

# Maven

**Maven** is an open-source project management tool that is based on the concept of a project object model (POM). Maven can manage a project's build, reporting and documentation from a central piece of information. Maven from Apache Organization and an opensource tool.

Maven is primarily used for Java projects, but it can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. **Standardization:** Maven provides a standard way to build and manage projects, which makes it easier to share code and collaborate with other developers.

If you are working on a Java project, I highly recommend using Maven. It is a powerful tool that can make your life as a developer much easier.

### Understanding the problem without Maven:

There are many problems that we face during the project development. They are discussed below:

**Adding set of Jars in each project:** In case of struts, spring, hibernate frameworks, we need to add set of jar files in each project. It must include all the dependencies of jars also.

**Creating the right project structure:** We must create the right project structure in servlet, struts etc, otherwise it will not be executed.

**Building and Deploying the project:** We must have to build and deploy the project so that it may work.

Maven is a popular tool because it provides a number of benefits, including:

- **Project Object Model (POM):** The POM is a central file that contains information about the project, such as its dependencies, build configuration, and reporting information.
- **Dependency Management:** Maven can automatically download and manage the dependencies of your project. This saves you time and effort, and it ensures that your project is always using the latest versions of its dependencies.
- **Build Automation:** Maven can automate the build process for your project. This includes tasks such as compiling the code, running unit tests, and packaging the project.
- **Reporting:** Maven can generate reports about your project, such as code coverage reports and unit test reports. This can help you to track the progress of your project and to identify areas that need improvement.
- **Documentation:** Maven can generate documentation for your project, such as Javadocs and project website. This can help you to share your project with others and to make it easier for them to understand how it works.

## What is Build Tool?

A build tool takes care of everything for building a process. It does following:

- Compiles source code
- Generates documentation from source code
- Packages compiled code into JAR or WAR files

## Project Creation With Maven in Eclipse:

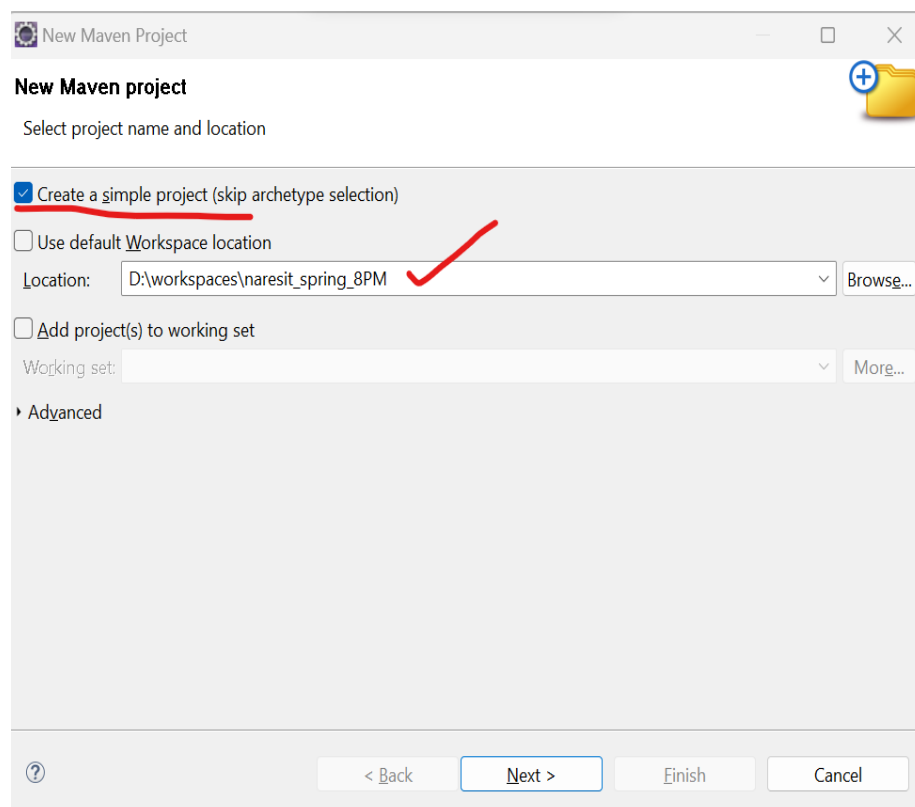
Eclipse integrated with Maven by default i.e. Not required any extra Maven setup inside Eclipse.

