Automation Testing

©Simplilearn. All rights reserved.

Introduction to Cucumber

©Simplilearn. All rights reserved.

A Day in the Life of an Automation Test Engineer

Sam is working in an organization as an Automation Testing Engineer.

He has been asked to use Cucumber to run the test cases. To learn more about it, he has to first set up an environment to work with Cucumber.

To achieve the above with some additional features, he will learn the basics of Cucumber, including BDD, Gherkins, and key features that will help him find the solution for the given scenario.

©Simplilearn. All rights reserved.

Learning Objectives

By the end of this lesson, you will be able to:

- Comprehend Cucumber and its uses
- Describe Gherkins and its use
- Identify Step Definition
- Illustrate Cucumber installation

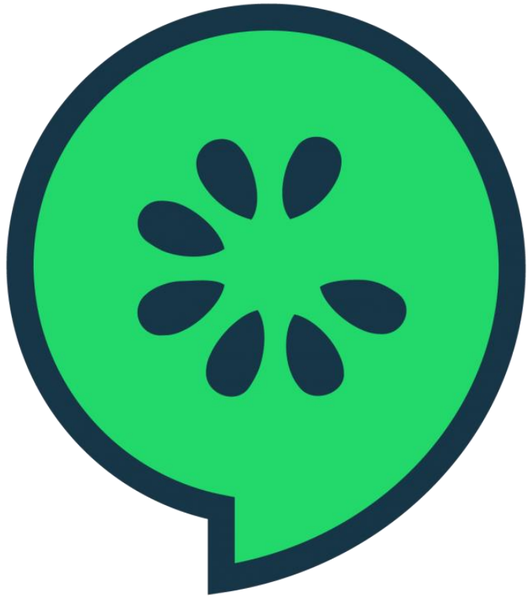
©Simplilearn. All rights reserved.

What Is Cucumber?

©Simplilearn. All rights reserved.

Introduction to Cucumber

Cucumber is a testing tool that supports Business Driven Development (BDD) and it provides a way to write tests that anyone, regardless of tech expertise can understand.



Before developers write their code, users write scenarios or acceptance tests that describe the system's behavior from the customer's perspective for review and approval by the product owners.

©Simplilearn. All rights reserved.

Why Cucumber?

These are the reasons why users should use the Cucumber framework:

- It supports different languages like Java.net and Ruby.
- It acts as a bridge between business and technical language.
- It serves the purpose of an end-to-end test framework, unlike other tools.
- It provides code reusability.

©Simplilearn. All rights reserved.

How Does Cucumber Work?

©Simplilearn. All rights reserved.

What is Gherkin?

©Simplilearn. All rights reserved.

Introduction to Gherkin

Gherkin is a business readable language that allows you to describe business behavior without getting entangled in the implementation details.

©Simplilearn. All rights reserved.

Why Gherkin?

These are the reasons why users should use the Gherkin language:

- It is easy to understand even for non-programmers.
- Programmers can use it as a solid starting point for their tests.
- The business requirements are the focus of Gherkin Testing.
- Gherkin Test cases link acceptance tests directly to automated tests.

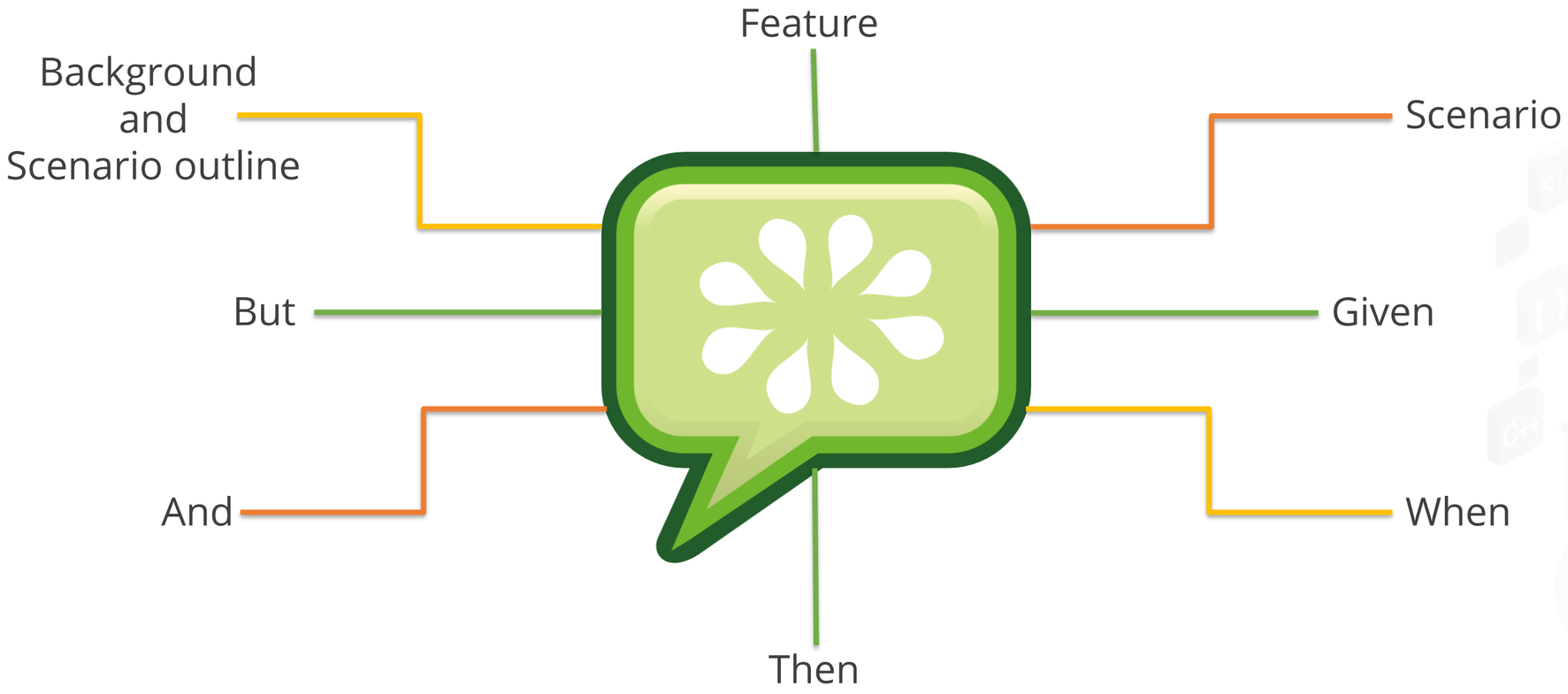
©Simplilearn. All rights reserved.

Gherkin Syntax


```
Feature: Title of the Scenario
Given [Preconditions or Initial
        Context]
When [Event or Trigger]
Then [Expected output]
```

©Simplilearn. All rights reserved.

Gherkin Keywords



©Simplilearn. All rights reserved.

Gherkin Keywords

Feature and **Scenario**

A **feature** is a self-contained unit or functionality that needs to be tested.

Each feature has the required number of tests to ensure that the feature is fully functional. A **scenario** is a name given to each test.

©Simplilearn. All rights reserved.

Gherkin Keywords

Given and **When**

Given keyword refers to the pre-condition of the test.

When usually refers to the actions of a user that is to be executed.

©Simplilearn. All rights reserved.

Gherkin Keywords

Then and **And**

Then keyword refers to the outcome of the previous step or upcoming action.

And keyword is used to add more conditions to your steps.

©Simplilearn. All rights reserved.

Gherkin Keywords

Example

```
Feature: Login
Scenario: Login verification
Given user navigates to the website simplilearn.com
And user logs in through Login Window by using Username as "USER" and Password as "PASSWORD"
Then login must be successful.
```

```
Given User is on Home Page
And Login Link is displayed
When User Navigates to Login form
And User enters email and Password
Then Login Successfully will be displayed
And Logout Link should be displayed
```

©Simplilearn. All rights reserved.

Gherkin Keywords

But

But keyword is used to add the negative conditions.

```
Scenario: Unsuccessful Login with Invalid entries
    Given user navigates to the website simplilearn.com
    And user logs in through Login Window by using Username as "USER" and Password as "1234erty"
    But user entered wrong password
    Then login must be unsuccessful.
```

©Simplilearn. All rights reserved.

Gherkin Keywords

Background

The steps that are common to all the tests in the feature file are defined by the **Background** keyword.

Feature file without Background

```
Feature: Registration, Login and MyAccount

  @smoke
  Scenario: Verify Login Functionality
    Given I am on the homepage
    And I follow "Sign in"
    When  I fill "email address textbox" with "goswami.tarun77@gmail.com"
    Then I fill "password textbox" with "Test1234"

  @smoke
  Scenario: Create New User
    Given I am on the homepage
    When I follow "Sign in"
    When I fill "registration email textbox" with "goswami.tarun77+1@gmail.com"
    Then I click "create an account button"
```

Feature file with Background

```
Feature: Registration, Login and MyAccount

  Background:
    Given I am on the homepage
    And I follow "Sign in"

  @smoke
  Scenario: Verify Login Functionality
    When I fill "email address textbox" with "goswami.tarun77@gmail.com"
    Then I fill "password textbox" with "Test1234"

  @smoke
  Scenario: Create New User
    When I fill "registration email textbox" with "goswami.tarun77+1@gmail.com"
    Then I click "create an account button"
```

©Simplilearn. All rights reserved.

simplilearn

Gherkin Keywords

Scenario Outline

The **scenario outline** basically replaces the variables/keywords with the value from the table. Each row in the table is considered to be a scenario.

```
Feature: Calculator As a user I want to use a calculator to
add numbers So that I don't need to add myself Scenario
Outline:
Add two numbers <num1> & <num2>
Given I have a calculator
When I add <num1> and <num2>
Then the result should be <total>
Examples: | num1 | num2 | total |
          | -2   |  3   | 1     |
          | 10   | 15   | 25    |
          | 99   | -99  | 0     |
          |-1    | -10  | -11   |
```

©Simplilearn. All rights reserved.

simplilearn

Basic Terms of Cucumber

These are the basic terms used in the Cucumber framework:

- Feature Files
- Feature

- Scenario
- Tags

- Gherkin Language
- Step Definition

©Simplilearn. All rights reserved.

