**Document for Automation of “**[**http://practice.automationtesting.in**](http://practice.automationtesting.in)**” with cucumber framework.**

1. **Execution flow**

Following flow chart will explain the execution flow.

Feature file

Step Definitions

Page Elements

Reports

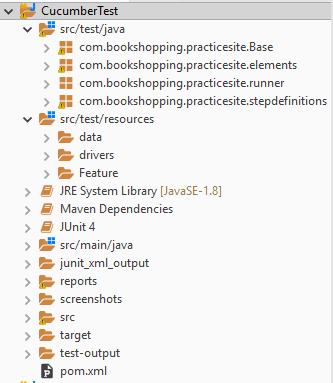
1. **Framework Details**

This section lists and explains about the sample project which has been used for POC and explains the details as mentioned below.

* 1. **Maven Project Name: CucumberTest**

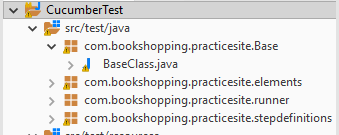
This project is created as Maven Project. This project is created with several packages and the complete structure of the project is present as mentioned in the screenshot present in next page. Also, the packages are explained in successive sections.

* 1. **Screenshot of the project packages**



* 1. **Details of the package com.bookshopping.practicesite.Base**

This package contains BaseClass.java.

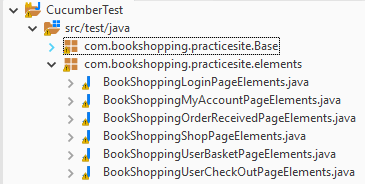


This class has method to set the URL to be launched, method to initialize the driver of a browser, method to capture a screenshot, method to wait, method to execute the javascriptexecutor to perform a scroll operation.

This class will be extended by every other class in the project to use the static variables and methods declared inside it.

* 1. **Details of package com.bookshopping.practicesite.elements**

* This package contains all the web elements of the pages that were used in automating the application.
* Each class under this package represents one web page.



* The page elements are identified by **FindBy** annotations and object identifier. Example of login button element is shown as below.

@FindBy(name="login")

**public** WebElement loginbutton;

* 1. **Details of package com.bookshopping.practicesite.runner**

This package contains a JUnit runner class which has multiple annotations as shown in the below screenshot.



@**RunWith** annotation tells JUnit, which is the test runner class.

@**CucumberOptions** has the following parameters specified in it.

* features: path of the feature file.
* glue: path of step definitions.
* monochrome: this will give the clear information of the execution time.
* dryrun: this will check whether mapping is correct between feature file and step definition.
* format: this will generate the JUnit reports in xml, json and html.
* tags: this is used to group multiple scenarios. To exclude/skip the execution of any tag ‘ ~ ‘ should be used in front of the annotation.

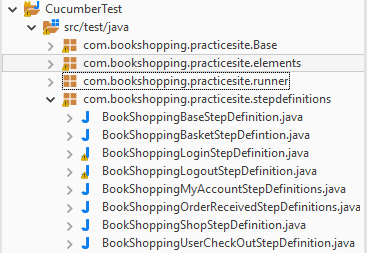
Ex: ~@shop.

On running the feature file as cucumber feature, it will generate the method stubs which should be implemented in the step definitions.

Post execution reports will be generated in different formats and will be stored under **test-output** folder name index.html, result.json and result.xml will be stored under **junit\_xml\_output** folder.

* 1. **Details of package com.bookshopping.practicesite.stepdefinitions**

This package contains classes that has the implementation of the step definition methods that were generated on running the feature file.



Each class represents one web page of the application.

Class: BookShoppingBaseStepDefinition.java

This class has two methods beforeExecutingScenario with @Before annotation and afterExecutingScenario with @After annotation.

Method beforeExecutingScenario with @Before annotation initializes ExtentHtmlReporter and configures the theme, title, system info to the report that will be generated post execution.

Method afterExecutingScenario with @After annotation verifies the status of the scenario under execution and if the scenario is failed, it will capture the screenshot along with the scenario name and timestamp and it will be saved under screeshots folder.

* 1. **Details of the folder src/test/resources/Data**

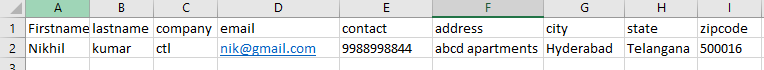
Data folder has an excel sheet TestData.xlsx

TestData.xlsx has two sheets with name Login and Customer.

Login sheet has the credentials of the registered user and these values will be used while logging in to the application.

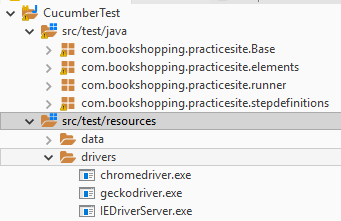


Customer sheet has the billing details of the customer which will be used while testing the checkout page of the application.



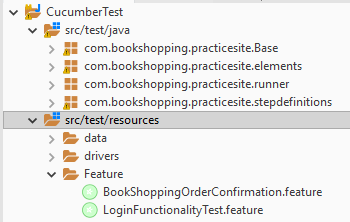
* 1. **Details of the folder src/test/resources/drivers**

Drivers folder has the .exe files of the drivers for different browsers.



* 1. **Details of the folder src/test/resources/features**

This folder has different feature files



Feature files will have all the steps to be executed.

Gherkin language is used to define a Feature file.

Feature, Scenario, Scenario Outline, Given, When, Then, And are the **Gherkin** keywords used in defining a feature file.

* 1. **Details of reports and screenshots folders**

Reports folder will consist of the extent reports that were generated after the execution. Reports will be named with the name that is supplied under BookShoppingBaseStepDefintion along with a timestamp to it.

Ex: BookShoppingReport \_06-20-2019-02-47-19.html

Screenshots folder will consist of the screenshots of the screen at which the test is getting failed. Name of the screenshot will be scenario name along with a timestamp.

Ex: login\_with\_valid\_username\_and\_password.\_06-20-2019-02-47-19.jpg

* 1. **Pom.xml**

The pom.xml will be created along with the creation of maven project. Here, the dependencies need to be added and maven install step needs to be run to get the dependencies added to the running project.

Sample dependency step is shown below to install the POI dependency for the project. Similarly, the project specific dependency needs to be added.

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>4.0.1</version>

</dependency>

Some of the mandatory dependencies that needs to be added are listed below

* selenium-java
* poi-ooxml

Link for finding the maven dependencies: <https://mvnrepository.com/>

1. **Steps to import and running the project**

* Download the project CucumberTest from the GitHub to your local system from the following link.

<https://github.com/Nikhilpuri93/TestingPractice.git>

* Import the project to the eclipse IDE.
* Open the **BaseClass.java** under com.bookshopping.practicesite.base package and the change the system properties for the drivers.
* Navigate to com.bookshopping.practicesite.runner package and open **BookShoppingRunner.java** and run as Junit Test and the test execution will start.
* Once the test is completed, default cucumber reports (index.html, result.json and result.xml) will be generated and extent report will be generated under reports folder.
* Screen shot of the extent report.

