Experiment 2 - Working with Maven: Creating a Maven Project, Understanding the POM File, Dependency Management and Plugins

Working with Maven: Creating a Maven Project, Understanding the POM File, Dependency Management and Plugins

Objective:

Maven is a build automation tool primarily used for Java projects. It simplifies the build process, manages project dependencies, and supports plugins for different tasks. It uses XML files (called pom.xml) for project configuration and dependency management.

Key Benefits of Maven:

1. Dependency management (automates downloading and including libraries).
2. Build automation (compiles code, runs tests, creates artifacts).
3. Consistent project structure (standardizes how Java projects are set up).
4. Integration with CI tools (like Jenkins).

Key Features of Maven:

1. **Project Management**: Handles project dependencies, configurations, and builds.
2. **Standard Directory Structure**: Encourages a consistent project layout across Java projects.
3. **Build Automation**: Automates tasks such as compilation, testing, packaging, and deployment.
4. **Dependency Management**: Downloads and manages libraries and dependencies from repositories (e.g., Maven Central).
5. **Plugins**: Supports many plugins for various tasks like code analysis, packaging, and deploying.

To understand build automation tools, compare **Maven** and **Gradle**, and set up both tools for software development.

Using Command Line:

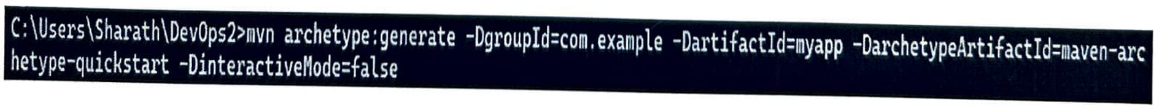
- Open command prompt.
- **mkdir program2** – this will create **program2** folder.
- **cd program2** – navigate program2 folder.
- After that, follow the below steps to work with Maven project.

Step 1: Creating a Maven Project

- You can create a **Maven project** using the **mvn** command (or through your **IDE**, as mentioned earlier). But here, I'll give you the essential **pom.xml** and **Java code**.

- Let's use the **Apache Commons Lang library** as a **dependency** (which provides utilities for **working with strings, numbers**, etc.). We will use this in a **simple Java program** to **work with strings.**

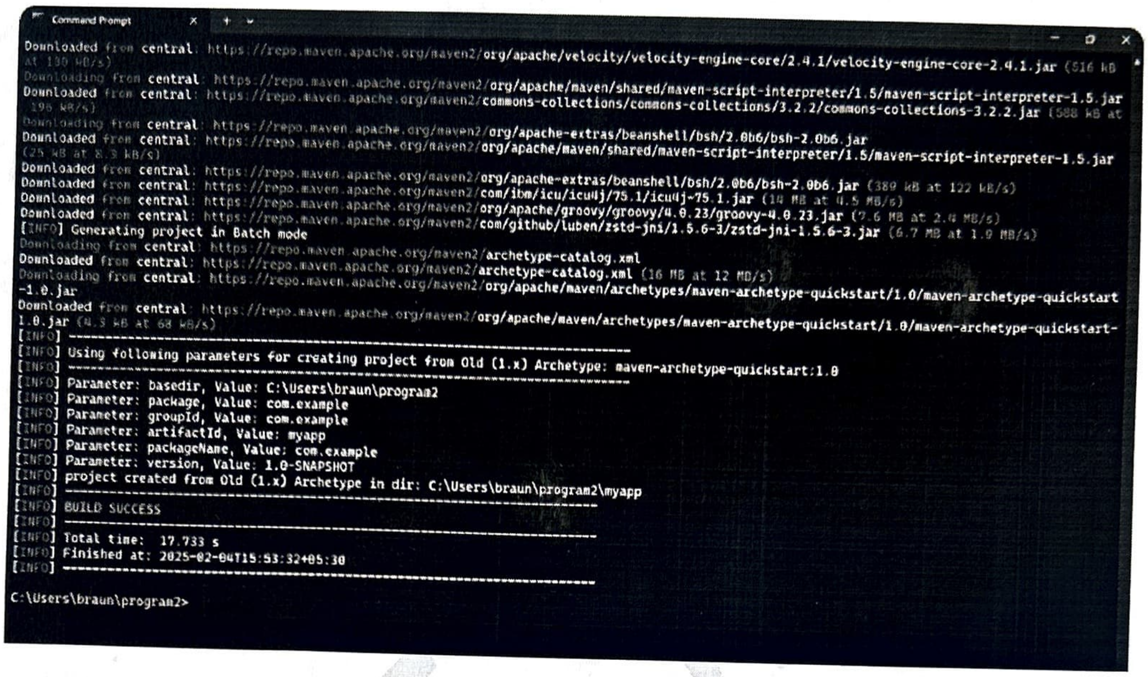
mvn archetype:generate -DgroupId=com.example -DartifactId=myapp -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false



```
C:\Users\Sharath\DevOps2>mvn archetype:generate -DgroupId=com.example -DartifactId=myapp -DarchetypeArtifactId=maven-arc
hetype-quickstart -DinteractiveMode=false
```

