Lesson 02 Demo 03

Executing Test Cases with Cucumber

Objective: To execute test cases with Cucumber to automate testing, enhance collaboration, and ensure the software meets specifications

Tool required: Eclipse IDE

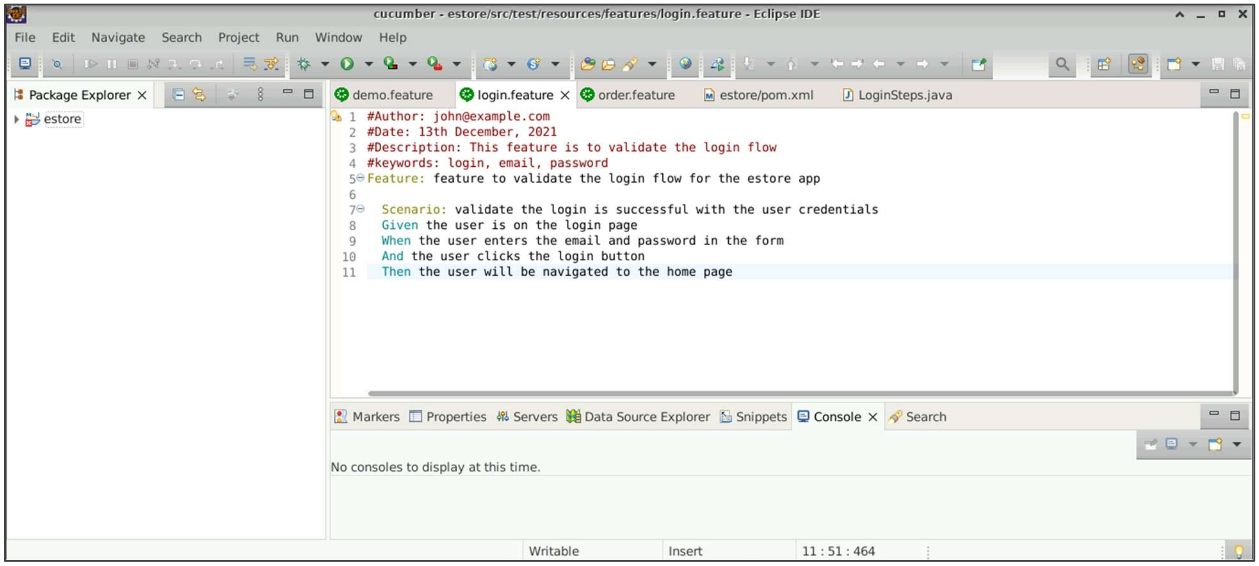
Prerequisites: None

Steps to be followed:

1. Add dependencies to the **pom.xml** file
2. Add annotations in the AppTest class
3. Execute the configuration

Step 1: Add dependencies to the pom.xml file

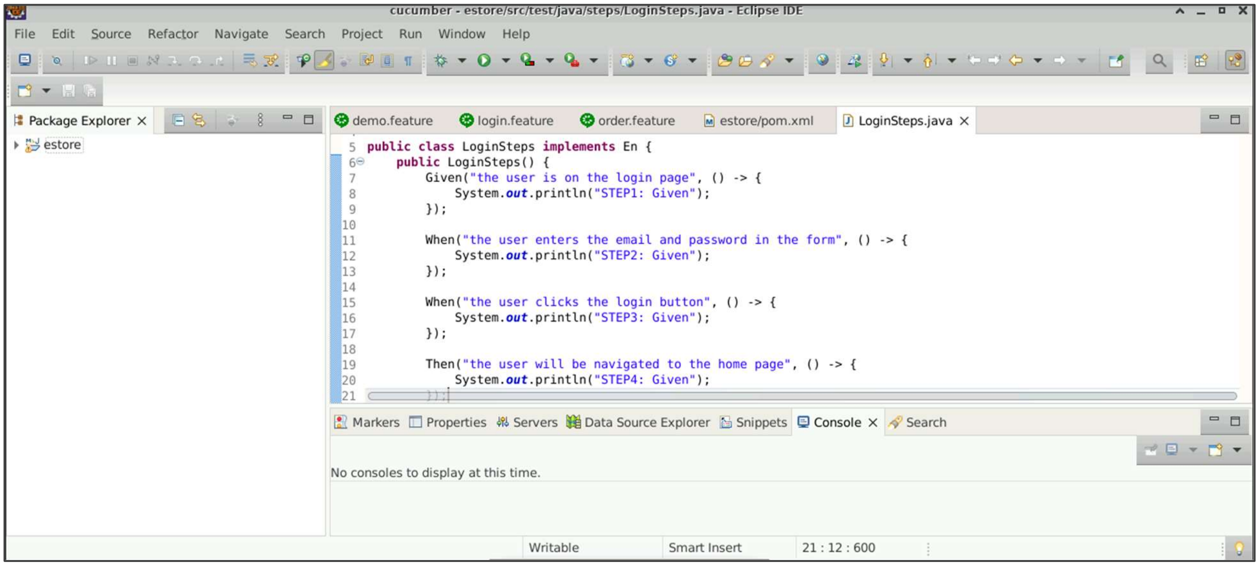
1.1 Navigate to the **estore** project and open the **login.feature** from the **resources** feature



```
#Author: john@example.com
#Date: 13th December, 2021
#Description: This feature is to validate the login flow
#keywords: login, email, password
Feature: feature to validate the login flow for the estore app

  Scenario: validate the login is successful with the user credentials
  Given the user is on the login page
  When the user enters the email and password in the form
  And the user clicks the login button
  Then the user will be navigated to the home page
```

```
No consoles to display at this time.
```

