

TCOE AUTOMATION

JAVA FRAMEWORK USER GUIDE



Kalluru, Venkat Suneel SBOBNG-ITC/I/PF

SHELL INDIA

**Table of Contents**

[**Introduction**](#page2)[**1**](#page2)

[Framework Architecture](#page3) [2](#page3)

[Why should you use this Framework?](#page3) [2](#page3)

[**Framework Setup**](#page4)[**3**](#page4)

[Prerequisites Solution](#page4) [3](#page4)

[Setup the Framework in less than 5mins](#page5) [4](#page5)

[Optional Features setup](#page8) [7](#page8)

[**Framework Usage**](#page9)[**8**](#page9)

[Creating Feature Files and Generating step definitions](#page9) [8](#page9)

[Creating Control Library and Function Library](#page12) [11](#page12)

[Using the Library functions and Test data in step definition](#page17) [16](#page17)

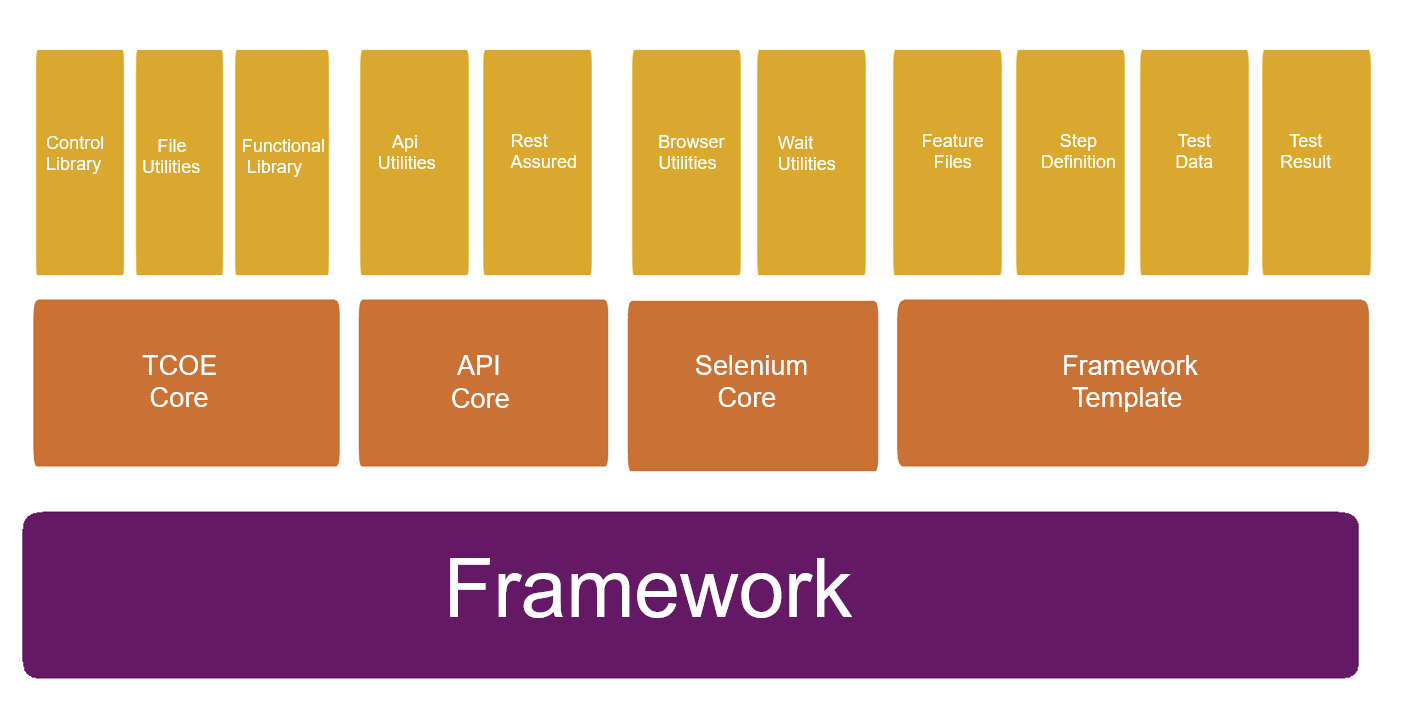
[Setting up Application Configuration(Update Properties)](#page18) [17](#page18)

[Analyzing Test Reports](#page18) [17](#page18)

**INTRODUCTION**

This Java Automation framework designed by TCoE is currently used to automate web applications and REST APIs. The framework is built on Selenium and Rest Assured as its core components. Its designed for ease of usability, maintainability and scalability. To use this framework, one must have fair understating of the OOPS programing concepts. The framework is written in Java, but even without any knowledge on Java one can use this framework with basic programing skills.

**Framework Architecture**



* The Framework consists of four major components.
* The TCOE Core, Selenium core and API core which are the core components in the form of JAR’s provided by TCoE.
* The Framework Template contains Application Library and Cucumber BDD tests which are the business components that contain the functionalities for Tests.
* Based on your requirements, you can use only selenium core or only API core or both.
* The Core components will be maintained and enhanced by TCoE while the business components will be maintained by the project teams.
* Selenium Core: It contains the Selenium base which initializes your local or remote driver. The Utilities contains selenium extension methods and various utilities for browser, SQL, Excel and Waits.
* API Core: It contains Rest Assured base which helps in registering the client and make all kind of Rest request calls. It also contains methods for Authentication and Utility methods for serialize and deserialize the json response content.
* Application Library: It contains the control library which is used to store the web element properties, function library that are designed by injecting control library to write business functions and Asserts, Models are property class to get and set Test Data.
* Cucumber/BDD Tests: This contains your test suites written in BDD using cucumber. The Feature files contain a feature with all related scenarios written in plain English using gherkins. The Step definitions are reusable test methods that map to the scenarios which in turn calls function library and test data. The runs will generate test results with screen shots.

**Why should you use this Framework?**

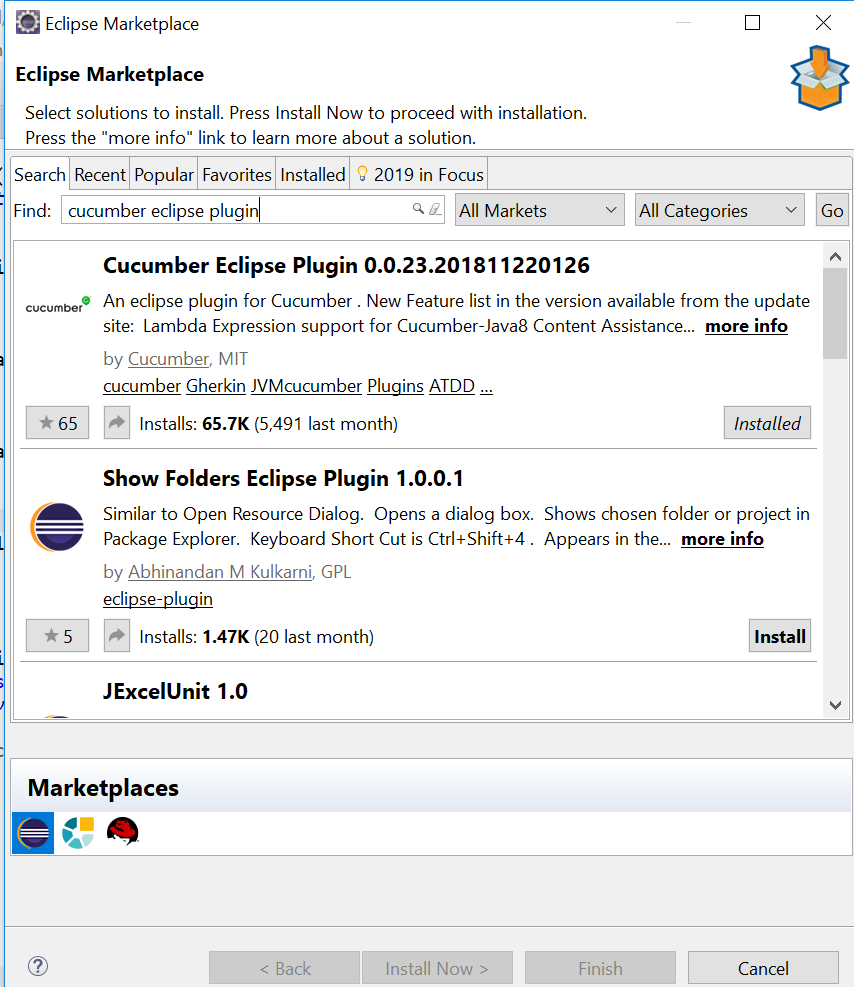
* A robust framework ready to adopt.

