**Maven Document:**

**Introduction:** 1. Maven is free and open software given by Apache Organization.

2. Maven software developed by using java language and it is used to build process automation for java projects.

3. Maven is called as java build tool.

**Definition:** 1. Maven is a build automation and project management tool for only java projects and streaming the development lifecycle.

2. It uses a declarative configuration approach through a project object model.

**Why We Need Maven?** : 1. Project Initialization: it mean’s create structure of the project folder.

2. Dependency Management: it means download automatically all required libraries and jar from repository.

3. Build Automation: Maven automate the build process, handling the task such as compile, testing, packaging, based on the configuration in Project Object Model.

4. Lifecycle Management: maven define set of build lifecycle ex. (Clean, validate, compile, test, package) that can be executed sequential.

5. Artifact Publishing: Maven can publish build artifacts ( ex. Jar or War) to repositories, allowing for easy to sharing distribution of project components across development teams.

**Maven Terminology:**

**1. Archetype:**  The archetype is project template that define the structure and initial configuration for a new project. We have two types that is

**1. Maven-archetype-quickstart:** It represents java standalone application **2.maven-archetype-webapp:** it represents java web application.

**2. Group Id:** Group Id is a naming convention that helps uniquely identify and organize projects with in specific domain or organization.

**3. Artifact Id:** Artifact id is represent name of the project that helps unique identifier for a project or module.

**4. Version:**  Version is element in the project object model specifies the version of the project.

We have two type

**1.** **Snapshot:** It represent under development

**2. Release:** It represent released to customer.

**5. Packaging:**  Package is used to the process of converting the compile source code and resource into distributable format, such as a JAR (Java Archive) or WAR (Web Archive) or POM (Project Object Model).

**6. Dependencies:**  Dependencies is used to external libraries or modules that help to load or download required jar or libraries, such as Junit, spring, Hibernate, Spring Securities etc.

**7. Repository:** Repository is a storage of dependencies and artifacts.

We have three type of repository, such as

**1. Central Repository:** its public, anyone can access. If we want to download dependencies then you have download from apache.

**2. Remote Repository:** its private, anybody can’t access, need permeation. If we join any company then we need download from remote only. That uses for third party server like **Nexus or JFrog.**

**Configuring Remote Repository:**

**<repositories>**

**<repository>**

**<id>id</id>**

**<url>jfrong-repo-url/</url>**

**</repository>**

**</repositories>**

**3. Local Repository:**  it represent dependencies in our system only. If we want setup in local IDE then we need to load from local repo.

**8. Goals:** Goals is used to perform build lifecycle that can be executed several project related actions, such as compilation, testing or packaging. We have different types

**1. Clean:** delete byte code in target

**2. Compile:** Compile Source code

**3. Test:** Execute Junit

**4. Package:** Packaging as JAR or WAR.

**9. Scope:**  Scope are represent when it include maven dependency in the class path then you have to use. We have several scopes are there.

**1. Compile:** it could be included dependency at compile time.

**2. Provided:** if we don’t want include at compile time, it required provided container at runtime like tomcat.

**3. Runtime:** It could be included dependency at runtime.

**4. Test:** it could be dependency included at test.

**5. System:** All files are loaded from system not maven.

**6. Import:** when we want require Multi model project that time it help.

**10. Transitive Dependency Management:**  Transitive dependency management means when we are downloading dependency then automatically, if anything having child dependency that are also downloaded.

**11. Dependency Exclusive:** Exclusive is used to remove un-wanted child dependencies from build path

Ex: When we add 'spring-context' we are getting other dependencies also like 'spring-core', 'spring-beans', 'spring-aop' etc..

-> I want to use 'spring-context' but i don't want 'spring-aop' in build path then we can exclude 'spring-aop' from 'spring-context' like below

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.23</version>

<exclusions>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

</exclusion>

</exclusions>

</dependency>

**How to Use Maven:** 1. Maven can use in two ways.

1. CLI (Command Line)
2. IDE (Eclipse, STS).

**Maven Installation:**

-> Download and install java software

-> When we install java we will below 2 things:

a) JDK (Java Development Kit)

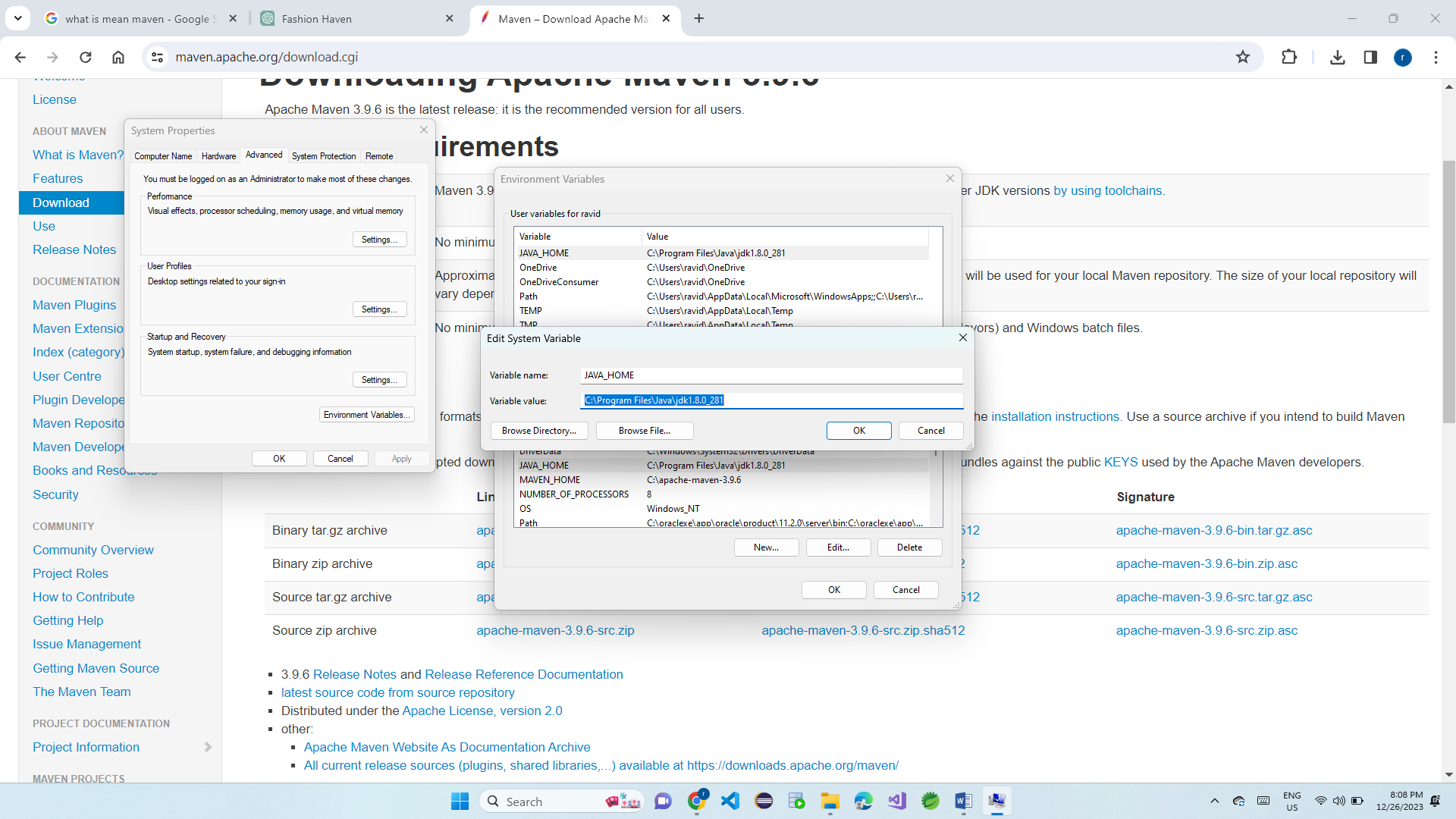
b) JRE (Java Runtime Environment).