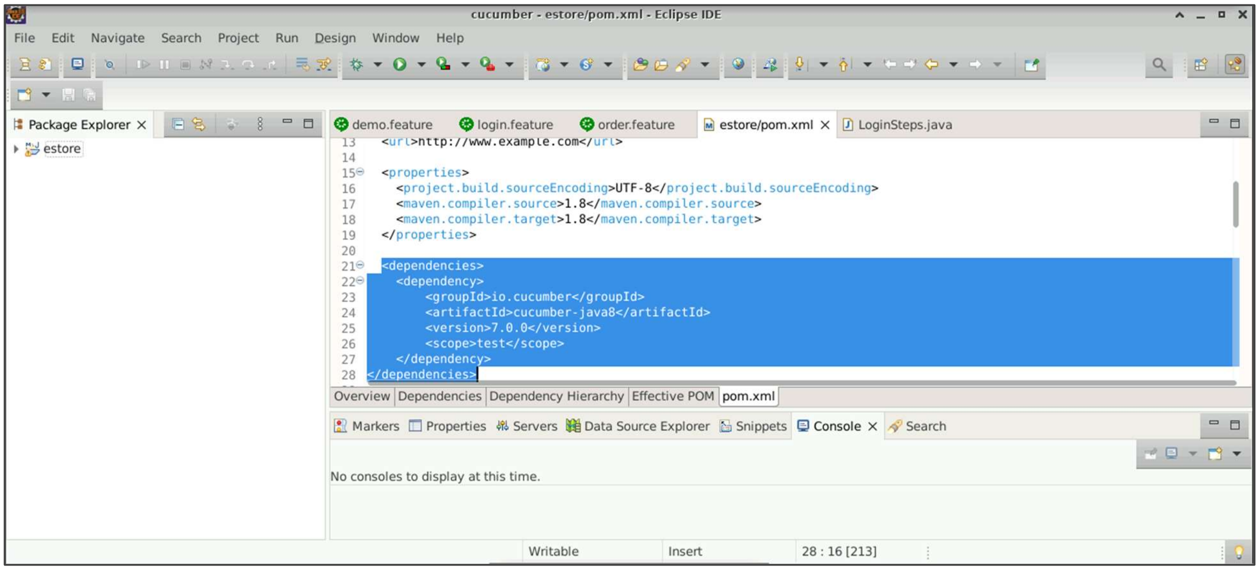
1.2 With respect to the steps given in the **Scenario** in the **login.feature** file, you can get the steps definition in the Java file called **LoginSteps** with the annotations **@Given**, **@When**, **@And**, and **@Then**.



```
public class LoginSteps implements En {
    public LoginSteps() {
        Given("the user is on the login page", () -> {
            System.out.println("STEP1: Given");
        });

        When("the user enters the email and password in the form", () -> {
            System.out.println("STEP2: Given");
        });

        When("the user clicks the login button", () -> {
            System.out.println("STEP3: Given");
        });

        Then("the user will be navigated to the home page", () -> {
            System.out.println("STEP4: Given");
        });
```

No consoles to display at this time.

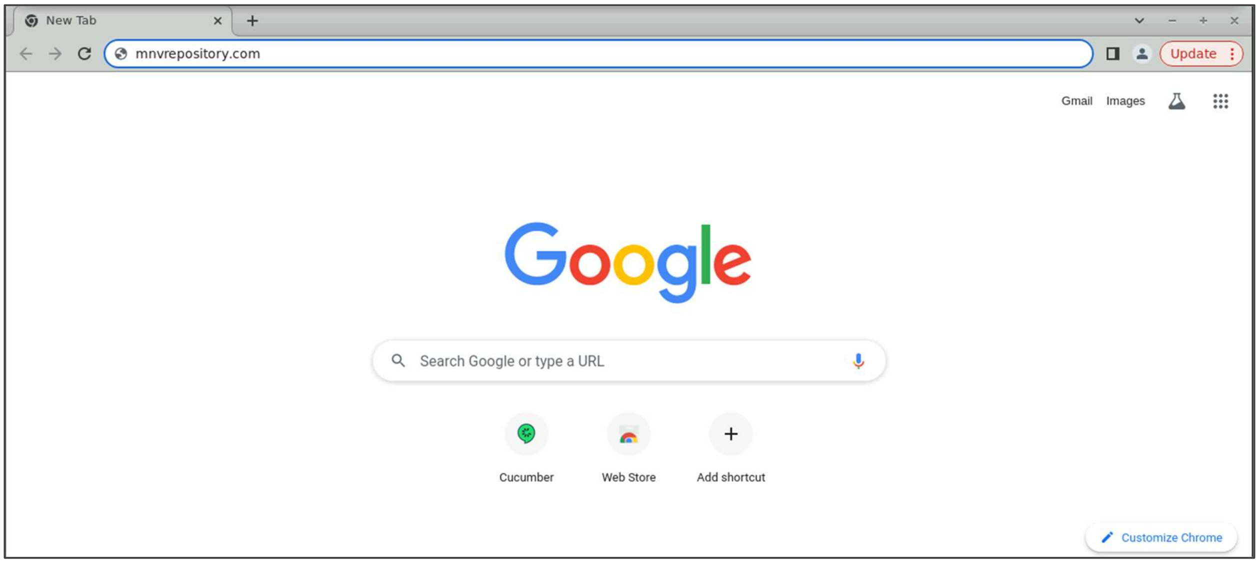
1.3 To execute the test, you need to add a few dependencies in the **pom.xml** file. Then check the current dependencies in the pom.xml.

