Maven is a Build tool

What Is Maven?

- Maven is a build tool that helps in project management. The tool helps in building and documenting the project

What Is Maven?

- Maven is a build tool that helps in project management. The tool helps in building and documenting the project

Maven

1. When we create something like an project java
2. When we want work with database we need database library need download binaries in the form of jar file
3. Similar if you want to work with selenium we need work with drivers and attach with selenium project
4.

What is Maven ?

- Maven is a software project management and comprehension tool primarily used with Java-based projects but that can also be used to manage projects in other programming languages like C# and Ruby. Maven helps manage builds, documentation, reporting, dependencies, software configuration management (SCM), releases and distribution.

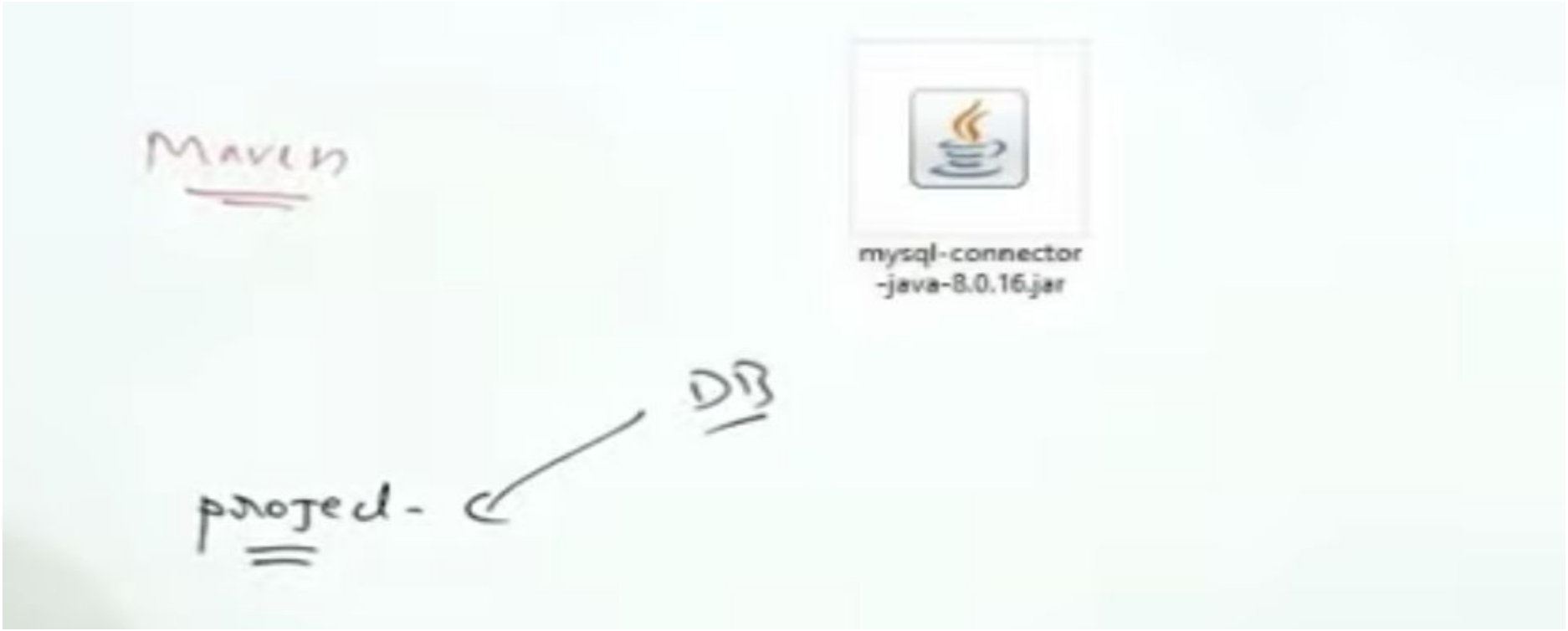
- Many integrated development environments (IDEs) provide plug-ins or add-ons for Maven, thus enabling Maven to compile projects from within the IDE.

Why Maven ?

- Before maven , we had Ant has build tool. This needs lot of configuration like where the source code exists , where the output should be stored and many more.
- Maven uses **Convention** over **Configuration**, which means developers are not required to create build process themselves.

Item	Default
source code	${basedir}/src/main/java
Resources	${basedir}/src/main/resources
Tests	${basedir}/src/test
Complied byte code	${basedir}/target/classes
distributable JAR	${basedir}/target

If you want to work with msql we need mysql jar files



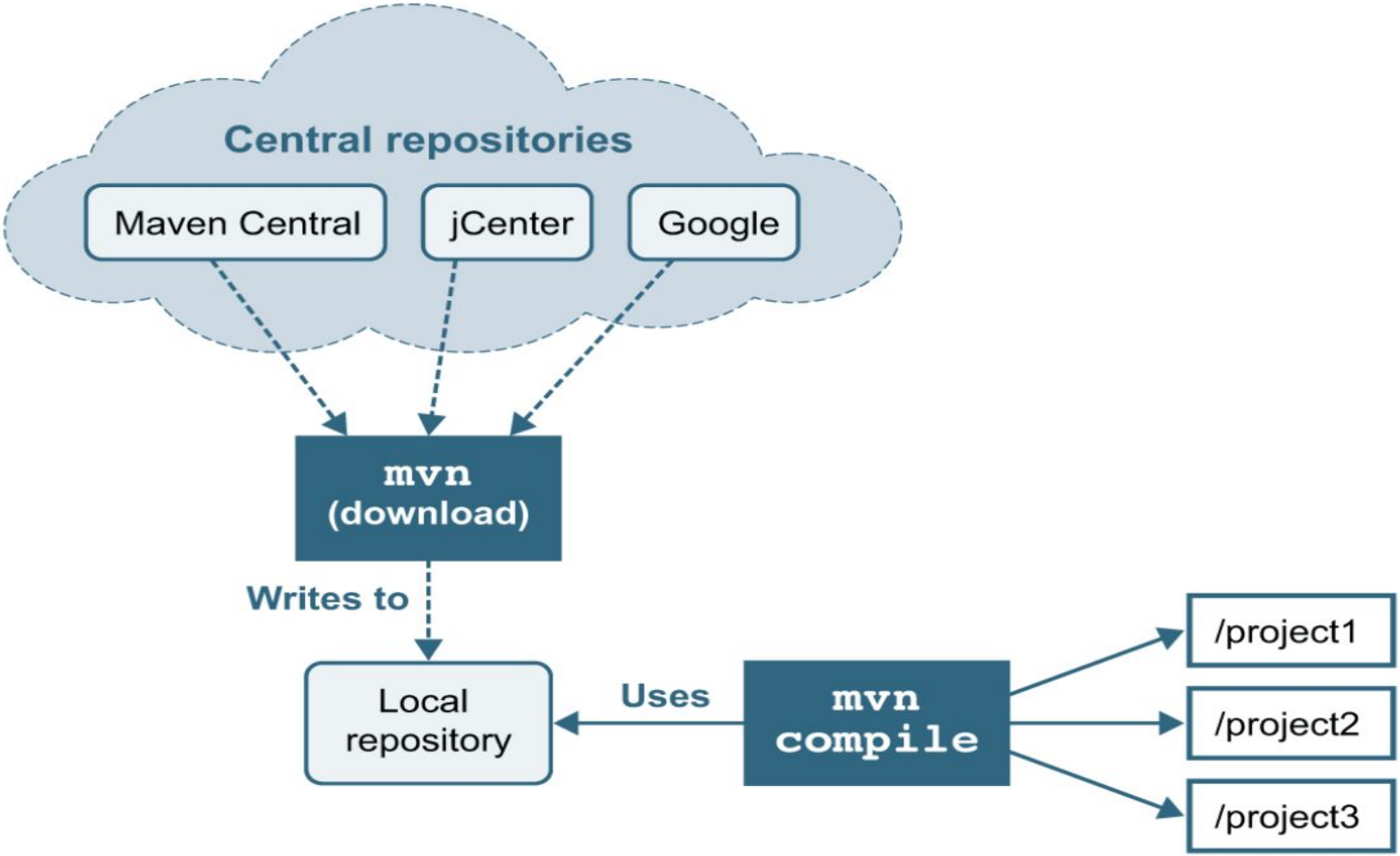
Maven

- Build automation is the act of scripting or automating a wide variety of tasks that software developers do in their day-to-day activities including things like

- Compiling computer source code into binary code
- Packaging binary code
- Running automated tests
- Deploying to production systems
- Creating documentation and/or release notes

List

- Apache Ant
- Apache Buildr
- Apache Maven
- Sbt
- Tup
- Gradle
- Visual build


Central repositories
Maven Central
jCenter
Google
mvn
(download)
Writes to
Local
repository
Uses
mvn
compile
/project1
/project2
/project3

What Is Maven?

- Project Object Model is an XML file that has all the information regarding project and configuration details
- When we tend to execute a task, Maven searches for the POM in the current directory

aven is written in Java and C# and is
sed on Project Object Model (POM)

The Need For Maven

Maven is chiefly used for Java-based projects. It helps in downloading dependencies, which refers to the libraries or JAR files

The Problems That Maven Solved:

- Getting right JAR files for each project as there may be different versions of separate packages
- To download dependencies visiting of the official website of different software is not needed. We can just visit “mvnrepositiry.com”
- Helps to create the **right project structure** which is essential for execution

1. We need to go for each and every website and download the dependency
2. If we need update every dependency in the project
3. Remove existing libraries again we need download and add them into the project
4. To overcome these problems we need maven tool

Maven will provide below benefits

1. Maven will provide project structure
2. Pom.xml in which we have dependencies and plugins when we have dependencies which will download all latest packages to your project
3. Plugins will have all configuration stuff related to the project
4. Maven we can generate reports and project related documentation
5. We can do packaging of the project
6. Main propose will be automatically download the dependencies in the project
7.