-> JDK contains set of tools to develop java programs.

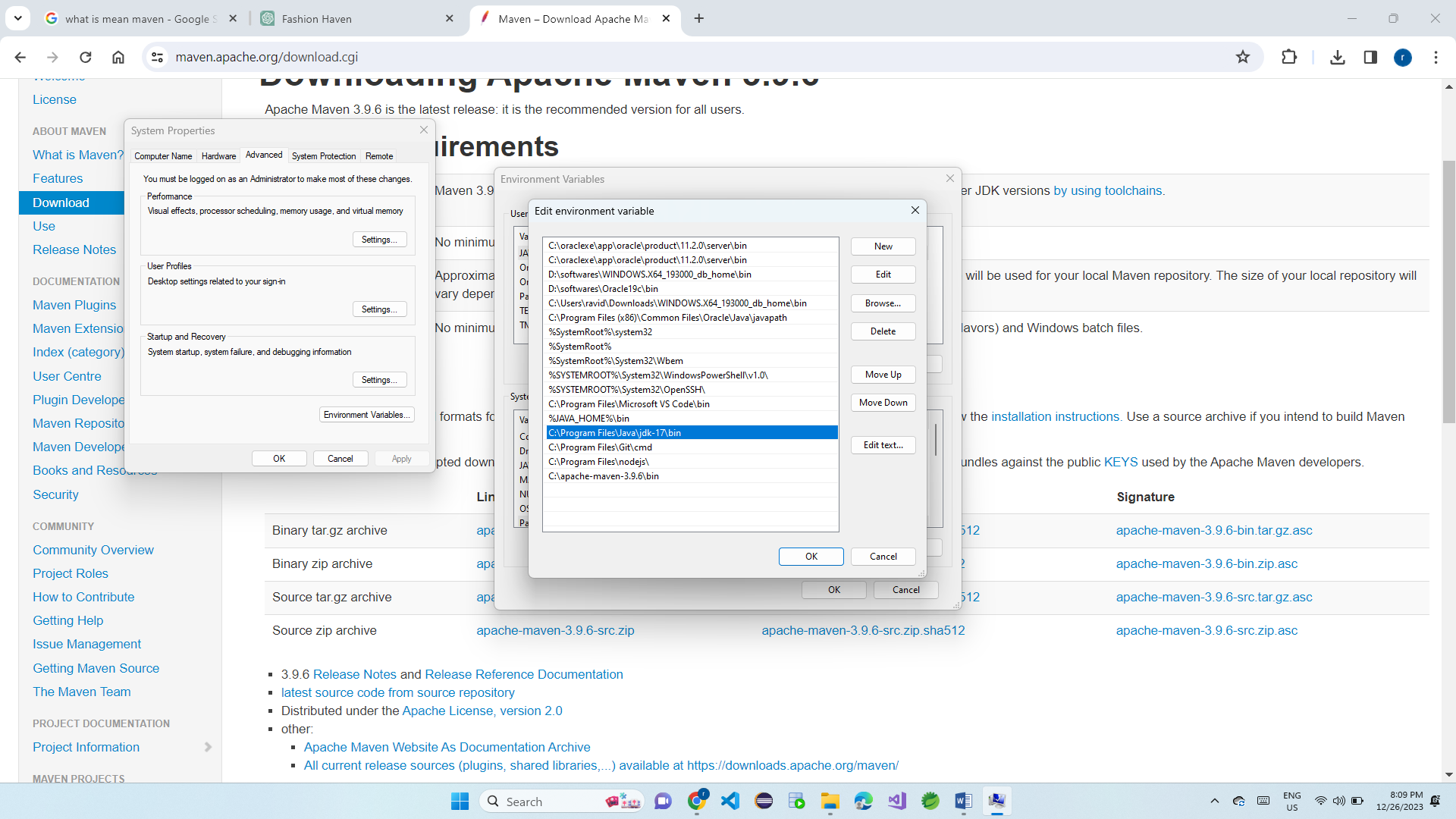
->JRE contains platform/environment which is used to run java programs.

-> Java Home Setup:

Open edit the system environment -> click on environment variable -> click on new in system environment -> give name and direction of Java



-> Java Path Setup: Open edit the system environment -> click on environment variable -> click on path in system environment ->click on new -> give direction of java path up to bin.