Basic Terms of Cucumber

Feature Files

- A feature file is a shared file that contains the features, scenarios, and feature descriptions that will be tested.
- The extension of the feature file is."**feature**."

©Simplilearn. All rights reserved.

Basic Terms of Cucumber

Tags

Tags are used to organize features and scenarios.

Purpose of Tags:

- Using tags in Cucumber allows us to create reports for scenarios under the same tag when we have many scenarios in the feature file and want to keep them all in one group.
- Cucumber runs every scenario inside the feature file by default, but if we need to run or skip any scenario inside a certain test, we can declare scenarios inside a tag.

©Simplilearn. All rights reserved.

Basic Terms of Cucumber

Tags syntax

Here is the syntax for single and multiple tests:

Single test

```
@TestName
Scenario: Mention the Scenario
```

Multiple test

```
@TestName@TestName
Scenario: Mention the scenario
```

Where,
@ is the symbol to declare a tag
TestName is the name of a specific test
Scenario is scenario

©Simplilearn. All rights reserved.

Basic Terms of Cucumber

Tags Example

```
  @ValidCredentials
  Scenario: Login with valid credentials

   Given User is on Home page
   When User enters username as "Admin"
   And User enters password as "admin123"
   Then User should be able to login successfully

  @InValidCredentials
  Scenario: Login with invalid credentials

   Given User is on Home page
   When User enters username as "username"
   And User enters password as "password"
   Then Login will be unsuccessful with error message "Invalid
credentials"
```

Note: To declare only those scenarios which are declared under @ValidCredentials, we will write: **tags={“@ValidCredentials”}**

©Simplilearn. All rights reserved.

Basic Terms of Cucumber

Tags

The testing through multiple tags can be done by using two operators:

OR

The next test should be examined if the first test for the application was unsuccessful. If the subsequent test also fails, a subsequent test should be examined, and so on.

AND

When testing an application like this, we should stop testing if the application fails the first test and continue testing only if the first test was successful.

©Simplilearn. All rights reserved.

simplilearn

Basic Terms of Cucumber

Tags

Syntax of OR and AND:

OR

```
Tags= {"@FirstTest,@SecondTest"}
```

AND

```
tags= {"@FirstTest", "@SecondTest"}
```

©Simplilearn. All rights reserved.

simplilearn

Basic Terms of Cucumber

Tags

Example of **OR**:

Debug | Project Explorer | JUnit

Finished after 16.384 seconds

Runs: 2/2 Errors: 0 Failures: 0

com.cucumber.demo.runner.CucumberTestRunner [Runner:
- Sample use of Tags in Cucumber (16.311 s)
  - Login with valid credentials (9.854 s)
  - Login with Blank credentials (6.457 s)

CucumberTestRunner.java | CucumberTagsExample.feature

```
1  package com.cucumber.demo.runner;
2
3  import org.junit.runner.RunWith;
4
5  import io.cucumber.junit.Cucumber;
6  import io.cucumber.junit.CucumberOptions;
7
8  @RunWith(Cucumber.class)
9
10 @CucumberOptions(features = { "src/test/resources/features/CucumberTagsExample.feature" }, glue = {
11         "com.cucumber.demo.definitions" }, tags = ("@BlankCredentials or @ValidCredentials"))
12
13 public class CucumberTestRunner {
14
15 }
16
```

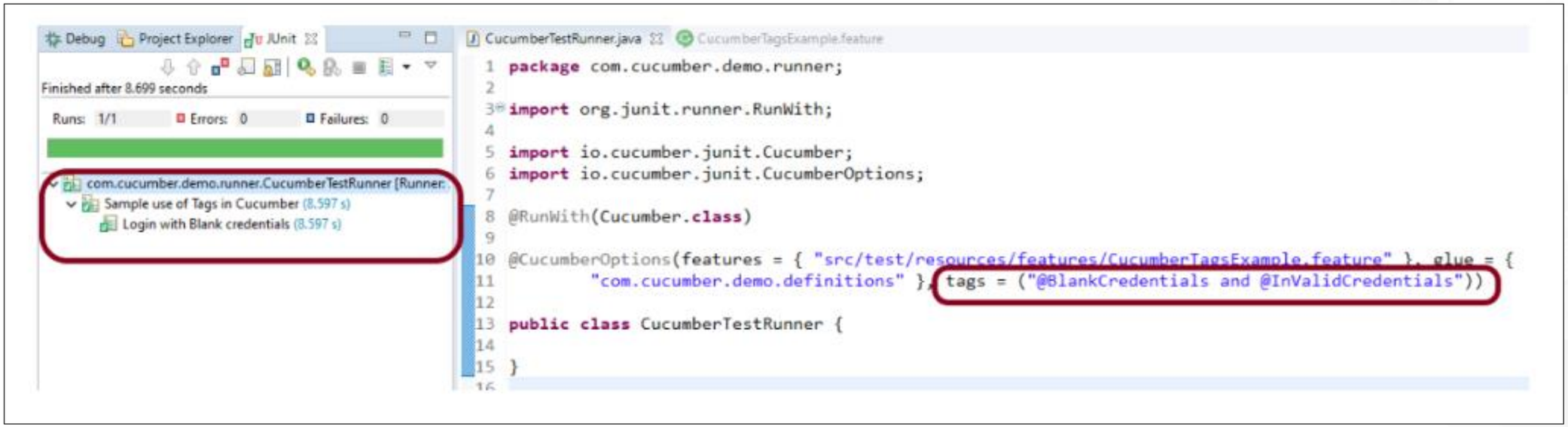
©Simplilearn. All rights reserved.

simplilearn

Basic Terms of Cucumber

Tags

Example of **AND**:



©Simplilearn. All rights reserved.

simplilearn

What is Hook?

©Simplilearn. All rights reserved.

Hook

The hook makes it easier for users to manage the workflow of the code and reduces code duplication.

The **hook is the block of code** that can be defined with each scenario in the step definition file by using the annotation **@Before** and **@After**.

©Simplilearn. All rights reserved.

Hook

The syntax of hook:

```
@Before setup ()
{
logic

} @

Scenario
Given
When
And
Then

@After cleanup (){
logic
}
```

©Simplilearn. All rights reserved.

simplilearn

Why Hook?

During testing, there may be situations where we must take a few conventional preparatory actions before running the test scenario:

To understand the types of prerequisites that may be encountered during testing, consider the following prerequisites:

- To start a webdriver
- Setup database connections
- Set up a test data
- Set up browser cookies
- Navigation to a certain page

©Simplilearn. All rights reserved.

Why Hook?

During testing, there may be situations where we must take a few conventional preparatory actions before running the test scenario:

Similar to that, there are always some steps that must be taken after testing:

- To stop the web driver
- To close the DB connections
- To clear the test data
- To clear the browser cookies
- To log out from the application
- Printing reports or logs
- Taking the screenshots of error

©Simplilearn. All rights reserved.

Hook

Hook annotations:

The cucumber only accepts two hooks, in contrast to the TestNG annotations:

•**@Before**
As the name suggests, we can use the **@Before** hook with the function/method after which we need to start the web driver.

•**@After**
As the name suggests, we can use the **@After** hook with the function/method after which we need to close the webdriver.

©Simplilearn. All rights reserved.

simplilearn

Hook

Hook Example:

TestRunner.java | Hooks.feature | Hooks_Steps.java | Hooks.java | LogIn_Test.feature

```
1  Feature: Test Hooks
2
3  Scenario Outline: This scenario is to test hooks functionality
4      Given this is the first step
5      When this is the second step
6      Then this is the third step
7  Examples:
8      |Scenario|
9      |First|
10     |Second|
```

Console

```
<terminated> Hooks.feature [Cucumber Feature] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Oct 3, 2017, 9:56:39 PM)
Feature: Test Hooks

  Scenario Outline: This scenario is to test hooks functionality # C:/ToolsQA/OnlineStore/Feature/Hooks.feature:3
    Given this is the first step
    When this is the second step
    Then this is the third step

    Examples:
This will run before the Scenario
This is the first step
This is the second step
This is the third step
This will run after the Scenario
This will run before the Scenario

  Scenario Outline: This scenario is to test hooks functionality # C:/ToolsQA/OnlineStore/Feature/Hooks.feature:9
    Given this is the first step                                 # Hooks_Steps.This_Is_The_First_Step()
    When this is the second step                                 # Hooks_Steps.This_Is_The_Second_Step()
    Then this is the third step                                  # Hooks_Steps.This_Is_The_Third_Step()
This is the first step
This is the second step
This is the third step
This will run after the Scenario

  Scenario Outline: This scenario is to test hooks functionality # C:/ToolsQA/OnlineStore/Feature/Hooks.feature:10
    Given this is the first step                                 # Hooks_Steps.This_Is_The_First_Step()
    When this is the second step                                 # Hooks_Steps.This_Is_The_Second_Step()
    Then this is the third step                                  # Hooks_Steps.This_Is_The_Third_Step()

2 Scenarios (2 passed)
6 Steps (6 passed)
0m0.121s
```

©Simplilearn. All rights reserved.

simplilearn

Hook

Tagged Hooks:

The hook can also be used with Tag.
We can use it before and after with a specific test.

Syntax:

Before

```
@Before ('@RegressionTest’)
```

After

```
@After('@RegressionTest,
@SmokeTest’)
```


©Simplilearn. All rights reserved.

simplilearn

What are Step Definitions?

©Simplilearn. All rights reserved.

Introduction to Step Definition

Step definition maps the test case steps in the feature files (introduced by Given/When/Then) to the code.