POM.XML

1. We have two things here dependencies and plugins
2. Dependencies -downloading third party libraries and packages
3. Plugins -different configuration which the project will run
4. Third party libraries are available in the form of jar file

Maven

project -

pom.xml

→ dependencies → download
Third party
Lib/jars

→ plugins
Configuration

Maven

Maven project

pom.XML

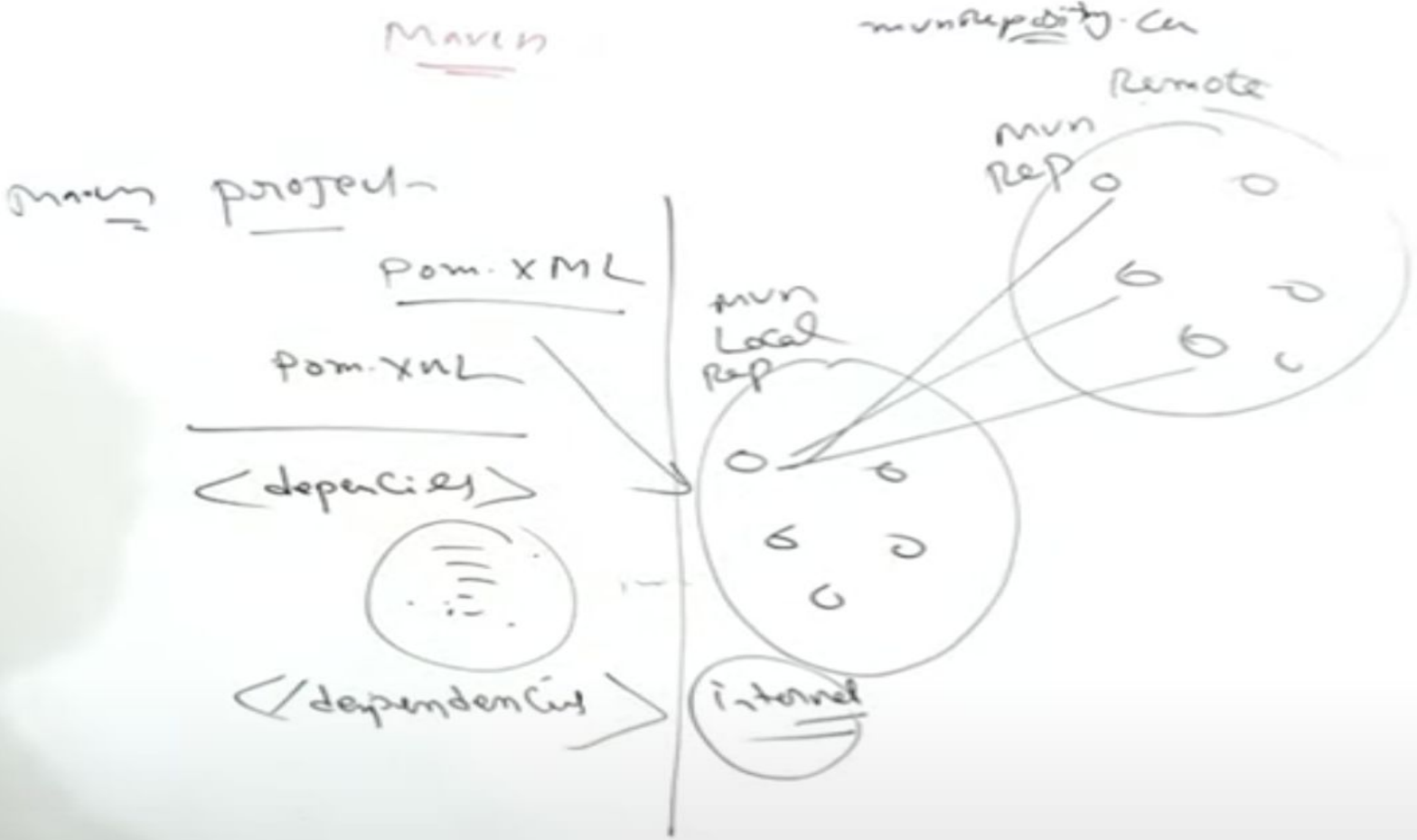
pom.XML

<dependency>

</dependency>

mvn Local Rep

1. Maven first it will create local repository
2. As soon as you add the dependency section it will download all the jar and third party libraries into maven local repo
3. From the remote maven repo it will download the jar file
4. To create all dependencies we need to have internet connection
5. Once we have internet connection we can get all the dependencies into our local file
6. Then pom.xml file refer those libraries
7. All the dependency will be downloaded and stored in local repository and from remote repository


Maven
mvnrepository.com
Remote
mvn
Rep
maven project
pom.xml
pom.xml
<dependency>
</dependencies>
mvn
Local
Rep
internet

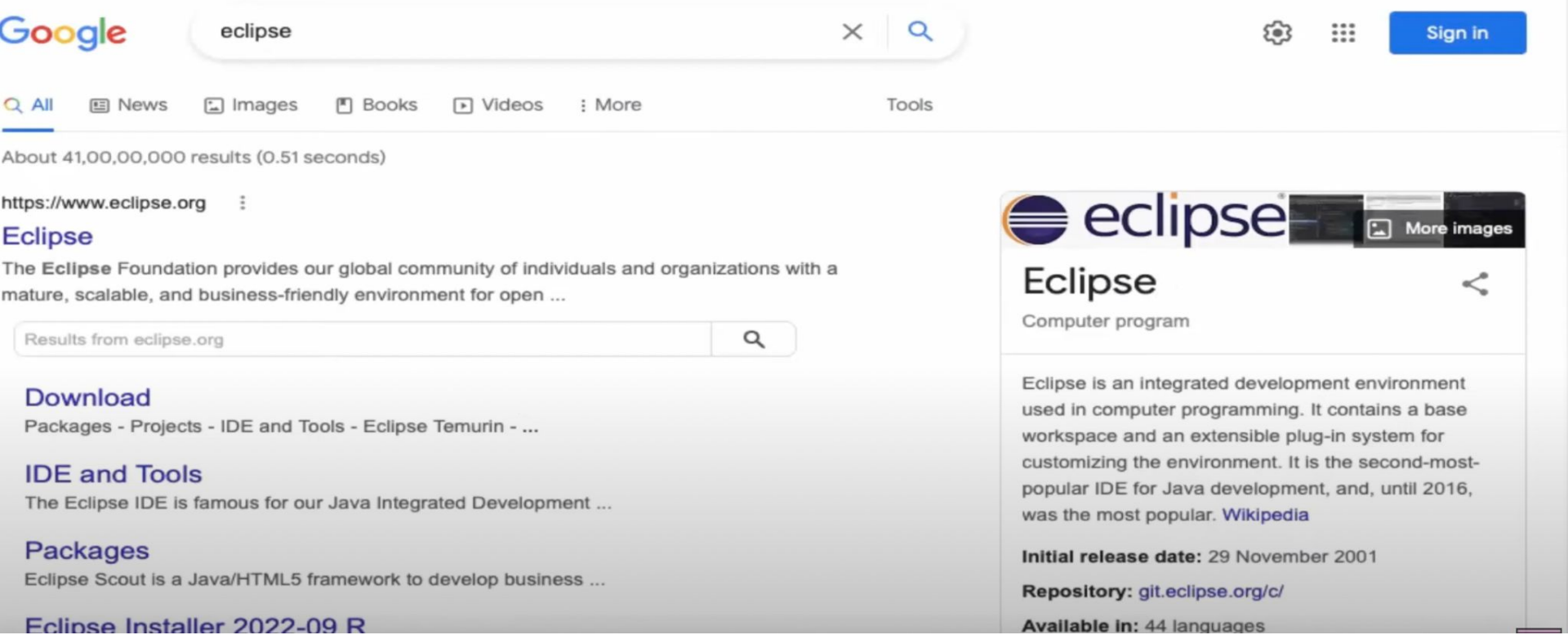
1. In the dependency section itself we need to remove the version number
2. Take the repository from remote repo and update local repo
3. There is also concept called plugins -project related configurations are specified in the plugins
4. Suppose if you want to add compiler project we need to have compiler plugin
5. We need to have whole project additional plugin is required for our application should start and stop for that we need additional plugins
6. Plugins will control entire project
7.

1) Create a maven project in Eclipse

2) Maven Project folder structure

3) Project Object Model(pom.xml)

Eclipse Install



Advertise Here

The Eclipse Installer 2022-09 R now includes a JRE for macOS, Windows and Linux.

Get **Eclipse IDE 2022-09**

Install your favorite desktop IDE packages.

Download x86_64

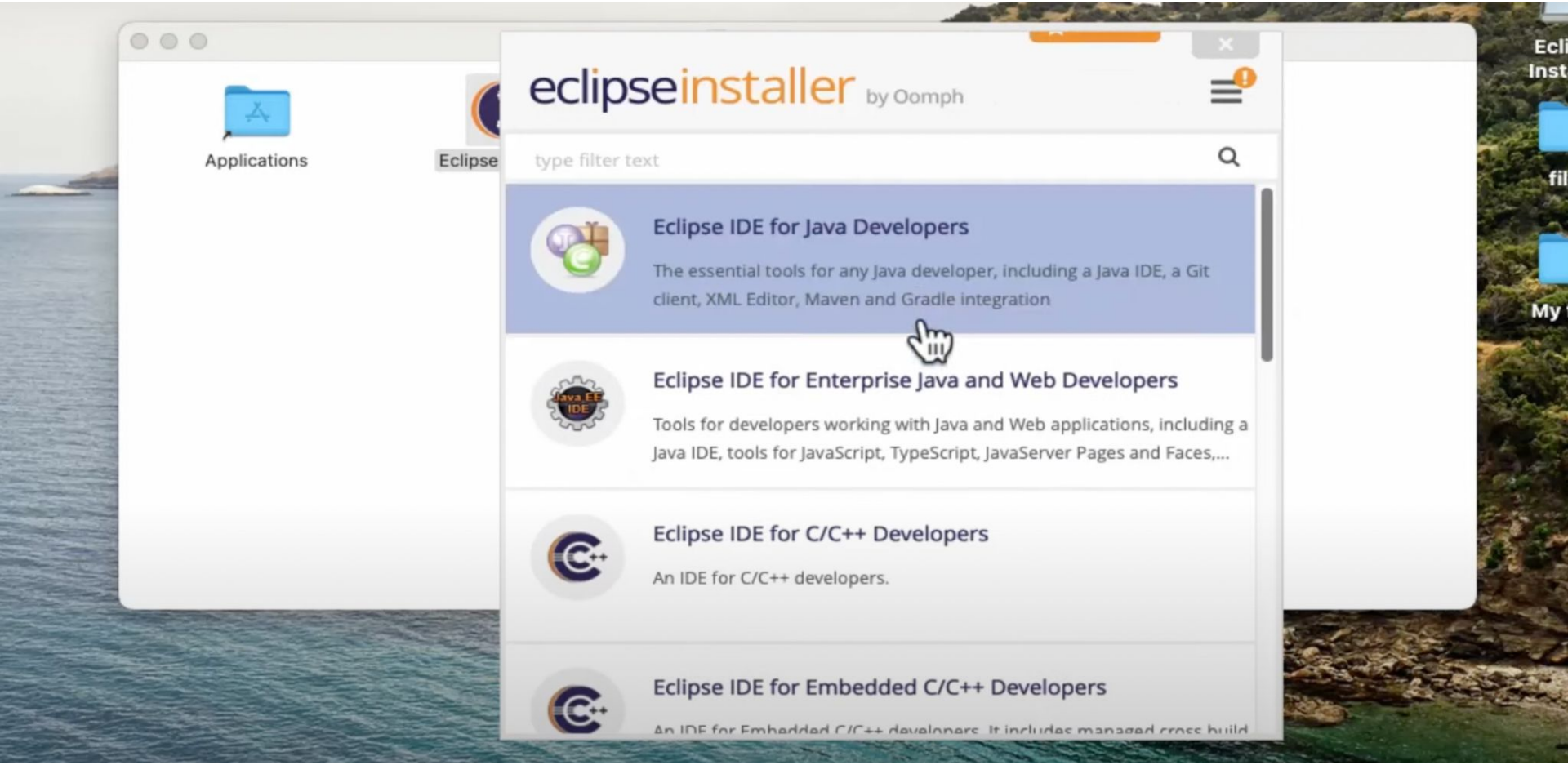
Download Packages | Need Help?