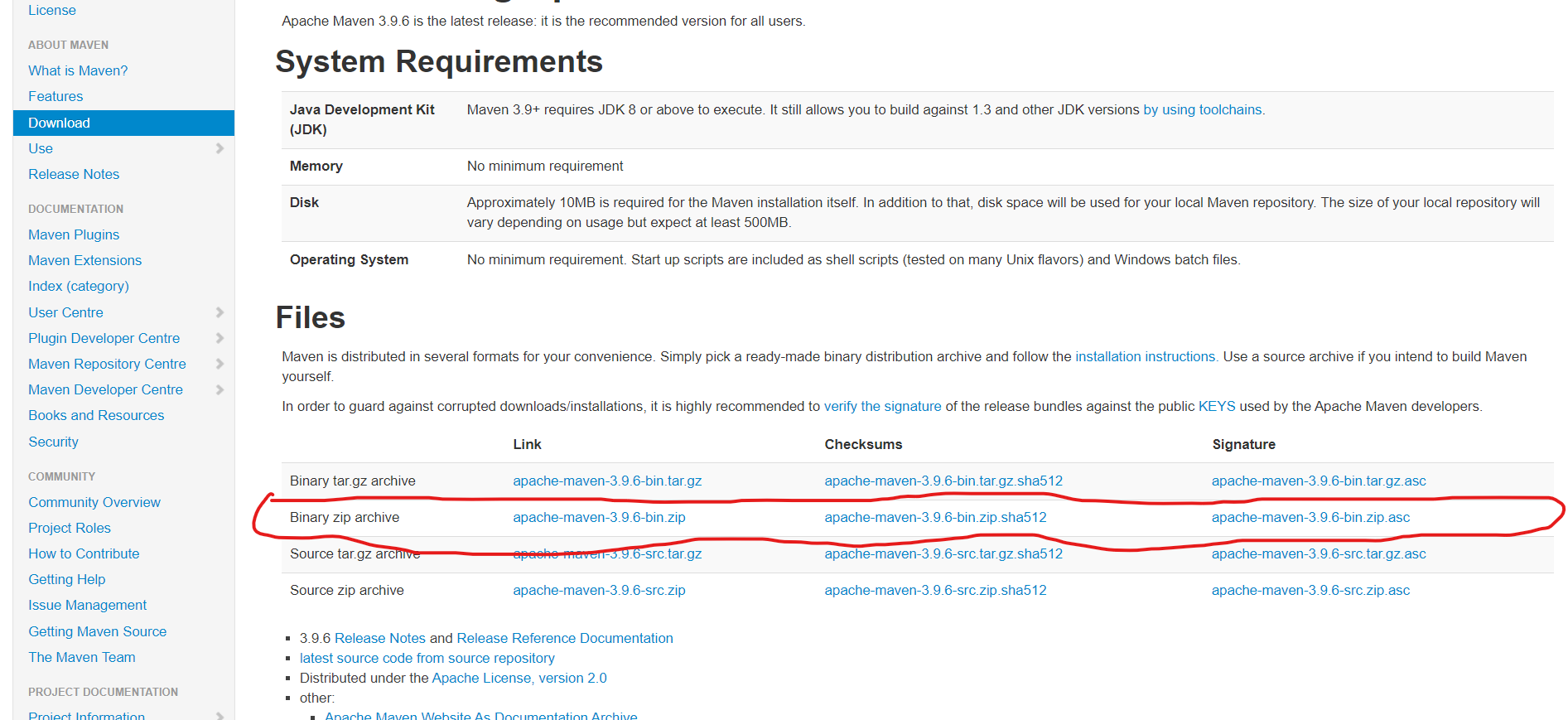


-> Verify java installation by executing below command in “command prompt”

-> Java –version

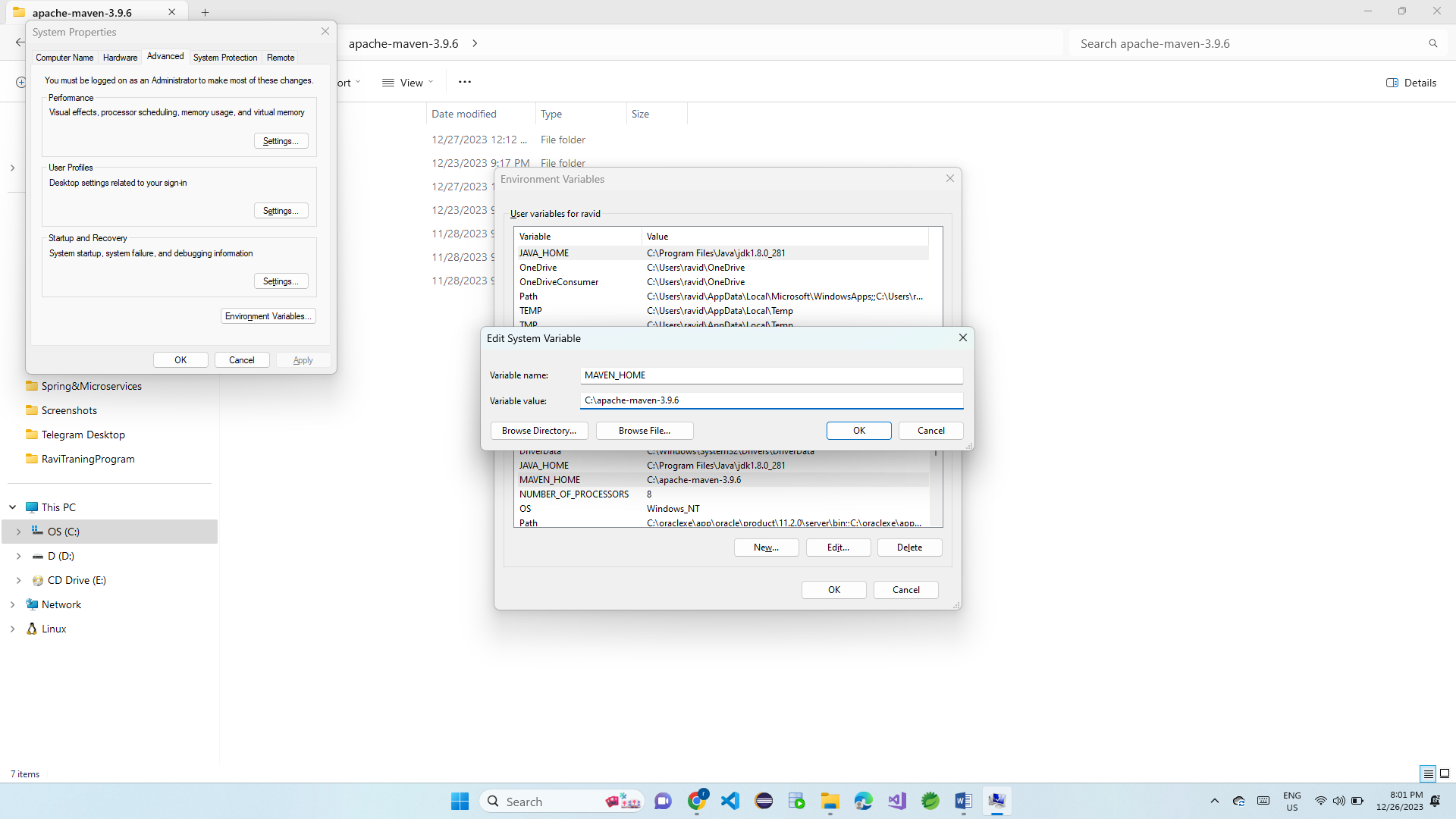
-> Download apache maven

Open browser -> type apache maven download for windows or Linux-> or click on this link <https://maven.apache.org/download.cgi>