Developer Tools - Node.js

Connection | Console | Profiler | Sources | Memory

nightwatch | nightwatch-api.js | steps.js ×

```
1  (function (exports, require, module, __filename, __dirname) { const { client
2  const { defineSupportCode } = require('cucumber');
3
4  defineSupportCode(({ Given, Then, When }) => {
5    Given(/^I open Google`s search page$/, () => {
6      return client
7        .url('http://google.com');
8    });
9
10   Given(/^I open DuckDuckGo search page$/, () => {
11     return client
12       .url('https://duckduckgo.com/');
13   });
14
15   Then(/^the title is "(.*?)"$/, (text) => {  text = "DuckDuckGo"
16     debugger;
17     return client.assert.title(text);
18   });
19
20   Then(/^the Google search form exists$/, () => {
21     return client.assert.visible('input[name="q"]');
22   });
23
24   Then(/^the DuckDuckGo search form exists$/, () => {
25     return client.assert.visible('input[name="q"]');
26   });
27 });
28
29 });
```

{} Line 16, Column 5

Debugger paused

Threads	
Node.js Main Context	paused

Watch

Call Stack	
Then	steps.js:16
(anonymous)	user_code_runner.js:20
tryCatcher	util.js:16
PromiseSpawn._promiseFulfilled	generators.js:97
(anonymous)	generators.js:201
run	user_code_runner.js:120
Cucumber.UserCodeRunner.run	cucumber-api.js:71
(anonymous)	step_runner.js:39
tryCatcher	util.js:16
PromiseSpawn._promiseFulfilled	generators.js:97
Async._drainQueue	async.js:138
Async._drainQueues	async.js:143
Async.drainQueues	async.js:17

Console | Search

Node.js Main C... | Filter | Default levels

>

©Simplilearn. All rights reserved.

Introduction to Step Definition

This is an example of step definition in cucumber:

```
Step 1:

Given (/^ I am on simplilearn.com$/) do

Browser.goto "http://simplilearn.com" -This will visit simplilearn.com on browser

end

Step 2:
When (/^ click on course menu$/) do

Browser.text (:name, " course" ).click – This will click "course menu"

end

Step 3:

Then (/^ I should see course page$/) do

Browser.goto "http://simplilearn.com/category/course/" - It will visit "course
page"

end
```

©Simplilearn. All rights reserved.

What are Comments ?

©Simplilearn. All rights reserved.

Comments

A comment is essentially a chunk of code intended just for documentation and not for execution.

Simply place the "#" symbol at the beginning of the statement to include comments.

©Simplilearn. All rights reserved.

Comments

This is an example of comments in cucumber:

```
Feature: annotation

#This is how background can be used to eliminate duplicate steps
Background:
User navigates to Facebook
Given I am on Facebook login page

#Scenario with AND
Scenario:
When I enter username as "TOM"
And I enter password as "JERRY"
Then Login should fail
```

©Simplilearn. All rights reserved.

Why Comments?

These are the reasons why users should use comments in cucumber:

- To create files that are simple to read and understand, whether they are a feature file or a step definition file.
- They also help while debugging the code.

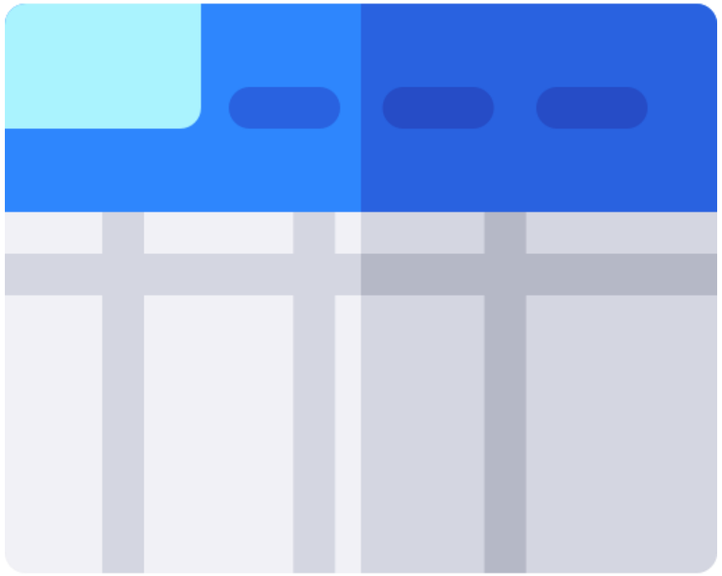
©Simplilearn. All rights reserved.

What is a Data Table ?

©Simplilearn. All rights reserved.

Data Table

A data table is used when users need to test numerous input parameters.



©Simplilearn. All rights reserved.

simplilearn

Data Table vs Scenario Outline

These are the few differences between a data table and a scenario outline:

Scenario Outline

- It uses Example keyword to define the test data for the Scenario.
- It works for the whole test.
- Cucumber automatically runs the complete test the number of times equal to the number of data in the test set.

Data Table

- No keyword is used to define the test data.
- It works only for the single step, under which it is defined.
- The test data must first be understood by a separate piece of code before it runs once or many times, but just for single step not for complete test.

©Simplilearn. All rights reserved.

simplilearn

Data Table

This is an example of a data table in cucumber:

```
Given the user on the user registration page.
When user enter invalid data on the page
| Fields|| Values|
| First Name            | User Name           |
| Last Name             | User Last Name      |
| Email Address         | someone@gmail.com    |
| Re-enter Email Address | someone@gmail.com   |
| Password              |PASSWORD|
| Birth-date              | 02|
Then the user registration should be successful.
```

©Simplilearn. All rights reserved.

How to Create a Step Definition?

These are the steps to create a step definition:

Step 01:

Click on the File menu in Eclipse > Then select the option New > Next click on Other

Step 02:

Click on Maven Project from the Maven folder > Then click on Next

Step 03:

Proceed with the further steps

Step 04:

Select maven-achetype-quickstart template > Then click on Next

©Simplilearn. All rights reserved.

How to Create a Step Definition?

These are the steps to create a step definition:

Step 05:

Add GroupId as Automation, Artifact Id as Cucumber, and proceed

Step 06:

The scripts relevant to cucumber should be written in the src/test/java folder, and the project should be built with a structure similar to a Cucumber project.

Step 07:

Create a new package called stepDefinitions inside the src/test/java folder

©Simplilearn. All rights reserved.

How to Create a Step Definition?

These are the steps to create a step definition:

Step 08:

Create a Java class file within the stepDefinations package > Right-click the **stepDefinations package** > then select option **New->Class**

Step 09:

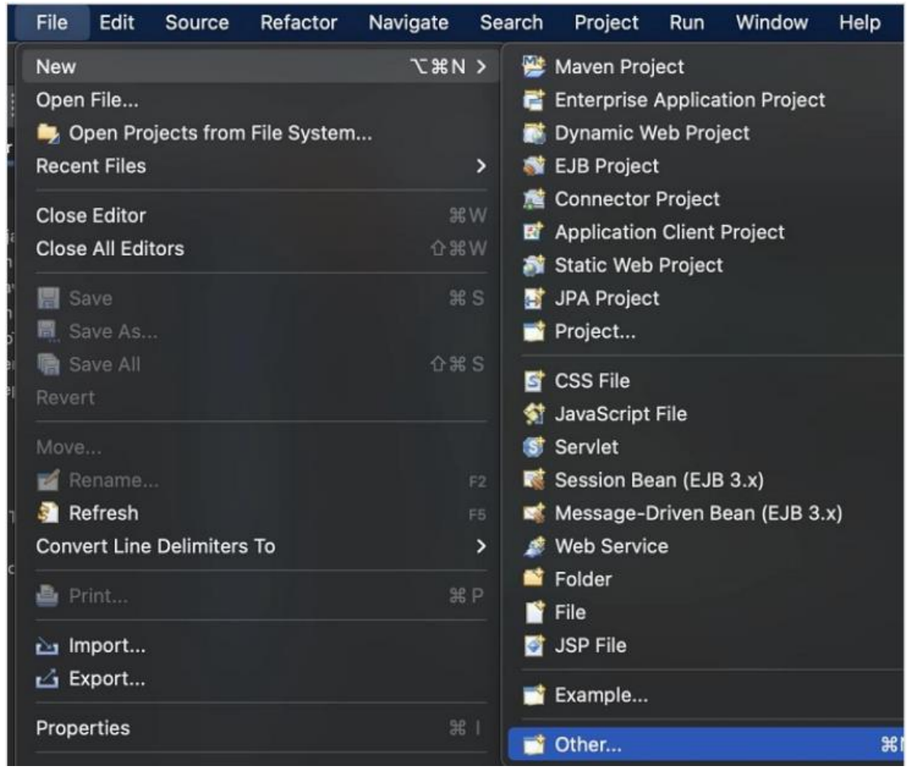
Give a Java class file name, say stepDefination > Then click on **Finish**

©Simplilearn. All rights reserved.

How to Create a Step Definition?

Step 01:

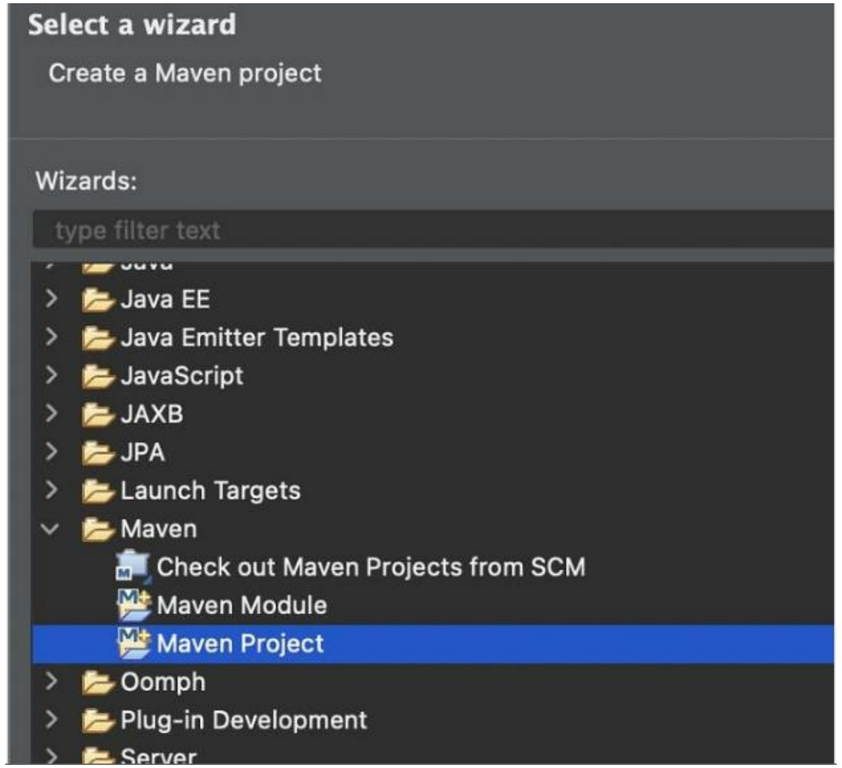
Click on the **File menu** in Eclipse > Then select the option **New** > Next click on **Other**



©Simplilearn. All rights reserved.

