ReadMe

Thursday, June 9, 2022 9:43 PM

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

Running tests using CMD:

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

2. Extract the zip file content on a location on hard drive

3. Use the address bar in the directory and type CMD

4. Run the command **mvn clean** on CMD

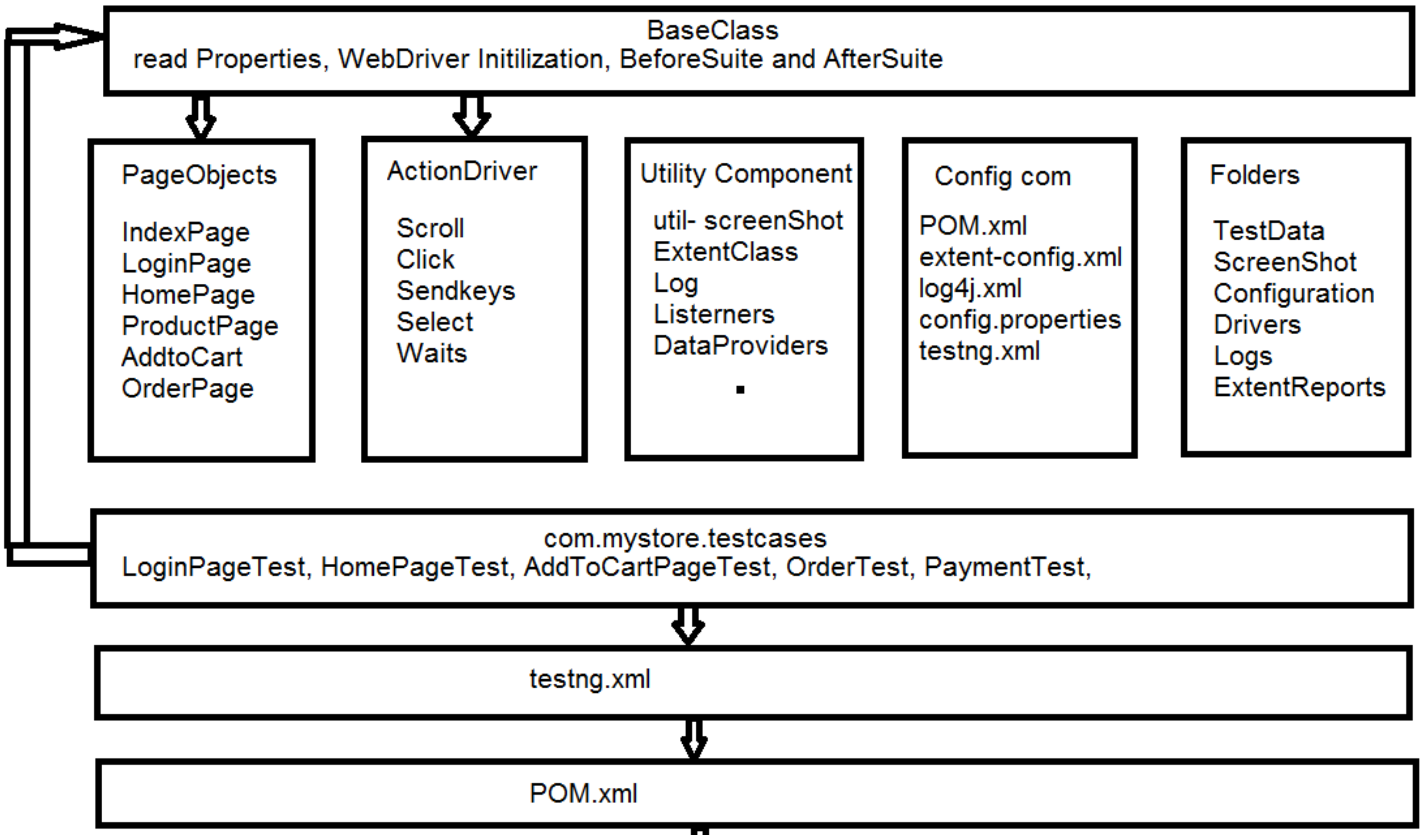
5. Run the command **mvn compile** on CMD

6. Run the command **mvn test** on CMD
   testng_all.xml is configured to run:

```
<?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.myamazon.utility.ListenerClass"></listener>
    </listeners>
    <test thread-count="5" name="Test">
        <parameter name="browser" value="chrome"></parameter>
        <classes>
            <class name="com.myamazon.testcases.LoginTest" />
            <class name="com.myamazon.testcases.EndToEndTest" />
            <class name="com.myamazon.testcases.CartAddTest" />
            <class name="com.myamazon.testcases.TitleTest" />
            <class name="com.myamazon.testcases.ProductAvailableTest" />
            <class name="com.myamazon.testcases.OrderHistoryWishlistLabelTest" />
            <class name="com.myamazon.testcases.TotalPriceTest" />
            <class name="com.myamazon.testcases.CartAddTest" />
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

Running tests using TestNG.xml files:

1. Open the project in Eclipse
2. Select the folder from the location where the ZIP file is extracted
3. On opening, Project structure is shown
1. Project is divided into the following components:



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:

```
<class name="com.myamazon.testcases.LoginTest" />
<class name="com.myamazon.testcases.EndToEndTest" />
<class name="com.myamazon.testcases.CartAddTest" />
<class name="com.myamazon.testcases.TitleTest" />
<class name="com.myamazon.testcases.ProductAvailableTest" />
<class name="com.myamazon.testcases.OrderHistoryWishlistLabelTest" />
<class name="com.myamazon.testcases.TotalPriceTest" />
<class name="com.myamazon.testcases.CartAddTest" />
```

- **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
- **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

```
newemail = vineet1234@test.com
email = vineet123456@gmail.com
password = vineet123
url = http://automationpractice.com/index.php
expectedUrl = http://automationpractice.com/index.php?controller=my-account
expectedMessage = Your order on My Store is complete.
browser = Chrome
implicitWait= 20
```

- **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

- **Implement proper waits.**
    a. Following waits and timeouts have been incorporated in the project:
        i. Explicit Wait
        ii. Implicit Wait

    b. Implicit Wait is used in BaseClass.java when loading URL in browser:
    c. Explicit Wait is used in the methods used in Action.java class in the click and isDisplayed methods

- **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 as stated above
    b. **TestData.xlsx** file is stored under **src\test\resources\testdata** folder and used to store data
- **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
    c. **testng_crossbrowserparallel.xml** file is used to run the tests in parallel in Browsers - Chrome, Firefox and IE
    d. 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
    b. On Test failure, the failure message is captured with screenshot in ListenerClass.java
- **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 root directory
    b. In BaseClass.java under @AfterSuite annotation, endReport() method defined in ExtentManager.java, is called
    Extent reports are located at **test-output\extentreport\**:

    c. On start of every test cycle, the report from **extentreport** gets moved to a**rchivedtestresults** and when the test completes, the current report gets generated in **extentreport** folder with name MyExtentReport.html
- **The tests should be runnable from command line**
    a. This is depicted in the beginning of this document

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

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

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

    d. Logs are written to file - **applog.log**