**John lewis Technical Test**

provide list of products having price reduction and apply appropriate data decoration and provide as a rest output

project Introduction 👍

project is build using maven and consists of different modules to facilitate easier code development and separate concerns

**jl-rest-clients** - used to create connection to rest end points and extract the product catalog

**product-pricing-engine-endpoint-ms** - this project exposes the massaged data in rest format and call the rest client services

**jl-utils -** is responsible to convert json string to respective java POJO classes, as models were not provided

product-pricing-test-suit - contains BDD user stories and acceptance criteria, this is live test framework to test all moving parts and assure that all acceptance criteria's are meet

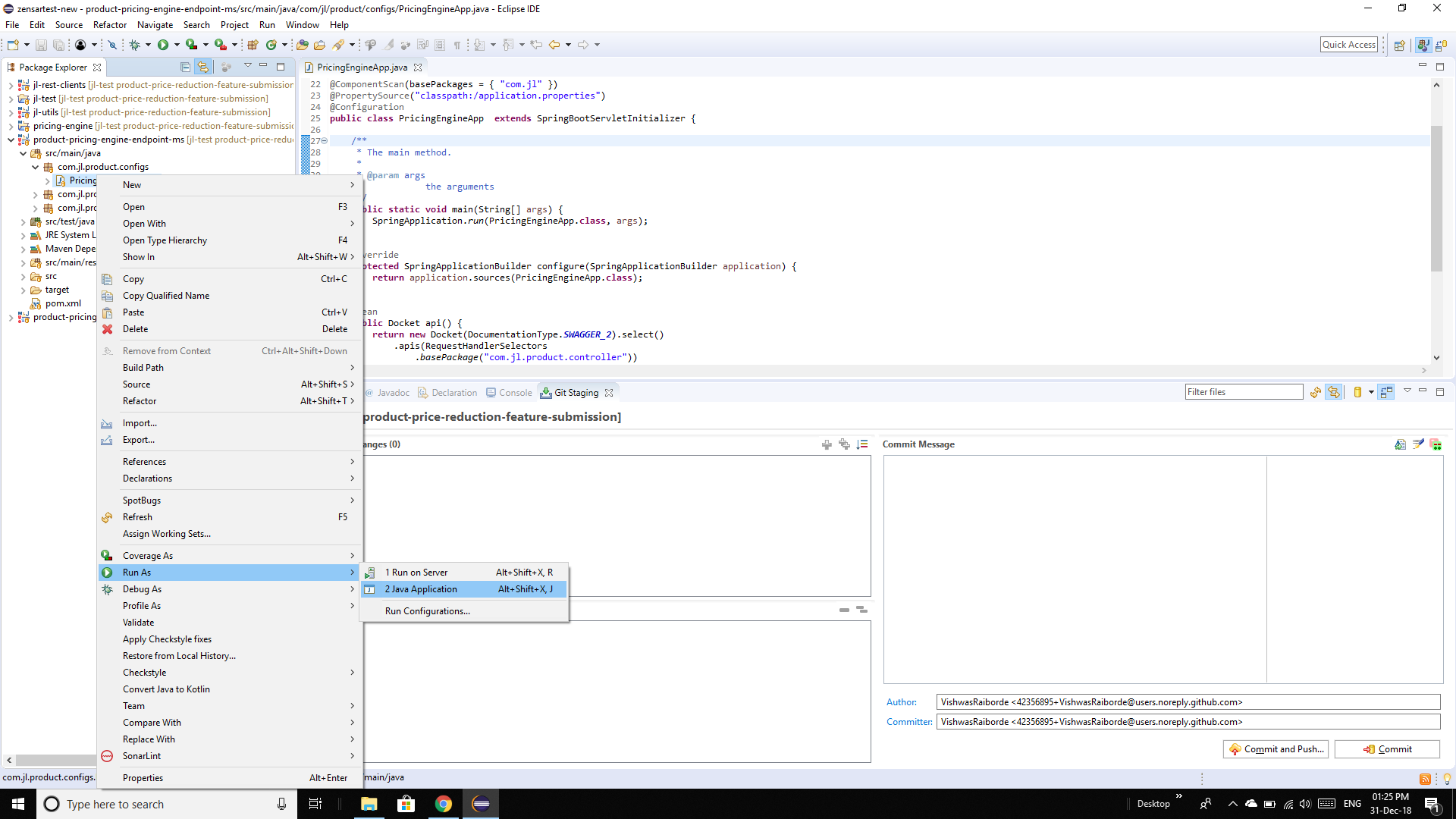
pricing-engine - parent maven project encapsulating all sub modules