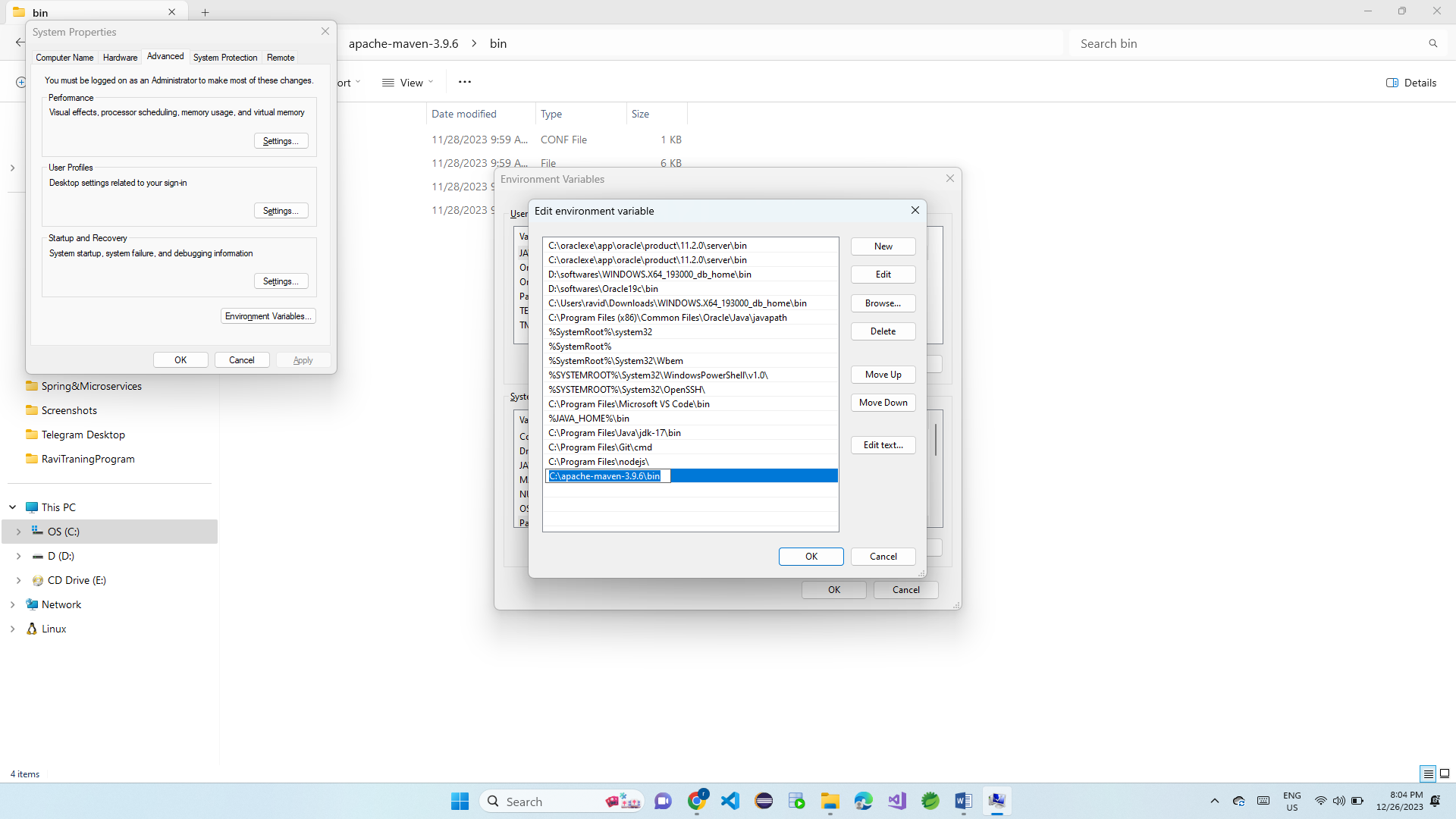
2. Maven Home Setup.

Open edit the system environment -> click on environment variable -> click on new in system environment -> give name and direction of maven



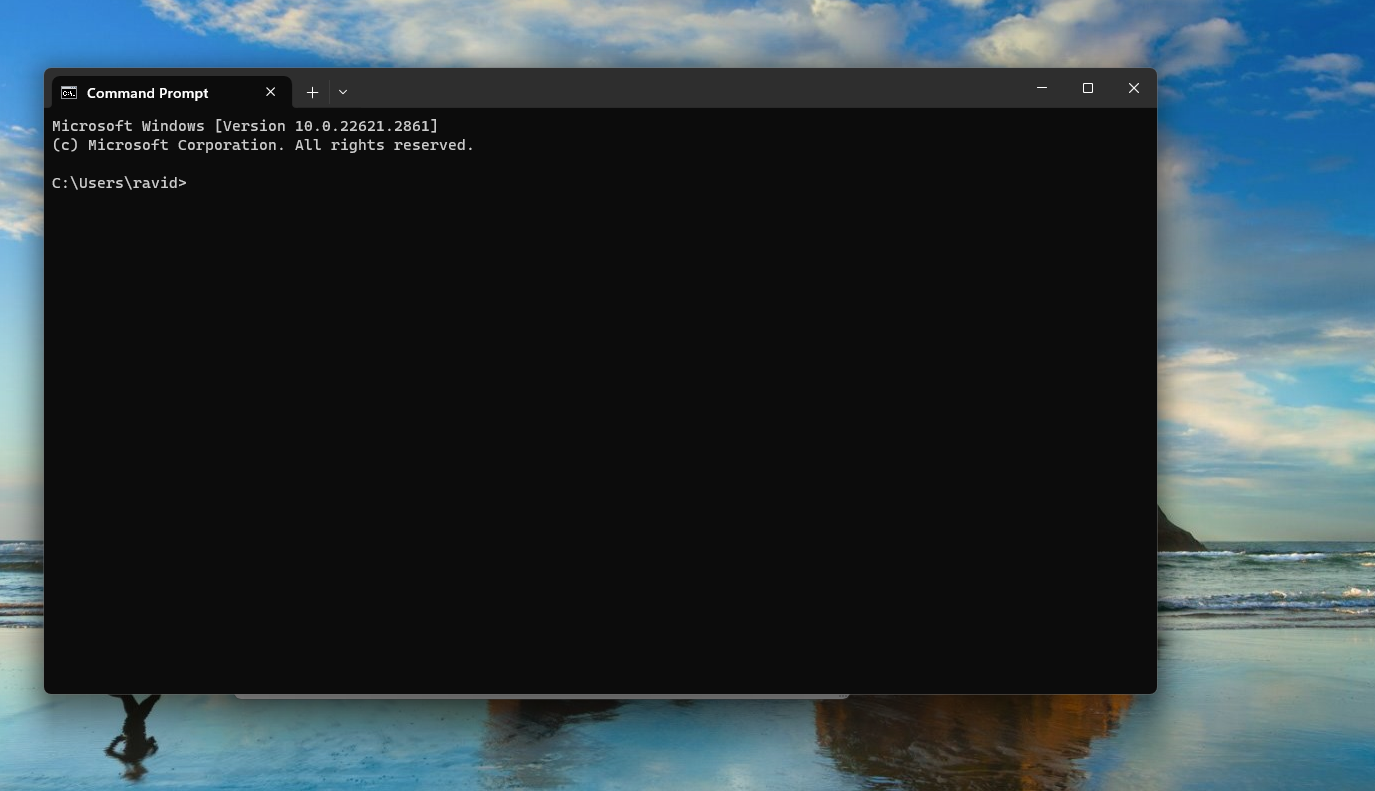
-> Click on OK.

3. Maven Path Setup: Open edit the system environment -> click on environment variable -> click on path in system environment ->click on new -> give direction of maven path up to bin.