OpenJDK Runtimes

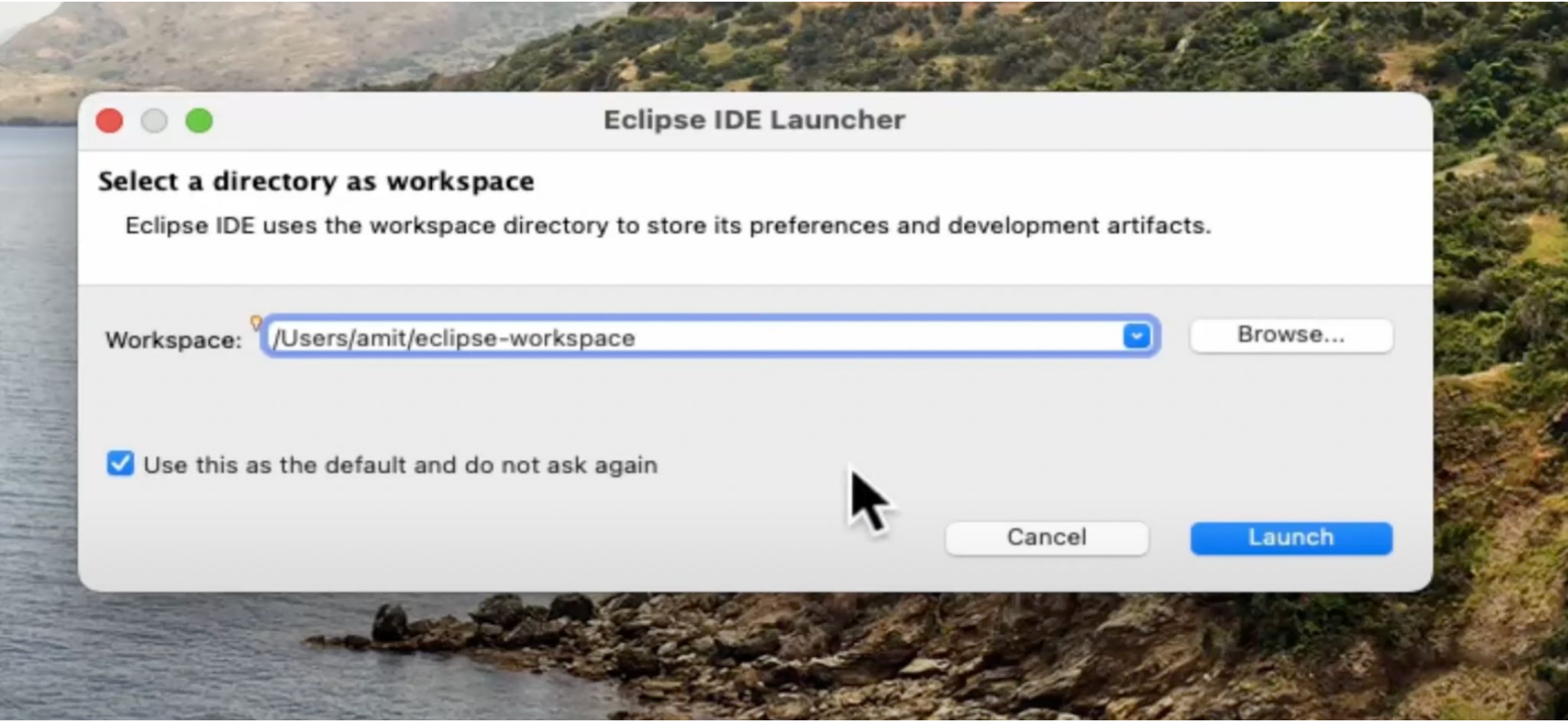
The Eclipse Temurin™ project provides high-quality, TCK certified OpenJDK runtimes and associated technology for use across the Java™ ecosystem.

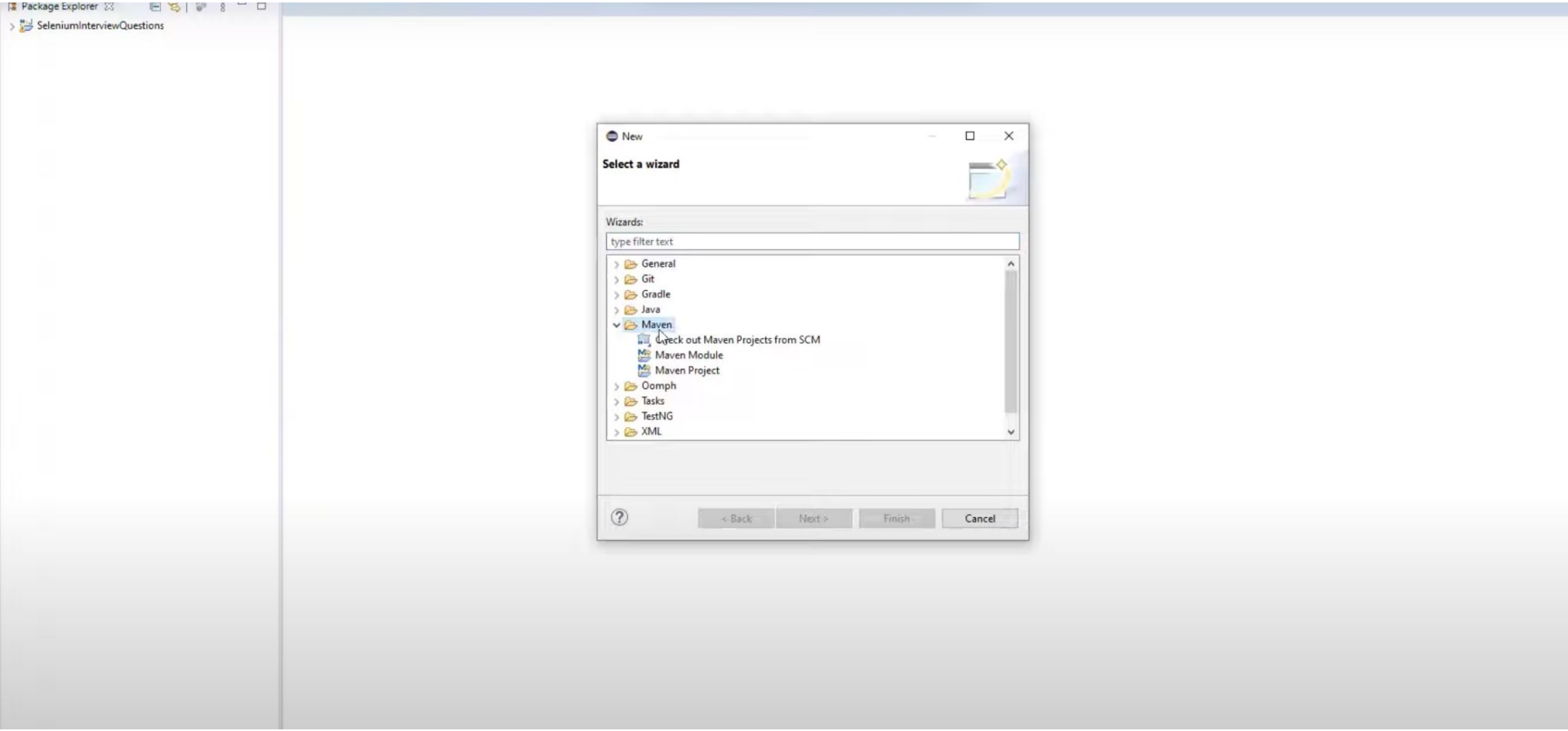
Download Now

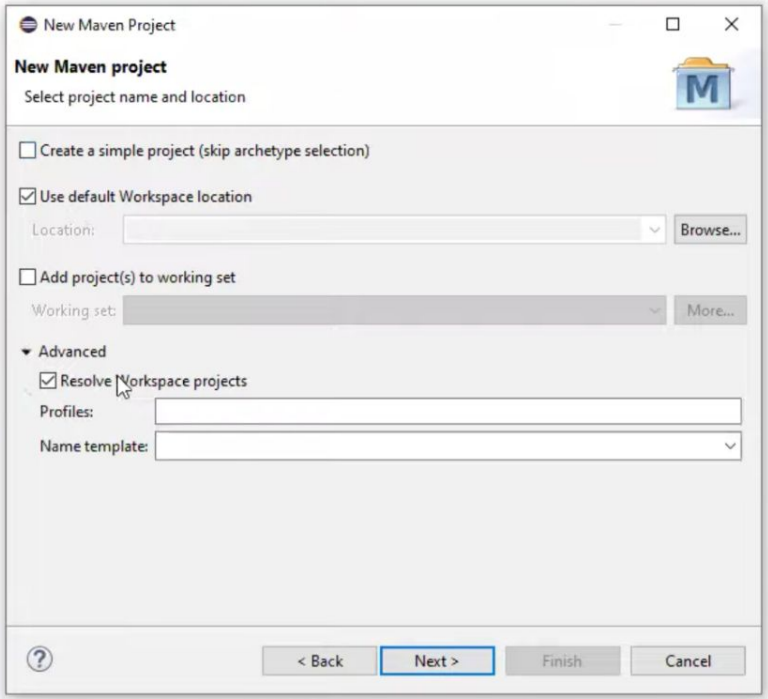
Learn More

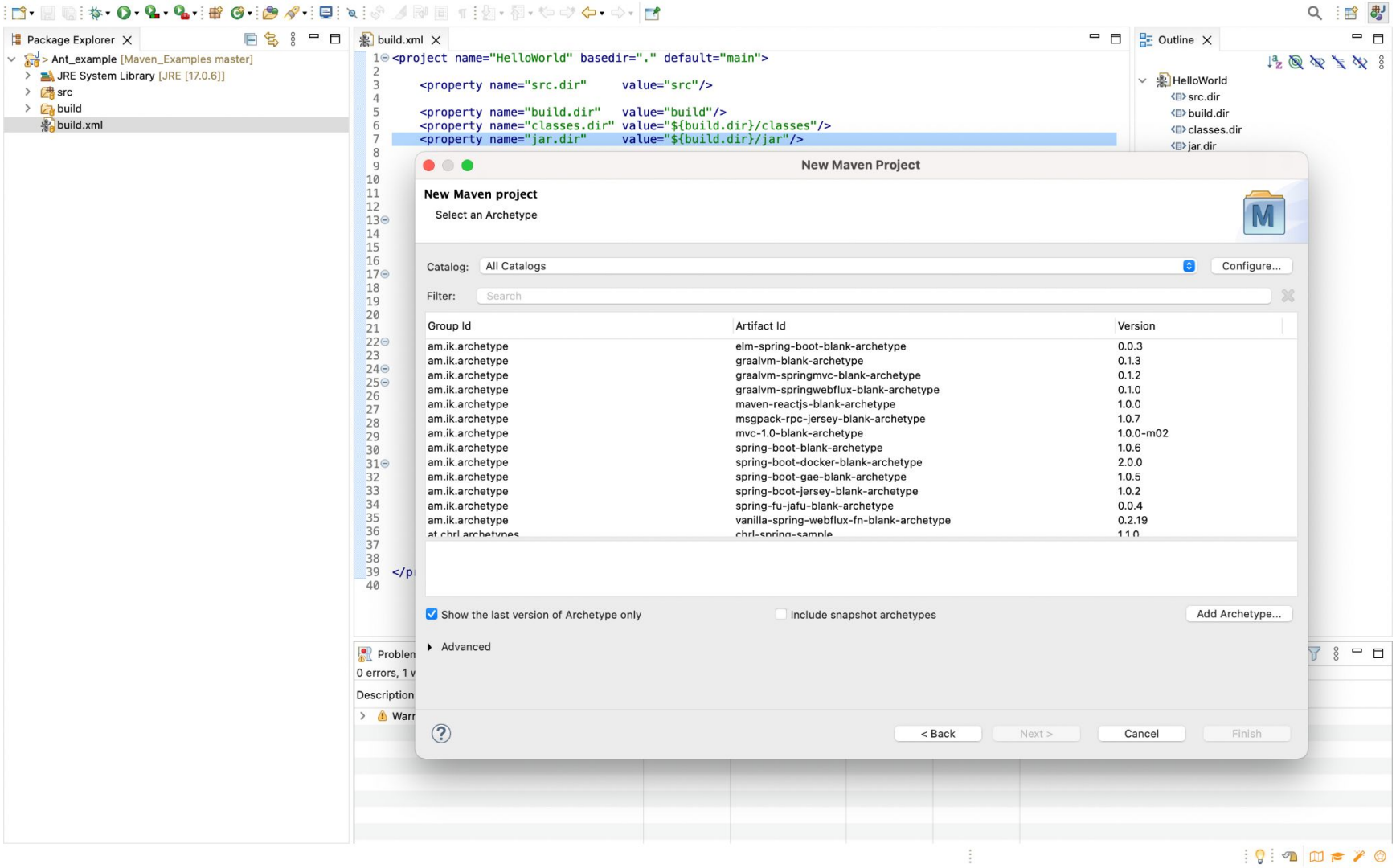

Applications
Eclipse
eclipseinstaller by Oomph
type filter text
Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration
Eclipse IDE for Enterprise Java and Web Developers
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces,...
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers.
Eclipse IDE for Embedded C/C++ Developers