Running the project

a) make a git check out using eclipse from the feature branch https://github.com/VishwasRaiborde/jl-test/tree/product-price-reduction-feature-submission

b) build the parent project using : mvn clean package install

c) once the project is build and all dependencies are resolved

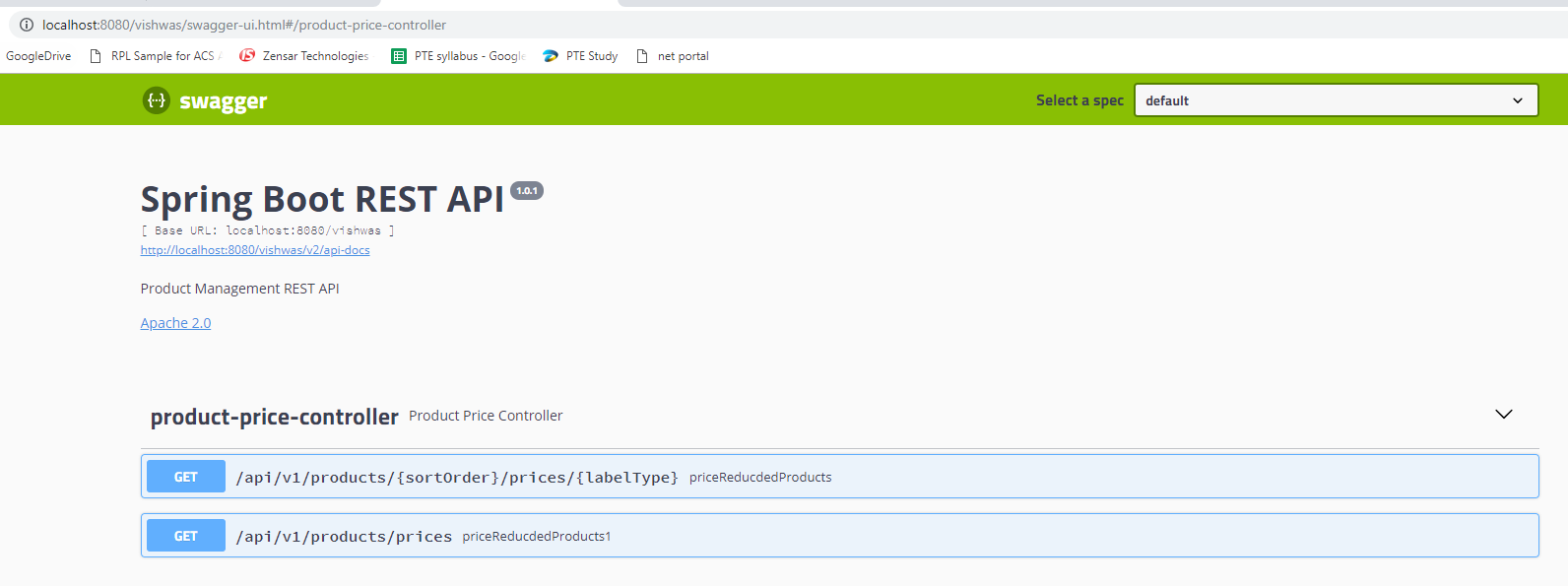
d) run

this will start the spring context and start the application

e) open a browser of your choice and open the url /http://localhost:8080//product-price-ms/swagger-ui.html