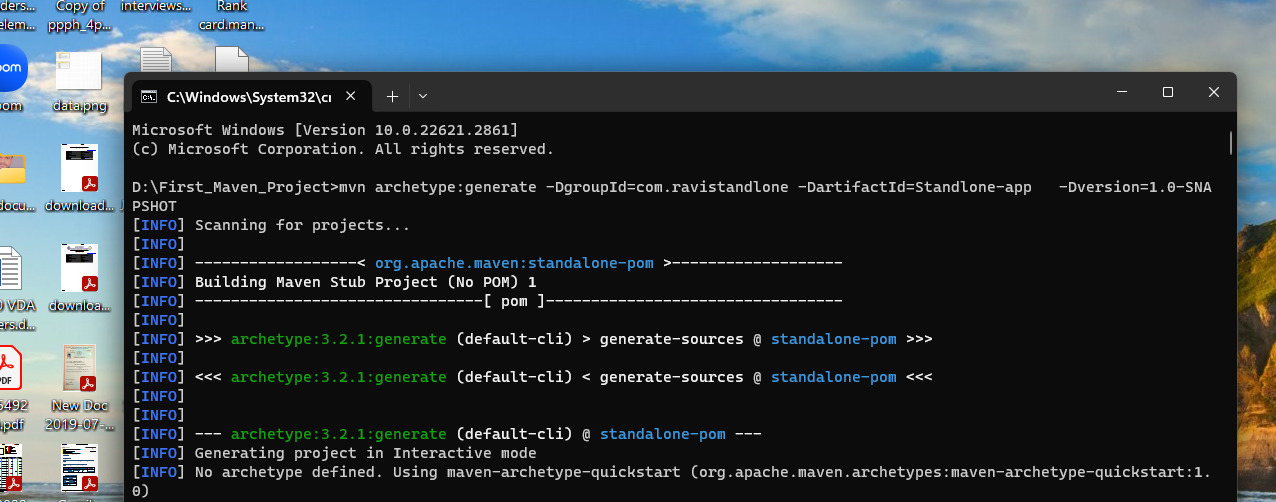


**MAVEN INTSALL PROJECT THROUGH COMMAND LINE:**

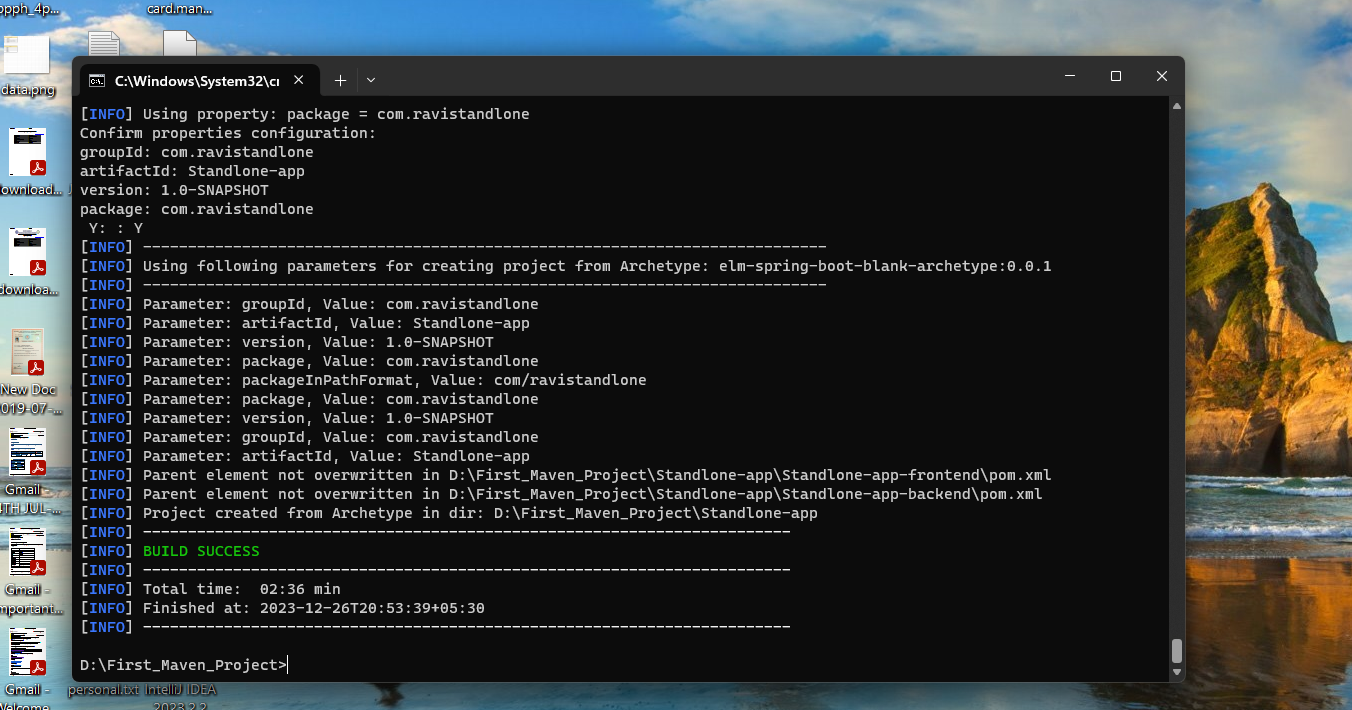
**Maven Install:** 1. Create one folder and Open command line in your system.



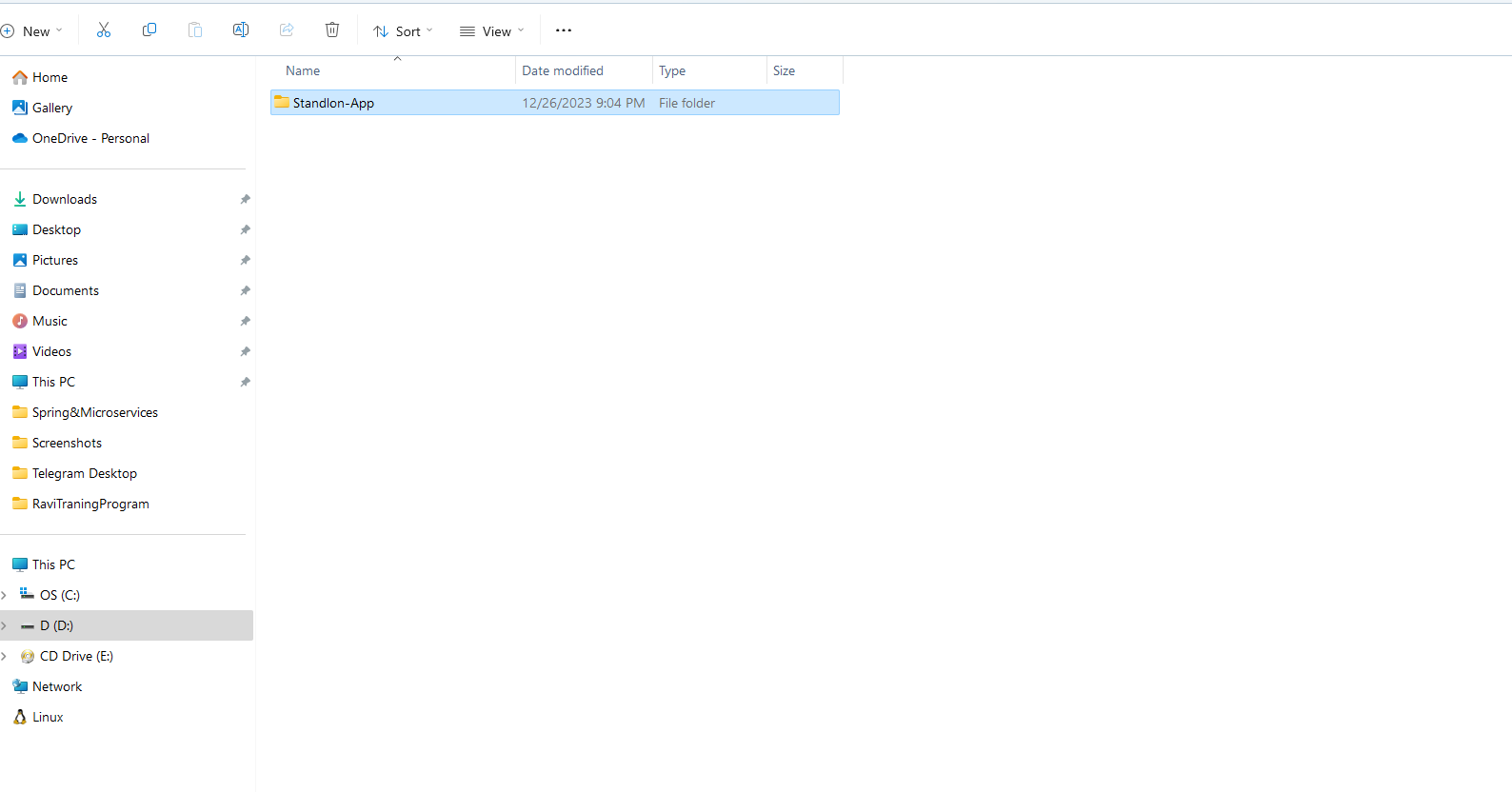
-> type this command: mvn archetype: generate -DgroupId=com.ravistandlone -DartifactId=Standlone-app -Dversion=1.0-SNAPSHOT



-> Once done installation you will see success build like below diagram.



