Framework Design

1. Create a Maven project with simple archetype
2. Create the framework folder structure, by creating the necessary packages
3. Under src/test/java – base, listeners, rough, testCases, utilities
4. Under src/test/resources – excel, executables, logs, properties, runner.
5. Now update the POM.xml file with dependencies – Selenium-Java, extent-report, log4J, javax.mail,

How to implement extent report in Selenium TDD framework.

1. Download extent report latest maven dependency

**<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->-**

**<dependency>**

**<groupId>com.aventstack</groupId>**

**<artifactId>extentreports</artifactId>**

**<version>5.1.2</version>**

**</dependency>**

1. package SeleniumSessions;  
     
   import com.aventstack.extentreports.ExtentReports;  
   import com.aventstack.extentreports.ExtentTest;  
   import com.aventstack.extentreports.reporter.ExtentSparkReporter;  
   import com.aventstack.extentreports.reporter.configuration.Theme;  
   import org.testng.annotations.AfterTest;  
   import org.testng.annotations.BeforeTest;  
   import org.testng.annotations.Test;  
     
   import static com.aventstack.extentreports.reporter.ExtentSparkReporter.\*;  
     
   public class TestExtentReports {  
     
    *//note- test pass or fail status should be handled from extent  
    // if its from testng then it will not be displayed in extent report* public ExtentSparkReporter htmlReporter; *//create html file and add configurations* public ExtentReports extent; *//attaching reporter, creating tests, logs, etc* public ExtentTest test; *// maintaining the test cases, add log status, etc* @BeforeTest  
    public void setReport(){  
     
    htmlReporter = new ExtentSparkReporter("./ExtentReports/extent.html");  
    htmlReporter.config().setEncoding("utf-8");  
    htmlReporter.config().setDocumentTitle("Automation Test Report");  
    htmlReporter.config().setReportName("Functional Test Report");  
    htmlReporter.config().setTheme(Theme.*STANDARD*);  
    htmlReporter.config().setTimeStampFormat("EEEE, MMMM dd, yyyy, hh:mm a '('zzz')'");  
    htmlReporter.config().setTimelineEnabled(true);  
     
    extent = new ExtentReports();  
    extent.attachReporter(htmlReporter);  
     
    extent.setSystemInfo("Hostname", "localhost");  
    extent.setSystemInfo("Browser", "Chrome");  
    extent.setSystemInfo("Environment", "QA");  
    extent.setSystemInfo("Tester", "Rahul");  
     
     
    }  
     
    @AfterTest  
    public void endReport(){  
    extent.flush();  
    }  
     
    @Test  
    public void createTest(){  
    */\*\*  
    \* Creates a test in the ExtentReports instance and logs the steps of the test.  
    \*/* test = extent.createTest("Login Test"); *// Create a test named "Login Test" in the ExtentReports instance* test.pass("URL is opened"); *// Log that the URL is opened* test.pass("Title is verified"); *// Log that the title is verified* test.pass("Login test"); *// Log that the login test is completed* }  
     
   }

Cucumber BDD Framework Design

Create maven project in eclipse

update POM.xml file

cucumber-java

cucumber-TestNG

Selenium

install cucumber eclipse plugin

restart eclipse after plugin installation

Right click on the project > configure > convert to cucumber project

cucumber green symbol will be displayed in project

create feature file and keep it under src test resources as feature files doesn't have any code, its plain english using gherkin language

create package features

create file login.feature

Feature: Login Feature

In order to perform successful login

As a user

I have to enter correct username and password

Scenario: Login to the Application

Given user navigates to the application

When user validates the home page title

Then user enters username

And user enters password

And user clicks on SignIn button

right click and select pretty format

yellow highlight as glue code is not there

run the feature file , step def methods will be shown in console

To Run - Run Confifuration > select feature option > enter feature file path

Create steps package under src/test/java

under this create a new java class > loginSteps.java

paste the auto suggested glue code from console here.

import the relevant packages

file > maven > update project, yellow sign will go

Again run the feature file and check

you will see green sign for each step once feature file is mapped to steps definitions