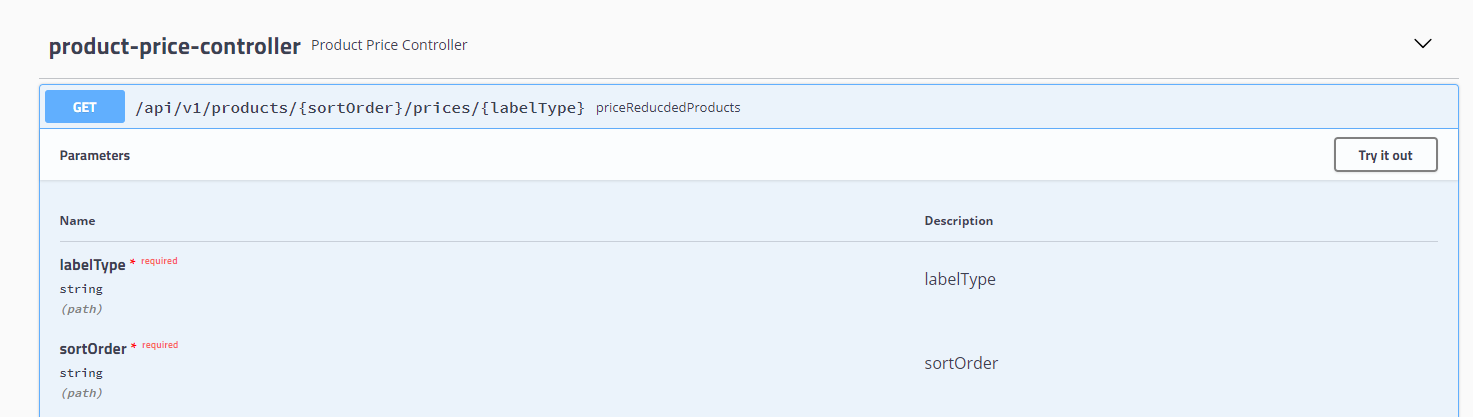
f) once the swagger page is open the application is ready for use

g) click on product price controller and two interfaces should be visible

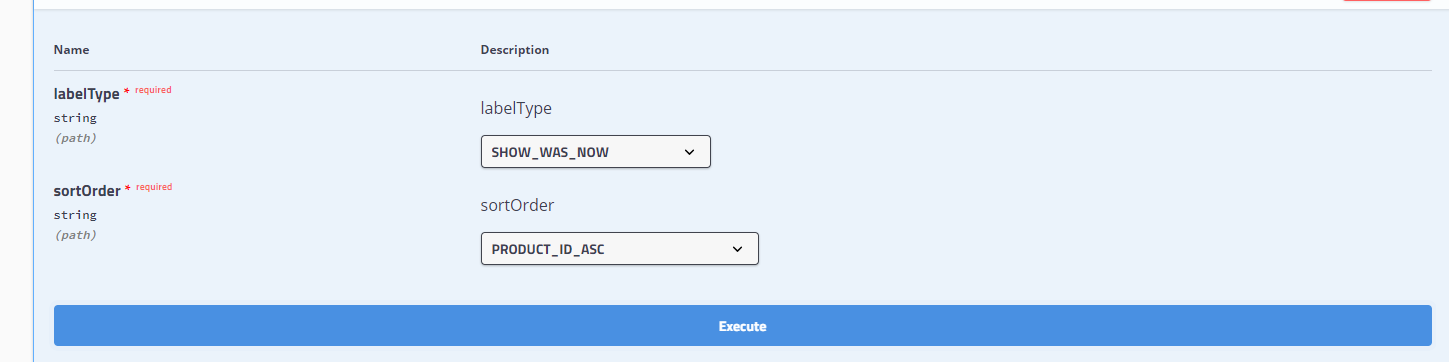


h) click on [**/api/v1/products/{sortOrder}/prices/{labelType}**](http://localhost:8080/vishwas/swagger-ui.html#/operations/product-price-controller/priceReducdedProductsUsingGET)