- **groupId:** A unique identifier for the group (usually the domain name).
- **artifactId:** A unique name for the project artifact (your project).
- **archetypeArtifactId:** The template you want to use for the project.
- **DinteractiveMode=false:** Disables prompts during project generation.

This will create a basic Maven project with the required directory structure and **pom.xml** file.

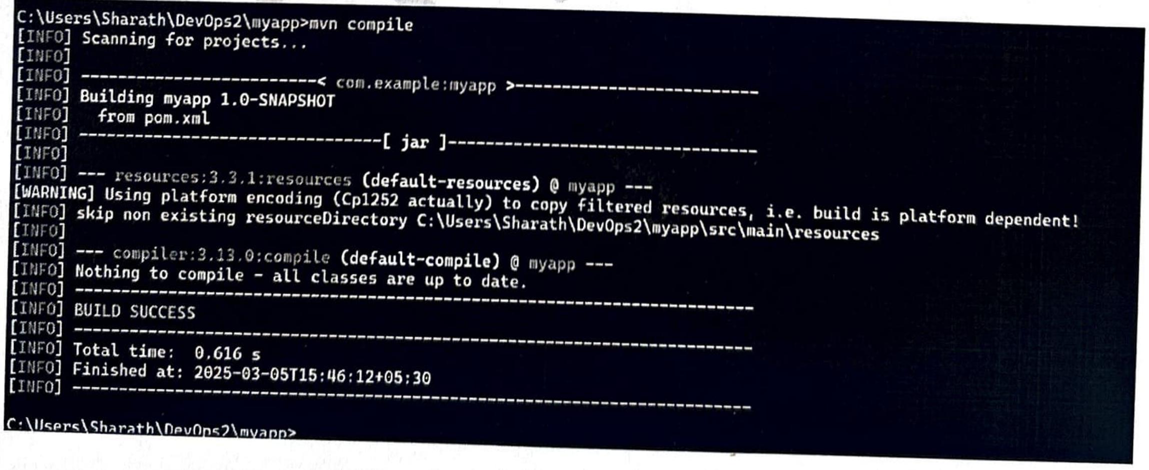


```
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/velocity/velocity-engine-core/2.4.1/velocity-engine-core-2.4.1.jar (516 kB
at 130 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-script-interpreter/1.5/maven-script-interpreter-1.5.jar
Downloaded from central: https://repo.maven.apache.org/maven2/commons-collections/commons-collections/3.2.2/commons-collections-3.2.2.jar (588 kB at
 196 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache-extras/beanshell/bsh/2.0b6/bsh-2.0b6.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-script-interpreter/1.5/maven-script-interpreter-1.5.jar
(25 kB at 8.3 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache-extras/beanshell/bsh/2.0b6/bsh-2.0b6.jar (389 kB at 122 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/ibm/icu/icu4j/75.1/icu4j-75.1.jar (14 MB at 4.5 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/groovy/groovy/4.0.23/groovy-4.0.23.jar (7.6 MB at 2.4 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/github/luben/zstd-jni/1.5.6-3/zstd-jni-1.5.6-3.jar (6.7 MB at 1.9 MB/s)
[INFO] Generating project in Batch mode
Downloading from central: https://repo.maven.apache.org/maven2/archetype-catalog.xml
Downloaded from central: https://repo.maven.apache.org/maven2/archetype-catalog.xml (16 MB at 12 MB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart
-1.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-
1.0.jar (4.3 kB at 68 kB/s)
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: C:\Users\braun\program2
[INFO] Parameter: package, Value: com.example
[INFO] Parameter: groupId, Value: com.example
[INFO] Parameter: artifactId, Value: myapp
[INFO] Parameter: packageName, Value: com.example
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\Users\braun\program2\myapp
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  17.733 s
[INFO] Finished at: 2025-02-04T15:53:32+05:30
[INFO] ------------------------------------------------------------------------

C:\Users\braun\program2>
```