Applications
Eclipse
eclipseinstaller by Oomph
DONATE
Eclipse IDE for Java Developers
details
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration.
Java 17+ VM
JRE 17.0.4.1 - https://download.eclipse.org/justj/jre
Installation Folder
/Users/amit/eclipse/java-2022-09
INSTALLING
Cancel Installation
BACK

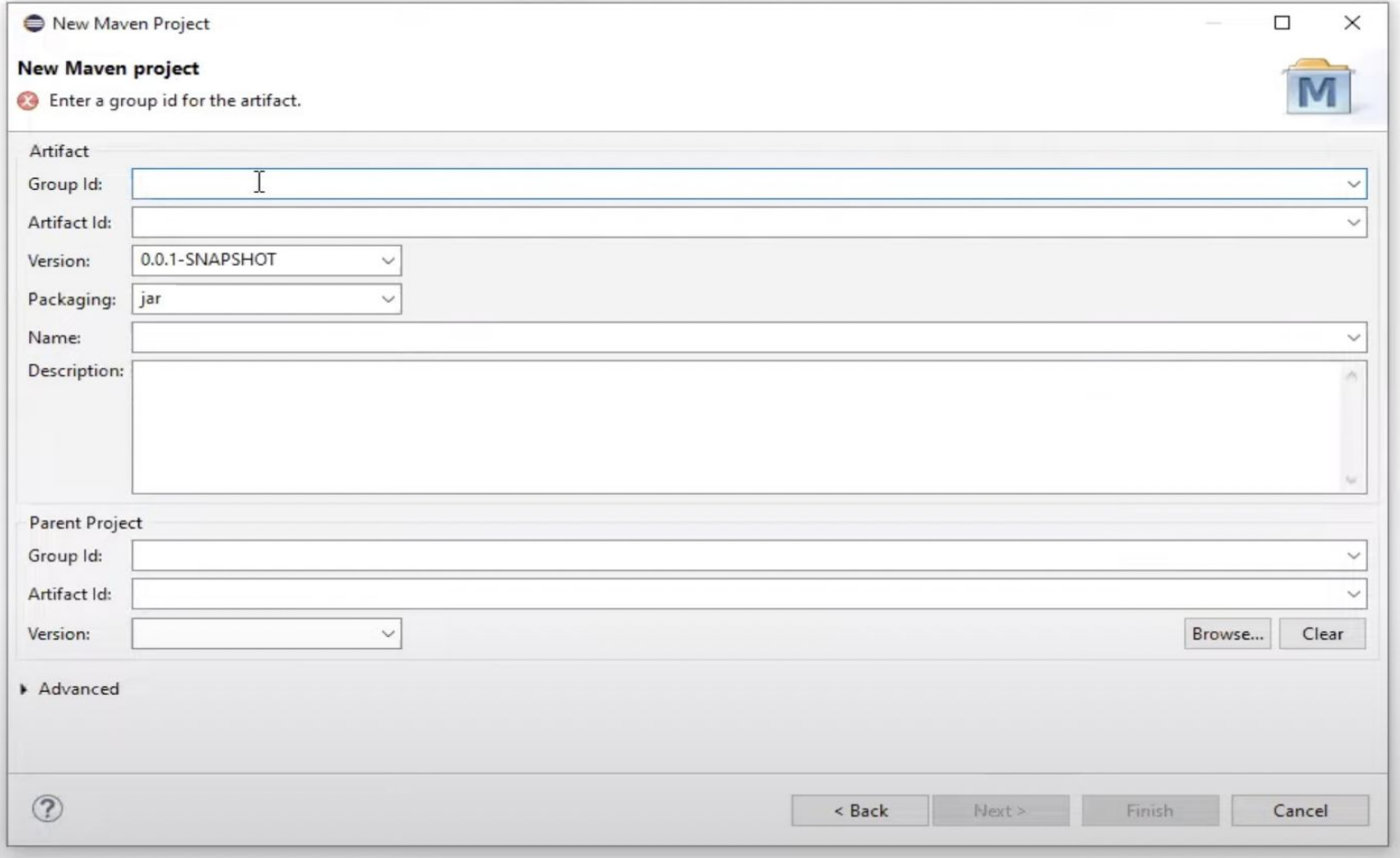

Eclipse IDE Launcher
Select a directory as workspace
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.
Workspace:
/Users/amit/eclipse-workspace
Browse...
Use this as the default and do not ask again
Cancel
Launch


Package Explorer
SeleniumInterviewQuestions
New
Select a wizard
Wizards:
type filter text
General
Git
Gradle
Java
Maven
Check out Maven Projects from SCM
Maven Module
Maven Project
Oomph
Tasks
TestNG
XML
< Back
Next >
Finish
Cancel

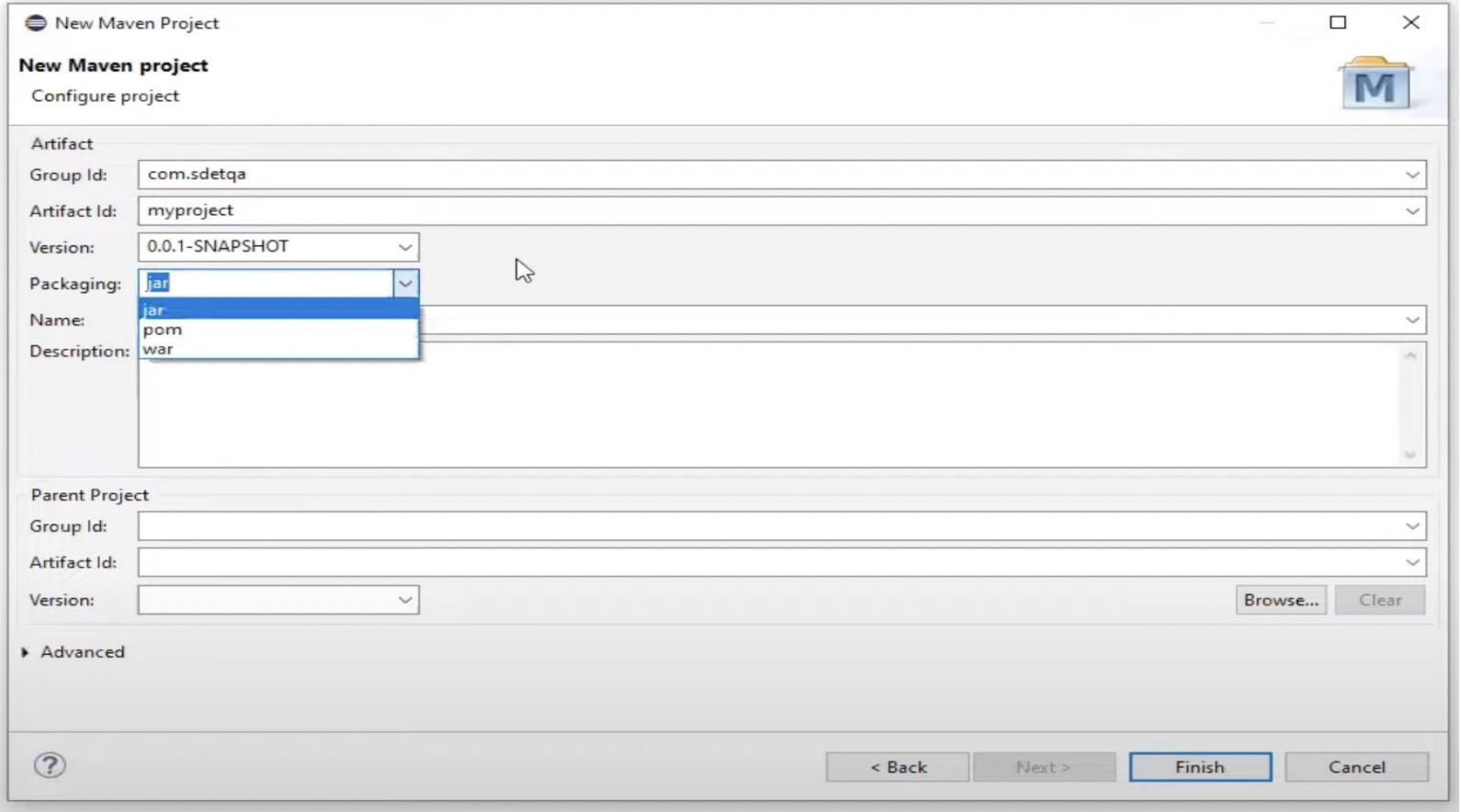

New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
Resolve Workspace projects
Profiles:
Name template:
< Back
Next >
Finish
Cancel


Package Explorer
Ant_example [Maven_Examples master]
JRE System Library [JRE [17.0.6]]
src
build
build.xml
build.xml
<project name="HelloWorld" basedir="." default="main">
<property name="src.dir" value="src"/>
<property name="build.dir" value="build"/>
<property name="classes.dir" value="${build.dir}/classes"/>
<property name="jar.dir" value="${build.dir}/jar"/>
Outline
HelloWorld
src.dir
build.dir
classes.dir
jar.dir
New Maven Project
New Maven project
Select an Archetype
Catalog: All Catalogs
Configure...
Filter: Search
Group Id
Artifact Id
Version
am.ik.archetype elm-spring-boot-blank-archetype 0.0.3
am.ik.archetype graalvm-blank-archetype 0.1.3
am.ik.archetype graalvm-springmvc-blank-archetype 0.1.2
am.ik.archetype graalvm-springwebflux-blank-archetype 0.1.0
am.ik.archetype maven-reactjs-blank-archetype 1.0.0
am.ik.archetype msgpack-rpc-jersey-blank-archetype 1.0.7
am.ik.archetype mvc-1.0-blank-archetype 1.0.0-m02
am.ik.archetype spring-boot-blank-archetype 1.0.6
am.ik.archetype spring-boot-docker-blank-archetype 2.0.0
am.ik.archetype spring-boot-gae-blank-archetype 1.0.5
am.ik.archetype spring-boot-jersey-blank-archetype 1.0.2
am.ik.archetype spring-fu-jafu-blank-archetype 0.0.4
am.ik.archetype vanilla-spring-webflux-fn-blank-archetype 0.2.19
Show the last version of Archetype only
Include snapshot archetypes
Add Archetype...
Advanced
< Back
Next >
Cancel
Finish

Above are the template which will help is the selection of the suitable project


New Maven Project
New Maven project
Enter a group id for the artifact.
Artifact
Group Id:
Artifact Id:
Version:
0.0.1-SNAPSHOT
Packaging:
jar
Name:
Description:
Parent Project
Group Id:
Artifact Id:
Version:
Browse...
Clear
Advanced
< Back
Next >
Finish
Cancel

1. Groupid is the one which is company name
2. Artificat id is name of the project
3. By combining groupid and artificat id packages will be created
4.


