver(), Integer.parseInt(prop.getProperty("explicitWait")));
wait.until(ExpectedConditions.visibilityOf(ele));
```

- **The global configuration values like browser name, test site URL, global wait value etc. should be read from a properties file**
  a. **config.properties** file is used to store the following data
     email = test12312312312@gmail.com
     password = ABC123
     url = http://automationpractice.com/index.php
     expectedUrl = http://automationpractice.com/index.php?controller=my-account
     expectedMessage = Your order on My Store is complete.
     implicitWait= 10
     explicitWait = 10
     pageLoadTimeOut=40
  b. **TestData.xlsx** file is stored under **src\test\resources\testdata** folder and used to store the following data
     Sheet names in excel file are given below
     Data available in sheets and test cases in which it is being used is given below
     - **Credentials** - username and password for loginTest, orderHistoryTest
     - **AccountCreationEmail** - accountCreationEmail for accountCreatePageTest
     - **SearchProduct** - searchProduct for addToCartTest , endToEndTest , totalPriceTest , productAvailabilityTest
     - **ProductDetails** - productName, qty and size for addToCartTest , endToEndTest, totalPriceTest
     - **ExpectedStoreTitle** - expectedStoreTitle for storeTitleTest
- **The test should run on following browsers Chrome, FF, IE (Safari, in case using Mac machine)**
- **Create multiple testing.xml file e.g., implement parallel execution, grouping, listeners etc**
  a. Tests are running on multiple browsers in BaseClass.java:
  b. Browser configuration is made available using Parameters annotation

```
@Parameters("browserName")
@BeforeMethod(groups = { "Smoke", "Sanity", "Regression" })
public void setup(String browserName) throws IOException, InterruptedException {
    openApplication(browserName);
}
```

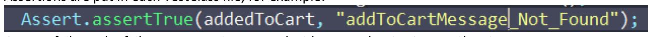
  c. **testng_crossbrowserparallel.xml** file is used to run the tests in parallel in Browsers - Chrome, Firefox and Edge
  d. Only **com.estore.testcases.IndexPageTest** is configured to run using this file and multiple cases can be included as required:



```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="CrossBrowserParallel" thread-count="5"
    parallel="tests">
    <listeners>
        <listener class-name="com.estore.utility.ListenerClass"></listener>
    </listeners>
    <test thread-count="5" name="ChromeTest">
        <parameter name="browserName" value="Chrome"></parameter>
        <classes>
            <class name="com.estore.testcases.IndexPageTest" />
        </classes>
    </test>
    <test thread-count="5" name="FirefoxTest">
        <parameter name="browserName" value="Firefox"></parameter>
        <classes>
            <class name="com.estore.testcases.IndexPageTest" />
        </classes>
    </test>
    <test thread-count="5" name="EdgeTest">
        <parameter name="browserName" value="Edge"></parameter>
        <classes>
            <class name="com.estore.testcases.IndexPageTest" />
        </classes>
    </test>
</suite>
```

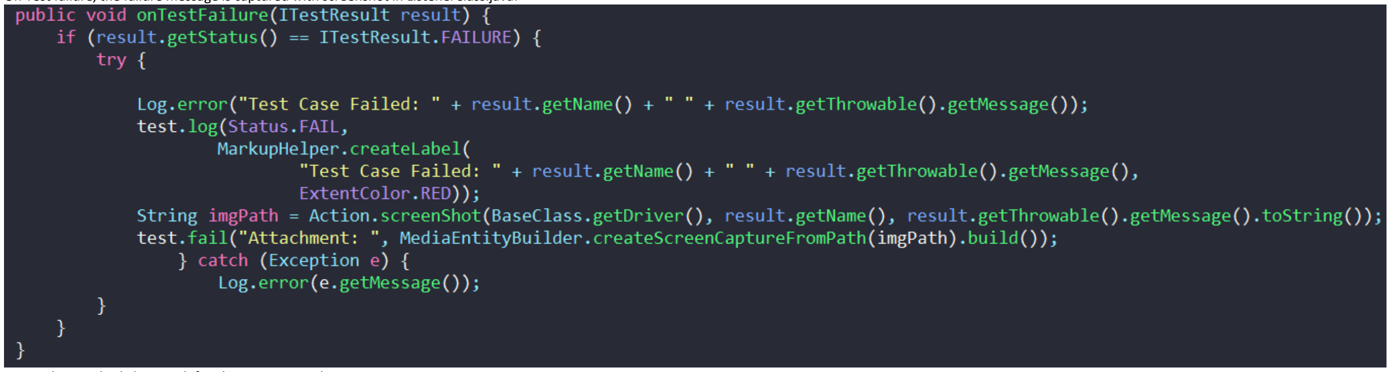
e. Testng.xml files for smoke, sanity and regression groups are also available in the project:
   i. testng_regression.xml
   ii. testng_sanity.xml
   iii. testng_smoke.xml

- **Put proper assertions with error description and if any test cases fail take a screenshot with the name same as test case and appended by a brief description of error in the screenshot file name (For e.g., TestCase1_Invalid_Credentials).**
  a. Assertions are put in each TestClass file, for example:



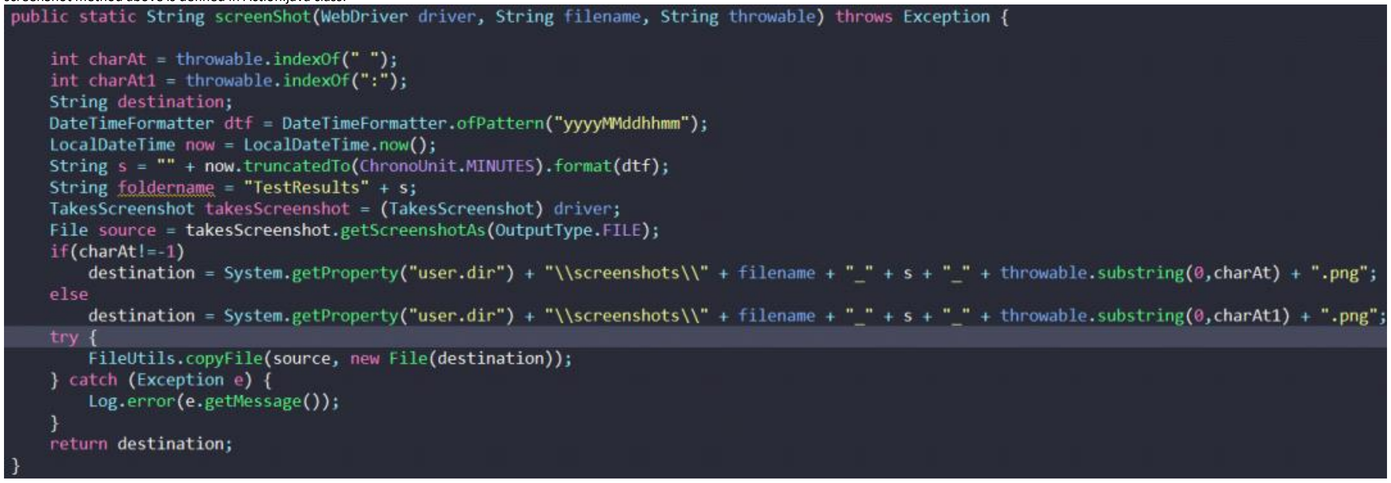
```java
Assert.assertTrue(addedToCart, "addToCartMessage_Not_Found");
```

  b. On Test failure, the failure message is captured with screenshot in ListenerClass.java:



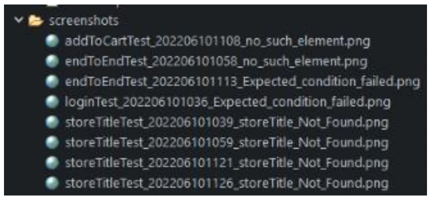
```java
public void onTestFailure(ITestResult result) {
    if (result.getStatus() == ITestResult.FAILURE) {
        try {