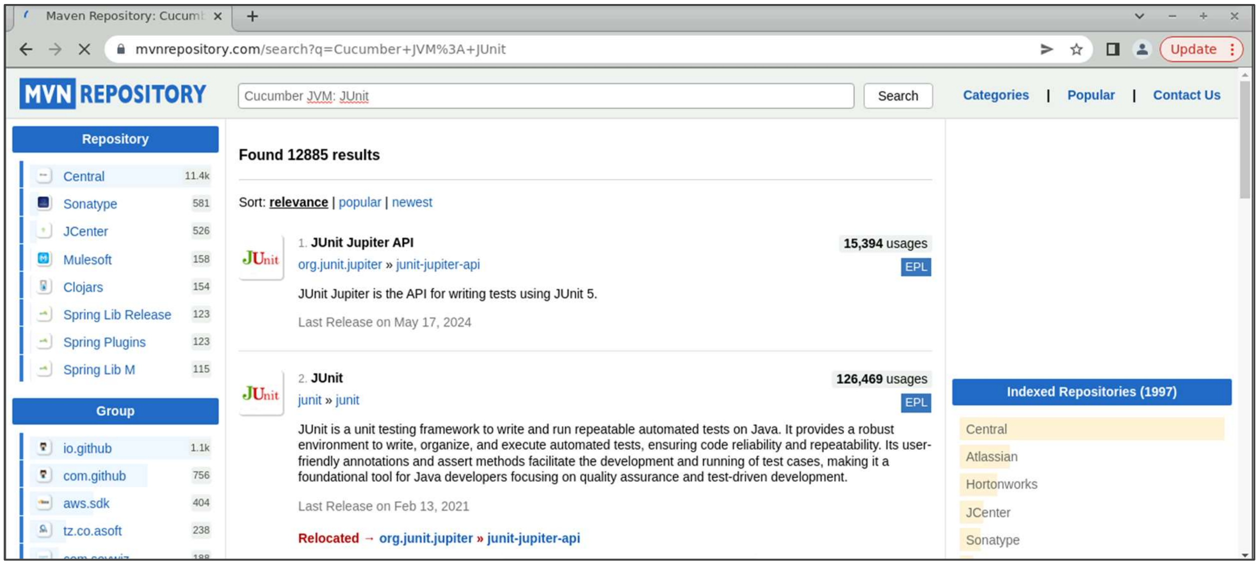


```
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-java8</artifactId>
        <version>7.0.0</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

No consoles to display at this time.

1.4 Now, you can add another dependency named **Cucumber-JUnit.** To do this, search for **Cucumber JVM: JUnit** in **mvnrepository.com**.




Maven Repository: Cucumb
mvnrepository.com/search?q=Cucumber+JVM%3A+JUnit
Update
MVN REPOSITORY
Cucumber JVM: JUnit
Search
Categories | Popular | Contact Us

Repository	
Central	11.4k
Sonatype	581
JCenter	526
Mulesoft	158
Clojars	154
Spring Lib Release	123
Spring Plugins	123
Spring Lib M	115

Group	
io.github	1.1k
com.github	756
aws.sdk	404
tz.co.asoft	238

Found 12885 results

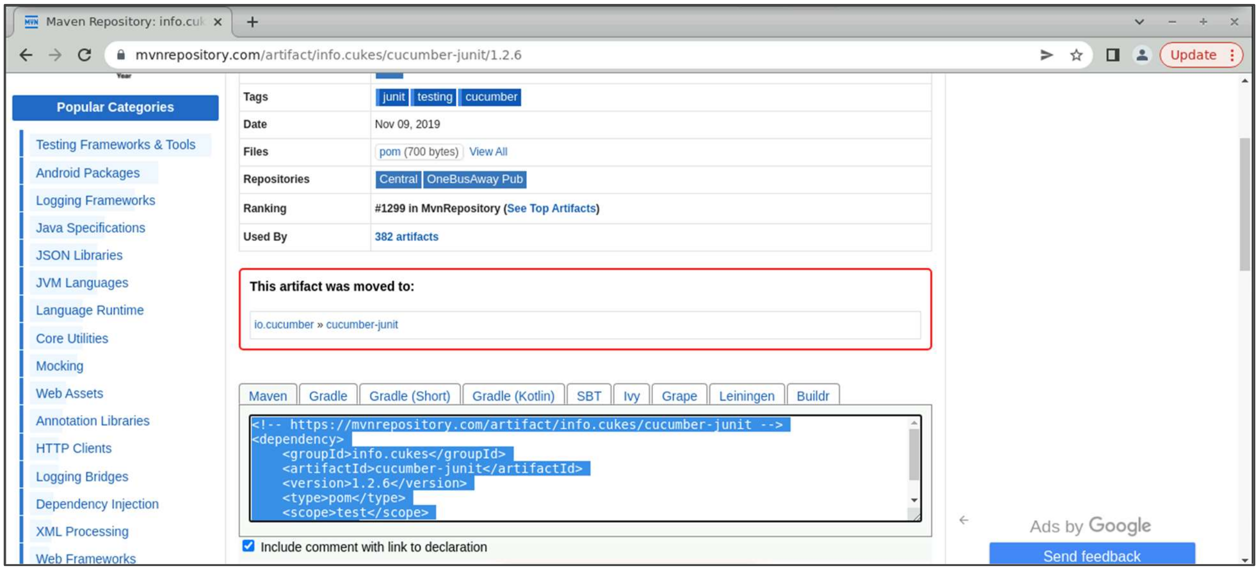
Sort: **relevance** | popular | newest

1. **JUnit Jupiter API** **15,394** usages
org.junit.jupiter » junit-jupiter-api EPL
JUnit Jupiter is the API for writing tests using JUnit 5.
Last Release on May 17, 2024