-> Once done installation then it can generate project like below diagram.

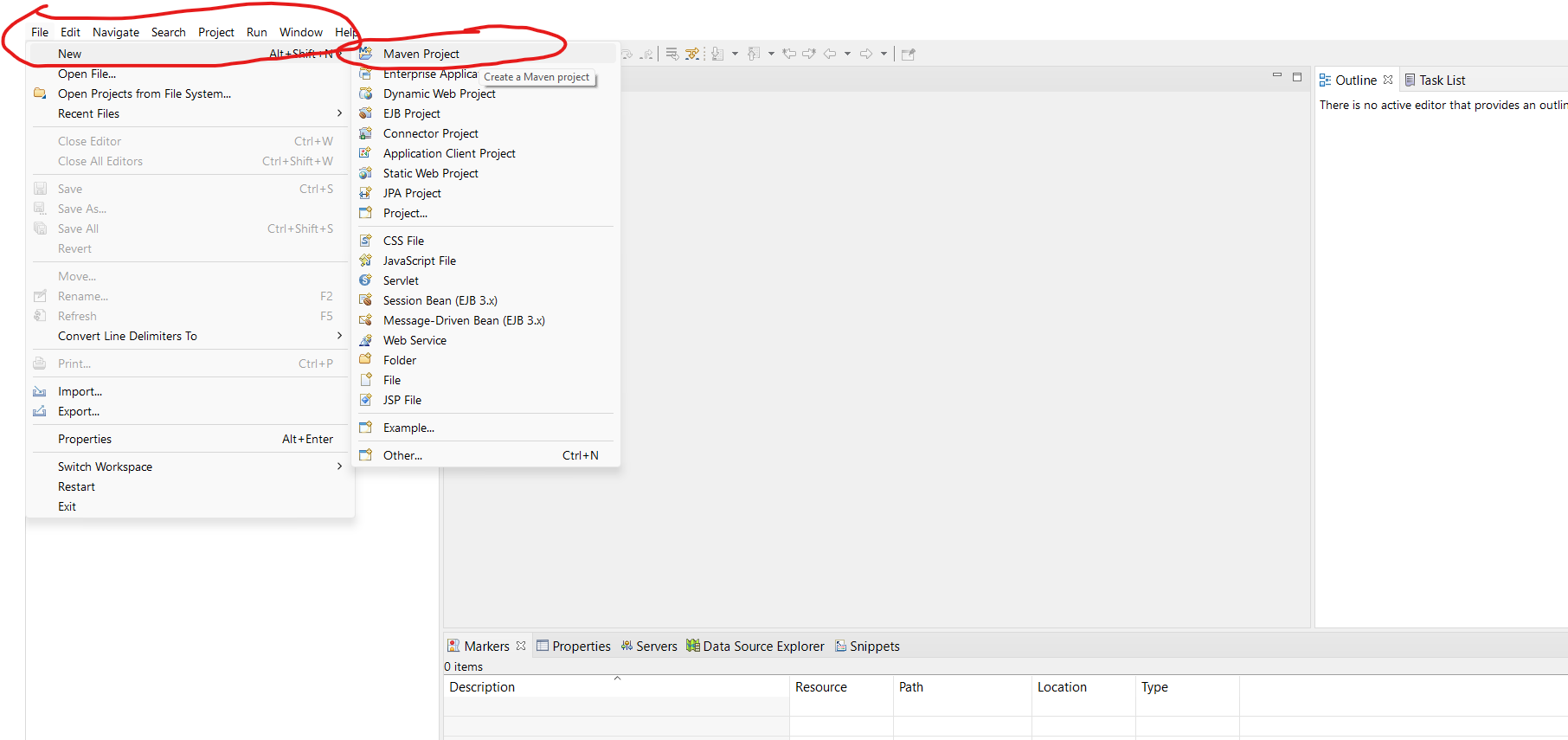


**Generally used Maven Commends:**

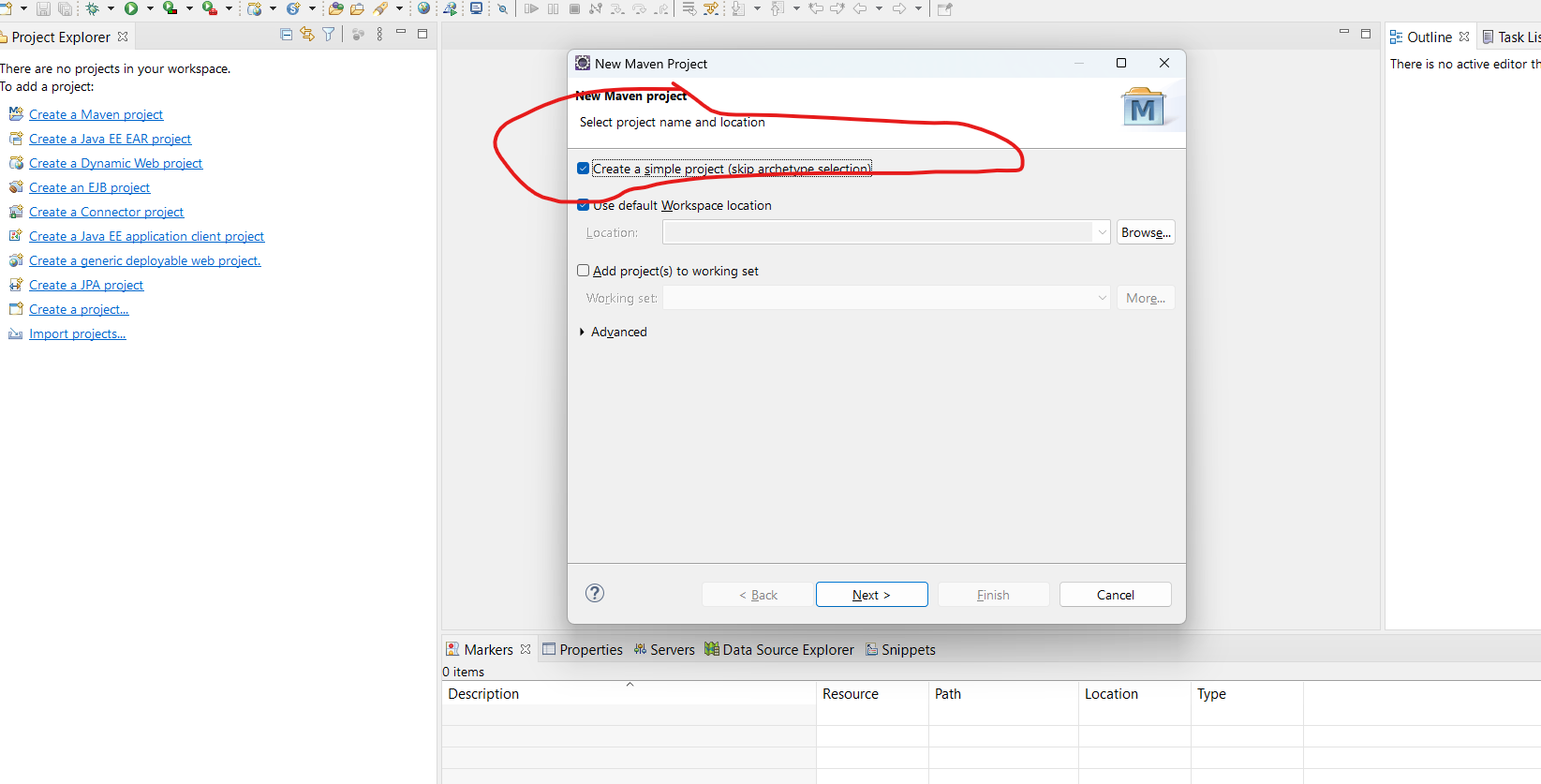
1. MVN Clean: delete the byte code in target folder
2. MVN Compile: compile the source code and converted to byte code
3. MVN Package: generated the package like JAR or WAR or POM
4. MVN Test: Test the Junit test cases
5. MVN Clean Compile: delete the byte code and compile the source code
6. MVN clean compile test: delete byte code and compile the source code and test the Junit test cases.
7. MVN clean package: delete byte code in target folder and compile the source code and test the Junit test cases and generate package either JAR or WAR, so mostly we can use this command.