            Log.error("Test Case Failed: " + result.getName() + " " + result.getThrowable().getMessage());
            test.log(Status.FAIL,
                    MarkupHelper.createLabel(
                            "Test Case Failed: " + result.getName() + " " + result.getThrowable().getMessage(),
                            ExtentColor.RED));
            String imgPath = Action.screenShot(BaseClass.getDriver(), result.getName(), result.getThrowable().getMessage().toString());
            test.fail("Attachment: ", MediaEntityBuilder.createScreenCaptureFromPath(imgPath).build());
                } catch (Exception e) {
                    Log.error(e.getMessage());
        }
    }
}
```

  c. screenShot method above is defined in Action.java class:



```java
public static String screenShot(WebDriver driver, String filename, String throwable) throws Exception {

    int charAt = throwable.indexOf(" ");
    int charAt1 = throwable.indexOf(":");
    String destination;
    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyyMMddhhmm");
    LocalDateTime now = LocalDateTime.now();
    String s = "" + now.truncatedTo(ChronoUnit.MINUTES).format(dtf);
    String foldername = "TestResults" + s;
    TakesScreenshot takesScreenshot = (TakesScreenshot) driver;
    File source = takesScreenshot.getScreenshotAs(OutputType.FILE);
    if(charAt!=-1)
        destination = System.getProperty("user.dir") + "\\screenshots\\" + filename + "_" + s + "_" + throwable.substring(0,charAt) + ".png";
    else
        destination = System.getProperty("user.dir") + "\\screenshots\\" + filename + "_" + s + "_" + throwable.substring(0,charAt1) + ".png";
    try {
        FileUtils.copyFile(source, new File(destination));
    } catch (Exception e) {
        Log.error(e.getMessage());
    }
    return destination;
}
```

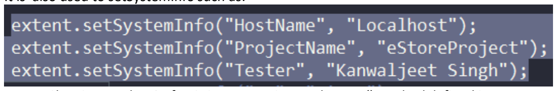
This would enable saving of screenshots with a short description included in screenshot file name:



```
screenshots
    addToCartTest_202206101108_no_such_element.png
    endToEndTest_202206101058_no_such_element.png
    endToEndTest_202206101113_Expected_condition_failed.png
    loginTest_202206101036_Expected_condition_failed.png
    storeTitleTest_202206101039_storeTitle_Not_Found.png
    storeTitleTest_202206101059_storeTitle_Not_Found.png
    storeTitleTest_202206101121_storeTitle_Not_Found.png
    storeTitleTest_202206101126_storeTitle_Not_Found.png
```

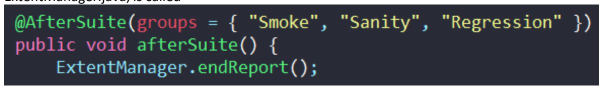
- **Create Extent report. Customize the report to append error screenshots. At the end of a test execution, move the results (the extent report and error screenshots) to a folder with the name "Current test results". The folder name should reflect the date time of the run. At the start of the results the result of the current test result should move to a folder with the name "Archived test results" and clean the "Current test results" folder to store the new results.**

  a. ExtentManager.java under com.estore.utility is used to initialize htmlReporter and load the extent-config.xml under **src\test\resources**
  b. This uses ExtentHtmlReporter object to load the extent-config.xml located at src\test \resources folder
  c. It is also used to setSystemInfo such as:

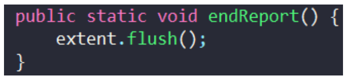


```
extent.setSystemInfo("HostName", "Localhost");
extent.setSystemInfo("ProjectName", "eStoreProject");
extent.setSystemInfo("Tester", "Kanwaljeet Singh");
```

  d. In BaseClass.java under @AfterSuite annotation, endReport() method defined in ExtentManager.java, is called

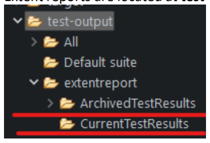