priceReducdedProducts and hit “ try out “



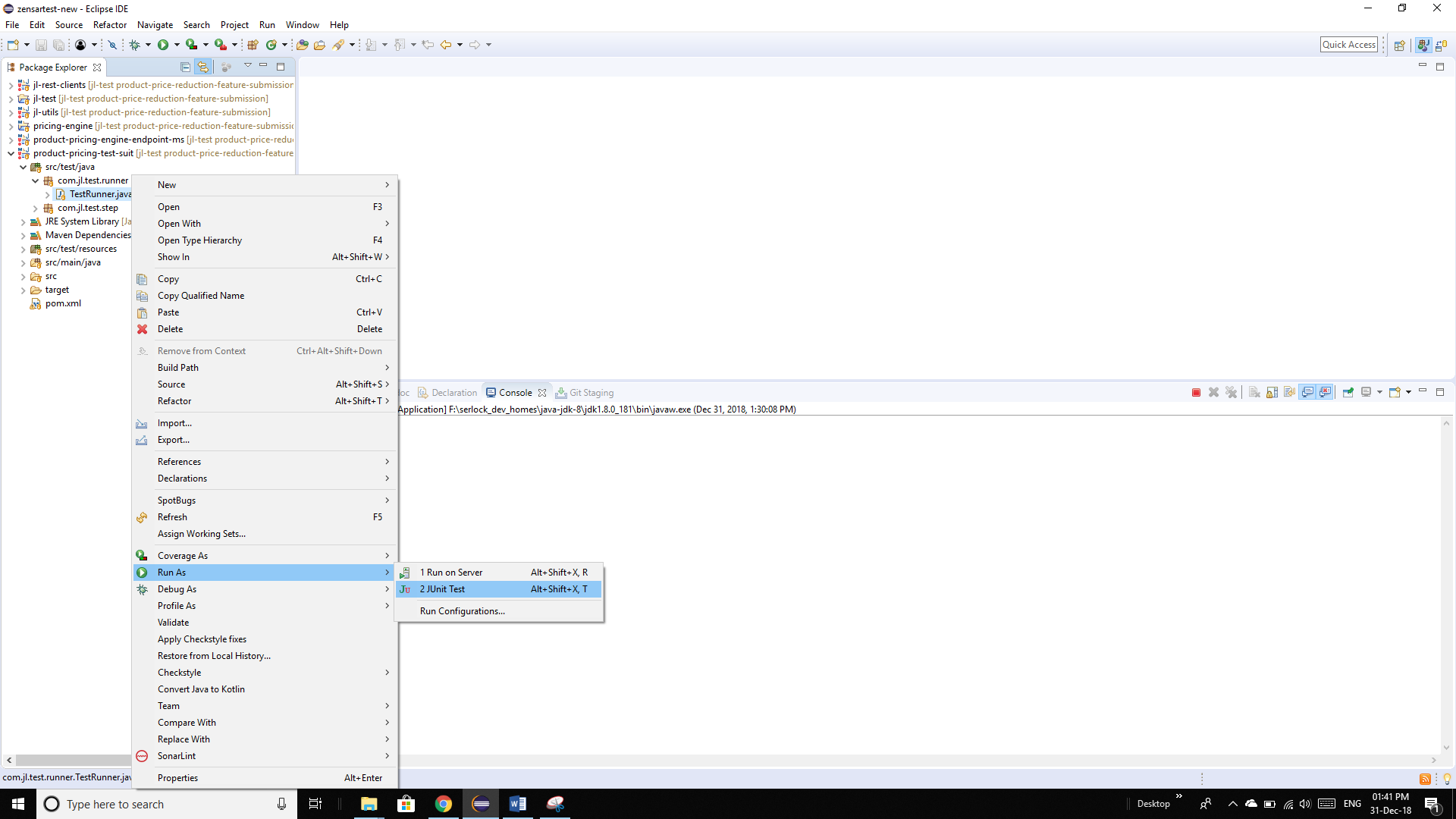
1. Select the required filter from the UI and click execute