**MAVEN CREATE PROJECT THROUGH IDE:**

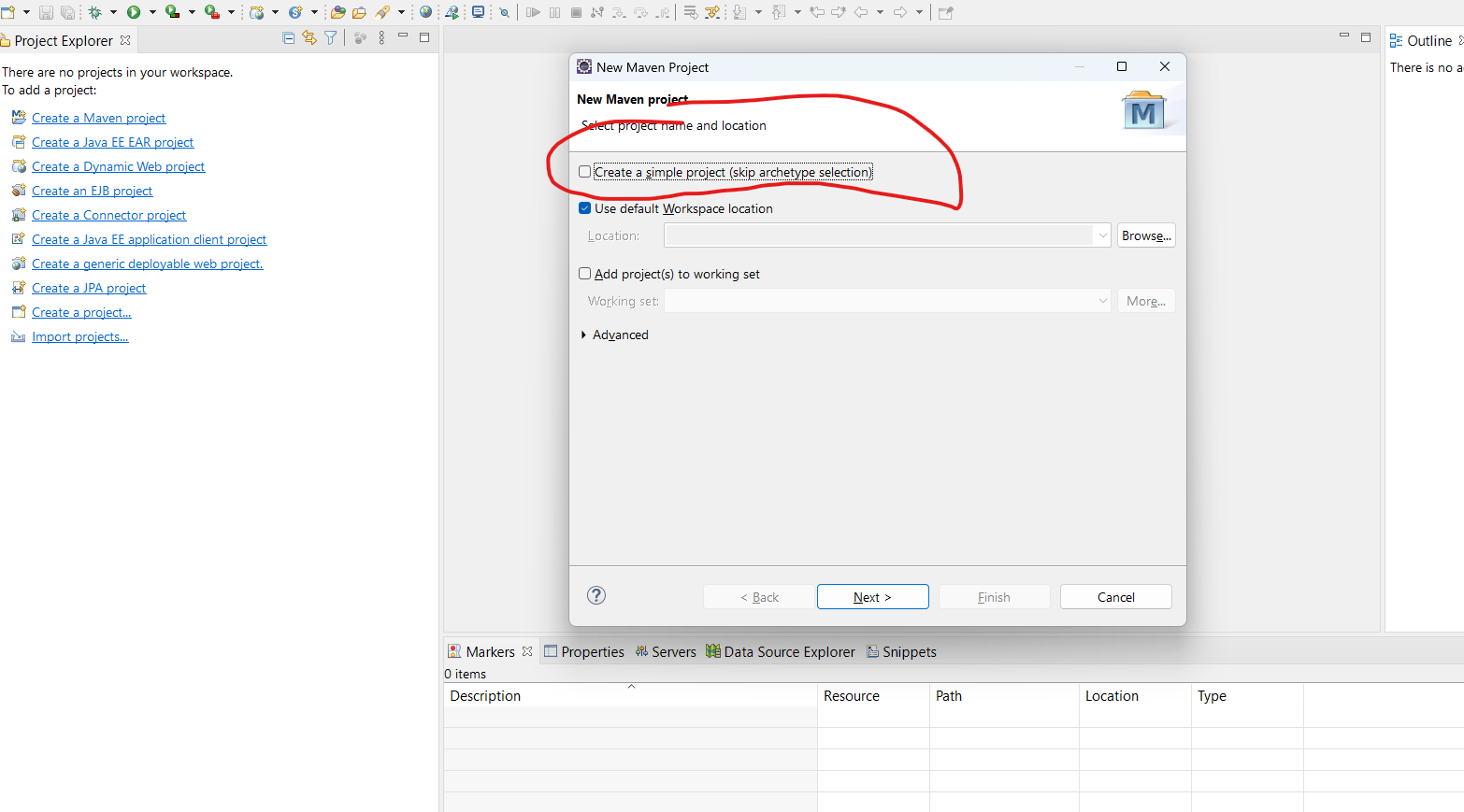
1. Open IDE and click on new -> select maven project->