```
@AfterSuite(groups = { "Smoke", "Sanity", "Regression" })
public void afterSuite() {
    ExtentManager.endReport();
```



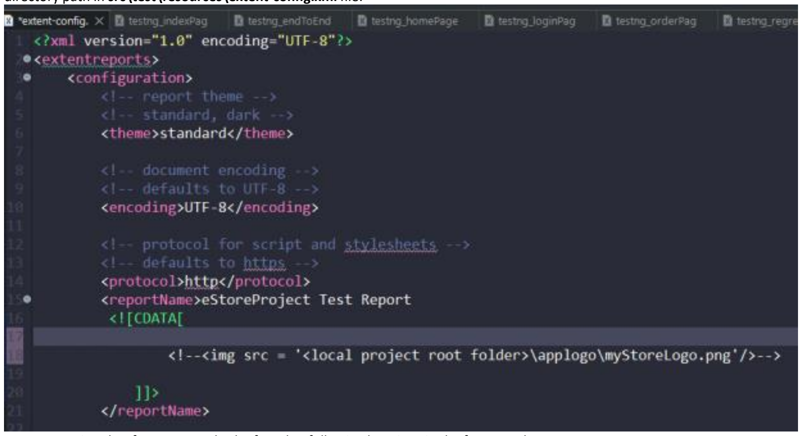
```
public static void endReport() {
    extent.flush();
}
```

Extent reports are located at **test-output\extentreport\**:



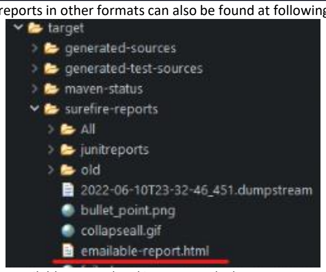
```
test-output
  All
  Default suite
  extentreport
    ArchivedTestResults
    CurrentTestResults
```

  e. On start of every test cycle, the report from **CurrentTestResults** gets moved to **ArchivedTestResults** and when the test completes, the current report gets generated in **CurrentTestResults** folder
  f. There is an appStoreLogo.png file located in applogo folder under project root directory This logo can be made to display in the generated extent report by passing the project local directory path in **src\test\resources\extent-config.xml** file:



```
<?xml version="1.0" encoding="UTF-8"?>
<extentreports>
    <configuration>
        <!-- report theme -->
        <!-- standard, dark -->
        <theme>standard</theme>

        <!-- document encoding -->
        <!-- defaults to UTF-8 -->
        <encoding>UTF-8</encoding>

        <!-- protocol for script and stylesheets -->
        <!-- defaults to https -->
        <protocol>http</protocol>
        <reportName>eStoreProject Test Report
         <![CDATA[

                <!--<img src = '<local project root folder>\applogo\myStoreLogo.png'/>-->

            ]]>
        </reportName>
```

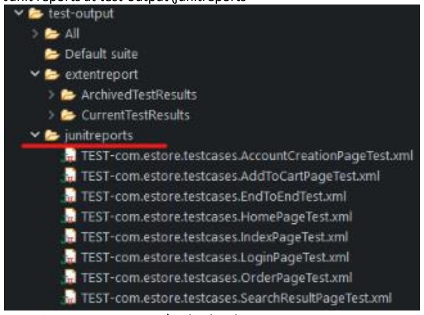
  g. Test reports in other formats can also be found at following locations in the framework:
     i.



```
target
  generated-sources
  generated-test-sources
  maven-status
  surefire-reports
    All
    junitreports
    old
    2022-06-10T23-32-46_451.dumpstream
    bullet_point.png
    collapseall.gif
    emailable-report.html
```

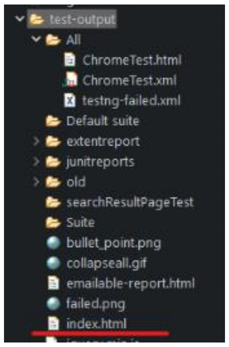
emailable-report.html is generated when tests are run from CLI using **mvn test** command

     ii. Junit reports at test-output\junitreports