* Write your Test cases using Gherkins in BDD format which can be understood by all stakeholders, PM/PO, Developers and testers.
* Good amount of utility methods and extensions methods to write your business functions.
* Easy maintenances of web elements in the control library which uses page factory pattern.
* A single framework for testing web applications and Rest APIs
* Interactive Reports in BDD format with screenshots on failure.
* Mongo DB Test Report server that gives insights and analytics on historic test runs for different projects.
* Supports customized parallel test executions.
* Auto generated Test data for non-specific data values
* Supports cloud execution with BrowserStack, Sauces Labs, Selenium grid or PhantomJS
* Full support for integration of CI/CD with VSTS or Jenkins
* This framework has loosely coupled components that is flexible and extensible
* TCoE will be upgrading and releasing updated versions of core components, so that there is no need to reinvent the wheel.

**FRAMEWORK SETUP**

**Prerequisites Solution**

* To use the framework, you would need to use Eclipse or Intellij or any IDE that supports Java development as the IDE. We would detail on using Eclipse in this document.
* User can download Eclipse or any other IDE to use the automation framework.
* Once IDE is installed you need to install the extension for Cucumber Eclipse – Plugin To do this in your Eclipse got to Help > Eclipse Marketplace. Search for Cucumber Eclipse – Plugin and install.

~~~~

* Users to download JDK8 and install it.
* Users to download maven and unzip it to location on your machine.
* User to add the Java JDK location to JAVA\_HOME
* User to add the Maven and JDK bin locations to PATH
* User to test Java and Maven installations as below and validate that both commands are getting proper results

Java -version

mvn --version

* Now you are all set to setup the framework

**Setup the Framework in less than 5 minutes**

* First TCoE would provide you with a basic template (to be downloaded from VSTS) and a settings.xml file
* Users to unzip the .zip file
* User to import the basic template in Eclipse as a Maven Project
* Users to copy the settings.xml file in the .m2 folder of the host machine
* Update the PAT token if required in the settings.xml file.
* Once above step is executed User to right click the Project of the template opened in Eclipse > Update Maven > Select the force update snapshots
* That’s it, you’re ready to automate.