New Maven Project
New Maven project
Configure project
Artifact
Group Id:
com.sdetqa
Artifact Id:
myproject
Version:
0.0.1-SNAPSHOT
Packaging:
jar
jar
pom
war
Name:
Description:
Parent Project
Group Id:
Artifact Id:
Version:
Browse...
Clear
Advanced
< Back
Next >
Finish
Cancel

Folder Structure in maven project

/src/main/java -development source code is present

/src/main/resources - while developing code if he need resource he can keep in resources folder

/src/main/test-once code is developer need to test code for that we have test folder

Src and target files which are down are used by maven for storing temporary files

POM XML is the main file which will control entire project

POM-Project object Model

Using pom we will control and organize the entire project

Maven Build Life Cycle

A Build Lifecycle is Made Up of Phases

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

For example, the default lifecycle comprises of the following phases (for a complete list of the lifecycle phases, refer to the Lifecycle Reference):

- `validate` - validate the project is correct and all necessary information is available
- `compile` - compile the source code of the project
- `test` - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- `package` - take the compiled code and package it in its distributable format, such as a JAR.
- `verify` - run any checks on results of integration tests to ensure quality criteria are met
- `install` - install the package into the local repository, for use as a dependency in other projects locally
- `deploy` - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

These lifecycle phases (plus the other lifecycle phases not shown here) are executed sequentially to complete the `default` lifecycle. Given the lifecycle phases above, this means that when the default lifecycle is used, Maven will first validate the project, then will try to compile the sources, run those against the tests, package the binaries (e.g. jar), run integration tests against that package, verify the integration tests, install the verified package to the local repository, then deploy the installed package to a remote repository.

Scope will help in deciding the dependency section phase

In which format we need to do packaging jar or war format we need select the format

1) Create a maven project in Eclipse
2) Maven Project folder structure
3) Project Object Model(pom.xml)

Maven Build Life Cycle

Build Life Cycle

Validate
Compile
Test
Package
Integration Test
Verify
Install - install in local rep
Deploy - install in remote rep's to use by other projects

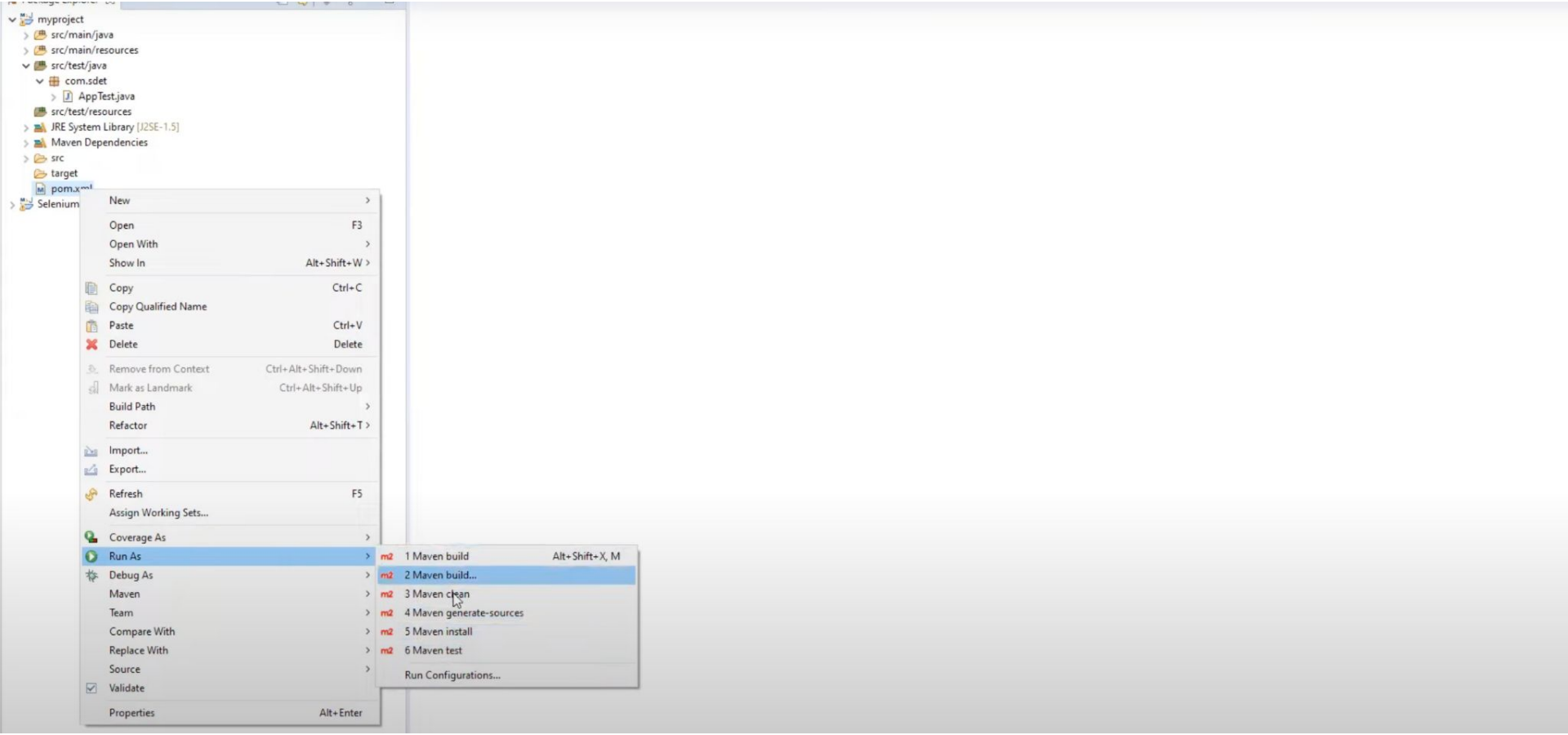
Maven Build Life Cycle

Build Life Cycle
Validate
Compile
Test
Package
Integration Test
Verify
Install - install in local rep
Deploy - install in remote rep's to use by other projects

Install -once the code is automatically tested we will create jar file in the local repo

By default

Deploy -Deploy the jar file in the remote repository


myproject
src/main/java
src/main/resources
src/test/java
com.sdet
AppTest.java
src/test/resources
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
pom.xml
Selenium
New
Open F3
Open With
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Mark as Landmark Ctrl+Alt+Shift+Up
Build Path
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Assign Working Sets...
Coverage As
Run As
Debug As
Maven
Team
Compare With
Replace With
Source
Validate
Properties Alt+Enter
m2 1 Maven build Alt+Shift+X, M
m2 2 Maven build...
m2 3 Maven clean
m2 4 Maven generate-sources
m2 5 Maven install
m2 6 Maven test
Run Configurations...