```
test-output
  All
  Default suite
  extentreport
    ArchivedTestResults
    CurrentTestResults
  junitreports
    TEST-com.estore.testcases.AccountCreationPageTest.xml
    TEST-com.estore.testcases.AddToCartPageTest.xml
    TEST-com.estore.testcases.EndToEndTest.xml
    TEST-com.estore.testcases.HomePageTest.xml
    TEST-com.estore.testcases.IndexPageTest.xml
    TEST-com.estore.testcases.LoginPageTest.xml
    TEST-com.estore.testcases.OrderPageTest.xml
    TEST-com.estore.testcases.SearchResultPageTest.xml
```

     iii. TestNG report: test-output\index.html

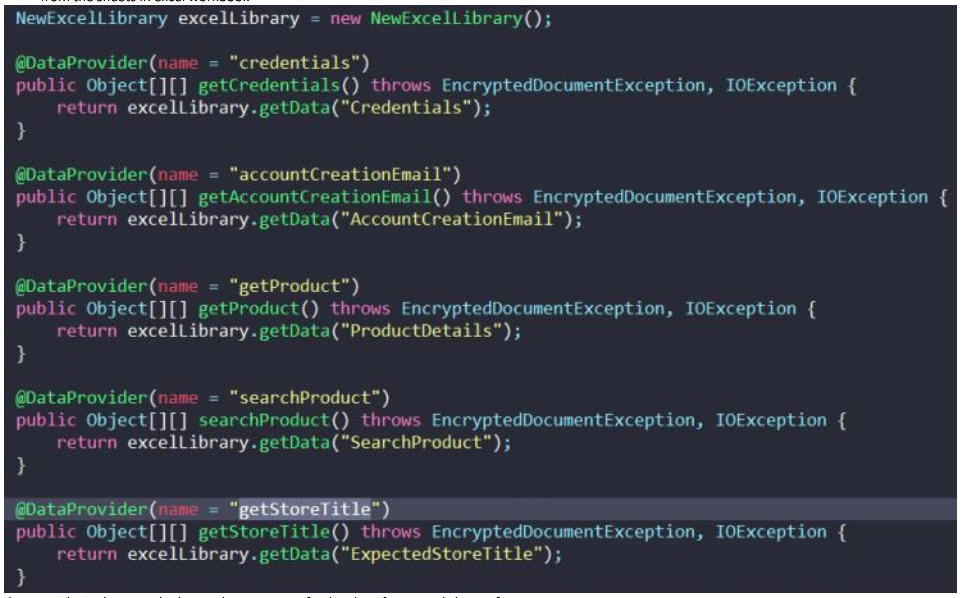


```
test-output
  All
    ChromeTest.html
    ChromeTest.xml
    testng-failed.xml
  Default suite
  extentreport
  junitreports
  old
  searchResultPageTest
  Suite
  bullet_point.png
  collapseall.gif
  emailable-report.html
  failed.png
  index.html
```

h. In the current project structure, there are sample reports placed to check the report view in case of **passed, failed** and **skipped** test cases

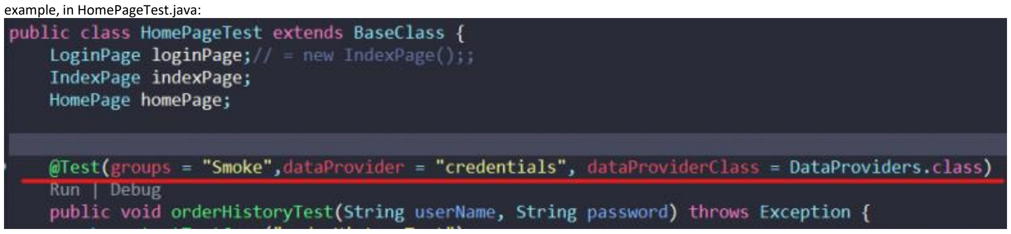
- **The tests should be runnable from command line**
  a. This is depicted in the Running tests using CLI part of this document above

- **Create at least one test using Data provider. (Bonus points for Excel Integration and reading data from Excel sheets.)**
  a. **TestData.xlsx** file is stored under **src\test\resources\testdata** folder and used to store the following data
     Sheet names in excel file are given below
     Data available in sheets and test cases in which it is being used is given below
     i. **Credentials** - username and password for loginTest, orderHistoryTest
     ii. **AccountCreationEmail** - accountCreationEmail for accountCreatePageTest
     iii. **SearchProduct** - searchProduct for addToCartTest , endToEndTest , totalPriceTest , productAvailabilityTest
     iv. **ProductDetails** - productName, qty and size for addToCartTest , endToEndTest, totalPriceTest
     v. **ExpectedStoreTitle** - expectedStoreTitle for storeTitleTest
  b. To use these excel sheets, **DataProviders.java** class is used under **com.estore.dataprovider** package
     i. This uses the excelLibrary object reference (of **NewExcelLibrary,java** class) to getData from the sheets in excel workbook



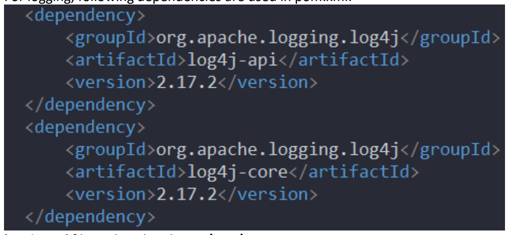
```java
NewExcelLibrary excelLibrary = new NewExcelLibrary();