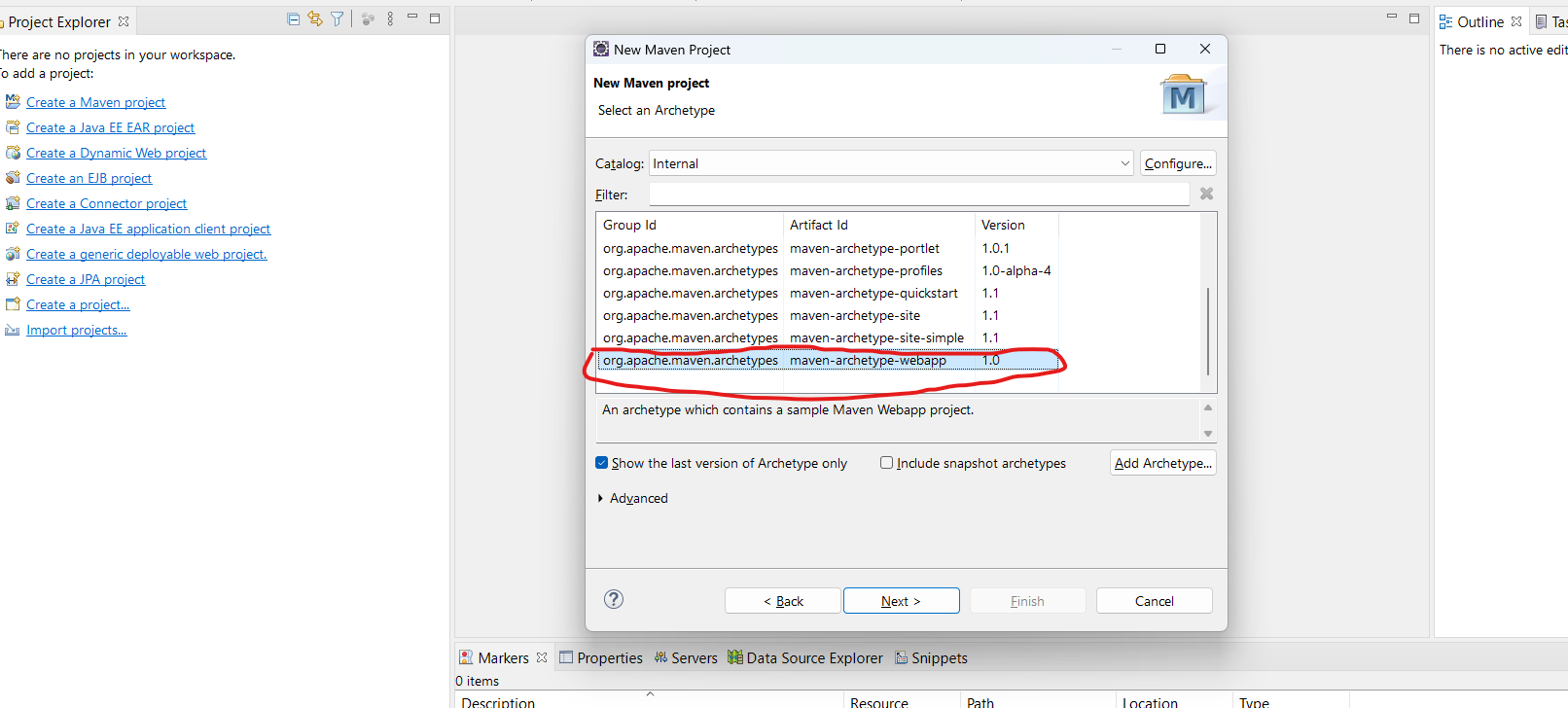
Then select archetype if it is Standlone project or Web project. If it is Standlone then you have to select simple project like below diagram.



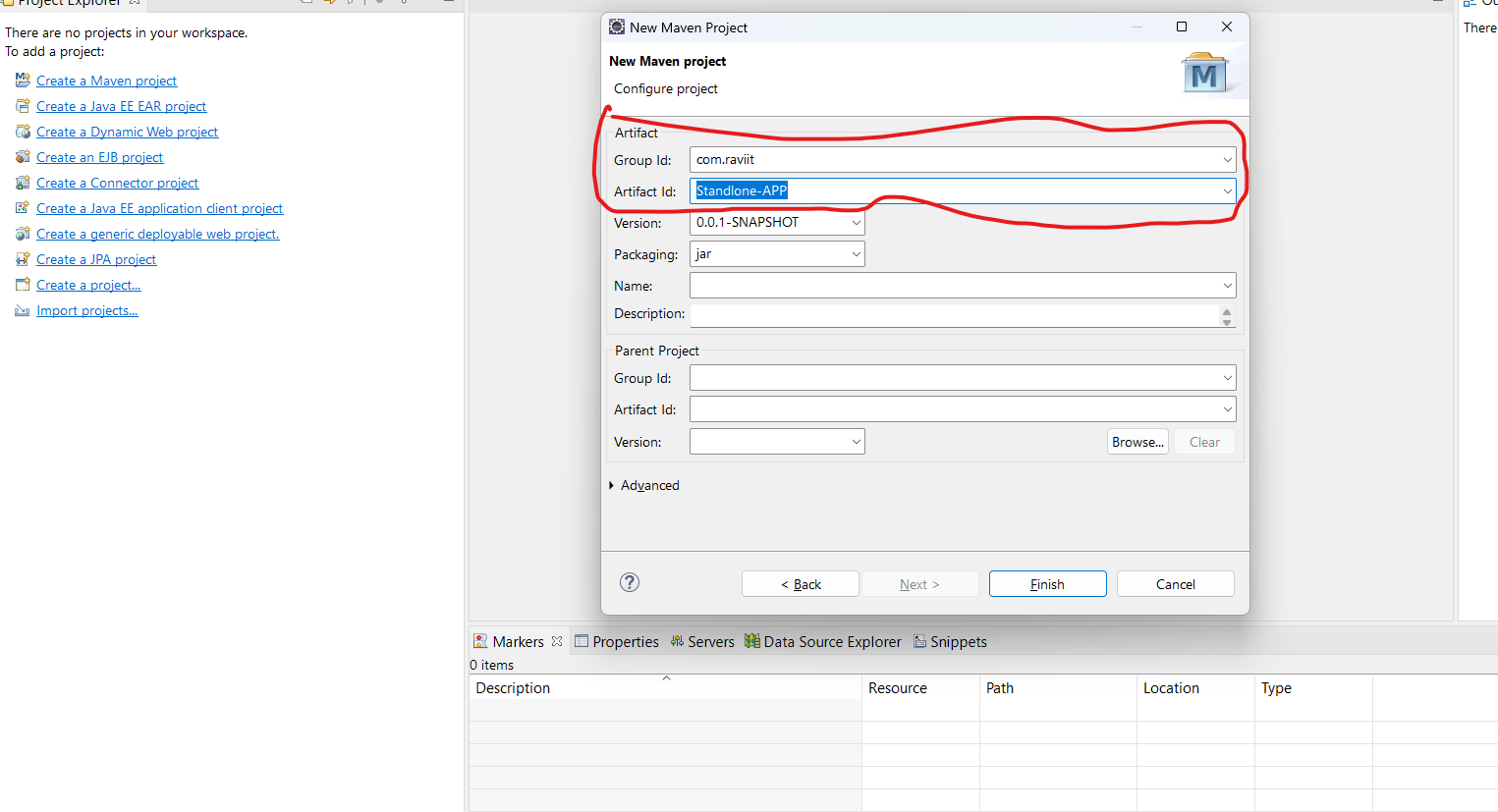
If it is web based project then you have to disable create a simple project and the select appropriate archetype.



Select archetype like web app.



Then click one next and type group id and artifact id like follow below diagram.. I am using only Standlone application.



And then click on finish button.

**Maven Multi Model Project:** 1. If we have several modules in our application then we create as parent and all related modules will be child. If you do like then if add any dependency or compile or test in parent then it’s automatically build for child modules or project.

2. It is easy to management in our application.

**Creating Maven Multi module project in IDE:**

-> create Maven Project with packaging type as ‘pom’

-> Right click on the project ->New -> select others ->search for Maven Module-> Create the Module

-> After Module got created, check parent project pom.xml and child project pom.xml.

**Parent Project pom.xml looks like below:**

<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>com.flipkart</groupId>

<artifactId>Flipkart\_App</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>pom</packaging>

<modules>

<module>ADMIN</module>

<module>REPORTS</module>

</modules>

</project>

**Child Project pom.xml looks like below:**

<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>

<parent>

<groupId>com.flipkart</groupId>

<artifactId>Flipkart\_App</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<artifactId>REPORTS</artifactId>

</project>

-> Right click on Parent Project -> Run as -> Maven Build -> Enter Goals and Execute.