**Optional Features setup**

* The framework has support to configure a report server
* To use the report server, navigate to [Klov](http://extentreports.com/docs/klov/) web page and follow the instructions to install MongoDB server and Klov jar file.
* There no need to do any setup for Redis.
* In the solution open pom.xml and in profile ids ‘seqexec’ and ‘parallelexec’ set ‘EnableKlovReport’ to true

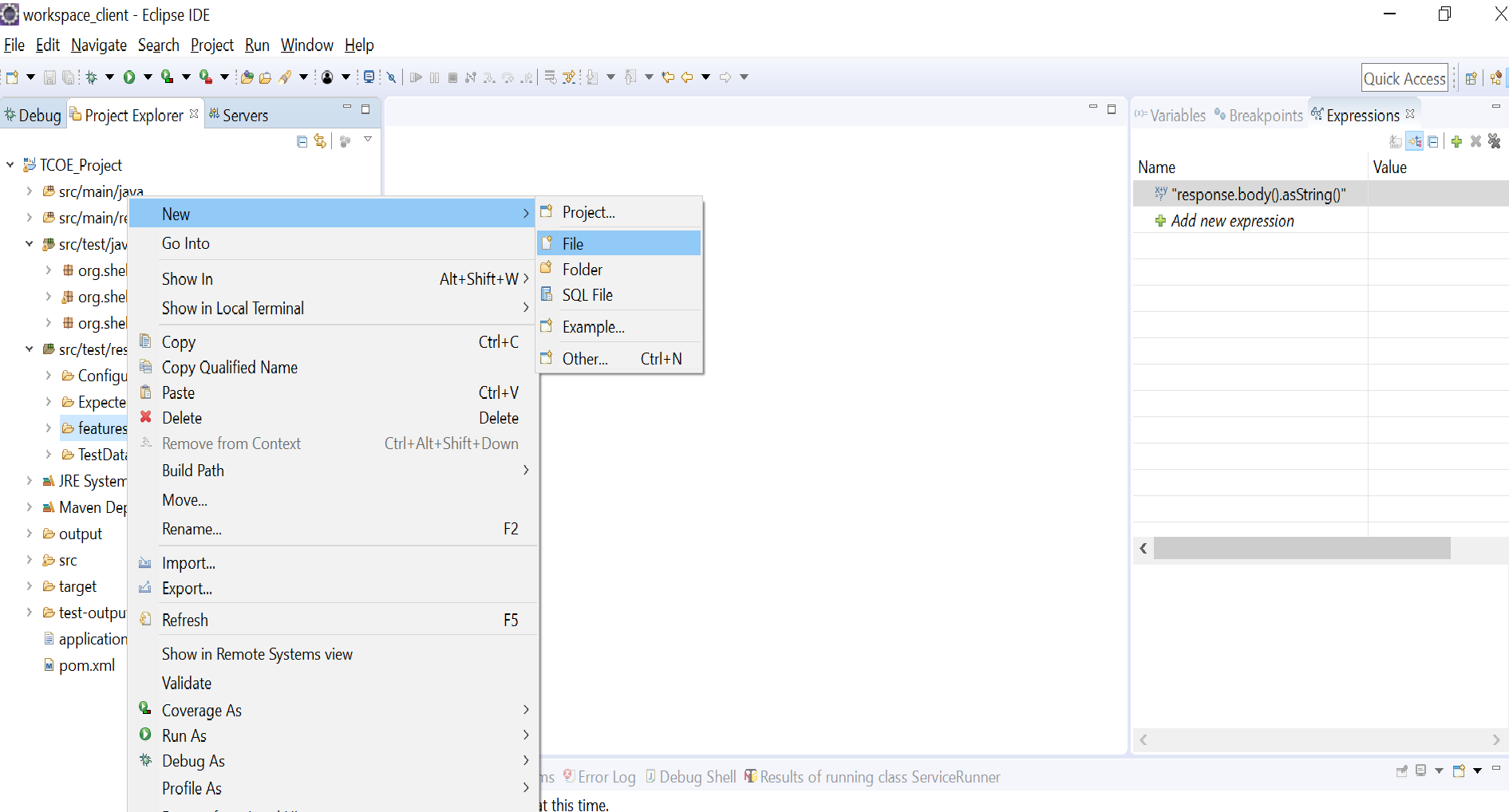
<EnableKlovReport>true</EnableKlovReport>

* Start the MongoDB server then the Klov jar
* Now run your test suits

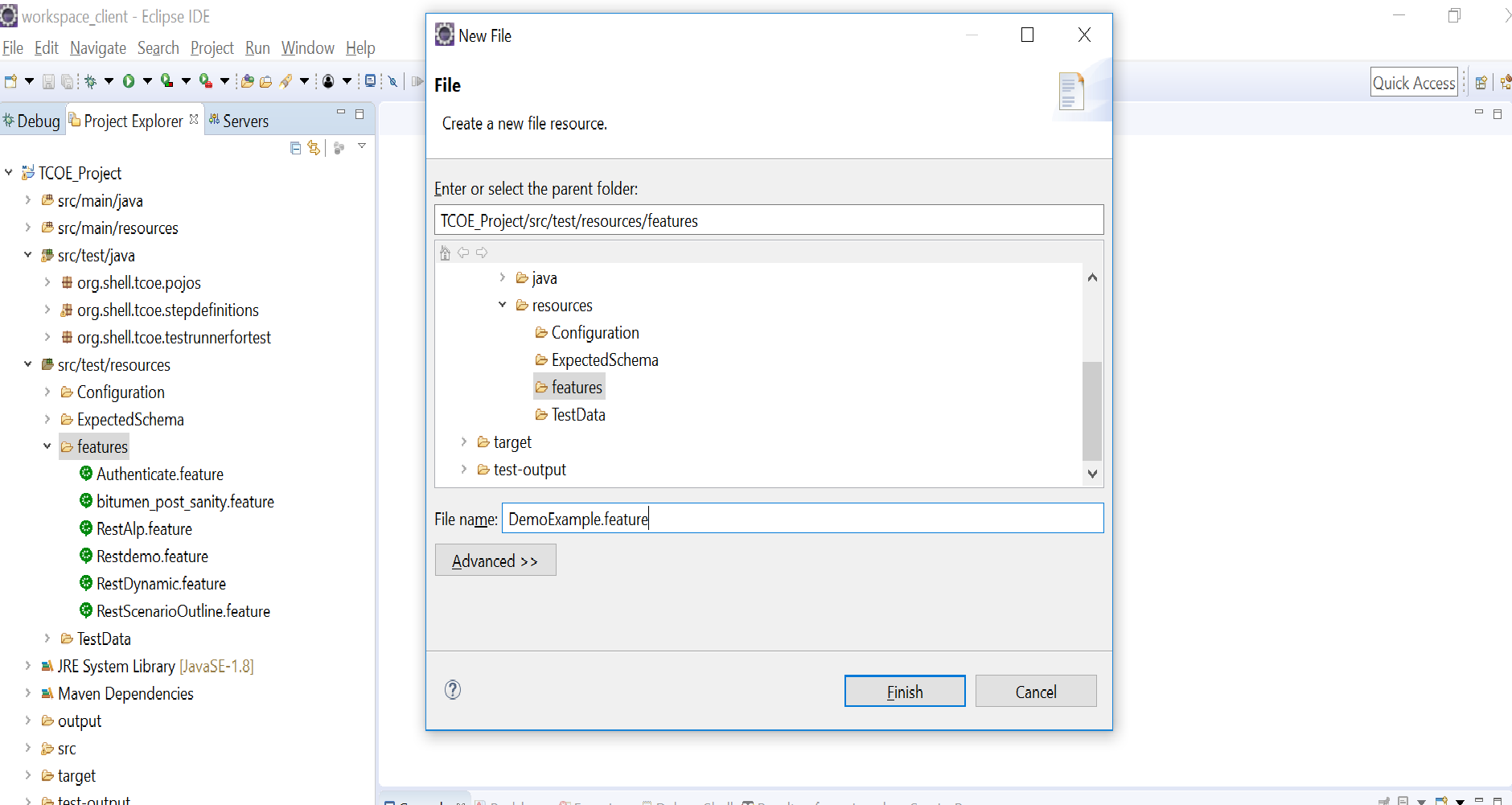
**FRAMEWORK USAGE**

**Creating Feature Files and Generating step definitions**

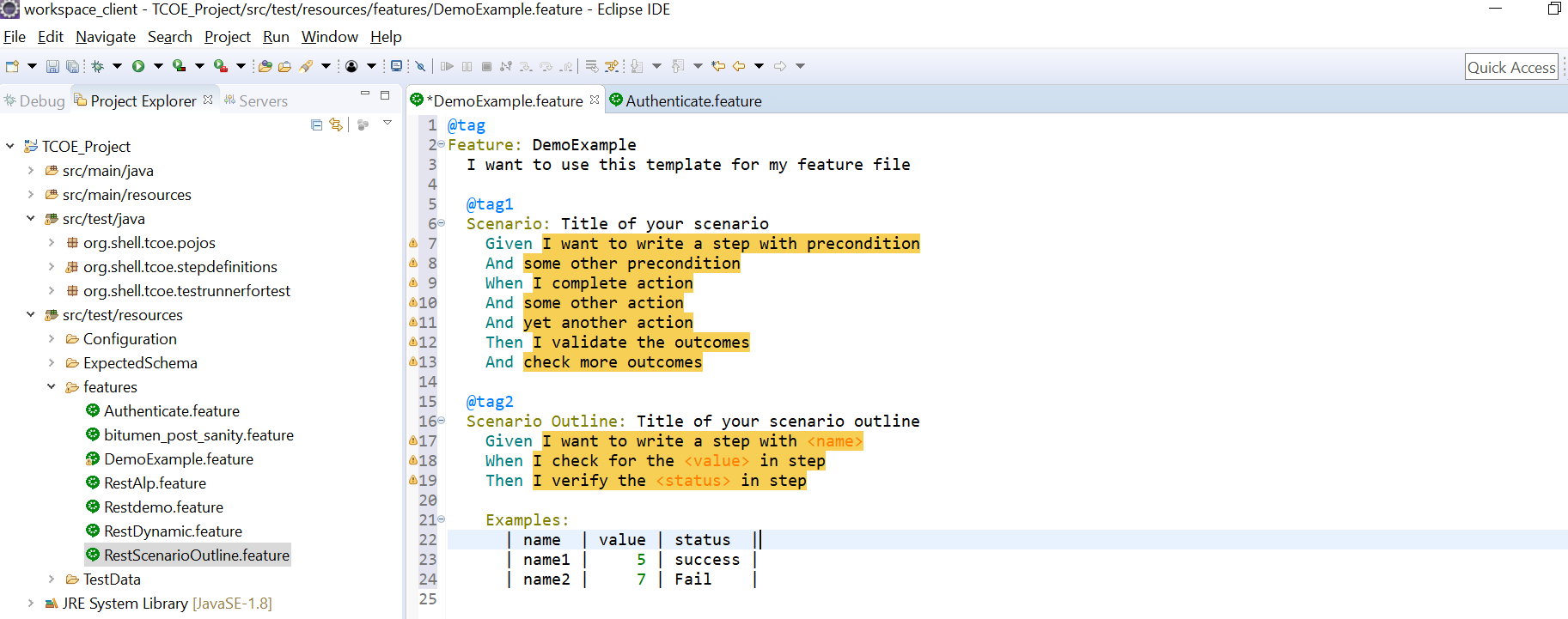
* Once your framework setup is done, your solution would look like this.



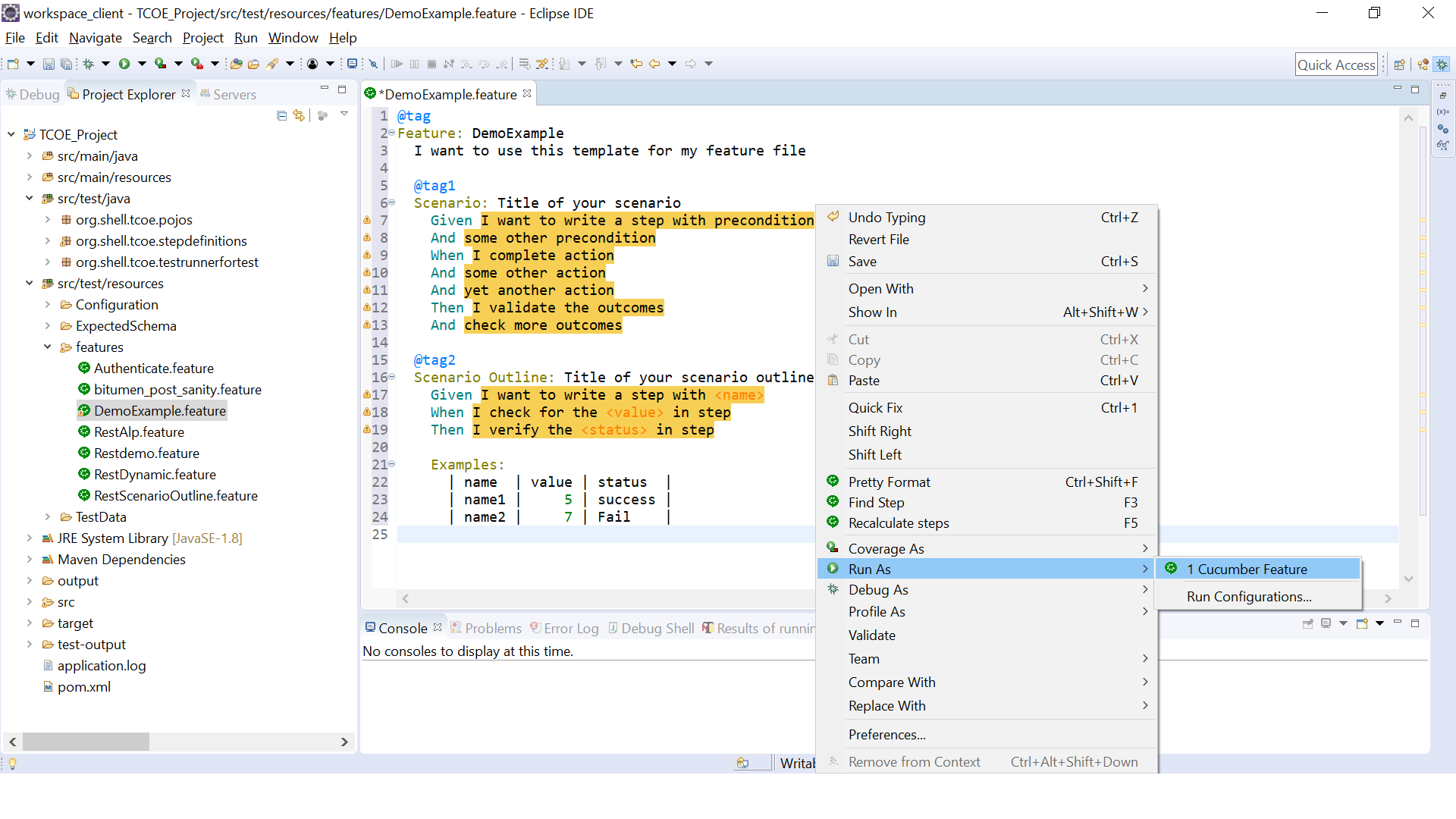
* As shown above right click on the Feature file folder and select Add> New Item
* Enter Feature File name and have extension as ‘.feature’ and click on Finish



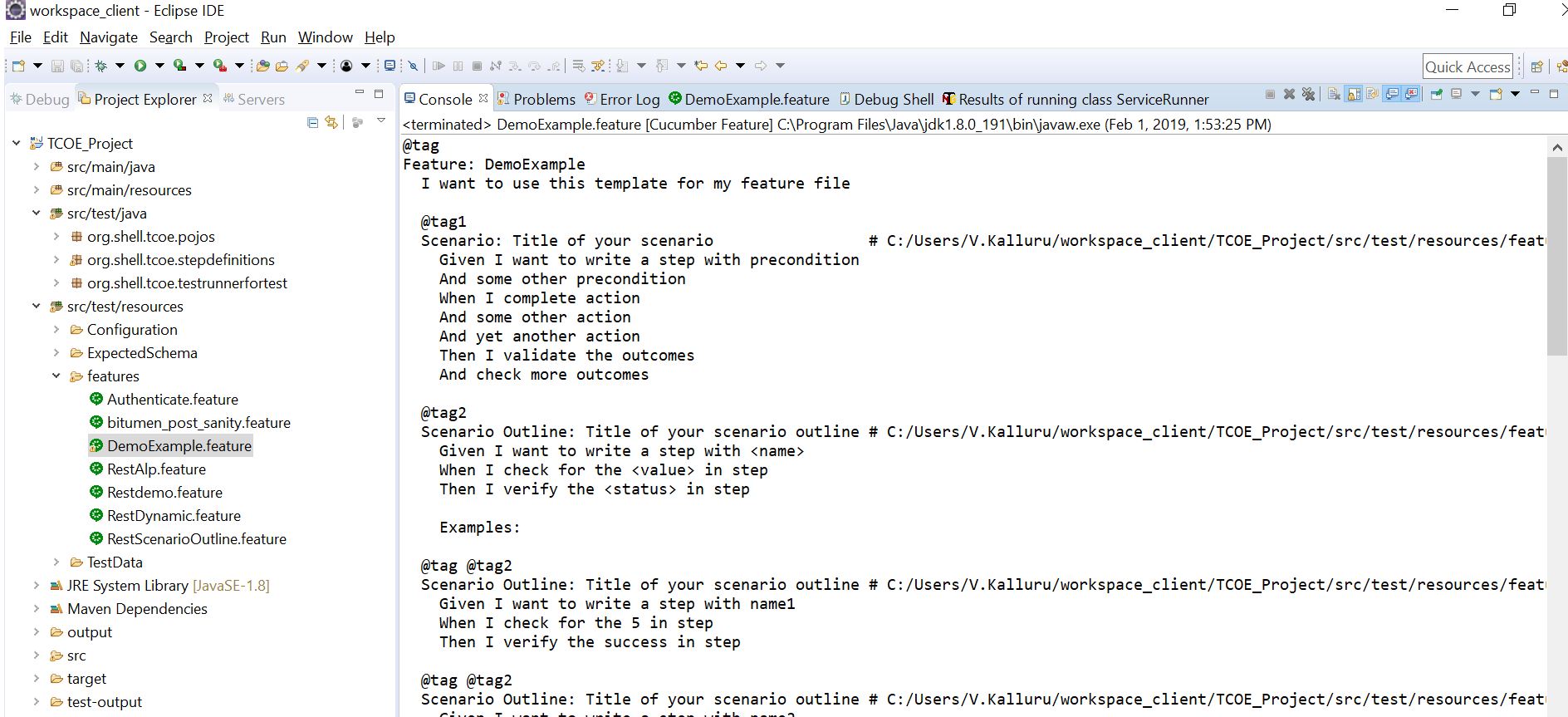
* Type in your feature and add the scenarios in Given, When and Then format (Gherkin format).
* Once all the feature scenarios are written
* Save the feature file



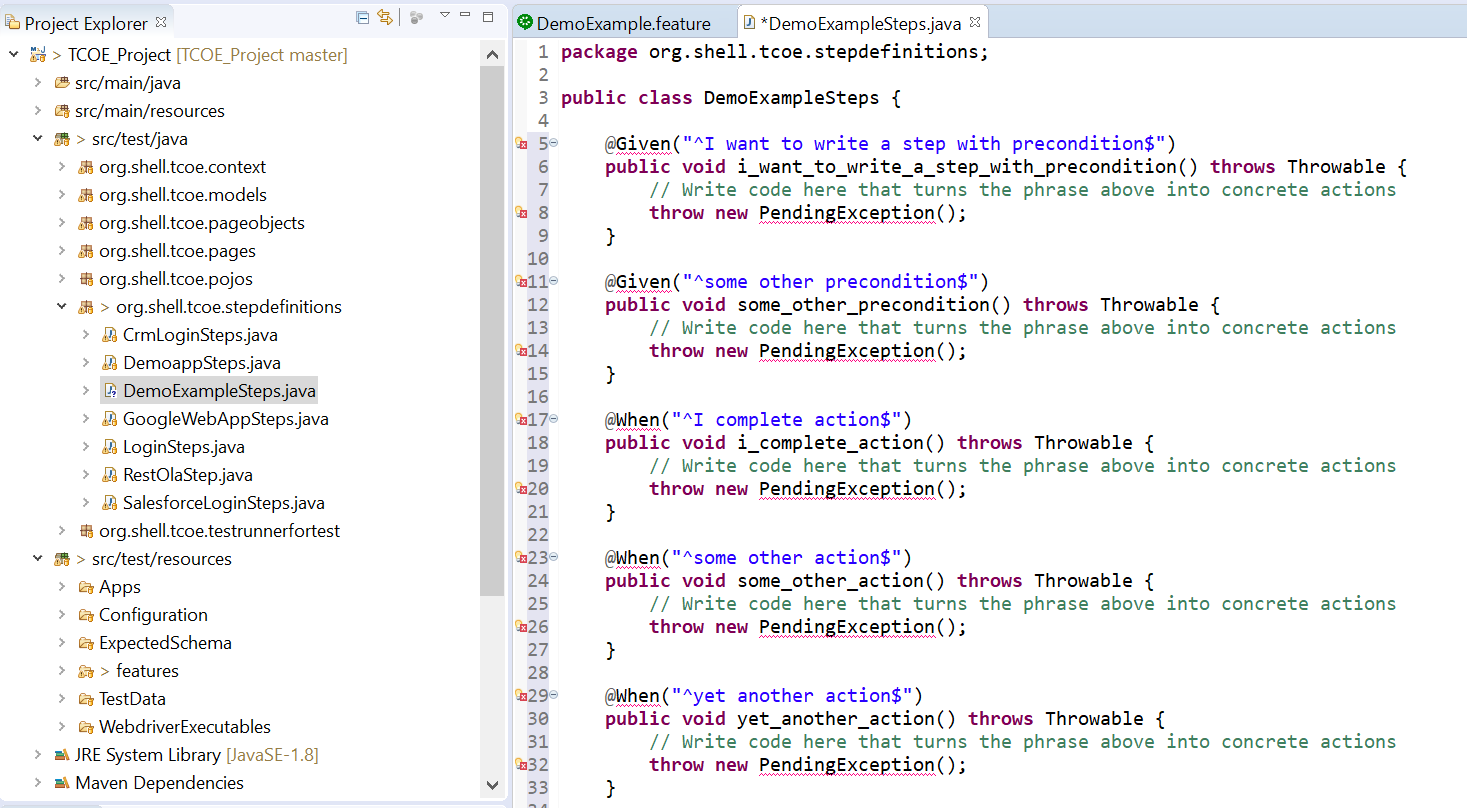