In the above page we are showing different commands for life cycle

Maven Build Life Cycle

Build Life Cycle

Validate

Compile

Test

Package

Integration Test

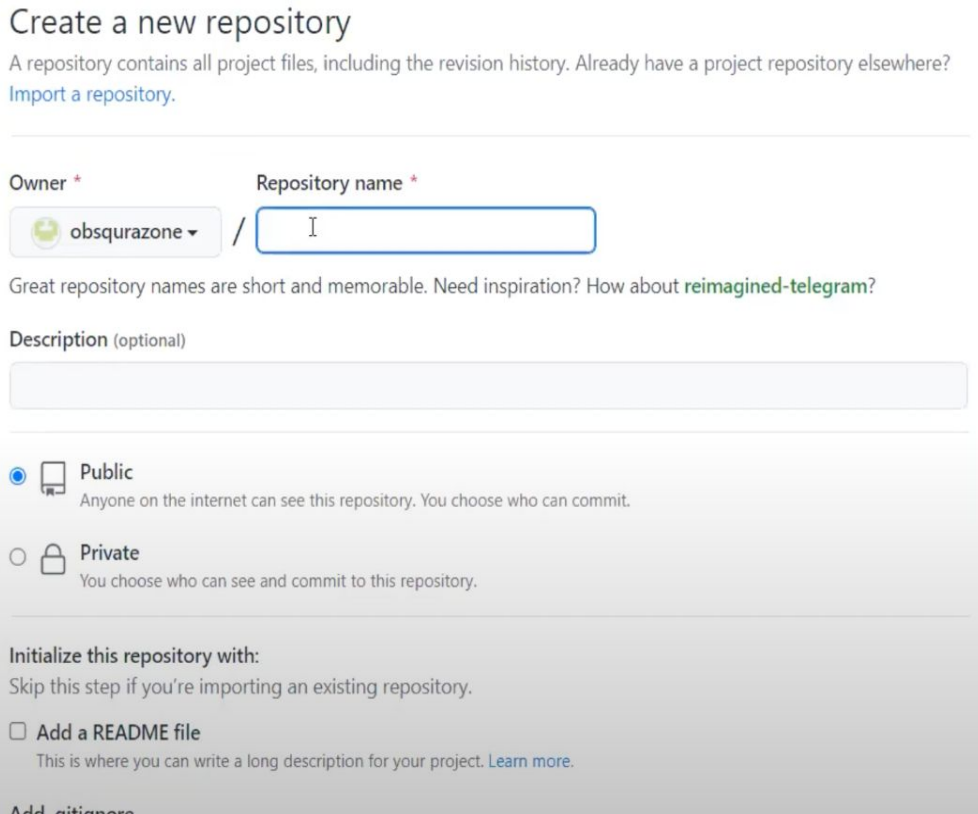
Verify

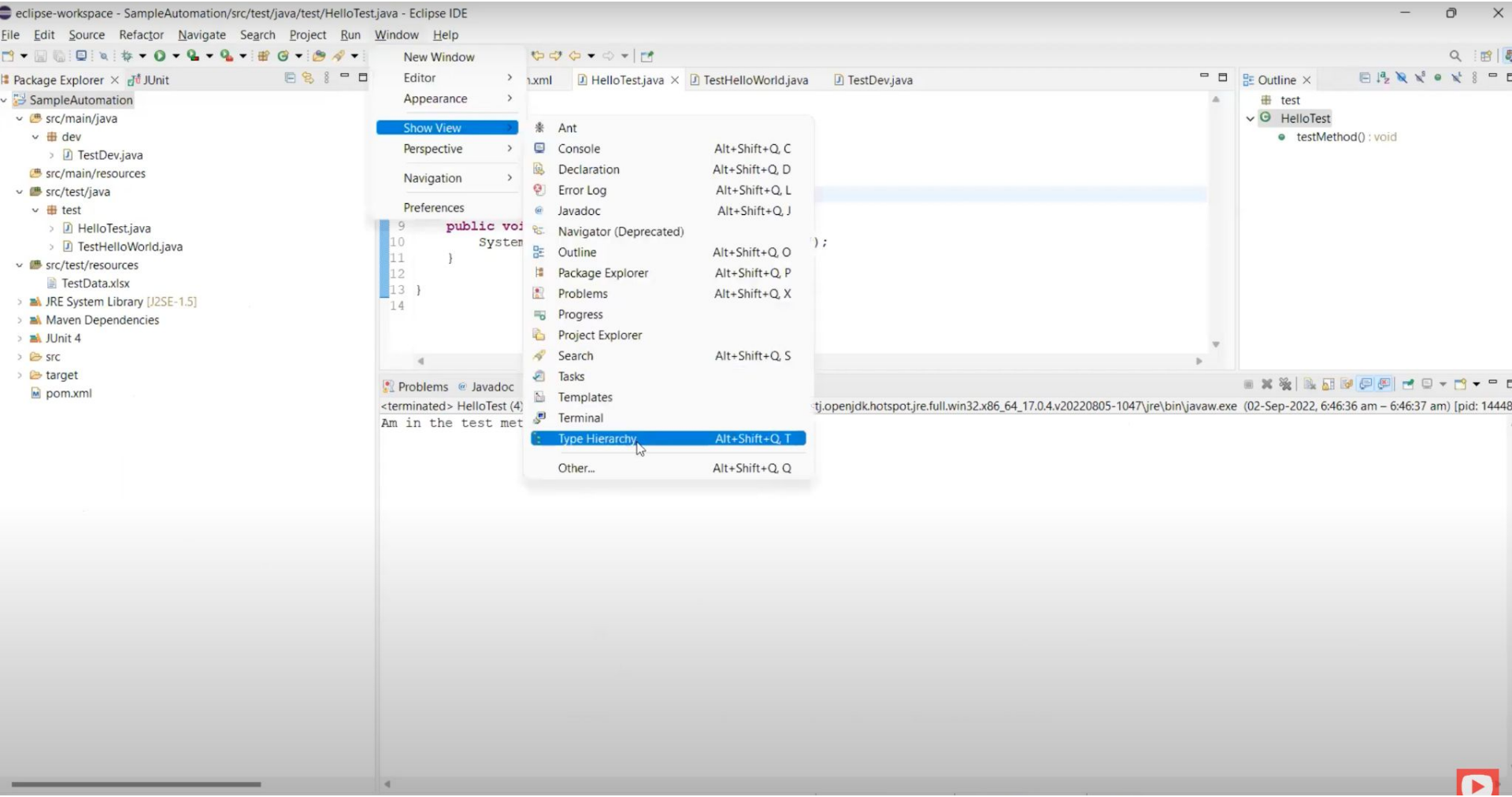
Install - install in local rep

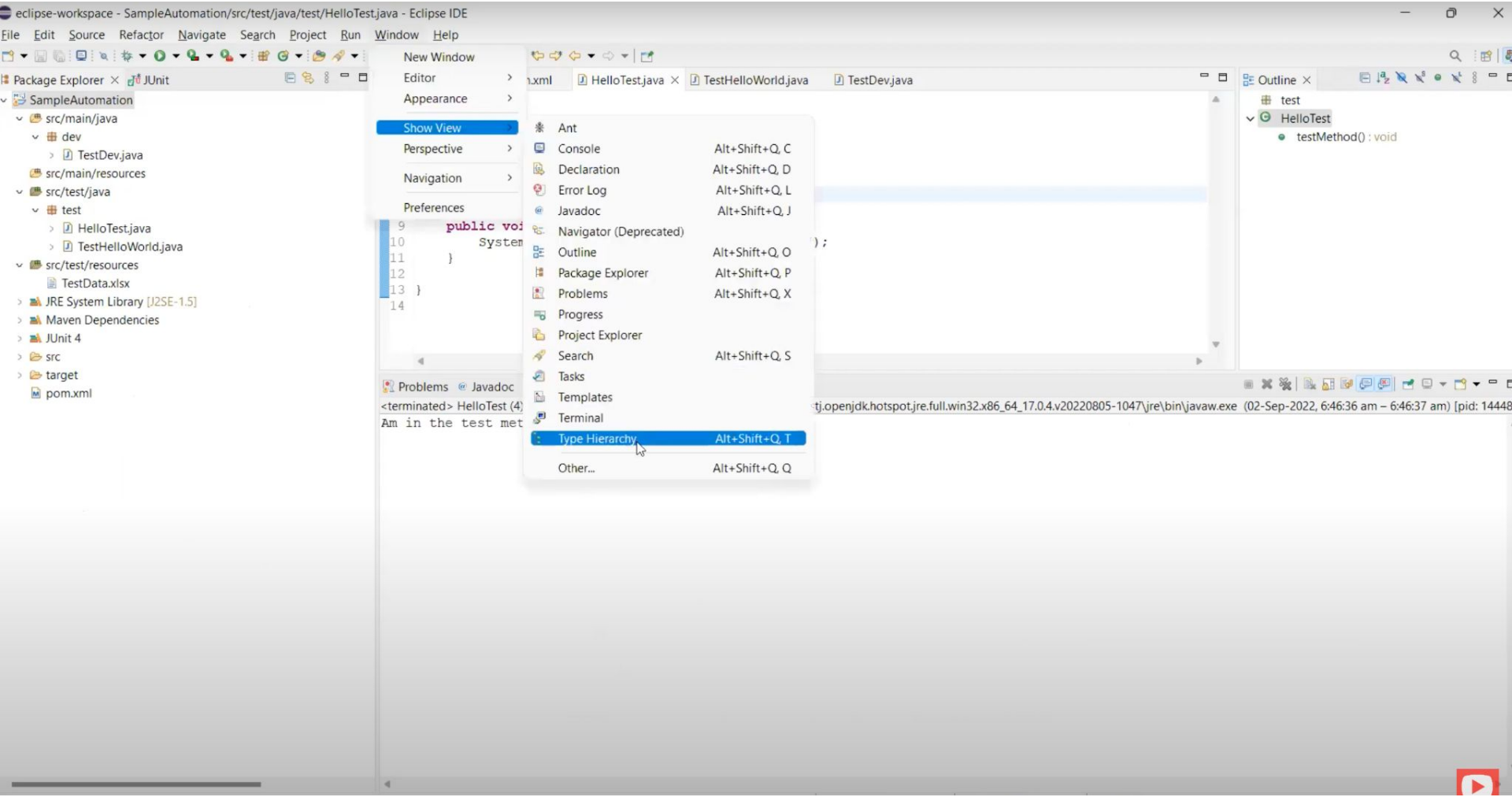
Deploy - install in remote rep's to use by other projects

1. We have validate and compile phase which will be executed after the test phase execution
2. Previous phase will be executed when we execute any phase in maven

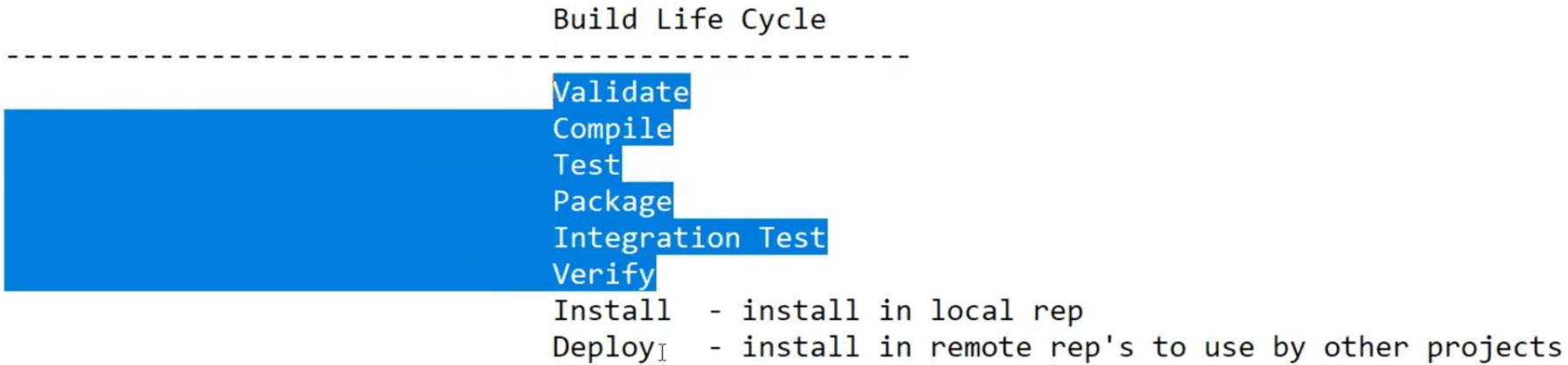
Export git repo to maven

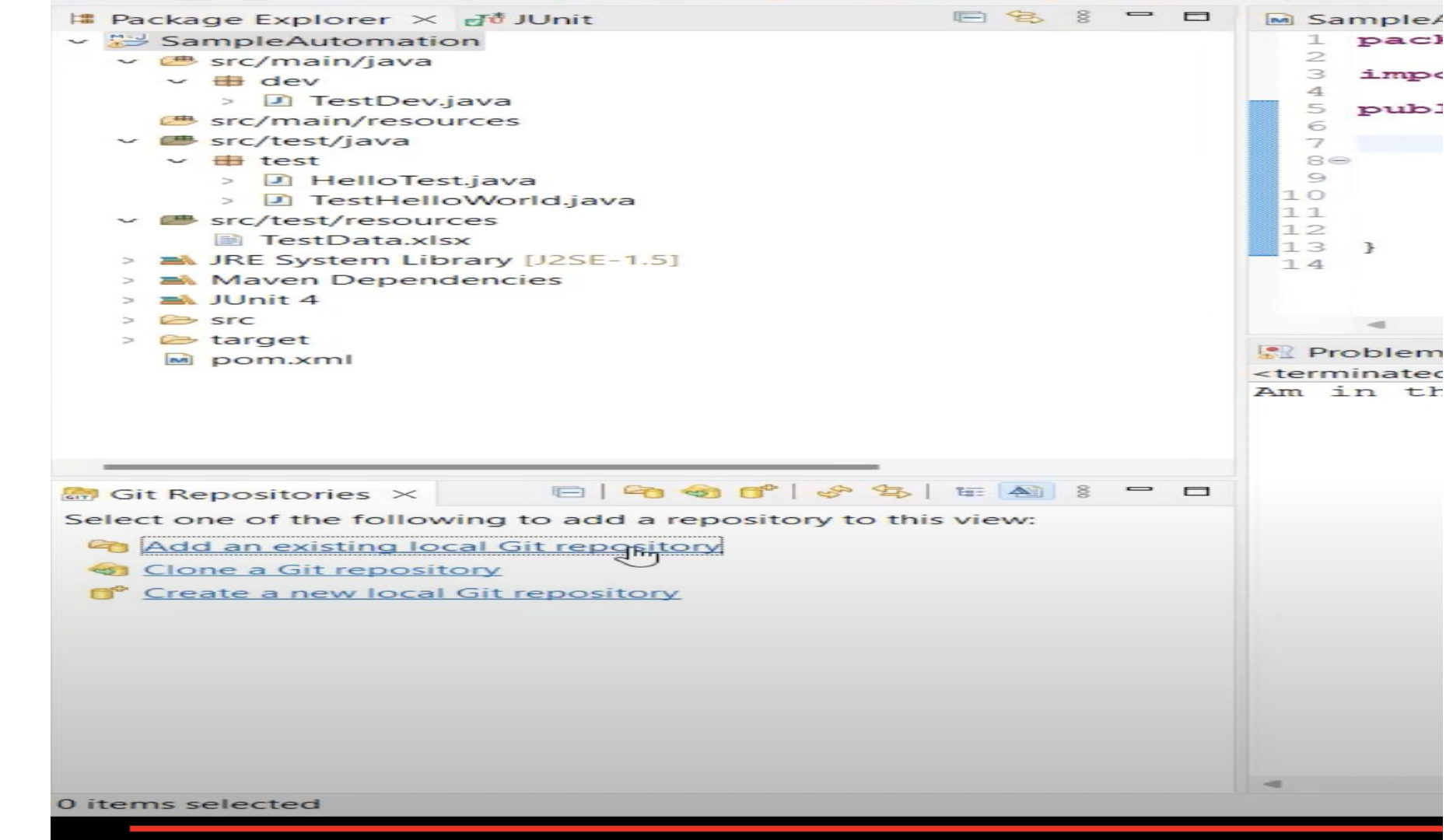



eclipse-workspace - SampleAutomation/src/test/java/test/HelloTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
New Window
Editor
Appearance
Show View
Perspective
Navigation
Preferences
Ant
Console Alt+Shift+Q, C
Declaration Alt+Shift+Q, D
Error Log Alt+Shift+Q, L
Javadoc Alt+Shift+Q, J
Navigator (Deprecated)
Outline Alt+Shift+Q, O
Package Explorer Alt+Shift+Q, P
Problems Alt+Shift+Q, X
Progress
Project Explorer
Search Alt+Shift+Q, S
Tasks
Templates
Terminal
Type Hierarchy Alt+Shift+Q, T
Other... Alt+Shift+Q, Q
Package Explorer
JUnit
SampleAutomation
src/main/java
dev
TestDev.java
src/main/resources
src/test/java
test
HelloTest.java
TestHelloWorld.java
src/test/resources
TestData.xlsx
JRE System Library [J2SE-1.5]
Maven Dependencies
JUnit 4
src
target
pom.xml
HelloTest.java
TestHelloWorld.java
TestDev.java
Outline
test
HelloTest
testMethod() : void
Problems
Javadoc
Am in the test met