@DataProvider(name = "credentials")
public Object[][] getCredentials() throws EncryptedDocumentException, IOException {
    return excelLibrary.getData("Credentials");
}

@DataProvider(name = "accountCreationEmail")
public Object[][] getAccountCreationEmail() throws EncryptedDocumentException, IOException {
    return excelLibrary.getData("AccountCreationEmail");
}

@DataProvider(name = "getProduct")
public Object[][] getProduct() throws EncryptedDocumentException, IOException {
    return excelLibrary.getData("ProductDetails");
}

@DataProvider(name = "searchProduct")
public Object[][] searchProduct() throws EncryptedDocumentException, IOException {
    return excelLibrary.getData("SearchProduct");
}

@DataProvider(name = "getStoreTitle")
public Object[][] getStoreTitle() throws EncryptedDocumentException, IOException {
    return excelLibrary.getData("ExpectedStoreTitle");
}
```

  c. dataProvider is then supplied to each test case to fetch values from excel sheets, for example, in HomePageTest.java:



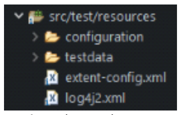
```java
public class HomePageTest extends BaseClass {
    LoginPage loginPage;// = new IndexPage();;
    IndexPage indexPage;
    HomePage homePage;


    @Test(groups = "Smoke",dataProvider = "credentials", dataProviderClass = DataProviders.class)
    Run | Debug
    public void orderHistoryTest(String userName, String password) throws Exception {
```

- **Logging in the framework level using log4j (Bonus points for logging in file)**
  ***** Kindly use the latest log4j dependency**
  a. For logging, following dependencies are used in pom.xml:



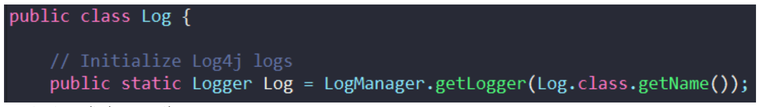
```xml
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.17.2</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.17.2</version>
</dependency>
```

  b. **log4j2.xml** file is placed under **src\test\resources**



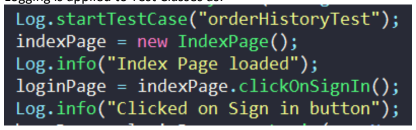
```
src/test/resources
  > configuration
  > testdata
    extent-config.xml
    log4j2.xml
```

c. **Log.java** class under **com.estore.utility** package is used to initialize the object of Logger



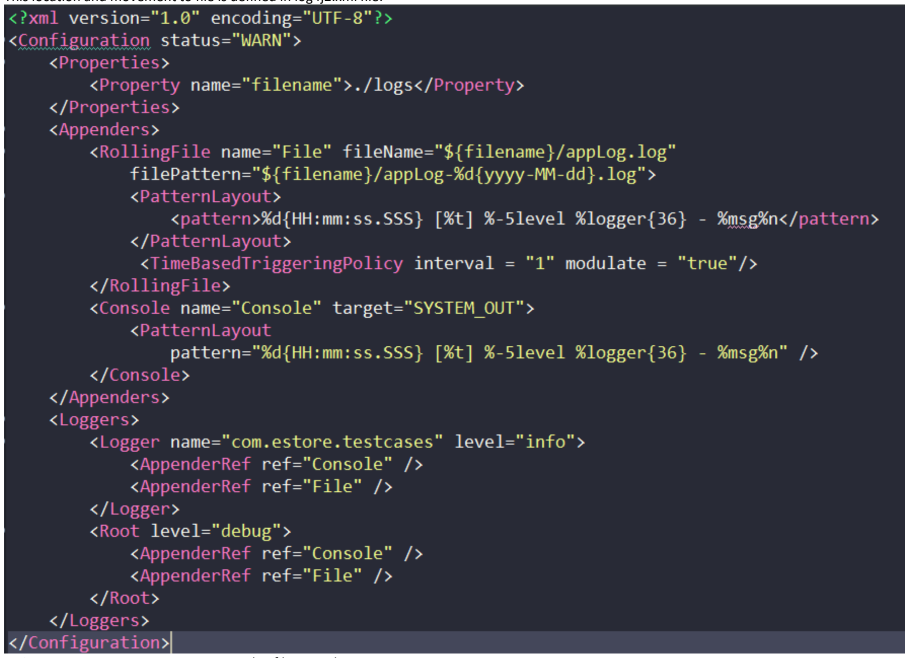
```java
public class Log {