2. **JUnit** **126,469** usages
junit » junit EPL
JUnit is a unit testing framework to write and run repeatable automated tests on Java. It provides a robust environment to write, organize, and execute automated tests, ensuring code reliability and repeatability. Its user-friendly annotations and assert methods facilitate the development and running of test cases, making it a foundational tool for Java developers focusing on quality assurance and test-driven development.
Last Release on Feb 13, 2021
Relocated → org.junit.jupiter » junit-jupiter-api

Indexed Repositories (1997)
Central
Atlassian
Hortonworks
JCenter
Sonatype

1.5 Select the dependency and then paste it into the **pom.xml** file



Tags	junit testing cucumber
Date	Nov 09, 2019
Files	pom (700 bytes) View All
Repositories	Central OneBusAway Pub
Ranking	#1299 in MvnRepository (See Top Artifacts)
Used By	382 artifacts

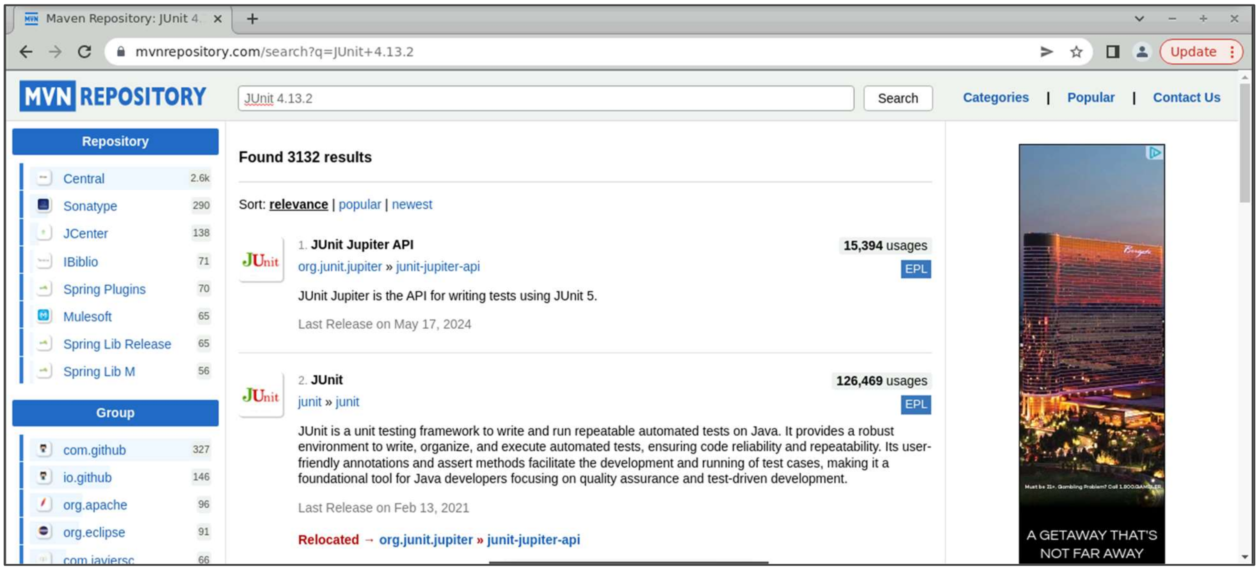
This artifact was moved to:

io.cucumber » cucumber-junit