1. Open Eclipse
2. File -> New -> Maven Project

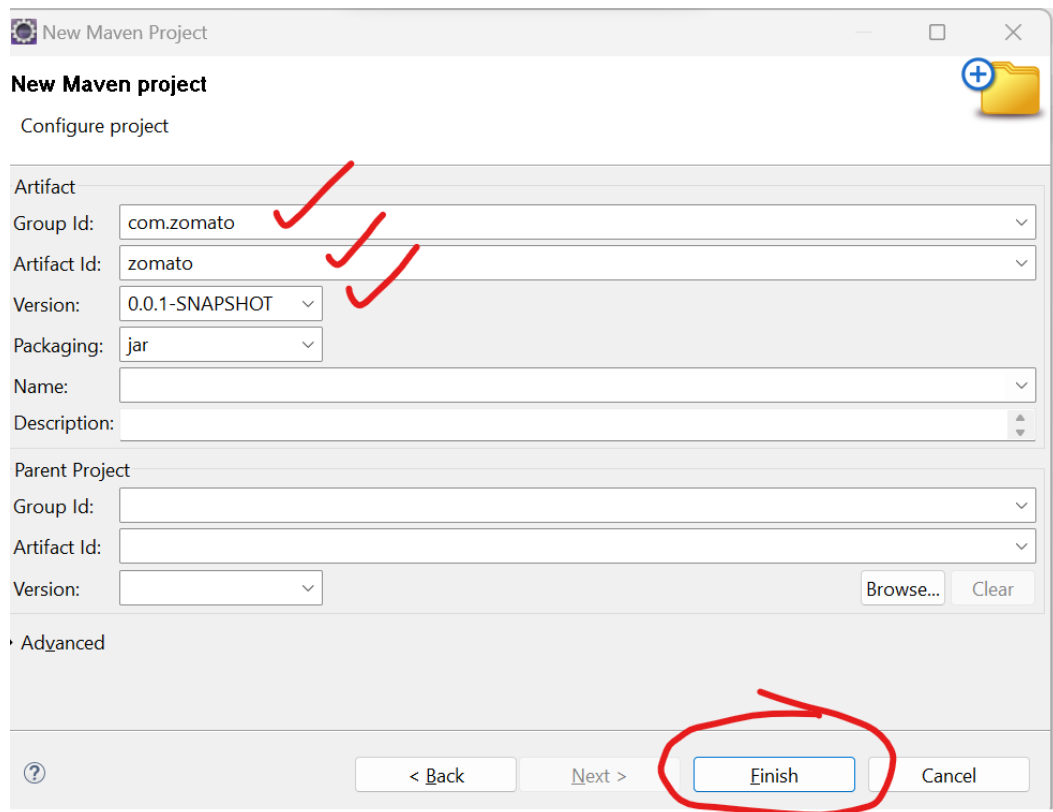
Select Create a Simple Project.

Select your workspace where project should be created.

Click on Next.



3. Now enter details of Project information and click on finish.



**New Maven project**

Configure project

Artifact

Group Id: com.zomato ✓

Artifact Id: zomato ✓

Version: 0.0.1-SNAPSHOT ✓

Packaging: jar

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:

Browse... Clear

Advanced

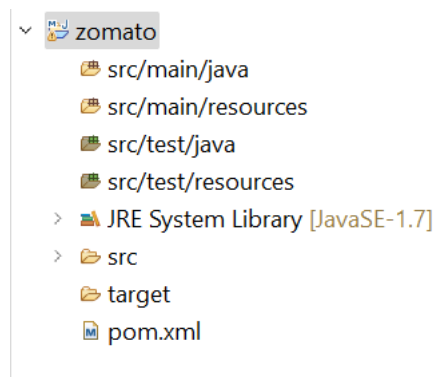
< Back Next > **Finish** Cancel

**Group Id:** This is an Id of project's group. This is generally unique amongst an organization of a project. For example, a project group com.zomato

**Artifact Id** This is an Id of the project. This is generally name of the project.

**Version:** This is the version of the project.

4. Project created with below folder structure.



## Understanding the Project Structure:

**pom.xml:** This is the Project Object Model (POM) file, which defines the project's configuration, dependencies, and other settings.

**src/main/java:** This directory contains the main Java source code of your project.

**src/test/java:** This directory contains the test Java source code.

## How to add Dependencies in Maven Project:

- Open **pom.xml** file

```
<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
    https://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.zomato</groupId>
  <artifactId>zomato</artifactId>
  <version>0.0.1-SNAPSHOT</version>

</project>
```

- Inside above **pom.xml** file, we are going to add a section called as dependencies. Inside dependencies we are adding individual dependency for every required jar files.

So, here we are not downloading jar files manually and not configuring with our project. Hand overing this to Maven tool.