How to Create a Step Definition?

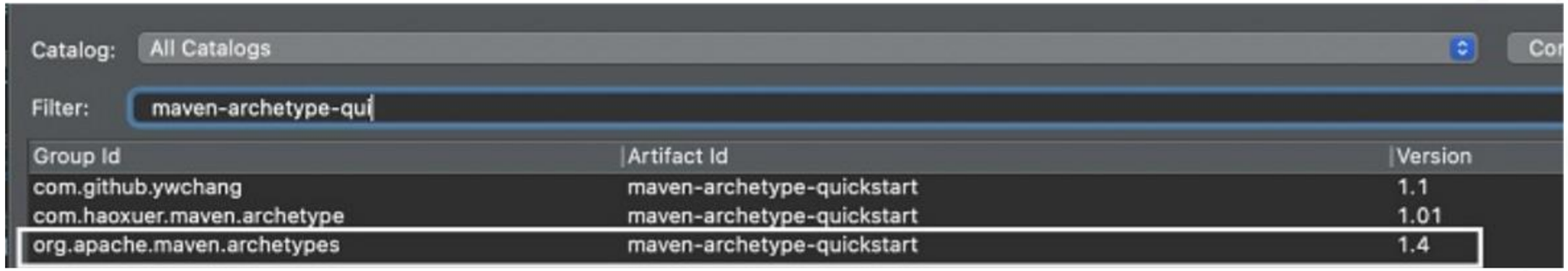
Step 02:
Click on **Maven Project** from the Maven folder > Then click on **Next**



©Simplilearn. All rights reserved.

How to Create a Step Definition?

Step 03:
Proceed with the further steps



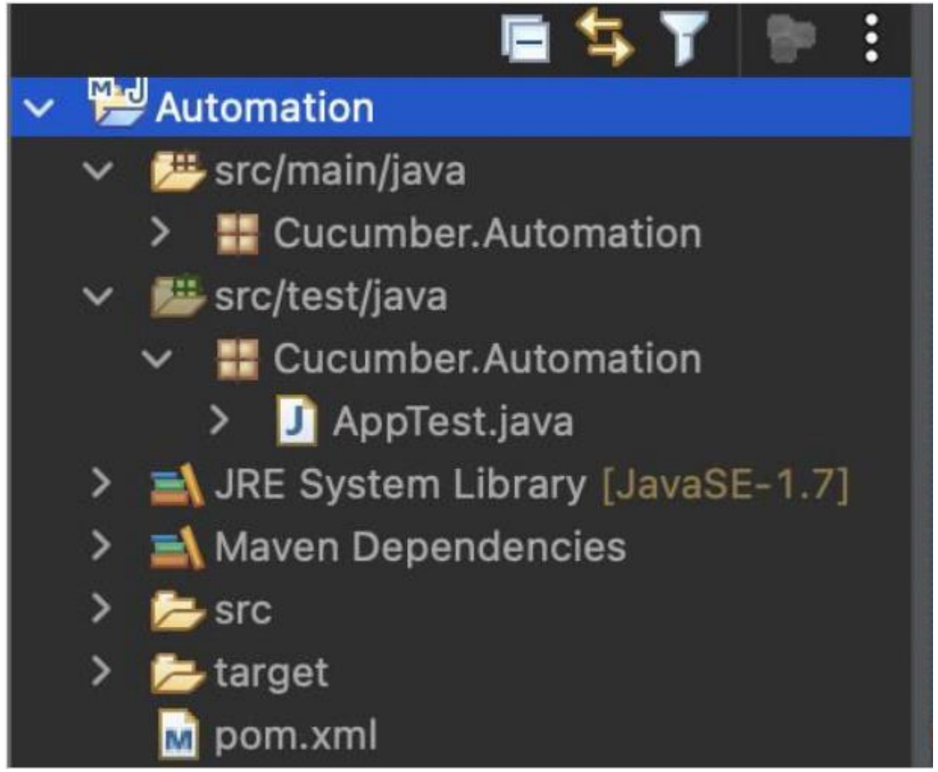
Step 04:
Select the **maven-achetype-quickstart** template > Then click on **Next**

©Simplilearn. All rights reserved.

How to Create a Step Definition?

Step 05:

Add GroupId as Automation, Artifact Id as Cucumber, and proceed



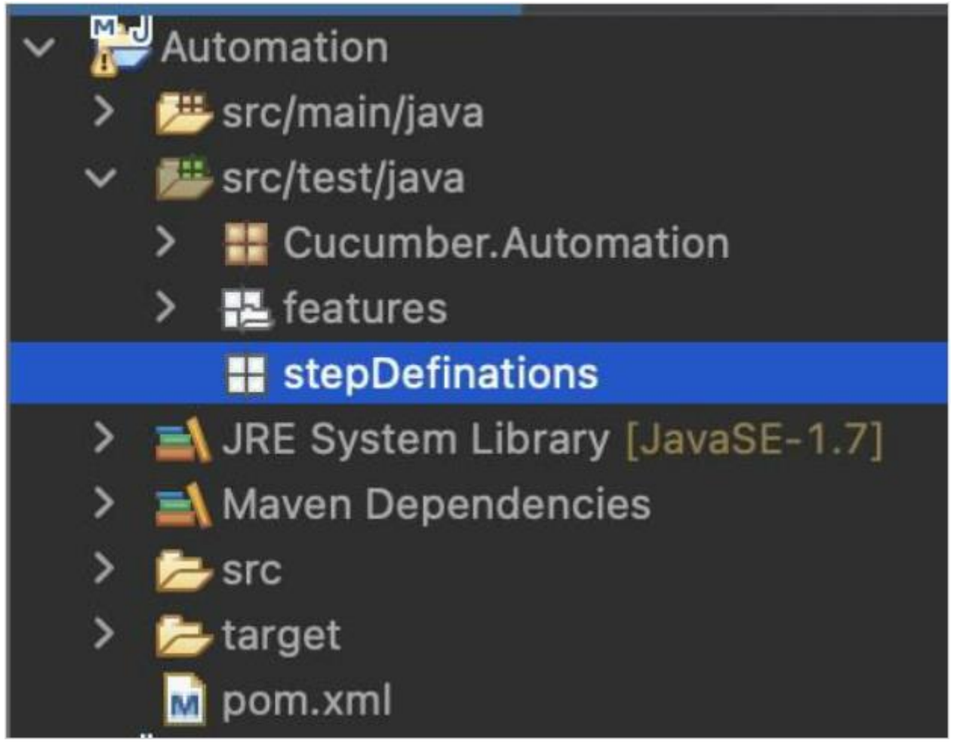
Step 06:

A project should get created with a Cucumber-type project structure. The cucumber-related scripts should be written within the src/test/java folder.

©Simplilearn. All rights reserved.

How to Create a Step Definition?

Step 07:
Create a new package called **stepDefinitions** inside the src/test/java folder

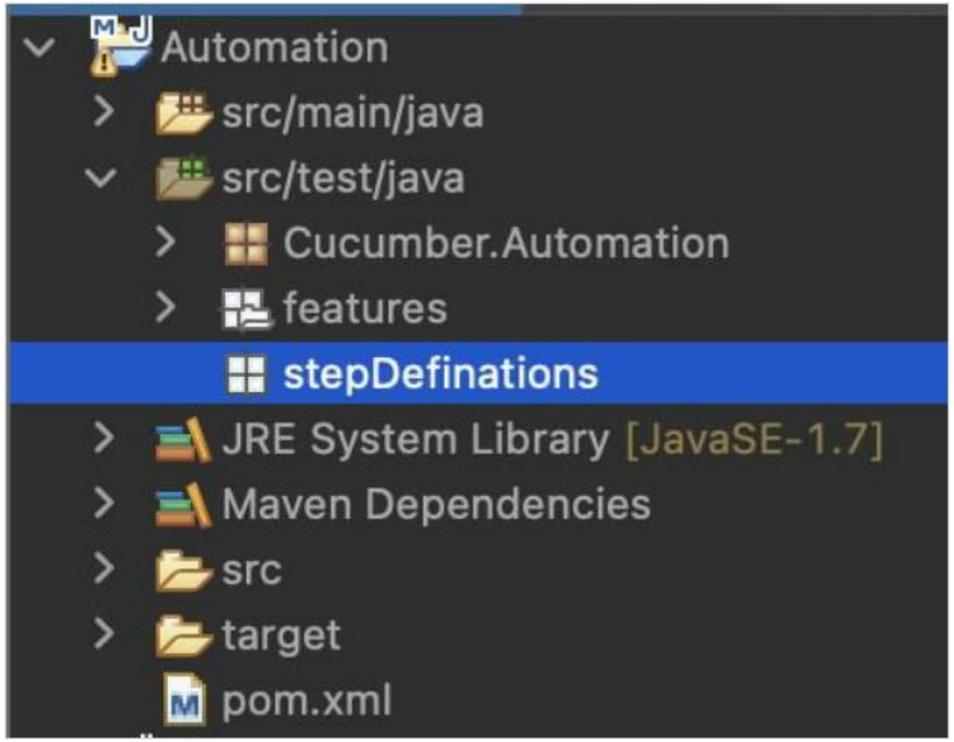


©Simplilearn. All rights reserved.

How to Create a Step Definition?