CTRL+click on feature steps will navigate to steps

code written inside steps will have selenium/ api/ appium any code will run

Now create the Runner file (java file ) to run our feature file

Under src/test/java create a new package called runner

Create java class RunLogin.java

package runner;

import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

@CucumberOptions(features="src/test/resources/features/login.feature",

glue= {"steps"}

)

public class RunLogin extends AbstractTestNGCucumberTests {

}

run the runner class as TestNG

After the runner class is implemented lets implement cucumber report in our framework

modified runner after implementing plugin

package runner;

import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

@CucumberOptions(features="src/test/resources/features/login.feature",

glue= {"steps"},

plugin= {"html:target/cucumber-reports/cucumber-html-report.html"}

)

public class RunLogin extends AbstractTestNGCucumberTests {

}

run the runner class and check for the report in Target folder after refreshing the project

extent report implementation from tech.grasshopper

search for - extentreports-cucumber 7 Adapter

Add the latest maven dependency in pom.xml file

add extent report latest version dependency also in pom.xml file

Now add extentreports-cucumber 7 Adapter as plugin in your runner file

https://extentreports.com/docs/versions/4/java/cucumber4.html

package runner;

import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

@CucumberOptions(features="src/test/resources/features/login.feature",

glue= {"steps"},

plugin= {"html:target/cucumber-reports/cucumber-html-report.html",

"com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"}

)

public class RunLogin extends AbstractTestNGCucumberTests {

}

Once the adapter plugin is configured in the runner class, under src/test/resources/ create extent.properties file

so basically the extent plugin is going to refer this extent.properties file.

So report activation write

Note- There are two files extent.properties and extent-config.xml keep these file under src/test/resources/ to make it work

extent-config.xml file will be referred from extent.properties file

# For report Activation

extent.reporter.spark.start=true

extent.reporter.spark.out=MyReports/Spark.html

extent.reporter.spark.config=src/test/resources/extent-config.xml

sample extent-config.xml

<?xml version="1.0" encoding="UTF-8"?>

<extentreports>

<configuration>

<!-- report theme -->

<!-- standard, dark -->

<theme>standard</theme>

<!-- enables timeline -->

<!-- defaults to true -->

<enableTimeline>true</enableTimeline>

<!-- document encoding -->

<!-- defaults to UTF-8 -->

<encoding>UTF-8</encoding>

<!-- protocol for script and stylesheets -->

<!-- defaults to https -->

<protocol>https</protocol>

<!-- title of the document -->

<documentTitle>Extent Framework</documentTitle>

<!-- report name - displayed at top-nav -->

<reportName>Build 1</reportName>

<!-- custom javascript -->

<scripts>

<CDATA[

$(document).ready(function() {

});

]]>

</scripts>

<!-- custom styles -->

<styles>

<![CDATA[

]]>

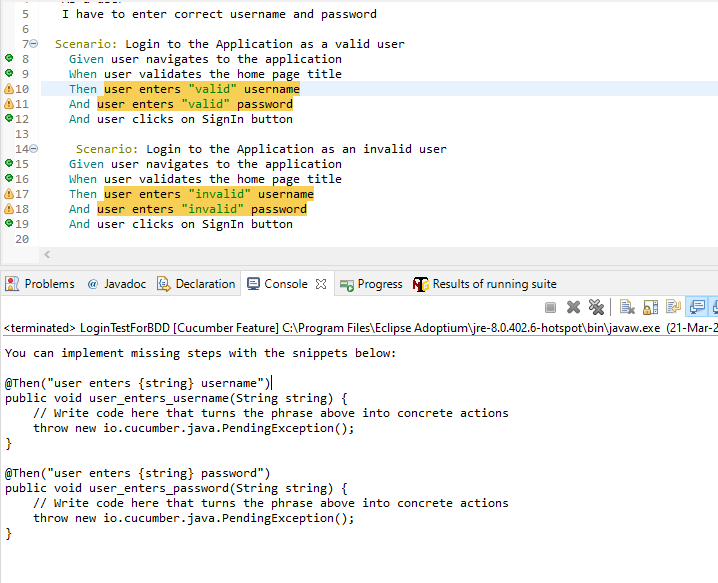
</styles>

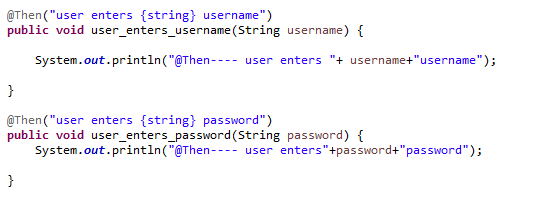
</configuration>

</extentreports>

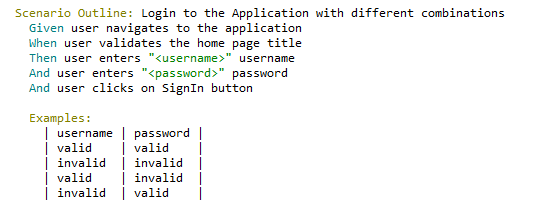
<https://extentreports.com/docs/versions/4/java/spark-reporter.html>

# Adding multiple scenarios to a feature file





No of {string} counts = no of parameters in the method.