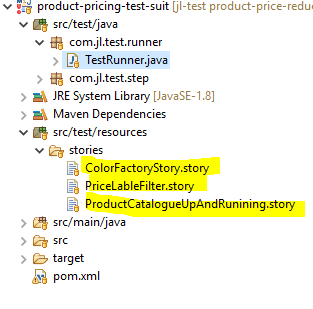
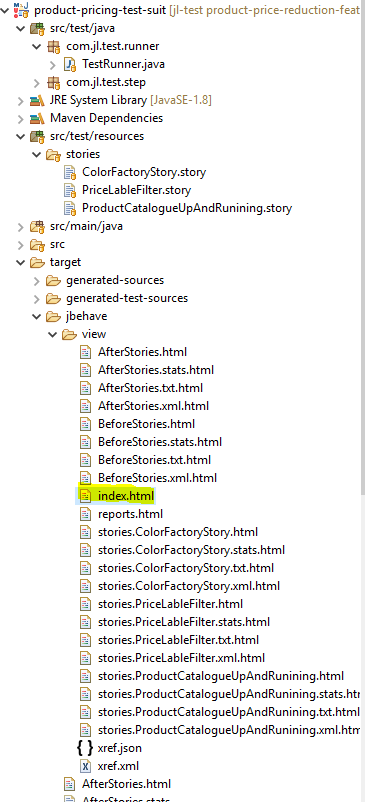
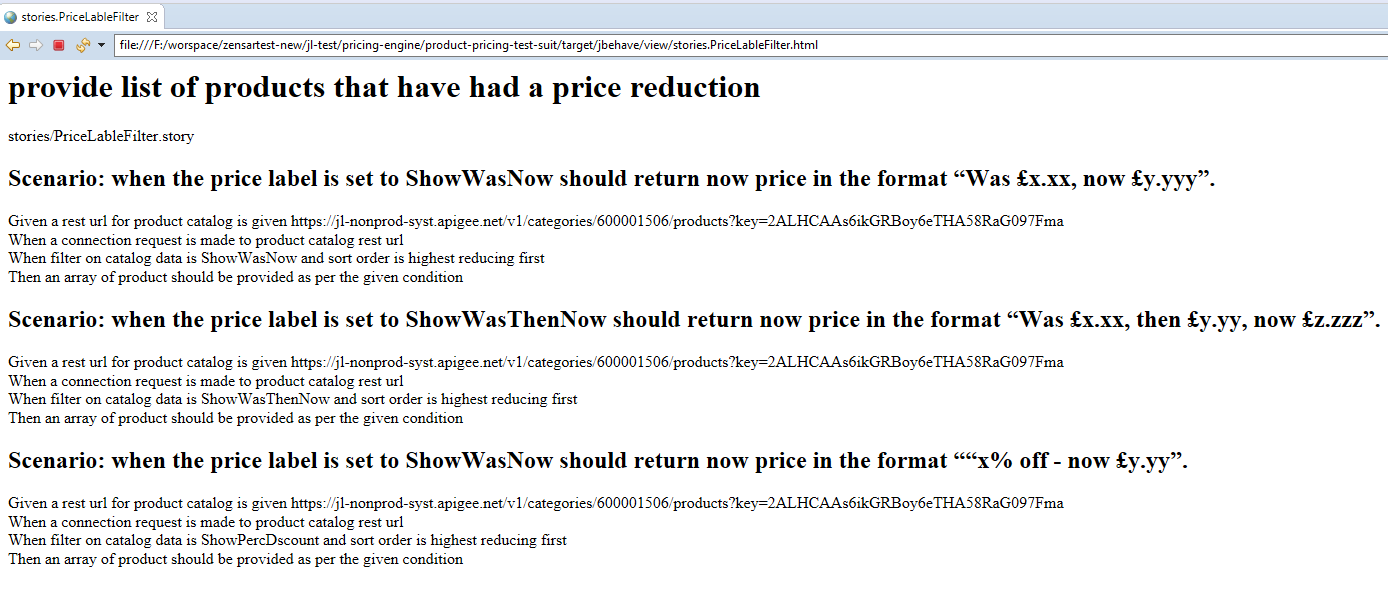


A combination of these filter will yield the output accordingly

Runnig the Live test : product-pricing-test-suit

This feature is testing using automation and Behavioral Driven Development as the code was complex and would result in fix one break one issue and hence needed a acceptance criteria based automated testing



1. Run the TestRunner as a unit test
2. Live test project will then execute the acceptance criteria for 3 stories
3. 
4. The test output can be found on console or a test report can be found at
5. 
6. 

Assumptions :

A ) A new product Catalogue is returned and hence there is no Caching done to fetch new products

B ) Micro service runs on a auto scalable and white listed environment and hence no O auth is required

c) An Empty Collection is required on request / method failure