Step 08:

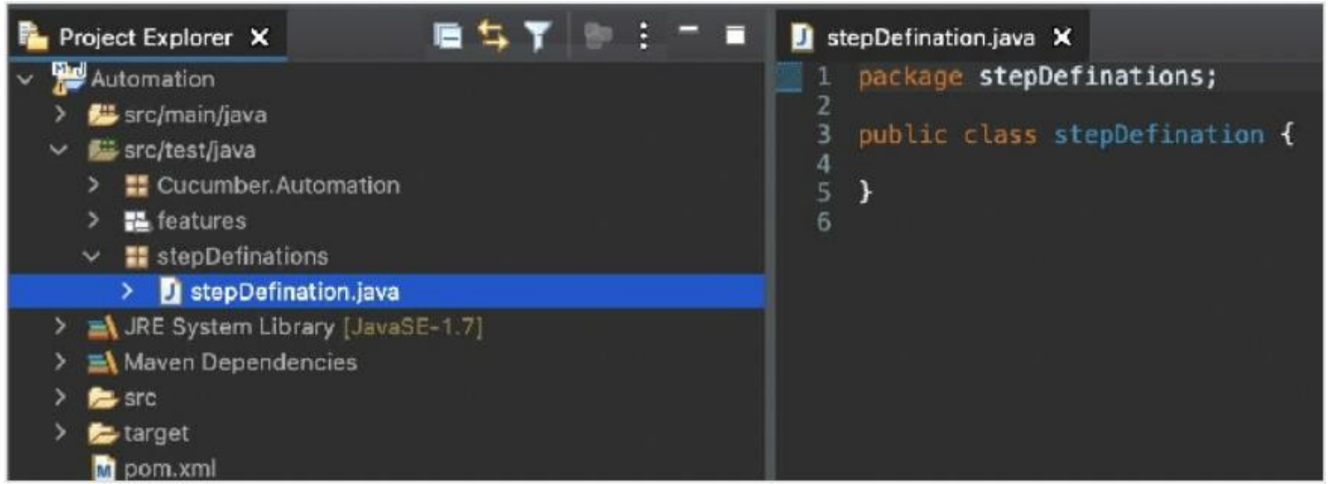
Create a Java class file within the **stepDefinitions package** > Right-click the **stepDefinations package** > Then select the option **New->Class**



©Simplilearn. All rights reserved.

How to Create a Step Definition?

Step 09:
Give a Java class file name, say stepDefinition > Then click on **Finish**



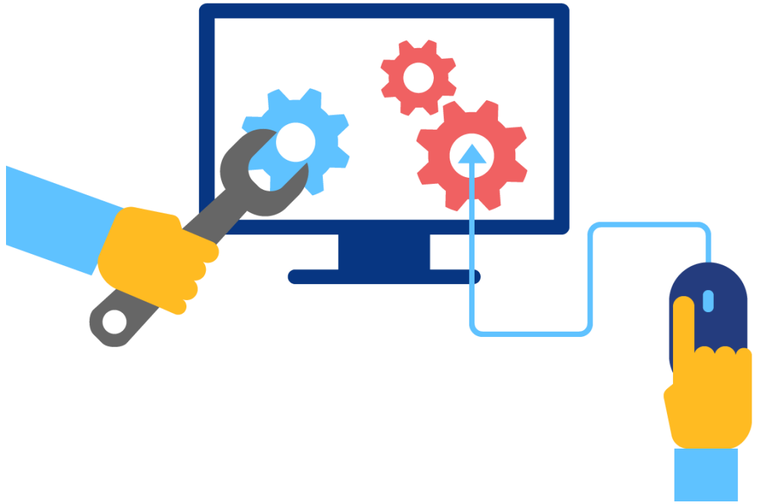
©Simplilearn. All rights reserved.

Development Environment Setup

©Simplilearn. All rights reserved.

Development Environment Setup

These are the steps to set up a Cucumber development environment:

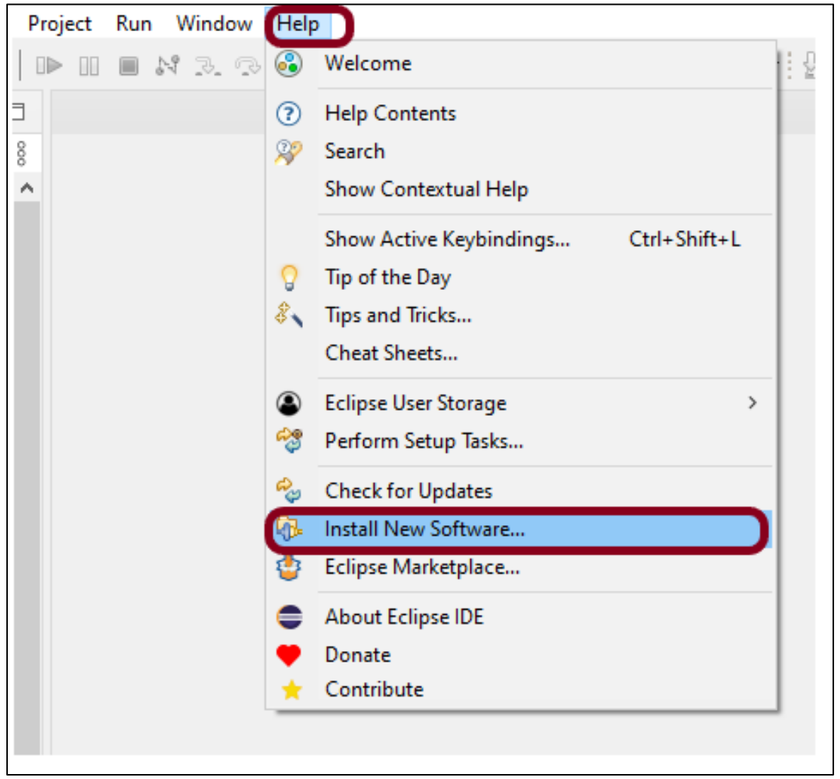


- Download and install Jdk and Jre
- Download and install Eclipse
- Download and install Maven
- Configure Cucumber with Maven
- Install Plugins
- Update dependencies

©Simplilearn. All rights reserved.

Development Environment Setup

These are the steps to install the Cucumber Plugins for Eclipse:



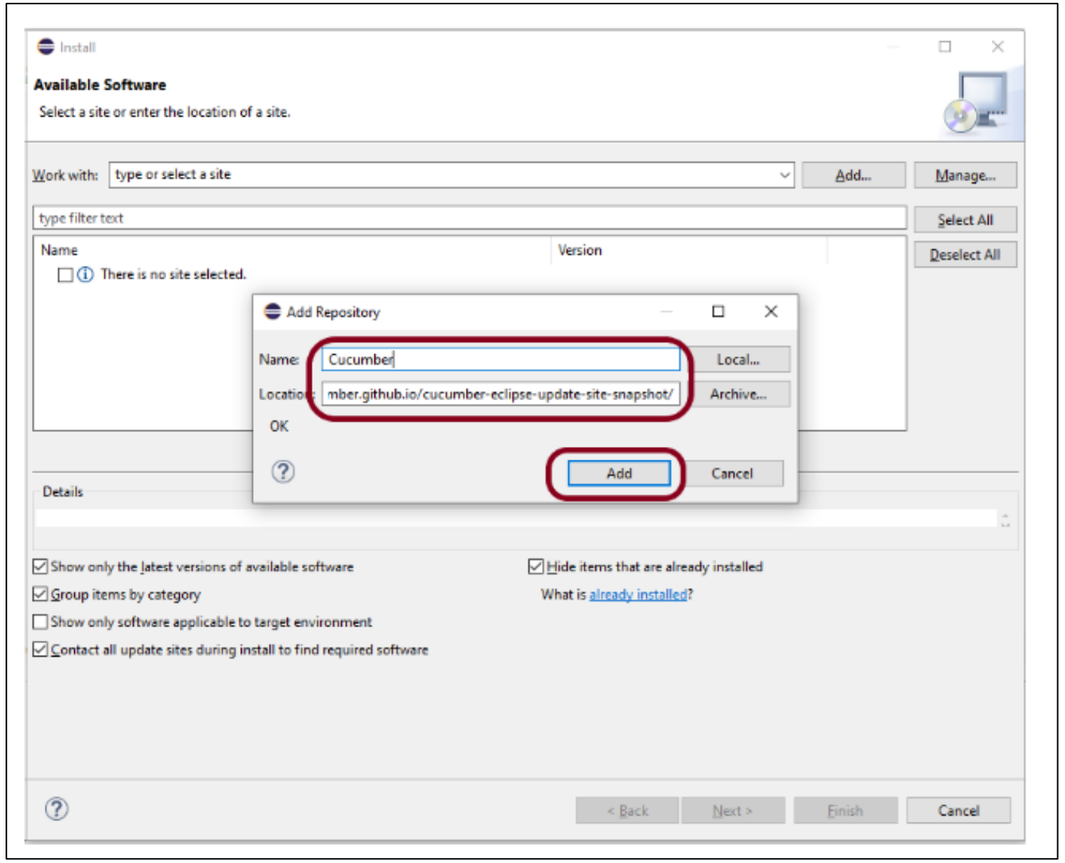
Step 1
- Launch Eclipse IDE

Step 2
- Click on Help

Step 3
- Then click “Install new software”

©Simplilearn. All rights reserved.

Development Environment Setup



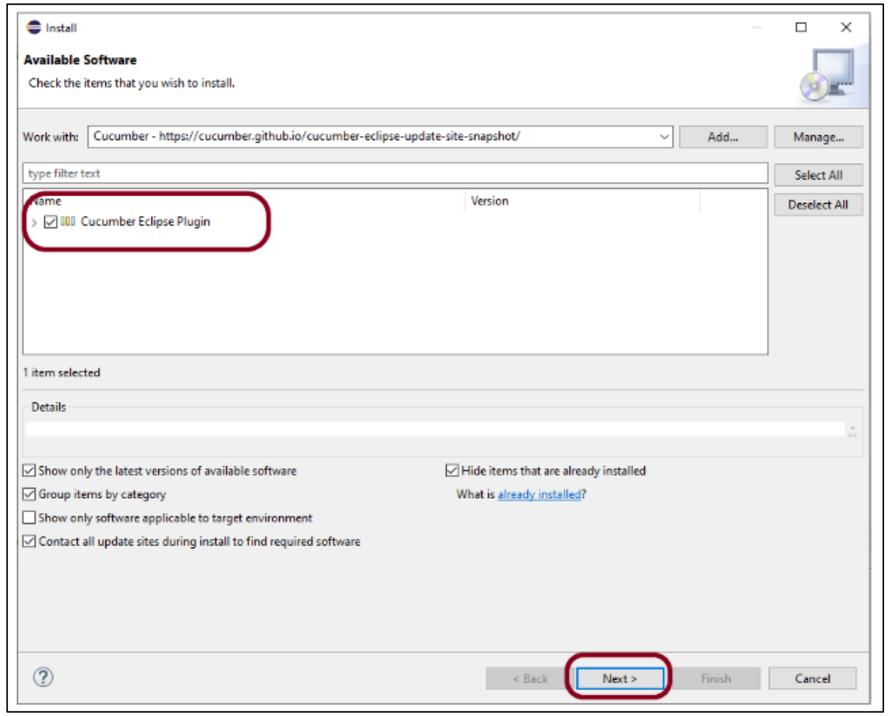
Step 4
- Click on Add

Step 5
- Write Cucumber in the name repository

Step 6
- Type this link “https://cucumber.github.io/cucumber-eclipse/update-site/main/.”