* Right click on the feature file in the editor and select Run as Cucumber feature



* Check the console to see if the missing steps with the snippets were generates
* Now from the console click on copy methods to clip board and paste it in a new or existing class.



* The Step definition class would look as below.

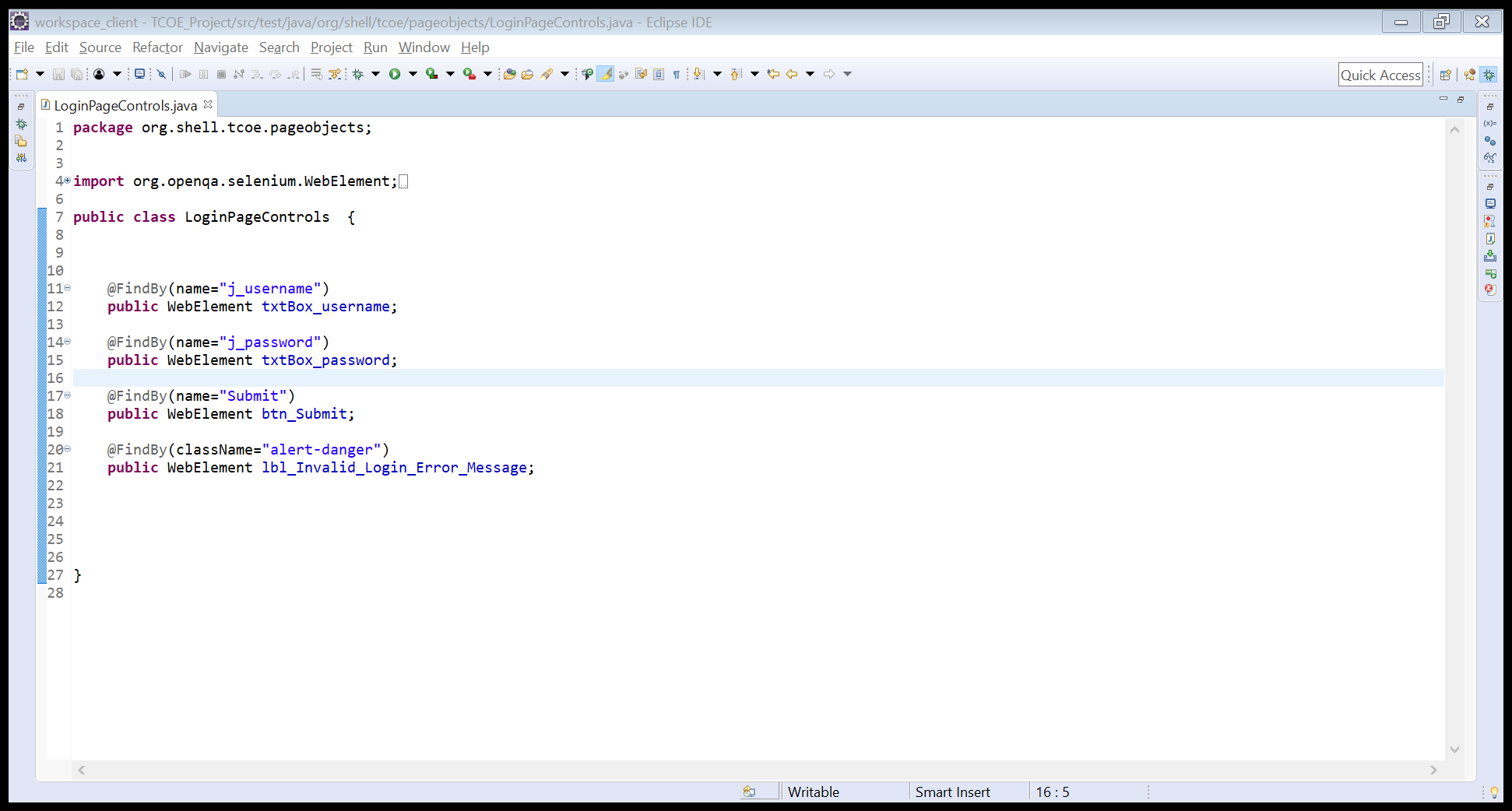


* Here your function library classes and Test data will be called, but before that let’s see how to design those.

**Creating Control Library and Function Library**

* In your src/test/java project, right click on the package - org. shell. tcoe. pageobjects and add a new class. Name the class as <PageName>Controls.java, where the PageName could be Home, Login or any functional name that the web page does.
* Now in the Control class create WebElement properties and use the FindBy annotations and assign attribute values like xpath or ID. These can be identified from the DOM explorer in the browser using F12.

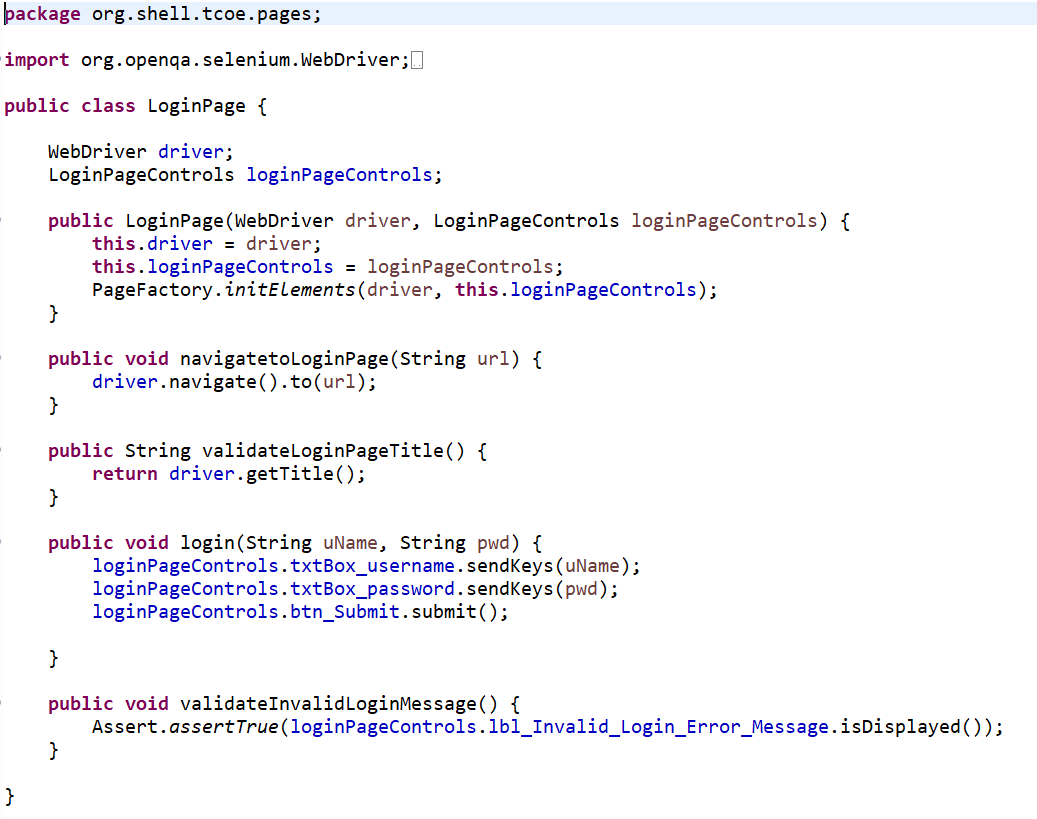
* Once this is done, in your src/test/java project, right click on the package - org. shell. tcoe. pages and add a new class. Name the class as <PageName>Page.java, where the PageName could be Home, Login or any functional name that the web page does.