eclipse-workspace - SampleAutomation/src/test/java/test/HelloTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
New Window
Editor
Appearance
Show View
Perspective
Navigation
Preferences
Ant
Console Alt+Shift+Q, C
Declaration Alt+Shift+Q, D
Error Log Alt+Shift+Q, L
Javadoc Alt+Shift+Q, J
Navigator (Deprecated)
Outline Alt+Shift+Q, O
Package Explorer Alt+Shift+Q, P
Problems Alt+Shift+Q, X
Progress
Project Explorer
Search Alt+Shift+Q, S
Tasks
Templates
Terminal
Type Hierarchy Alt+Shift+Q, T
Other... Alt+Shift+Q, Q
Package Explorer
JUnit
SampleAutomation
src/main/java
dev
TestDev.java
src/main/resources
src/test/java
test
HelloTest.java
TestHelloWorld.java
src/test/resources
TestData.xlsx
JRE System Library [J2SE-1.5]
Maven Dependencies
JUnit 4
src
target
pom.xml
HelloTest.java
TestHelloWorld.java
TestDev.java
Outline
test
HelloTest
testMethod() : void
Problems
Javadoc
Am in the test met

Maven Build Life Cycle




Package Explorer
JUnit
SampleAutomation
src/main/java
dev
TestDev.java
src/main/resources
src/test/java
test
HelloTest.java
TestHelloWorld.java
src/test/resources
TestData.xlsx
JRE System Library [J2SE-1.5]
Maven Dependencies
JUnit 4
src
target
pom.xml
Git Repositories
Select one of the following to add a repository to this view:
Add an existing local Git repository
Clone a Git repository
Create a new local Git repository
0 items selected

21 What is a Maven Artifact?

- Maven Artifact refers to a file, usually a JAR that gets deployed to a Maven repository

Basic concepts of Maven

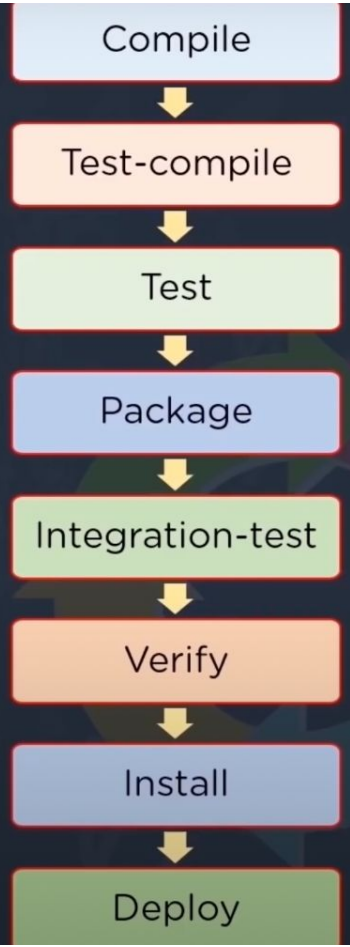
Dependencies and Repositories

- Dependencies refer to the Java libraries that are needed for the project. Repositories refer to the directories of packaged JAR files
- If the dependencies are not present in your local repository, then Maven downloads them from a central repository and stores them in the local repository

Maven Build Lifecyle

The Build Life Cycle has different build phases or stages:

- **Default**: handles project deployment
- **Clean**: handles project cleaning
- **Site**: handles the creation of project site's documentation



Advantages of Maven

- Apache Maven helps **manage all the processes**, such as building, documentation, releasing, and distribution in project management
- The tool **simplifies the process** of project building. It increases the performance of the project and the building process
- The task of **downloading Jar files** and other dependencies is done automatically

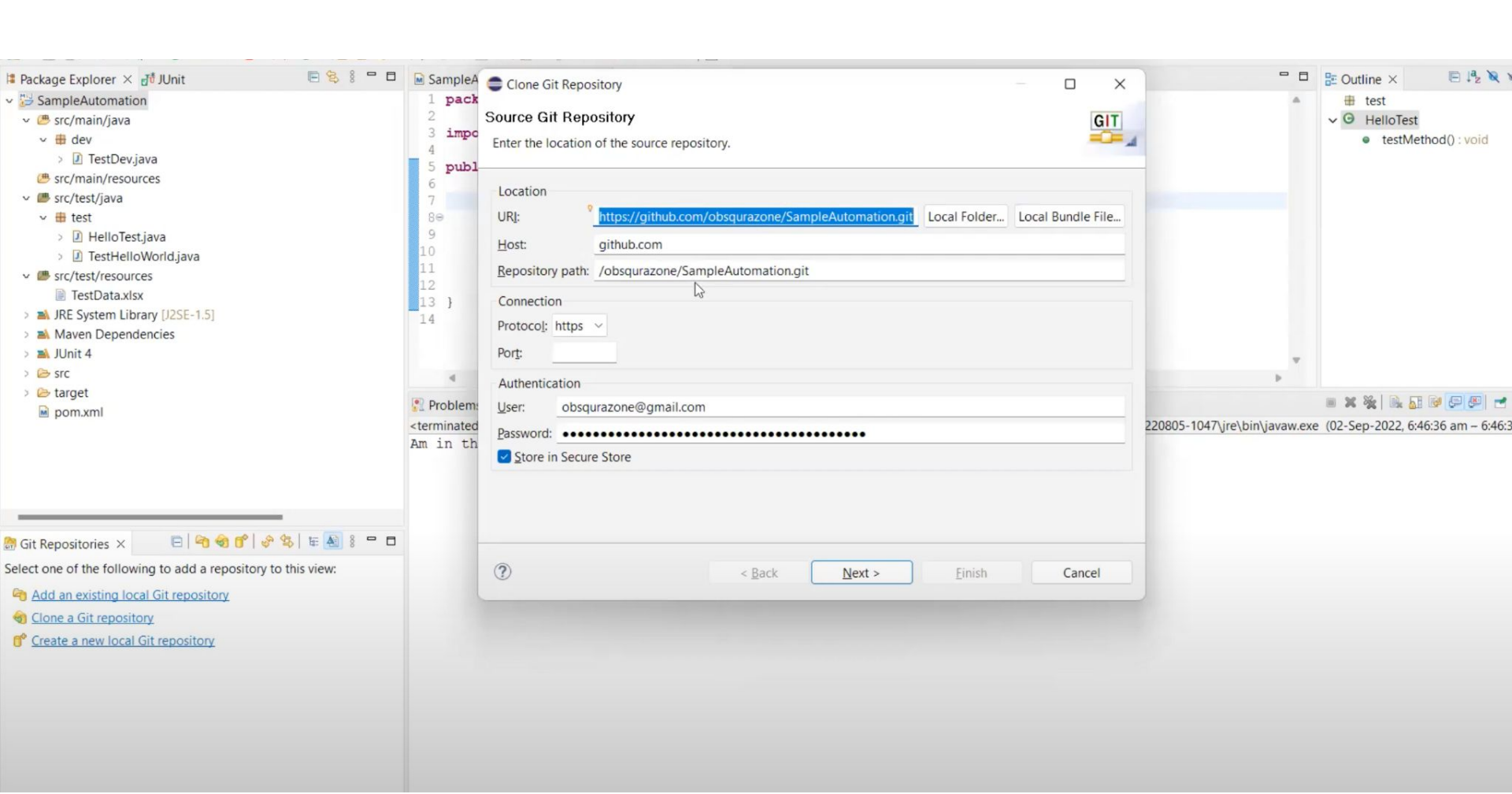
Build Lifecycle Basics

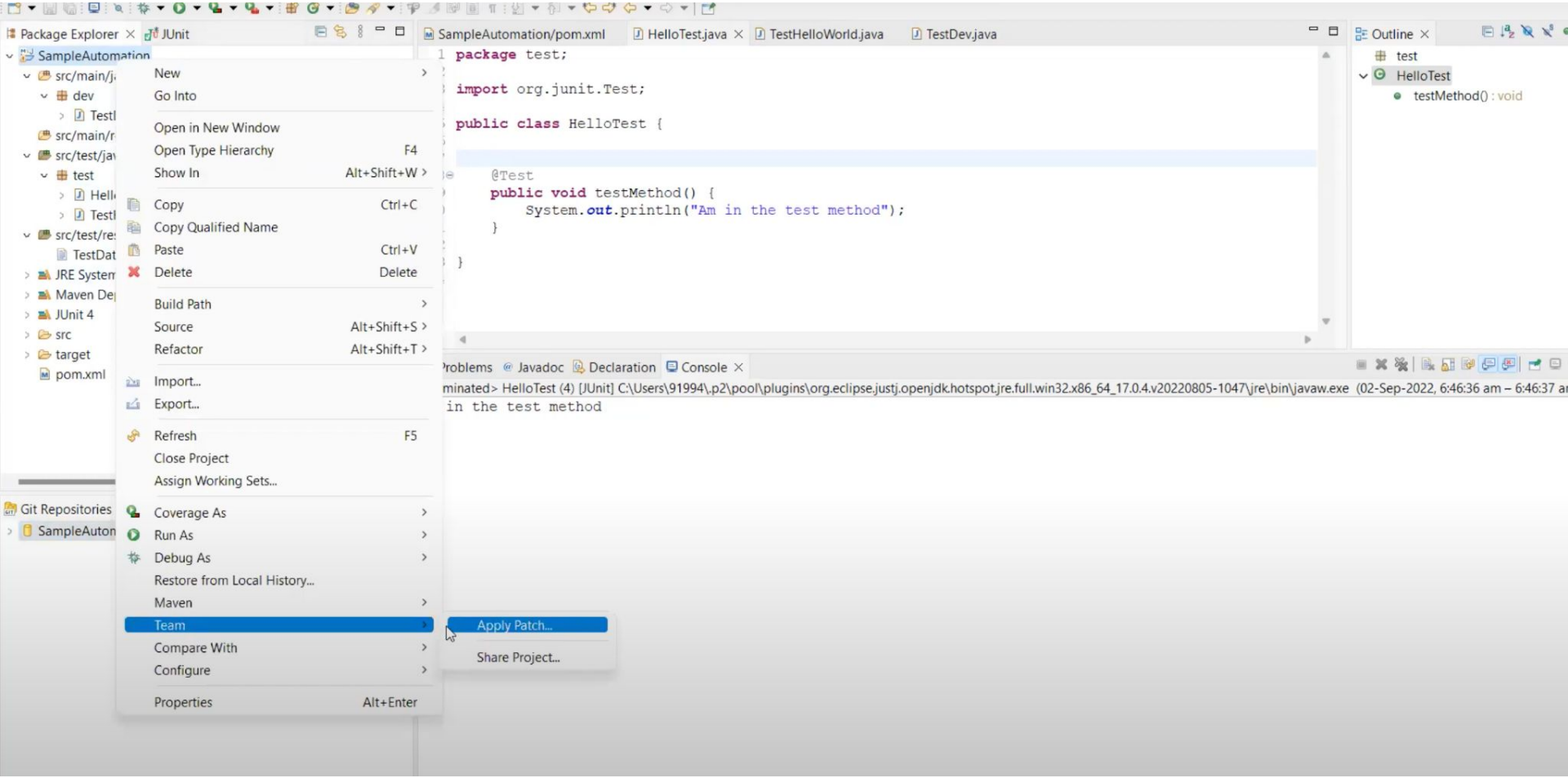
- There are three built-in build lifecycles: default, clean and site.
- The **default** lifecycle handles your project deployment, the **clean** lifecycle handles project cleaning, while the **site** lifecycle handles the creation of your project's site documentation.
- Phases of default lifecycle
  - ✓ ***validate*** *- validate the project is correct and all necessary information is available*
  - ✓ ***compile*** *- compile the source code of the project*
  - ✓ ***test*** *- test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed*
  - ✓ ***package*** *- take the compiled code and package it in its distributable format, such as a JAR.*
  - ✓ ***verify*** *- run any checks on results of integration tests to ensure quality criteria are met*
  - ✓ ***install*** *- install the package into the local repository, for use as a dependency in other projects locally*
  - ✓ ***deploy*** *- done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.*