    // Initialize Log4j logs
    public static Logger Log = LogManager.getLogger(Log.class.getName());
```

d. Logging is applied to Test Classes as:



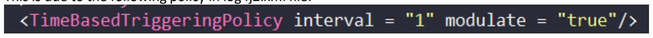
```java
Log.startTestCase("orderHistoryTest");
indexPage = new IndexPage();
Log.info("Index Page loaded");
loginPage = indexPage.clickOnSignIn();
Log.info("Clicked on Sign in button");
```

e. Logs are written to file - **applog.log** placed under **logs** folder at project root directory
This location and movement to file is defined in log4j2.xml file:



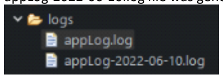
```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
    <Properties>
        <Property name="filename">./logs</Property>
    </Properties>
    <Appenders>
        <RollingFile name="File" fileName="${filename}/appLog.log"
            filePattern="${filename}/appLog-%d{yyyy-MM-dd}.log">
            <PatternLayout>
                <pattern>%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n</pattern>
            </PatternLayout>
             <TimeBasedTriggeringPolicy interval = "1" modulate = "true"/>
        </RollingFile>
        <Console name="Console" target="SYSTEM_OUT">
            <PatternLayout
                pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" />
        </Console>
    </Appenders>
    <Loggers>
        <Logger name="com.estore.testcases" level="info">
            <AppenderRef ref="Console" />
            <AppenderRef ref="File" />
        </Logger>
        <Root level="debug">
            <AppenderRef ref="Console" />
            <AppenderRef ref="File" />
        </Root>
    </Loggers>
</Configuration>
```

i. appLogs are set to generate a new log file everyday
This is due to the following policy in log4j2.xml file:



```xml
<TimeBasedTriggeringPolicy interval = "1" modulate = "true"/>
```

In the following structure, the appLog.log file was generated on 11 June 2022 and
appLog-2022-06-10.log file was generated on 10 June 2022



```
logs
  appLog.log
  appLog-2022-06-10.log
```

Test Cases Executed:

Smoke	**4**			
Sanity	**3**			
Regression	**3**			
Pages Classes	**TestClasses**	**TestCases**	**Grouping**	**Expected Results**
IndexPage	IndexPageTest	storeTitleTest	Smoke	Title should match with expected value
LoginPage	LoginPageTest	loginTest	Sanity, Smoke	User should be able to login
HomePage	HomePageTest	orderHistoryTest	Smoke	OderHistoryDetails button should be displayed
AccountCreationPage	AccountCreationPageTest	accountCreatePageTest	Sanity	User should be navigate to account creation page
SearchResultPage	SearchResultPageTest	productAvailabilityTest	Smoke	Search the product and product should be displayed
AddToCartPage	AddToCartPageTest	addToCartTest	Regression, Sanity	User should be able to add the product in the cart
OrderPage	OrderPageTest	totalPriceTest	Regression	
AddressPage				Validate the Price on Order Page
ShippingPage				
PaymentPage				
OrderSummaryPage				
OrderConfirmationPage	EndToEndTest	endToEndTest	Regression	User should be able to order the product