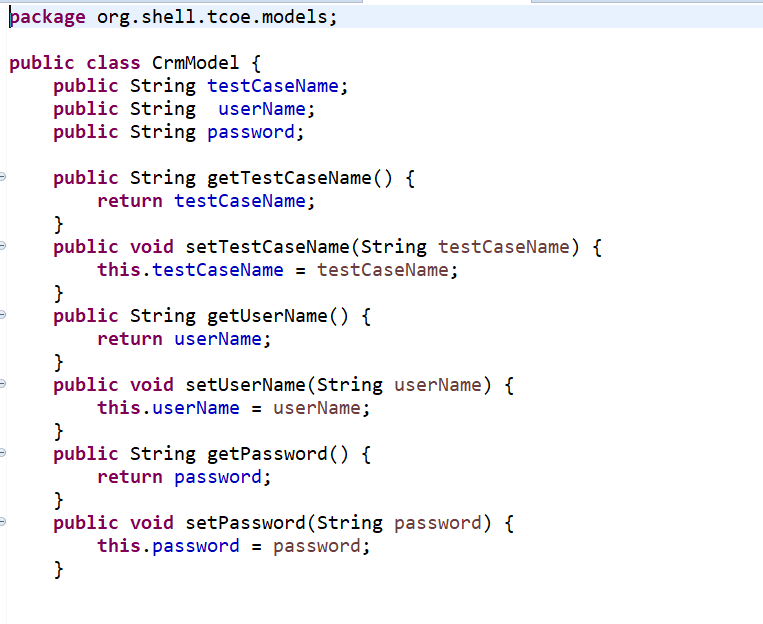


 Now let’s get back to Page class, declare a private read only instance of the corresponding control class and Inject it in

the constructor and initialize it with PageFactory as shown below.



* Now that you Page class is ready, before you start writing the methods for the business function, we need create a Model class with properties to inject test data.
* To do this, in your src/test/java project, right click on the package - org. shell. tcoe. models and add a new class. Name the class as <PageName>Model.java, where the PageName could be Home, Login or any functional name that the web page does.
* In the model class define properties for all the fields that require data input.

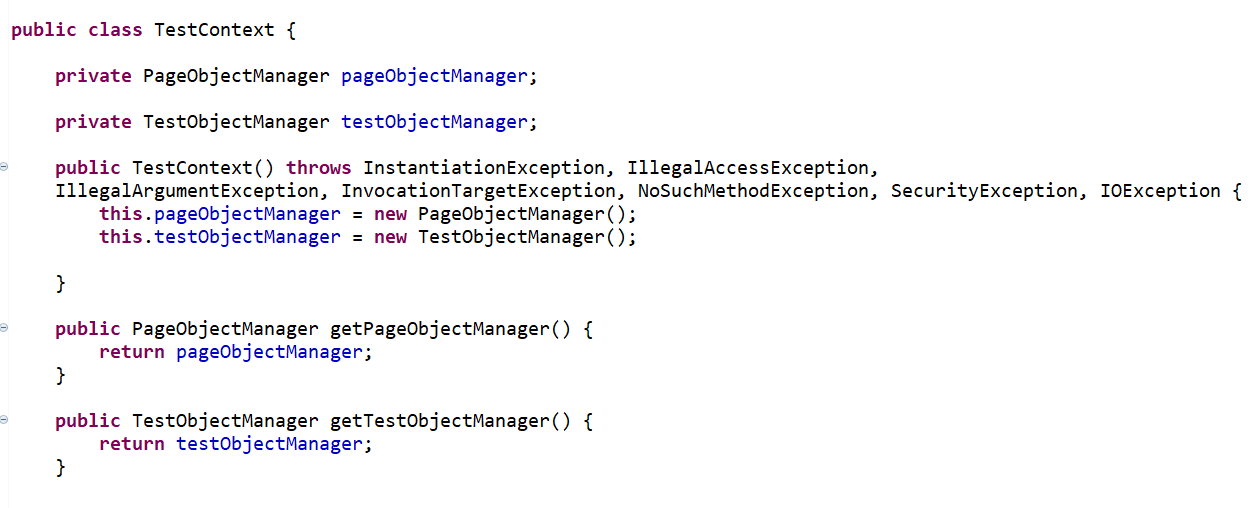


* If source of test data is an excel sheet, then the properties in the Model to match the column names in the excel sheet containing the test data.

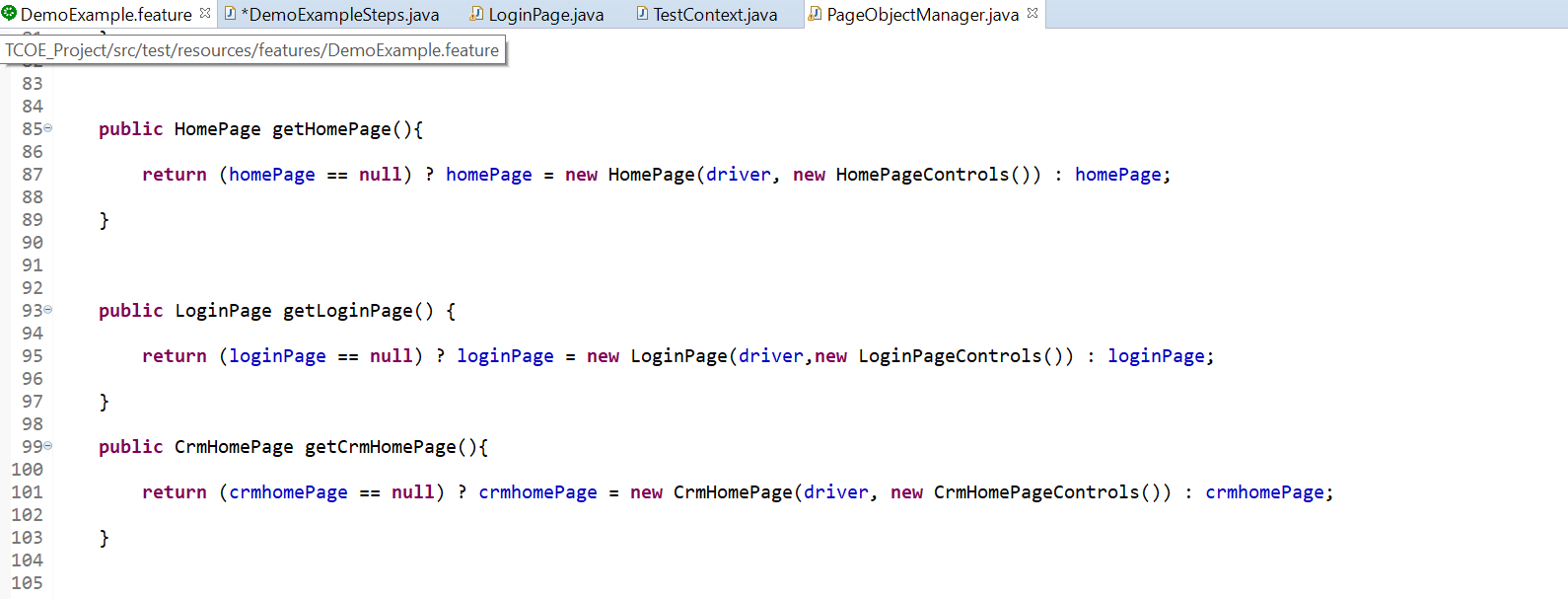
**Test Context**

* The TestContext looks like the image below

It has access to the PageObjectManager (enables access to all the page classes defined) and the TestObjectManager (enables access to the Test Data defined) objects.