©Simplilearn. All rights reserved.

Development Environment Setup

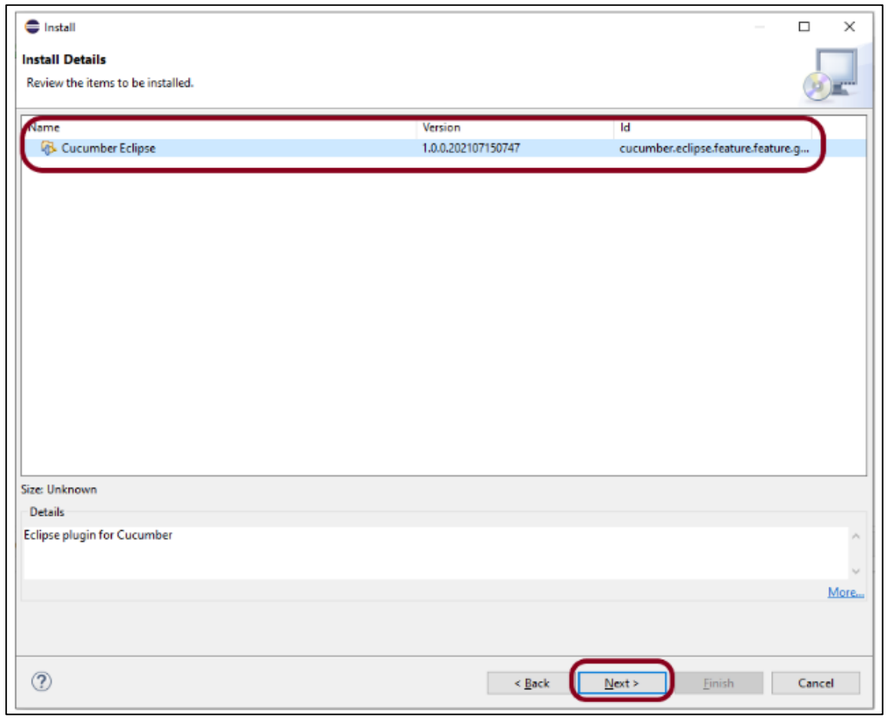


Step 7

- Click on Next

©Simplilearn. All rights reserved.

Development Environment Setup

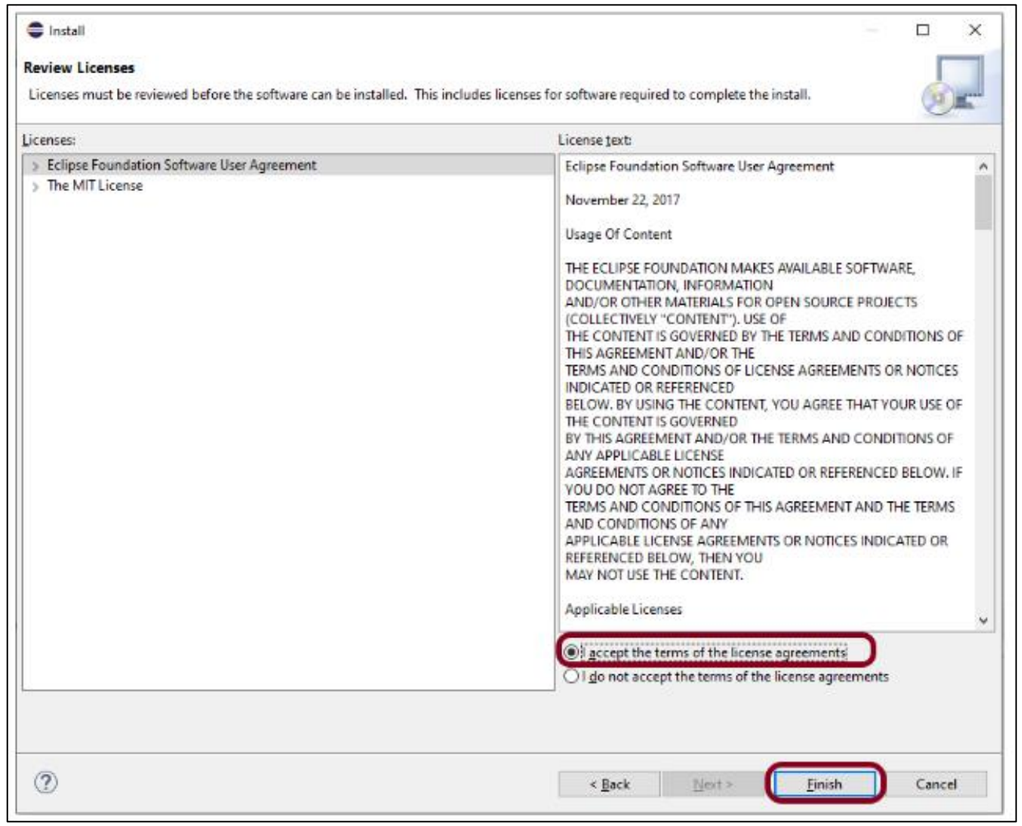


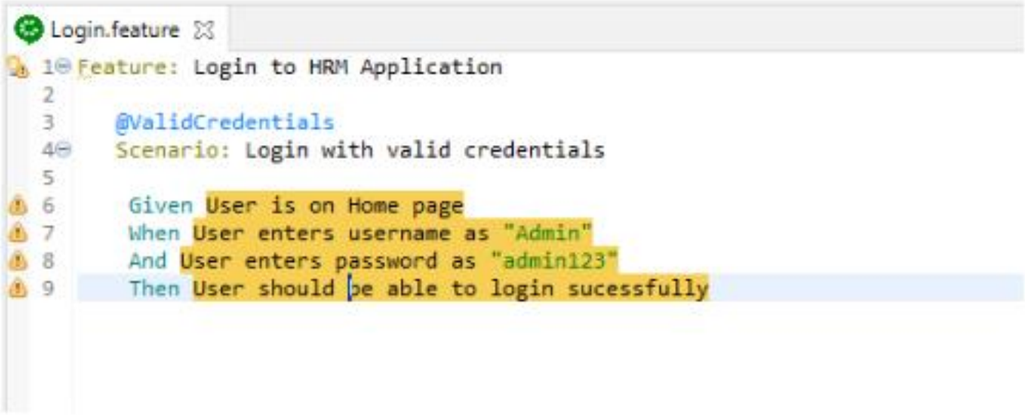
Step 8

- Click on Next

©Simplilearn. All rights reserved.

Development Environment Setup



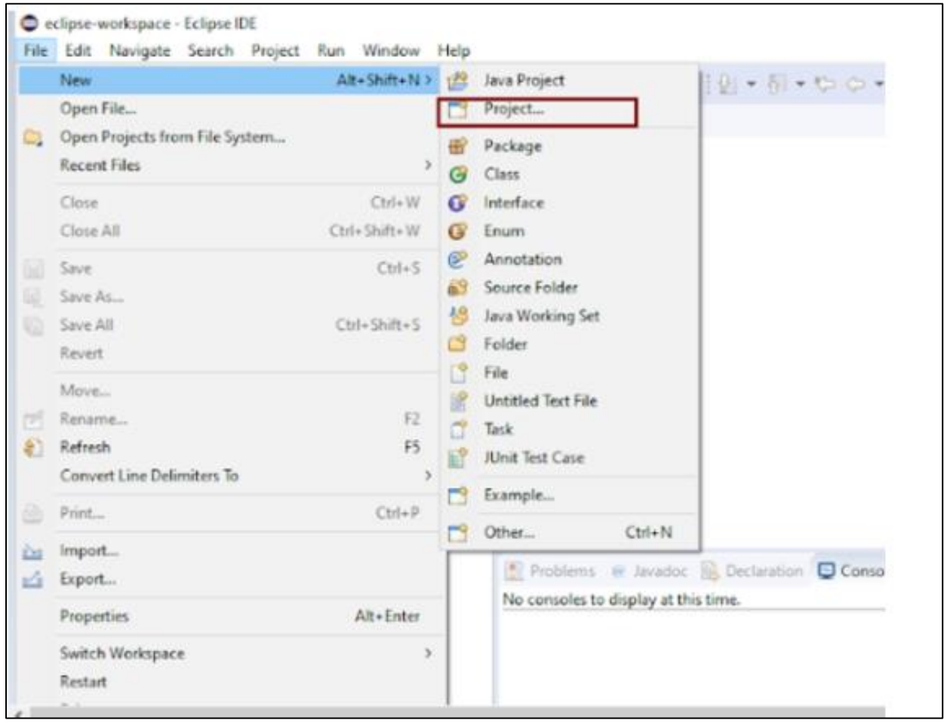


After the final step, this is how the interface will look.

©Simplilearn. All rights reserved.