We can get every jar file dependency from internet.

### Example Dependency:

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>5.3.9</version>
</dependency>
```

**Group Id:** This is an Id of project's group.

**Artifact Id** This is an Id of the project. This is generally name of the jar file.

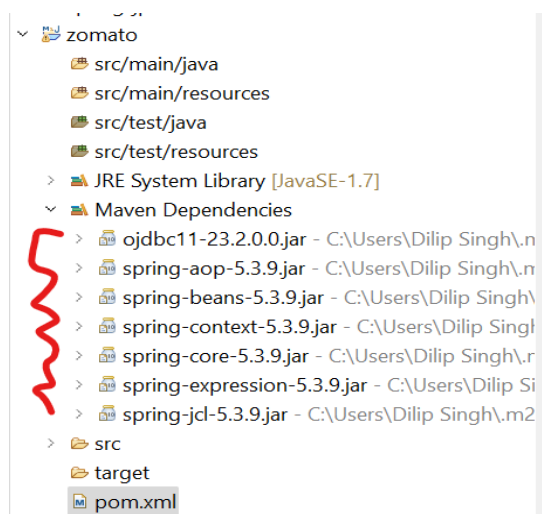
**Version:** This is the version of the jar file.

## Dependencies Declared inside pom.xml file:

```
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.zomato</groupId>
  <artifactId>zomato</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>5.3.9</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.9</version>
    </dependency>
    <dependency>
      <groupId>com.oracle.database.jdbc</groupId>
      <artifactId>ojdbc11</artifactId>
      <version>23.2.0.0</version>
    </dependency>
  </dependencies>
</project>
```

- Now observe under Maven Dependencies Section. All specified jar files and internal dependent jar files are downloaded in project.



Now we can start writing our programming in project.

## Maven Goals:

A Maven goal is a task that can be performed during the build process. Maven goals are defined in the project's POM file.

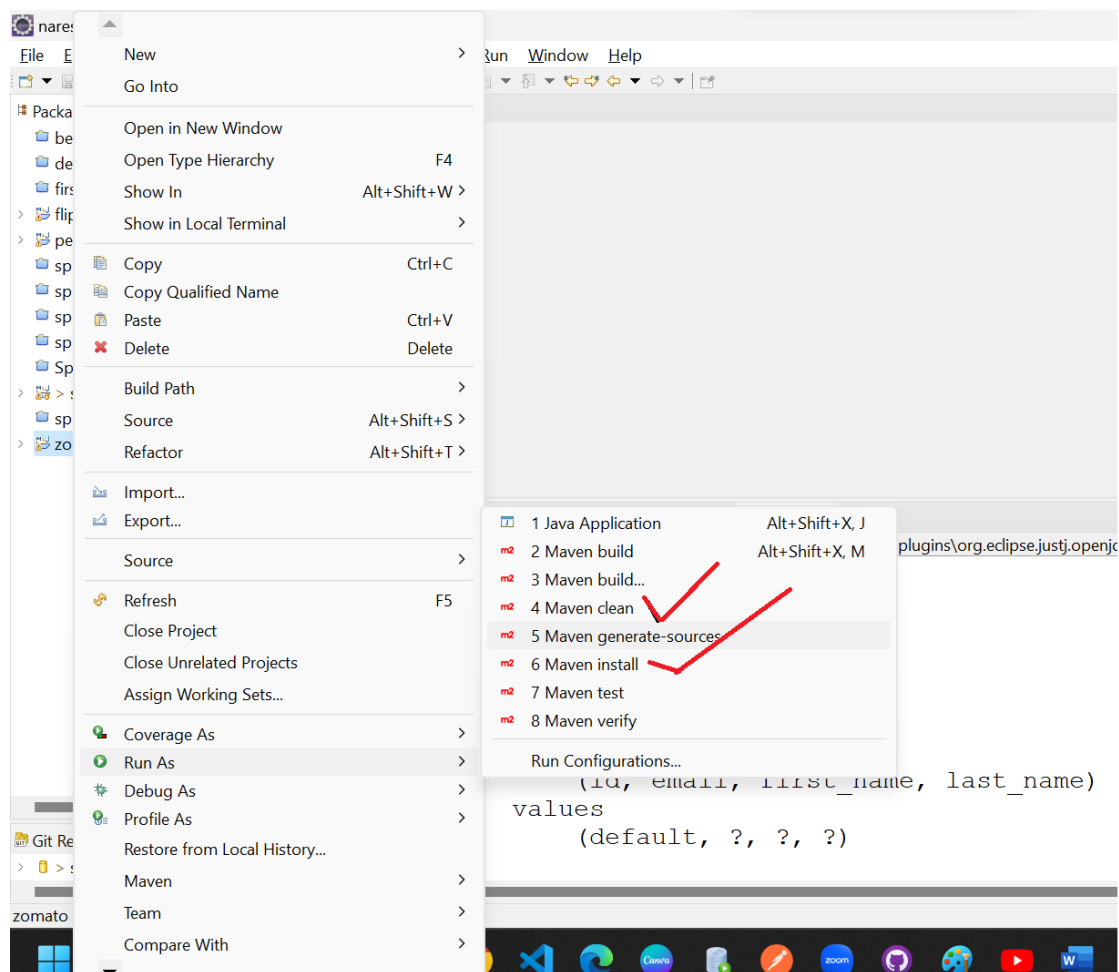
There are a number of built-in Maven goals, such as:

- clean: This goal cleans the project's output directory.
- install: This goal installs the project in the local Maven repository.