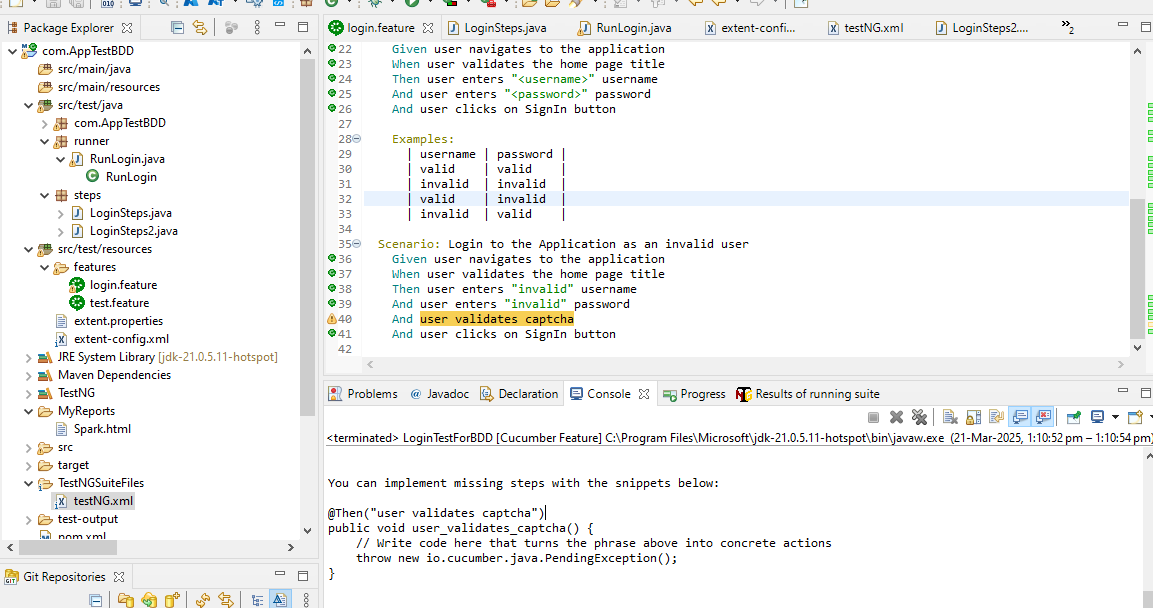


![A computer code with text

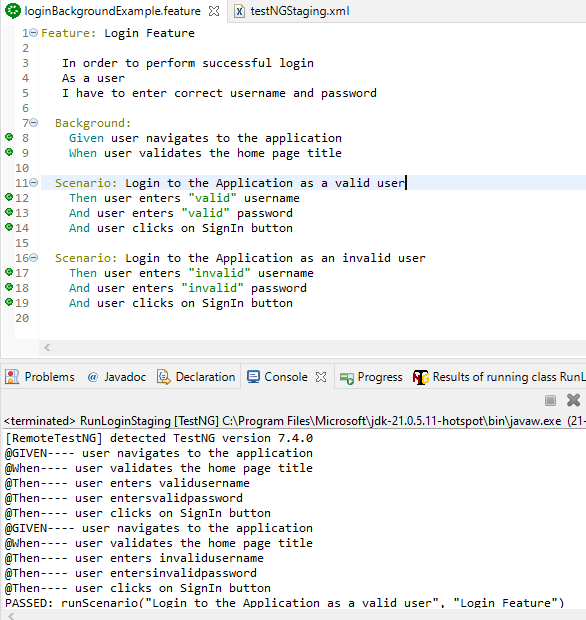
AI-generated content may be incorrect.

# Adding multiple features and Step Definitions

Under the steps package we can add multiple java classes for steps definitions and all the steps will be visible throughout the features.