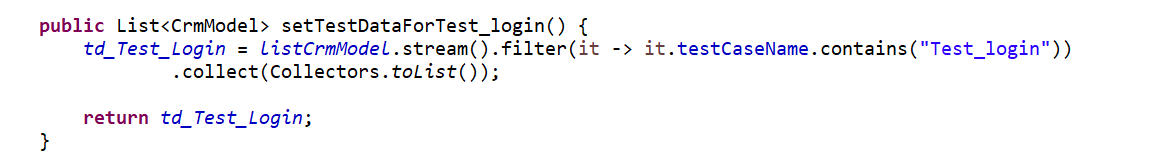


* The pages related to the respective application need to be configured in the PageObjectManager



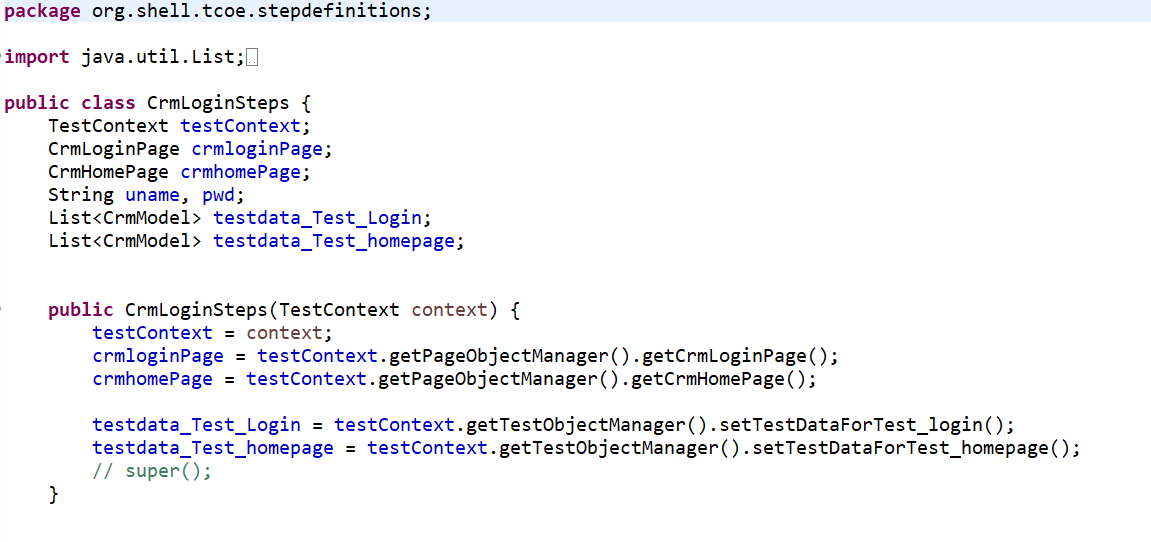
* User can retrieve the required respective test data from an excel sheet or a database in the form of a list using the test data models defined. The test data in the form of lists to be consumed in the Step definitions.

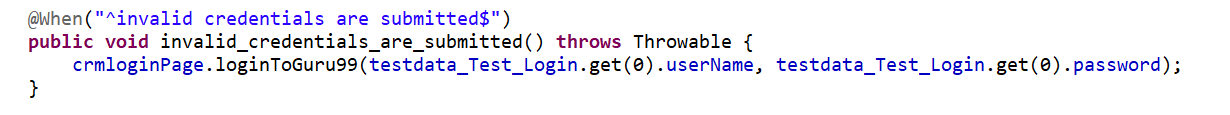




**Using the Library functions and Test data in step definition**

* Now open the step definition class that you generated early and remove below line from all methods.
* ScenarioContext.Current. Pending ();
* Now declare an instance of the necessary function classes and test data classes and Initialize them in a before scenario block as shown below.
* Now got to the respective steps methods and call the function library methods as shown below

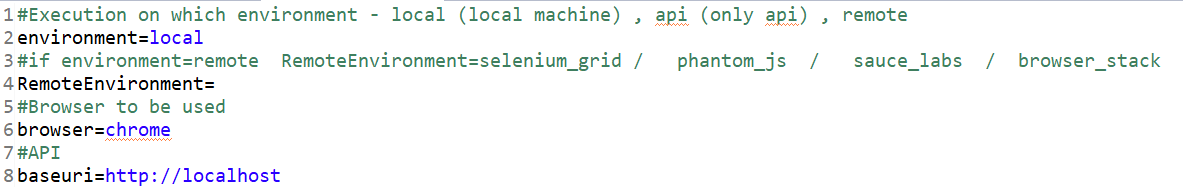




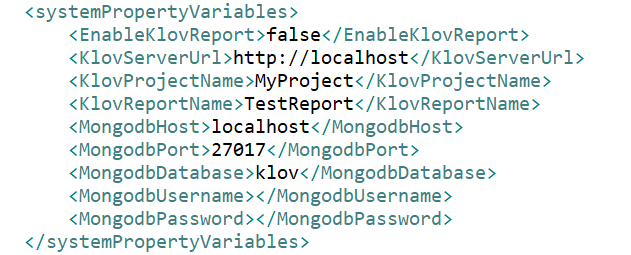
* Your almost done, now all you need to do is update your configurations which is discussed in the next section.

**Setting up Application Configuration (Update Properties)**

* One the Properties file and update the test URL, machine name and environment.