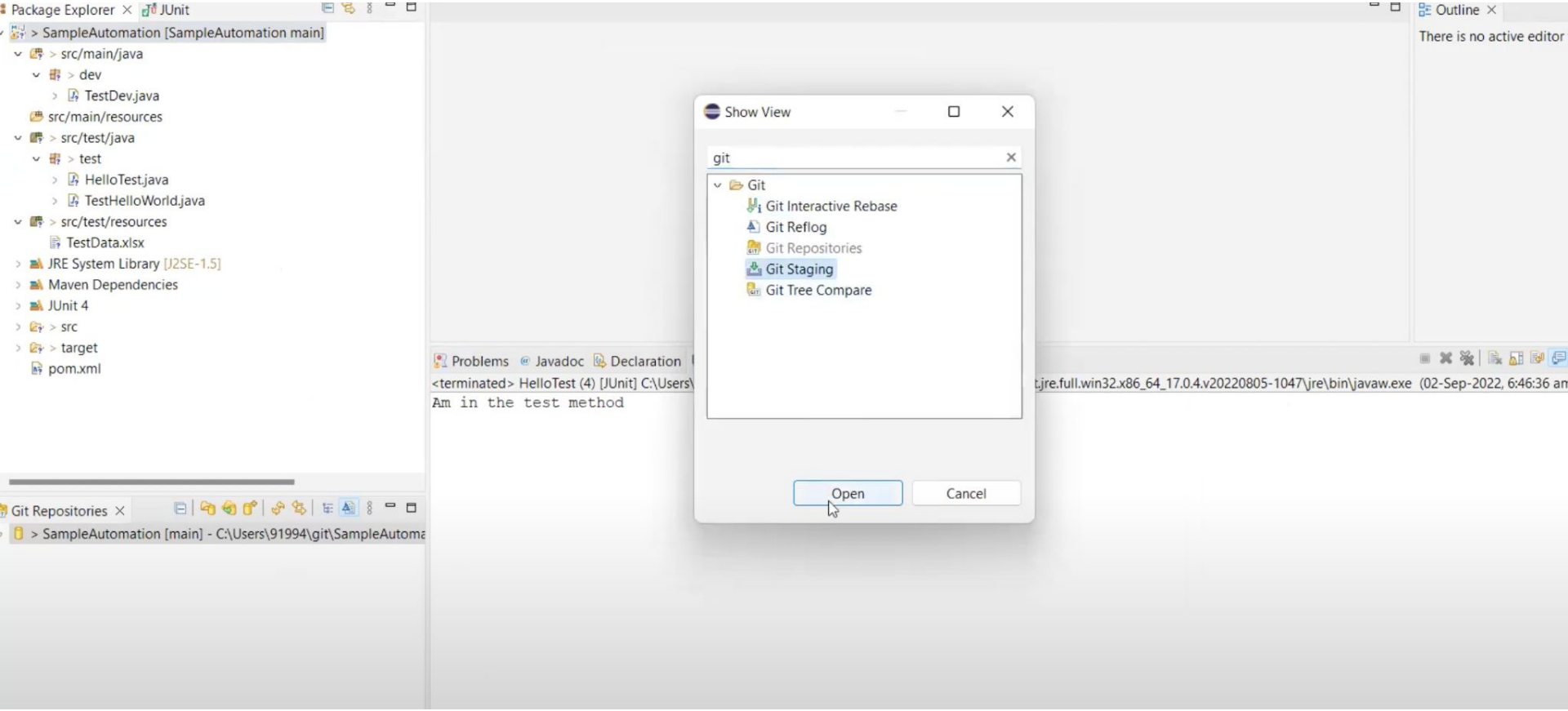
Pom.xml?

- POM stands for Project Object Model
- It is fundamental unit of work in Maven
- It is an XML file that resides in the base directory of the project as pom.xml
- POM also contains the goals and plugins. While executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, and then executes the goal. Some of the configuration that can be specified in the POM are following

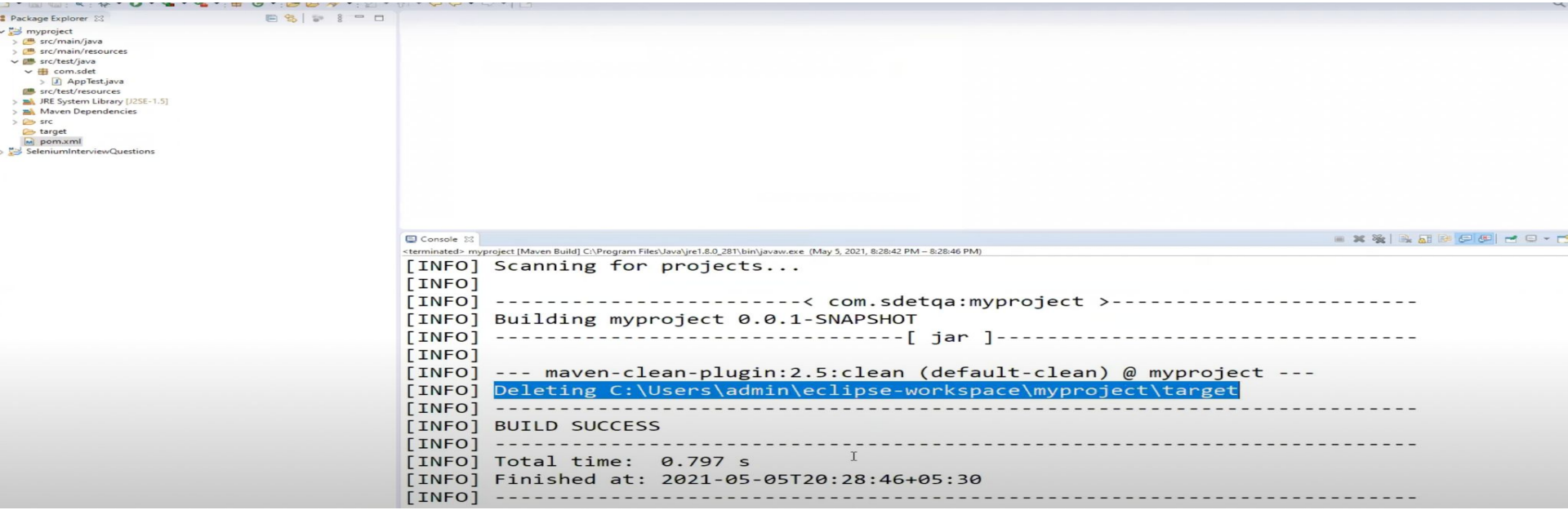
  - project dependencies
  - plugins
  - goals
  - build profiles
  - project version
  - developers
  - mailing list


Package Explorer
JUnit
SampleAutomation
src/main/java
dev
TestDev.java
src/main/resources
src/test/java
test
HelloTest.java
TestHelloWorld.java
src/test/resources
TestData.xlsx
JRE System Library [J2SE-1.5]
Maven Dependencies
JUnit 4
src
target
pom.xml
Git Repositories
Select one of the following to add a repository to this view:
Add an existing local Git repository
Clone a Git repository
Create a new local Git repository
Clone Git Repository
Source Git Repository
Enter the location of the source repository.
GIT
Location
URI:
https://github.com/obsqurazone/SampleAutomation.git
Local Folder...
Local Bundle File...
Host:
github.com
Repository path:
/obsqurazone/SampleAutomation.git
Connection
Protocol:
https
Port:
Authentication
User:
obsqurazone@gmail.com
Password:
Store in Secure Store
< Back
Next >
Finish
Cancel
Outline
test
HelloTest
testMethod() : void
Problems
(02-Sep-2022, 6:46:36 am – 6:46:3


Package Explorer
JUnit
SampleAutomation
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Assign Working Sets...
Coverage As
Run As
Debug As
Restore from Local History...
Maven
Team
Compare With
Configure
Properties Alt+Enter
Apply Patch...
Share Project...
SampleAutomation/pom.xml
HelloTest.java
TestHelloWorld.java
TestDev.java
package test;
import org.junit.Test;
public class HelloTest {
@Test
public void testMethod() {
System.out.println("Am in the test method");
}
}
Outline
test
HelloTest
testMethod() : void
Problems
Javadoc
Declaration
Console
in the test method


Package Explorer
JUnit
> SampleAutomation [SampleAutomation main]
> src/main/java
> dev
TestDev.java
src/main/resources
> src/test/java
> test
HelloTest.java
TestHelloWorld.java
> src/test/resources
TestData.xlsx
JRE System Library [J2SE-1.5]
Maven Dependencies
JUnit 4
> src
> target
pom.xml
Git Repositories
> SampleAutomation [main] - C:\Users\91994\git\SampleAutoma
Show View
git
Git
Git Interactive Rebase
Git Reflog
Git Repositories
Git Staging
Git Tree Compare
Open
Cancel
Outline
There is no active editor
Problems
Javadoc
Declaration
<terminated> HelloTest (4) [JUnit] C:\Users\
t.jre.full.win32.x86_64_17.0.4.v20220805-1047\jre\bin\javaw.exe (02-Sep-2022, 6:46:36 am
Am in the test method

When we execute the command automatically previous command is also executed



https://www.javaguides.net/2018/06/maven-cheat-sheet.html

https://medium.com/@TimvanBaarsen/maven-cheat-sheet-45942d8c0b86

https://www.bogotobogo.com/Java/tutorials/Spring-Boot/Maven-mvn-command-cheat-sheet.php

https://chenweixiang.github.io/docs/All_Cheat_Sheets_v2.pdf

PUSH

GitHub

BUILD

Maven

TEST

Maven

Se

DEPLOY

Apache TOMCAT

MONITOR

Jenkins