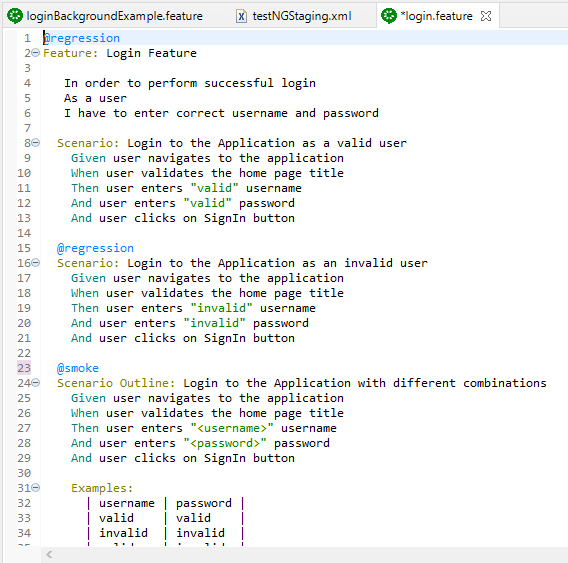
# Background



# Cucumber Tags

Used when we want to run some set of selected scenarios smoke,sanity, modulewise

To execute limited features, we can tag our features and scenarios.



To make this work under @cucumberOptions add tags=”regression”

If you don’t mention any tags then we can run all the features in features folder in case if we don’t want to run any specific feature file.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

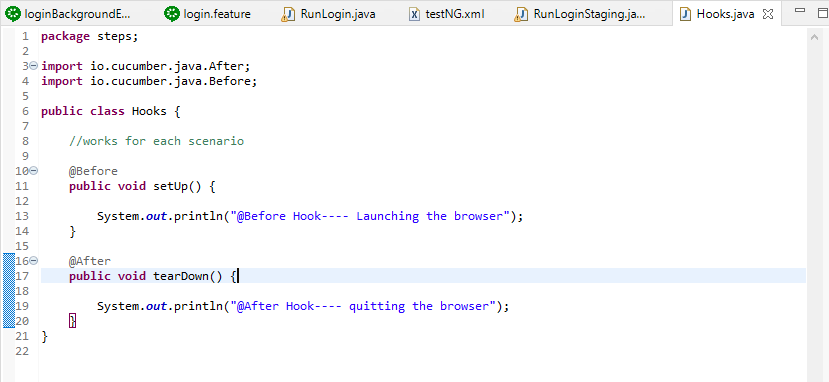
Tags=”@production or @staging or @sanity” run both

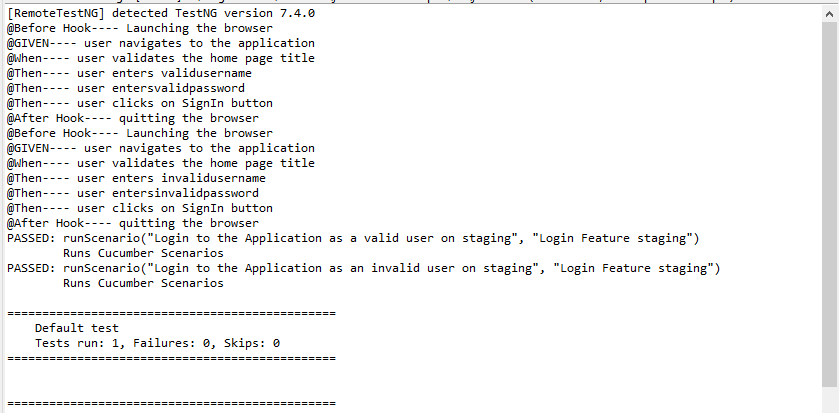
Tags=”@production or @staging and not @sanity”

# Cucumber Hooks

Hooks is similar to @Before and @After annotation in TestNG and junit

Create hooks class in your steps package





Note- @Before and @After – work before and after each scenario in a feature file.

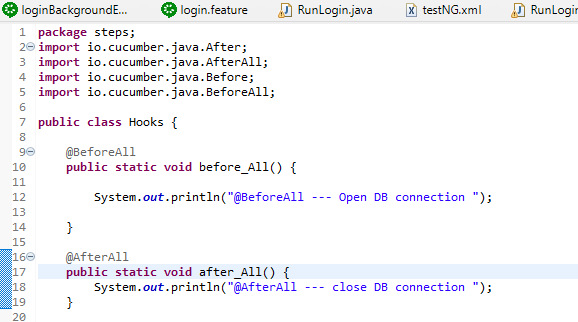
A screenshot of a computer program

AI-generated content may be incorrect.