```
<!-- https://mvnrepository.com/artifact/info.cukes/cucumber-junit -->
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>1.2.6</version>
    <type>pom</type>
    <scope>test</scope>
```

☑ Include comment with link to declaration

1.6 Now, you can add another dependency named **JUnit**. To do this, search for **JUnit** in mvnrepository.com.



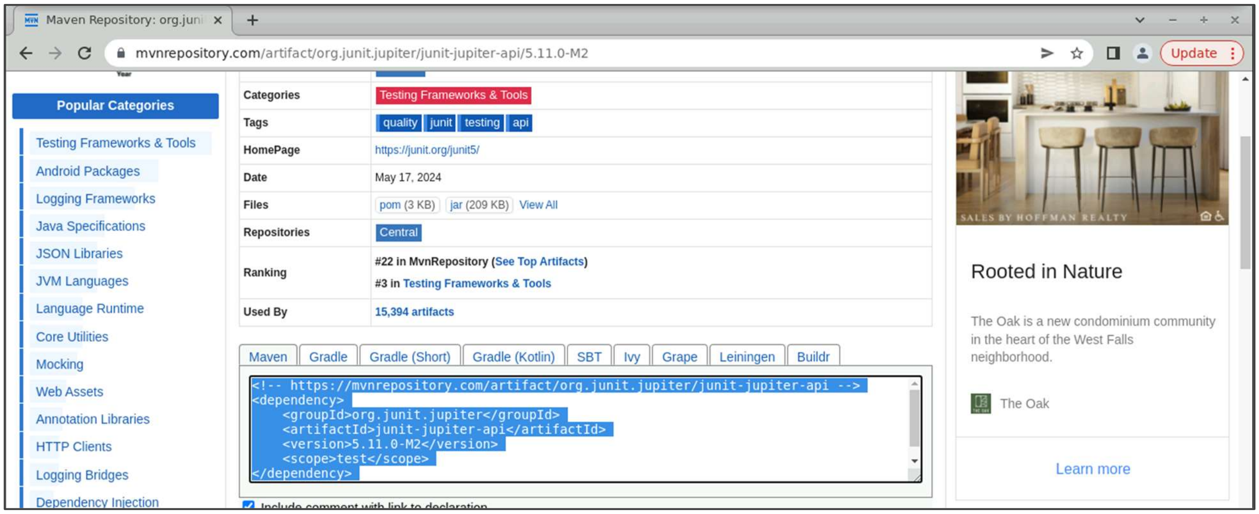
Found 3132 results

Sort: relevance | popular | newest

1. JUnit Jupiter API — 15,394 usages
org.junit.jupiter » junit-jupiter-api — EPL
JUnit Jupiter is the API for writing tests using JUnit 5.
Last Release on May 17, 2024

2. JUnit — 126,469 usages
junit » junit — EPL
JUnit is a unit testing framework to write and run repeatable automated tests on Java. It provides a robust environment to write, organize, and execute automated tests, ensuring code reliability and repeatability. Its user-friendly annotations and assert methods facilitate