- compile: This goal compiles the project's source code.
- test: This goal runs the project's unit tests.
- package: This goal packages the project into a distributable format, such as a JAR or WAR file.

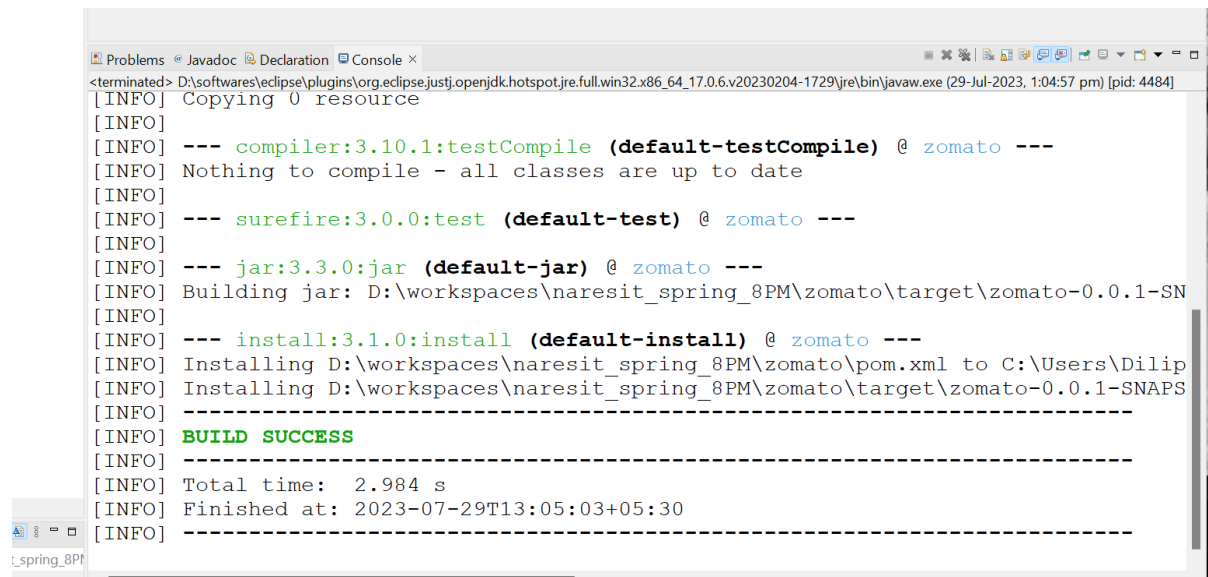
## Executing Goals From Eclipse:

Right click on Project -> Run As -> Maven Clean

Right click on Project -> Run As -> Maven install



Maven Goals execution status we can monitor from console view. If any errors occurred, then those will be printed in console.



```
<terminated> D:\softwares\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\bin\javaw.exe (29-Jul-2023, 1:04:57 pm) [pid: 4484]
[INFO] Copying 0 resource
[INFO]
[INFO] --- compiler:3.10.1:testCompile (default-testCompile) @ zomato ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- surefire:3.0.0:test (default-test) @ zomato ---
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ zomato ---
[INFO] Building jar: D:\workspaces\naresit_spring_8PM\zomato\target\zomato-0.0.1-SN
[INFO]
[INFO] --- install:3.1.0:install (default-install) @ zomato ---
[INFO] Installing D:\workspaces\naresit_spring_8PM\zomato\pom.xml to C:\Users\Dilip
[INFO] Installing D:\workspaces\naresit_spring_8PM\zomato\target\zomato-0.0.1-SNAPS
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.984 s
[INFO] Finished at: 2023-07-29T13:05:03+05:30
[INFO] -----
```