Development Environment Setup

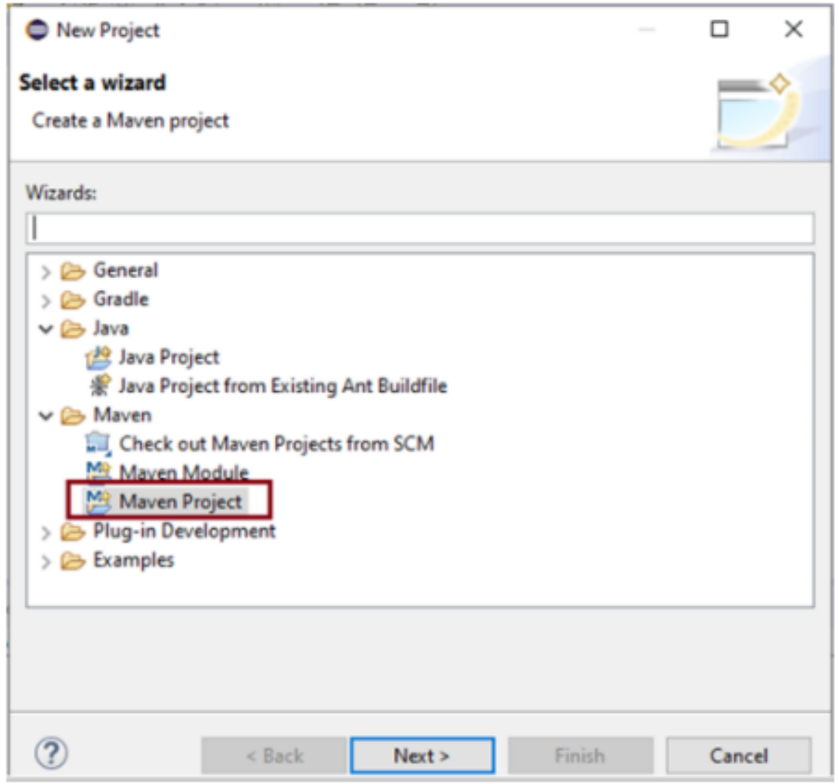
These are the steps to create a Maven Project in the Eclipse IDE:



Step 1:
Click on **New** and then select **Project.**

©Simplilearn. All rights reserved.

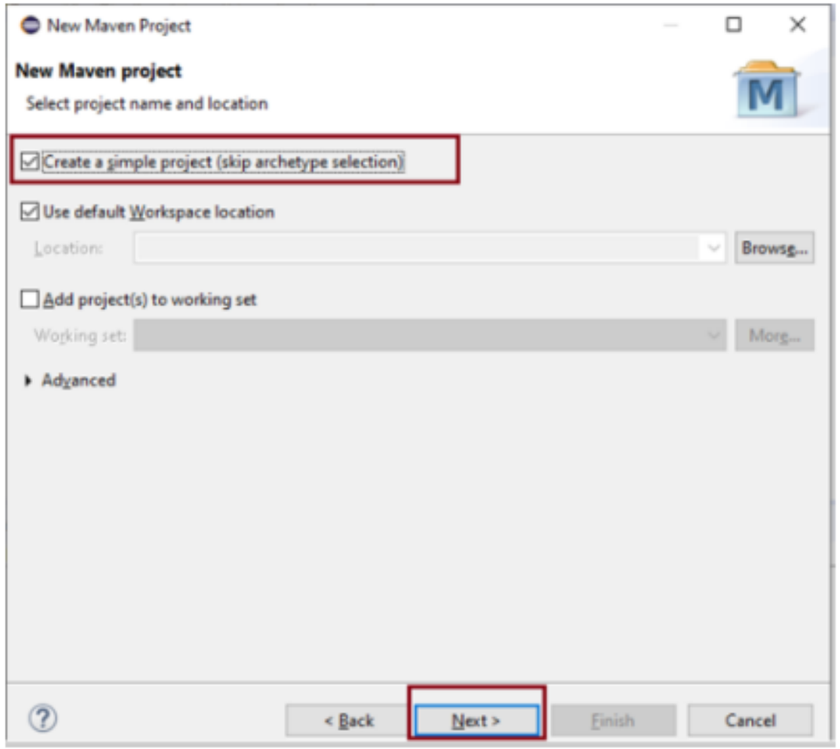
Development Environment Setup



Step 2:
Select **Maven project** and click on **Next.**

©Simplilearn. All rights reserved.

Development Environment Setup

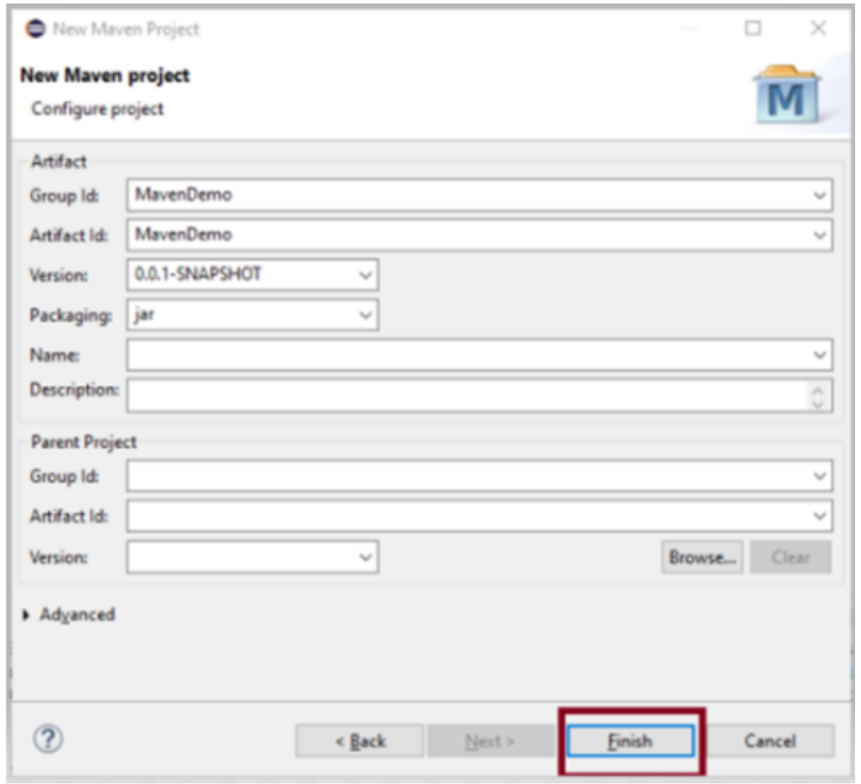


Step 3:

Select "**Create a simple project**", check box, and then click on **Next.**

©Simplilearn. All rights reserved.

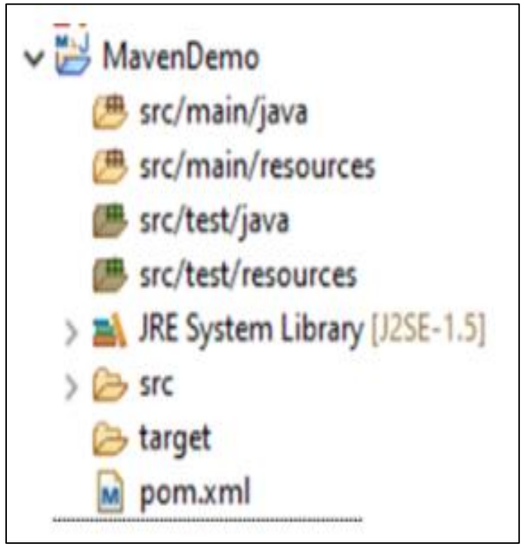
Development Environment Setup



Step 4:
Mention the **Group Id**, **Artifact Id,** and click the **Finish** button.

©Simplilearn. All rights reserved.

Development Environment Setup



Step 5:
The structure of the image looks as shown in the image.

©Simplilearn. All rights reserved.

Development Environment Setup

This is what a POM.xml file looks like:

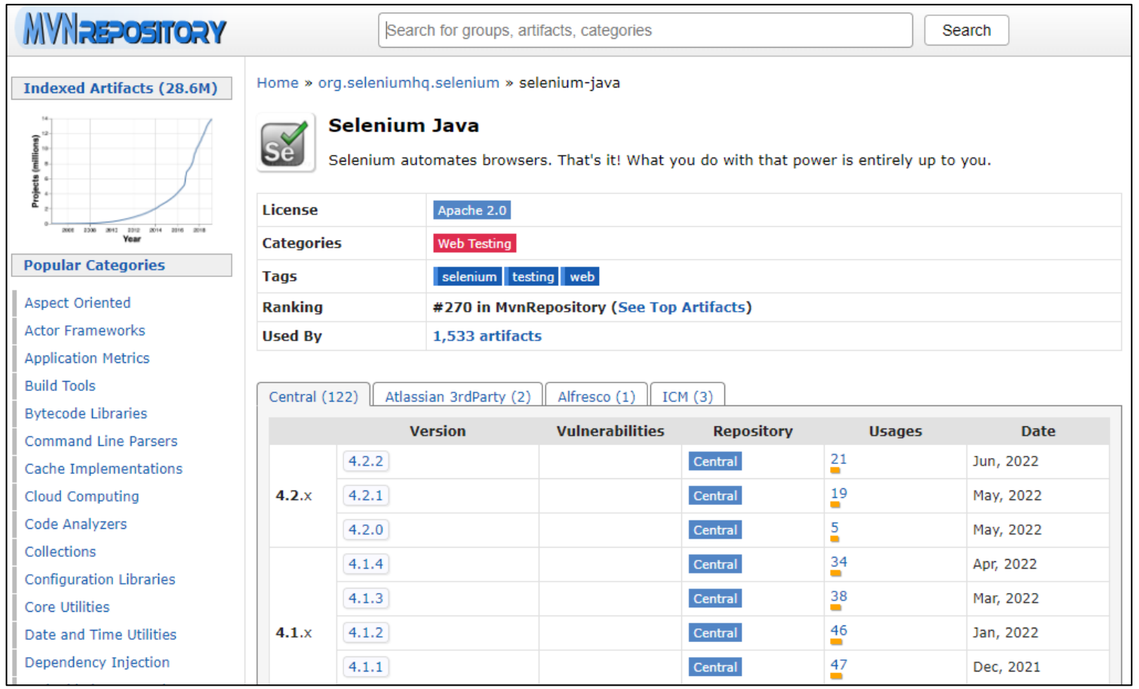
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>MavenDemo</groupId>
  <artifactId>MavenDemo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</project>
```

Step 6:
How to add dependencies to the POM.xml file?

©Simplilearn. All rights reserved.

Development Environment Setup

These are the steps to add dependencies to a POM.xml file:



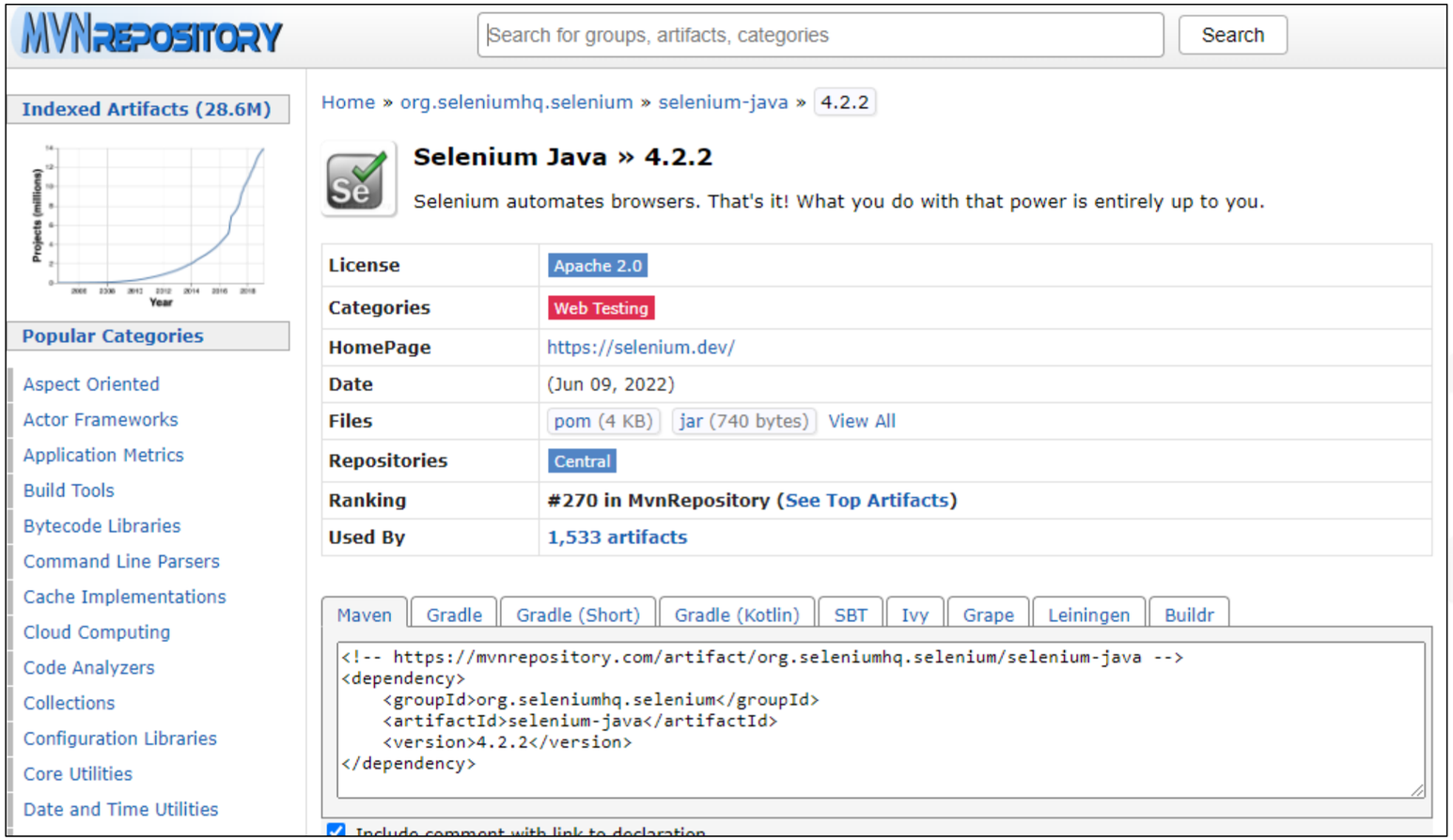
- Please visit website MVN Repository – https://mvnrepository.com/

- Type for Selenium Java in the search box

- Click on the latest version

©Simplilearn. All rights reserved.

Development Environment Setup



Copy the content as shown below and paste it in the pom.xml of the project.

©Simplilearn. All rights reserved.

Development Environment Setup

The updated pom.xml will look like this.

*CucumberDemo/pom.xml

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>CucumberDemo</groupId>
5   <artifactId>CucumberDemo</artifactId>
6   <version>0.0.1-SNAPSHOT</version>
7 </project>
```

- Clean and build the project. Click on **Project** and then select **Clean**. This will add Maven dependencies folder to the project.

©Simplilearn. All rights reserved.

Development Environment Setup

Adding selenium dependency :

- This will indicate to Maven that Selenium jar files need to be downloaded from the central repository to the local repository.

```
<dependency>

<groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
</dependency>
```

- Open the pom.xml in the edit mode
- Create dependencies tag () inside the project tag
- Inside the dependencies tag, create a dependency tag ()

©Simplilearn. All rights reserved.

Development Environment Setup

Adding cucumber-java dependency:

- This will indicate Maven on which Cucumber files need to be downloaded from the central repository to the local repository.

```
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>6.8.1</version>
</dependency>
```

- Create one more dependency tag

©Simplilearn. All rights reserved.

Development Environment Setup

Adding cucumber-junit dependency:

- This will indicate Maven on which Cucumber JUnit files need to be downloaded from the central repository to the local repository.

```
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>6.8.1</version>
    <scope>test</scope>
</dependency
```

- Create one more dependency tag

©Simplilearn. All rights reserved.

Development Environment Setup

Adding cucumber-junit dependency:

- This will indicate Maven on which Cucumber JUnit files need to be downloaded from the central repository to the local repository.

```
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>6.8.1</version>
    <scope>test</scope>
</dependency
```

- Create one more dependency tag

©Simplilearn. All rights reserved.

Development Environment Setup

Adding JUnit dependency:

- This will indicate Maven on which JUnit files need to be downloaded from the central repository to the local repository.

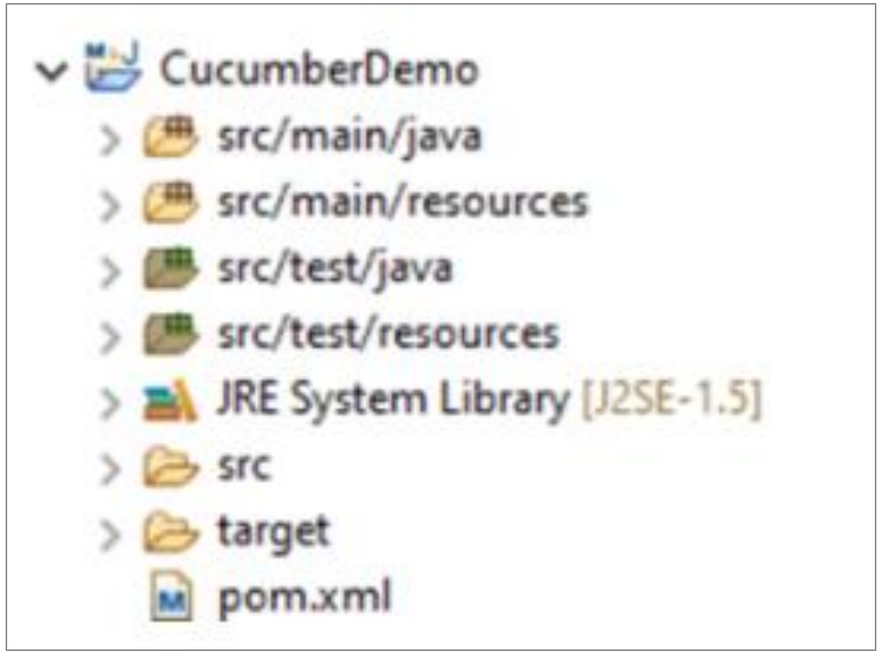
```
<dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>4.12</version>
   <scope>test</scope>
</dependency>
```

- Create one more dependency tag

©Simplilearn. All rights reserved.

Development Environment Setup

The image shows the Maven Project that is called CucumberDemo:



©Simplilearn. All rights reserved.

Development Environment Setup

This is the image of the pom.xml created:

*CucumberDemo/pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>CucumberDemo</groupId>
  <artifactId>CucumberDemo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</project>
```

©Simplilearn. All rights reserved.

Development Environment Setup

After adding the above mentioned dependencies, the pom.xml looks like this:

```
5   <groupId>com.cucumber</groupId>
6   <artifactId>demo</artifactId>
7   <version>0.0.1-SNAPSHOT</version>
8   <packaging>jar</packaging>
9
10  <name>demo</name>
11  <url>http://maven.apache.org</url>
12
13  <properties>
14    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15  </properties>
16
17  <dependencies>
18    <dependency>
19      <groupId>junit</groupId>
20      <artifactId>junit</artifactId>
21      <version>4.12</version>
22      <scope>test</scope>
23    </dependency>
24
```

```
25
26  <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
27  <dependency>
28      <groupId>io.cucumber</groupId>
29      <artifactId>cucumber-java</artifactId>
30      <version>6.8.1</version>
31  </dependency>
32
33  <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->
34  <dependency>
35      <groupId>io.cucumber</groupId>
36      <artifactId>cucumber-junit</artifactId>
37      <version>6.8.1</version>
38      <scope>test</scope>
39  </dependency>
40
41
42
43  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
44  <dependency>
45      <groupId>org.seleniumhq.selenium</groupId>
46      <artifactId>selenium-java</artifactId>
47      <version>3.141.59</version>
48  </dependency>
```

©Simplilearn. All rights reserved.

Development Environment Setup

Entire jar files get added to the Maven dependency:

Congratulations!! We are done with the setup of Cucumber in Eclipse, Happy Learning.

©Simplilearn. All rights reserved.

Key Takeaways

- Business and technical language are connected by Cucumber.
- Gherkin is easy to understand even for non-programmers.
- Tags are used to organize Feature and Scenario.
- A feature file is used to write acceptance steps for automation testing.

©Simplilearn. All rights reserved.

Key Takeaways

- The hook is the block of code that can be defined with each Scenario.
- Step definition is a java method class with an annotation above it.
- Hook is defined by using annotation @After and @Before.
- Testing through multiple tags can be done by using And, OR operators.

©Simplilearn. All rights reserved.