the development and running of test cases, making it a foundational tool for Java developers focusing on quality assurance and test-driven development.
Last Release on Feb 13, 2021
Relocated → org.junit.jupiter » junit-jupiter-api

1.7 copy the dependency and paste it to the **pom.xml** file

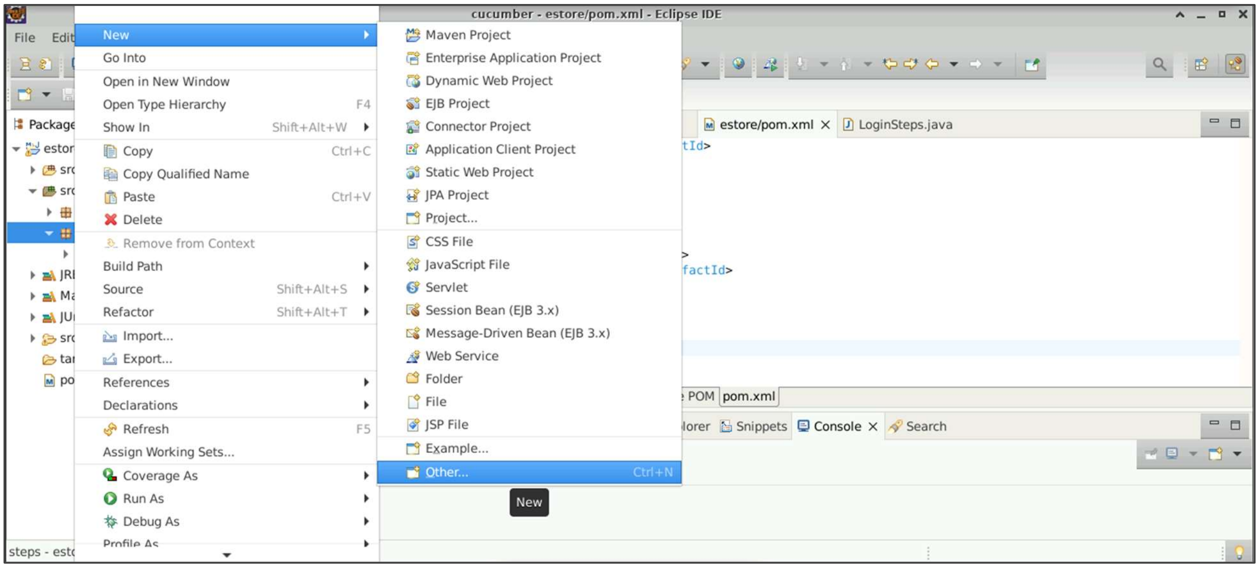


Categories	Testing Frameworks & Tools
Tags	quality junit testing api
HomePage	https://junit.org/junit5/
Date	May 17, 2024
Files	pom (3 KB) jar (209 KB) View All
Repositories	Central
Ranking	#22 in MvnRepository (See Top Artifacts) #3 in Testing Frameworks & Tools
Used By	15,394 artifacts

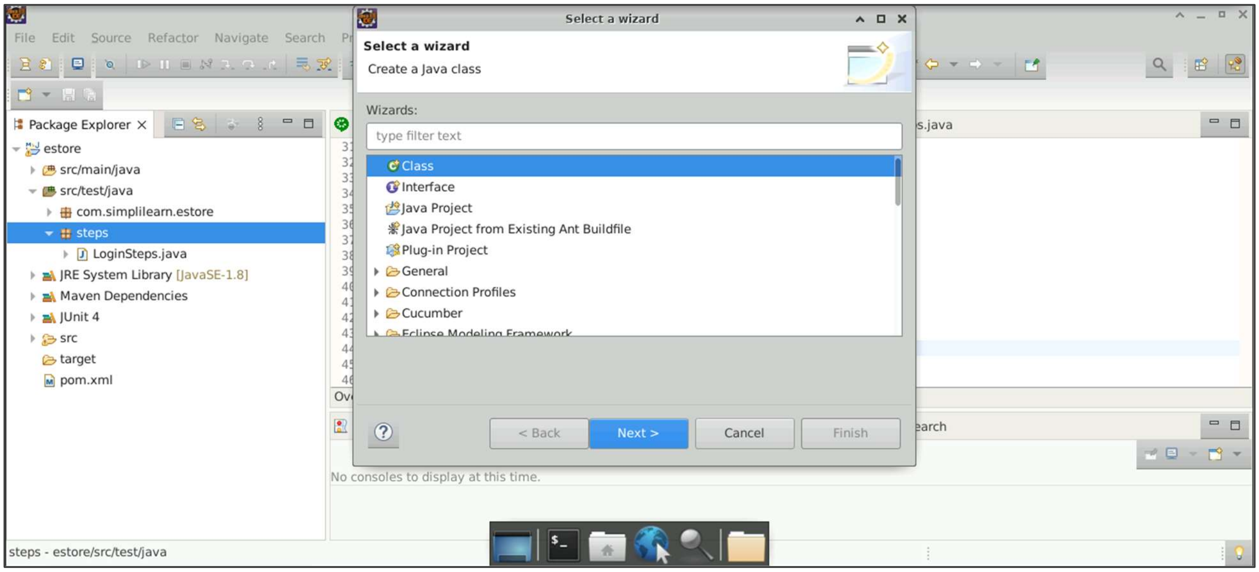
```
<!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.11.0-M2</version>
    <scope>test</scope>
</dependency>
```

Step 2: Add annotations in the AppTest class

2.1 Navigate to the **src/test/java**, Select **New**, and then click on **Other**