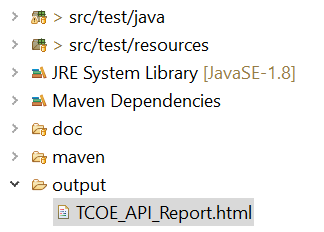
* You can update the klov report settings in the pom.xml if your using the report server.



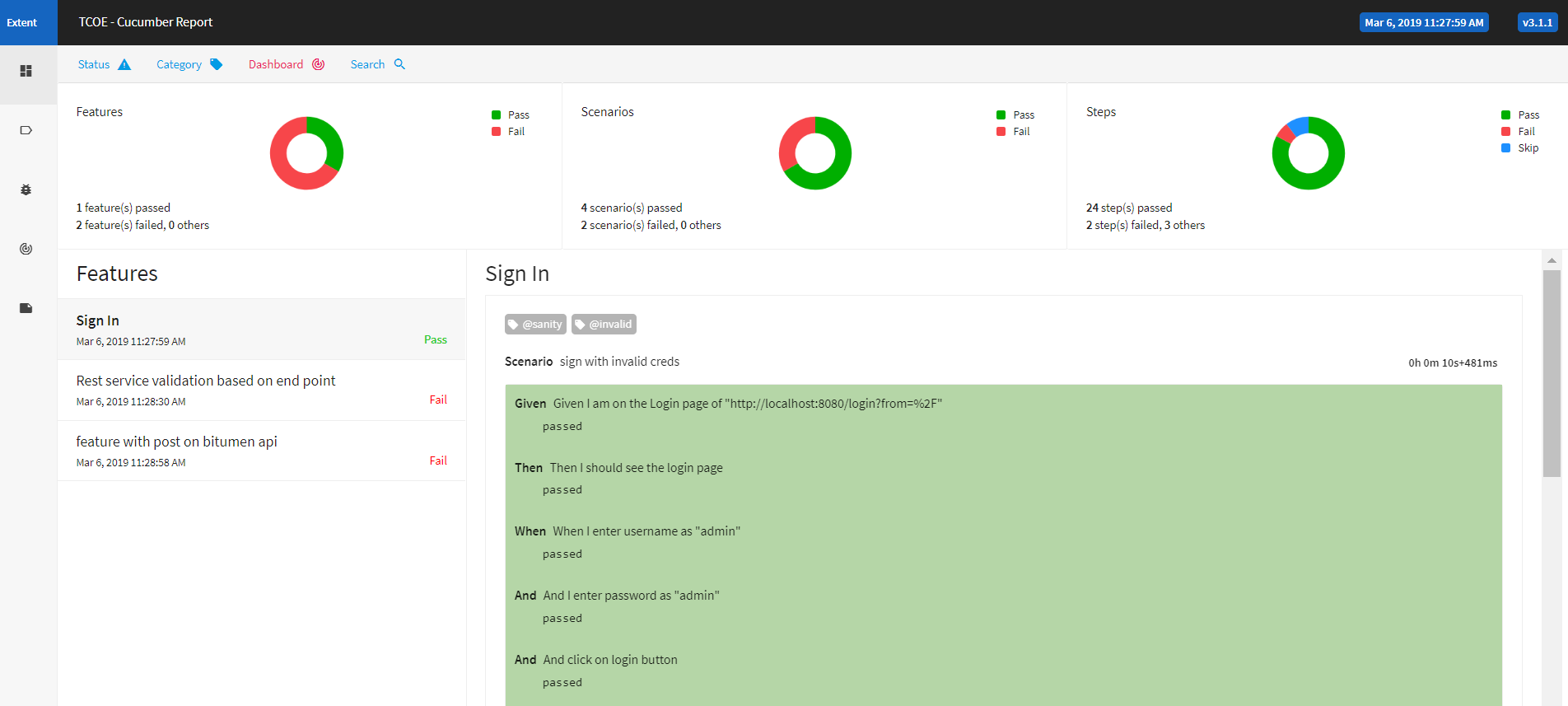
* You could run the suite sequentially using the ServiceRunner as TestNG Test
* Also, you could run the mvn test if you would like to run from the pom.xml
* If you want to run the tests parallelly you could run the command mvn clean test -Pparallelexec
* Now you are all set to run your test cases, go ahead and next we will discuss how to analyze test reports.

**Analyzing Test Reports**

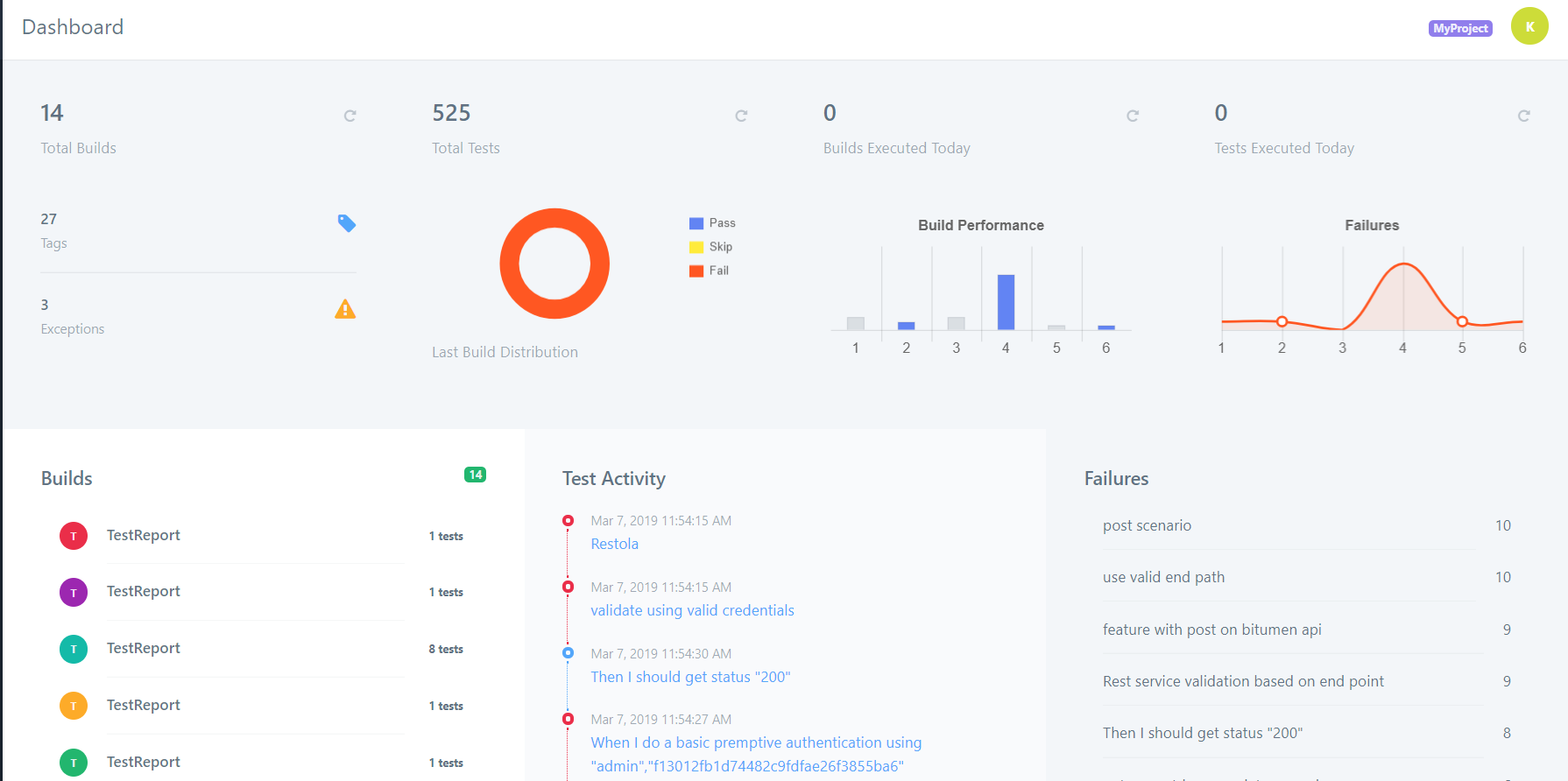
* When it comes to test reports you can use the extent reports that are in BDD format and view the same on Klov report server.
* Once the tests are run, navigate to the projects physical location and one the test results folder
* The reports would look like as shown below.



* The report is grouped by features and there is screen shot attached for any failed tests.



* The report server will give you analytical information about past test run, trends and build information.



Now go ahead and enjoy automating your application. For any question or query please feel free to reach out to TCoE team.

**THANK YOU**