A close-up of a computer screen

AI-generated content may be incorrect.

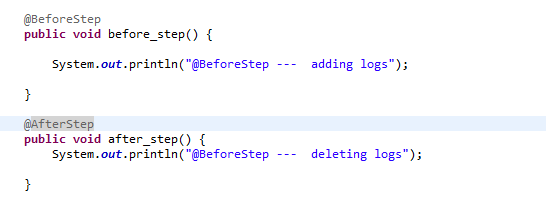
Make the BeforeAll and AfterAll method as static.

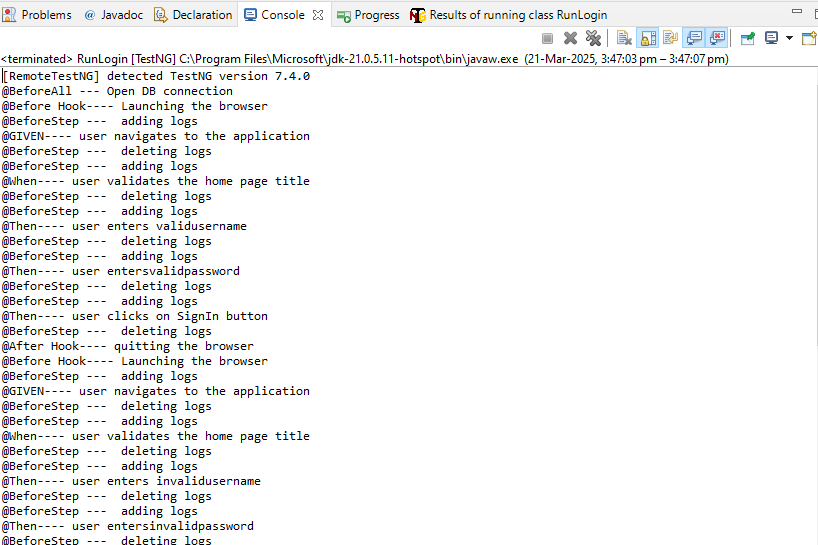


A screenshot of a computer

AI-generated content may be incorrect.

@BeforeStep and @AfterStep – works before and after each steps

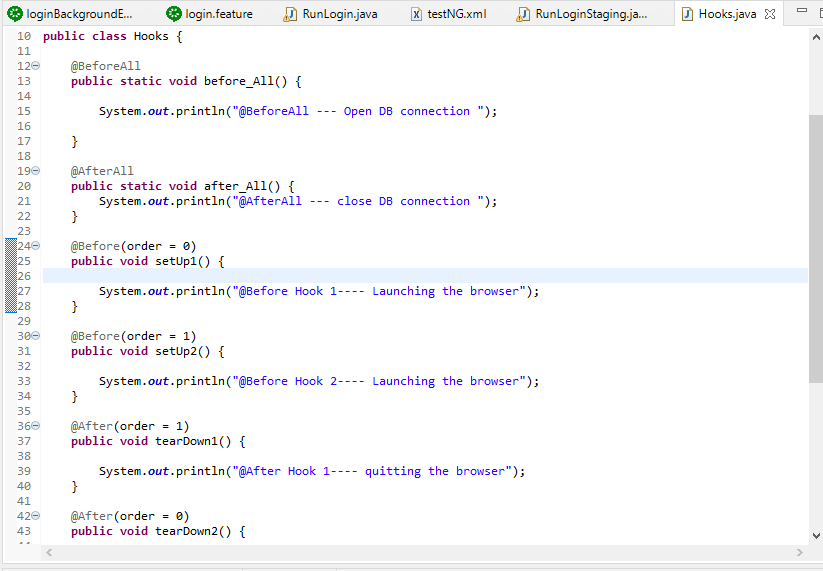




Defining the execution order in Cucumber Hooks

We can have multiple @Before hook and we can define the execution order

User order=0 , 0 being max priority for Before in case of after it will be reverse.



Tagged Hooks

Before or after condition for a particular tag, make sure runner class is also updated.

A close-up of a message

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Data Table as List

Create a feature file “dataTableAsList.feature

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.

It is a data table so we can read all the data as row column format.

# Data Table As Map

A screenshot of a computer

AI-generated content may be incorrect.

A computer code with black text

AI-generated content may be incorrect.

# Parameterize data table as a Map

A computer code with text

AI-generated content may be incorrect.

Taking Screenshot on failure

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.