2.2 Select **Class** and click on **Next**. Then, create a class named **AppTest.**



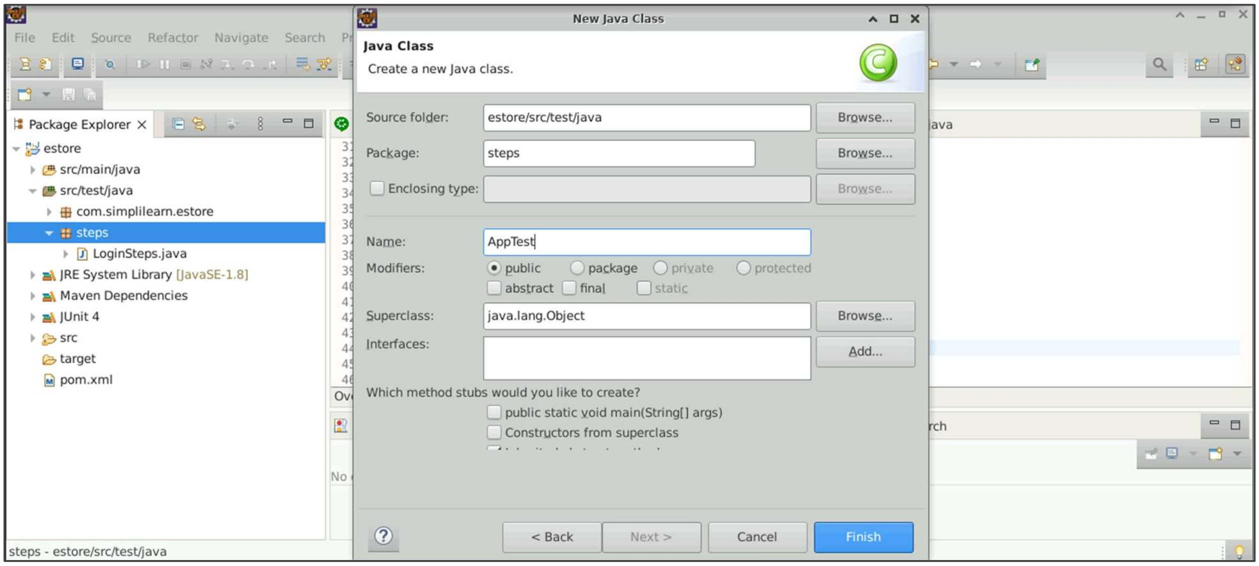
Select a wizard

Create a Java class

Wizards:

type filter text

- Class
- Interface
- Java Project
- Java Project from Existing Ant Buildfile
- Plug-in Project
- General
- Connection Profiles
- Cucumber
- Eclipse Modeling Framework



New Java Class

Java Class

Create a new Java class.

Source folder: estore/src/test/java

Package: steps

□ Enclosing type:

Name: AppTest

Modifiers: ● public ○ package ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

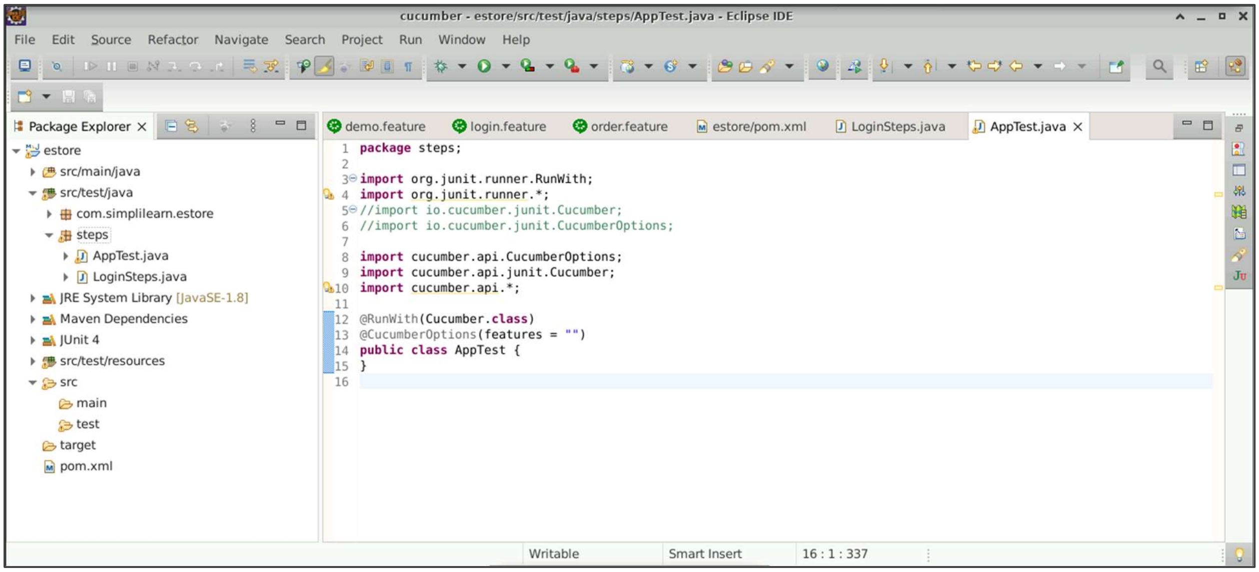
Interfaces:

Which method stubs would you like to create?

□ public static void main(String[] args)

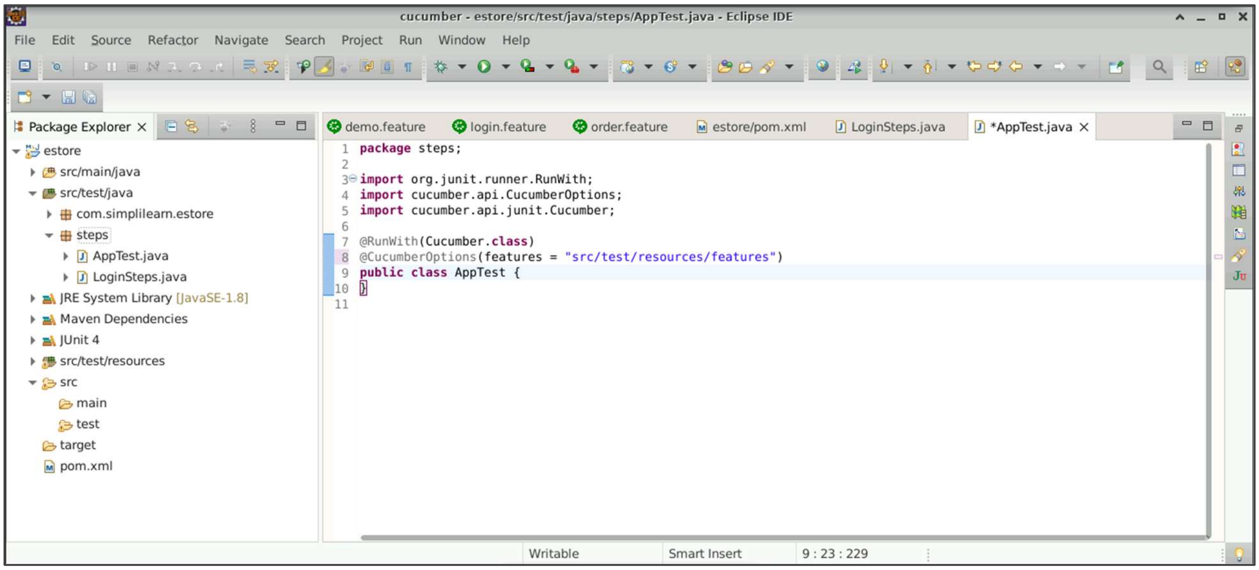
□ Constructors from superclass

2.3 In the **AppTest.java** file, add two annotations **@RunWith** and **@CucumberOptions**



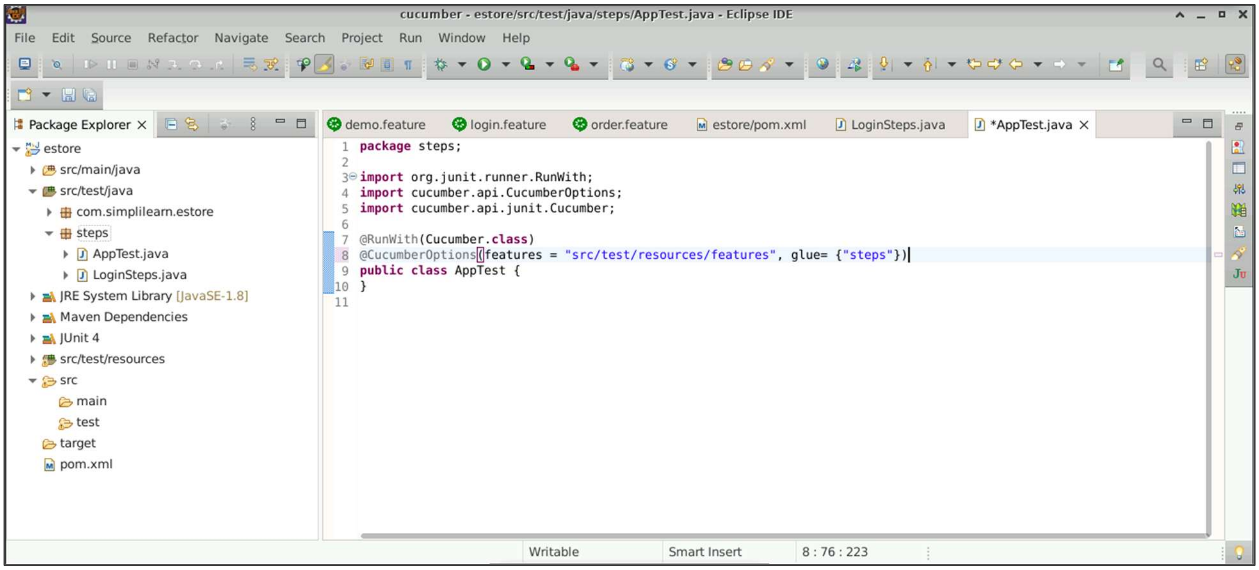
```
package steps;

import org.junit.runner.RunWith;
import org.junit.runner.*;
//import io.cucumber.junit.Cucumber;
//import io.cucumber.junit.CucumberOptions;

import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;
import cucumber.api.*;

@RunWith(Cucumber.class)
@CucumberOptions(features = "")
public class AppTest {
}
```

2.4 Copy the **src/test/resources** path to the **feature** file



```
package steps;

import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(features = "src/test/resources/features")
public class AppTest {
}
```

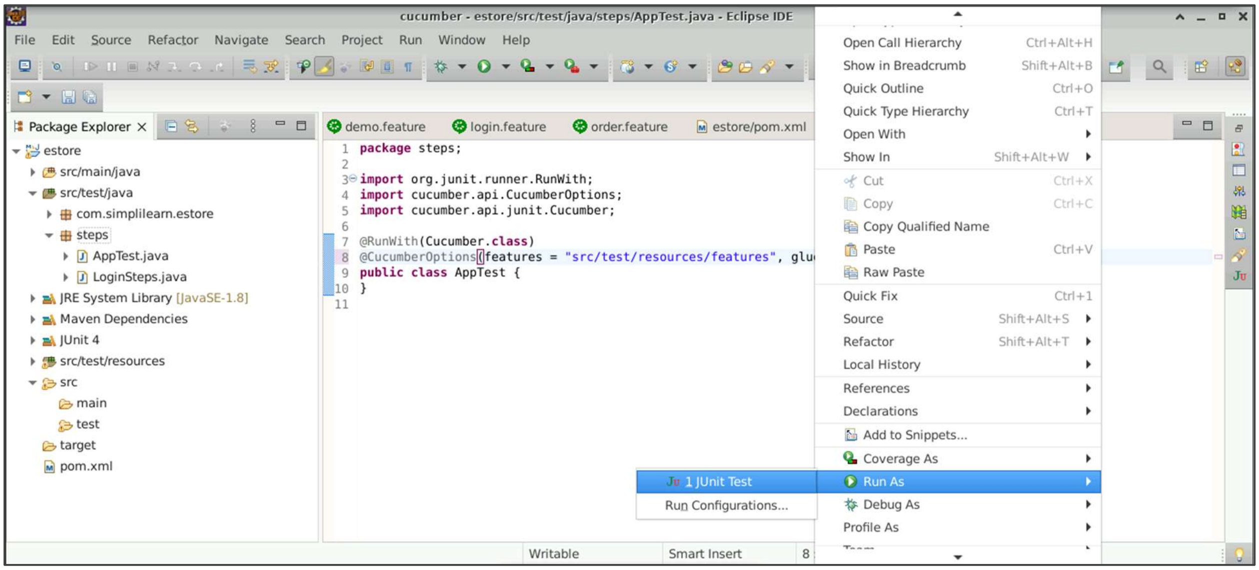
2.5 Specify another attribute as **glue**



```
package steps;

import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(features = "src/test/resources/features", glue= {"steps"})
public class AppTest {
}
```

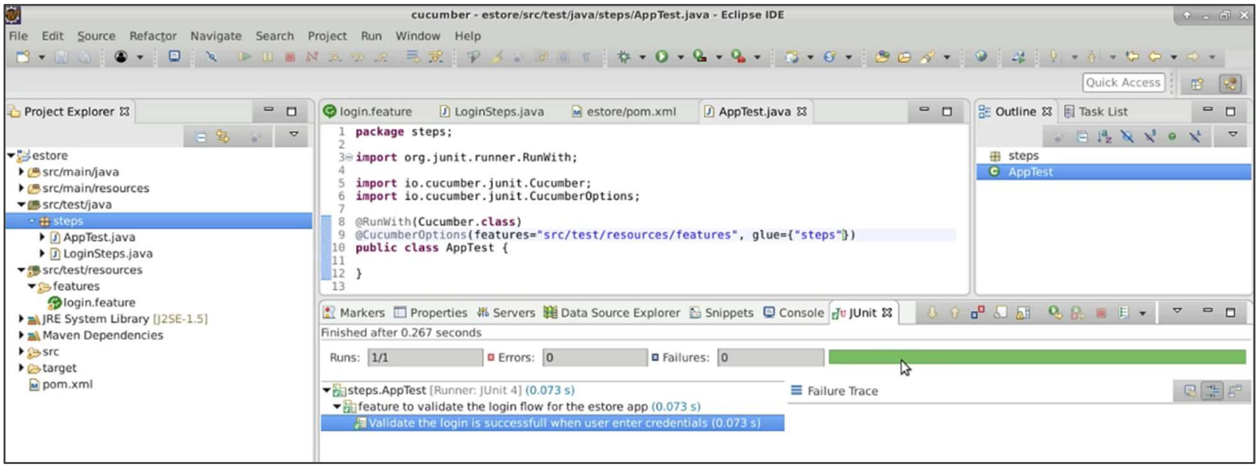
Step 3: Execute the configuration

3.1 Now, run this configuration, right-click and select **Run As**, and then click on **JUnit Test**



```
package steps;

import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(features = "src/test/resources/features", glu
public class AppTest {
}
```

3.2 You can see the green color. This shows that the execution is running properly.



```
package steps;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
@CucumberOptions(features="src/test/resources/features", glue={"steps"})
public class AppTest {

}
```

```
Finished after 0.267 seconds
Runs: 1/1    Errors: 0    Failures: 0

steps.AppTest [Runner: JUnit 4] (0.073 s)
  feature to validate the login flow for the estore app (0.073 s)
    Validate the login is successfull when user enter credentials (0.073 s)
```

You have successfully executed the test cases with Cucumber by following these steps.