Step 2: Building the Project

To build and run this project, follow these steps:

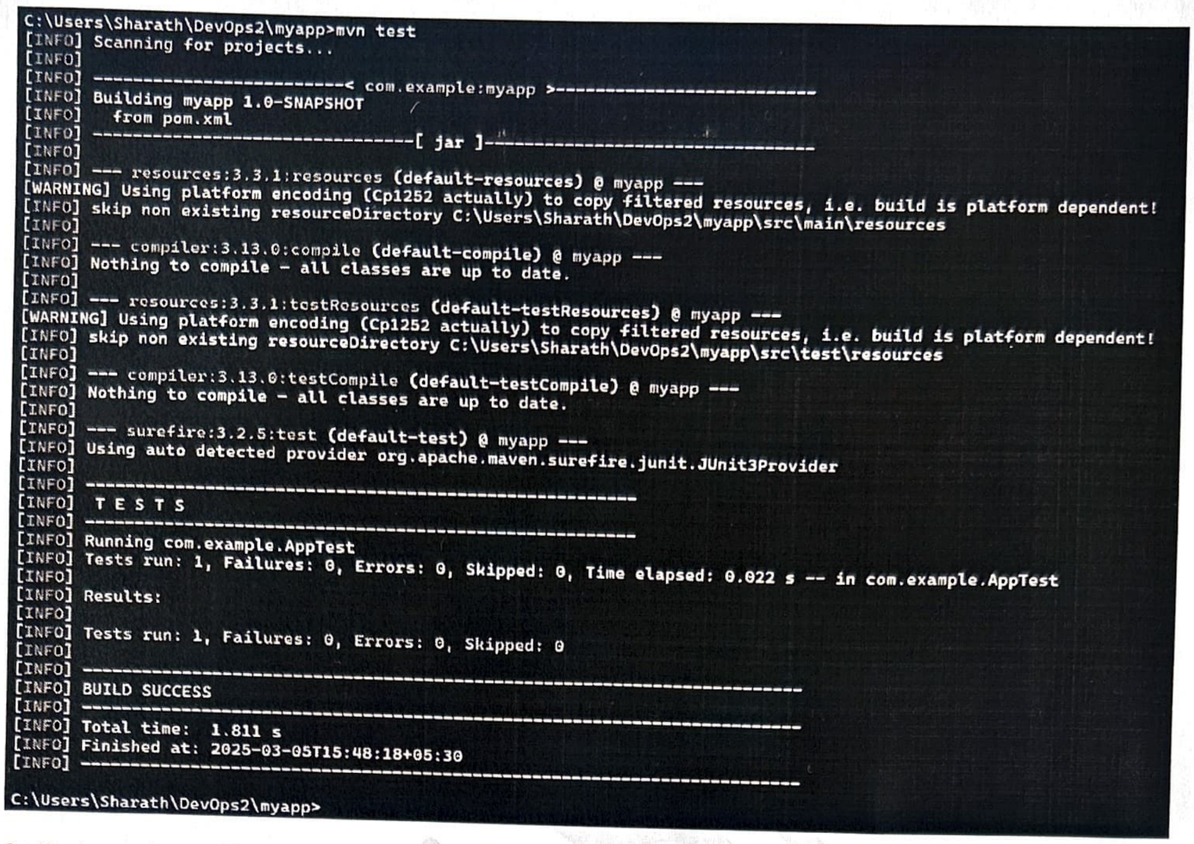
1. **Compile the Project**



```
C:\Users\Sharath\DevOps2\myapp>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------------< com.example:myapp >--------------------------
[INFO] Building myapp 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ myapp ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\Sharath\DevOps2\myapp\src\main\resources
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ myapp ---
[INFO] Nothing to compile - all classes are up to date.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.616 s
[INFO] Finished at: 2025-03-05T15:46:12+05:30
[INFO] ------------------------------------------------------------------------

C:\Users\Sharath\DevOps2\myapp>
```

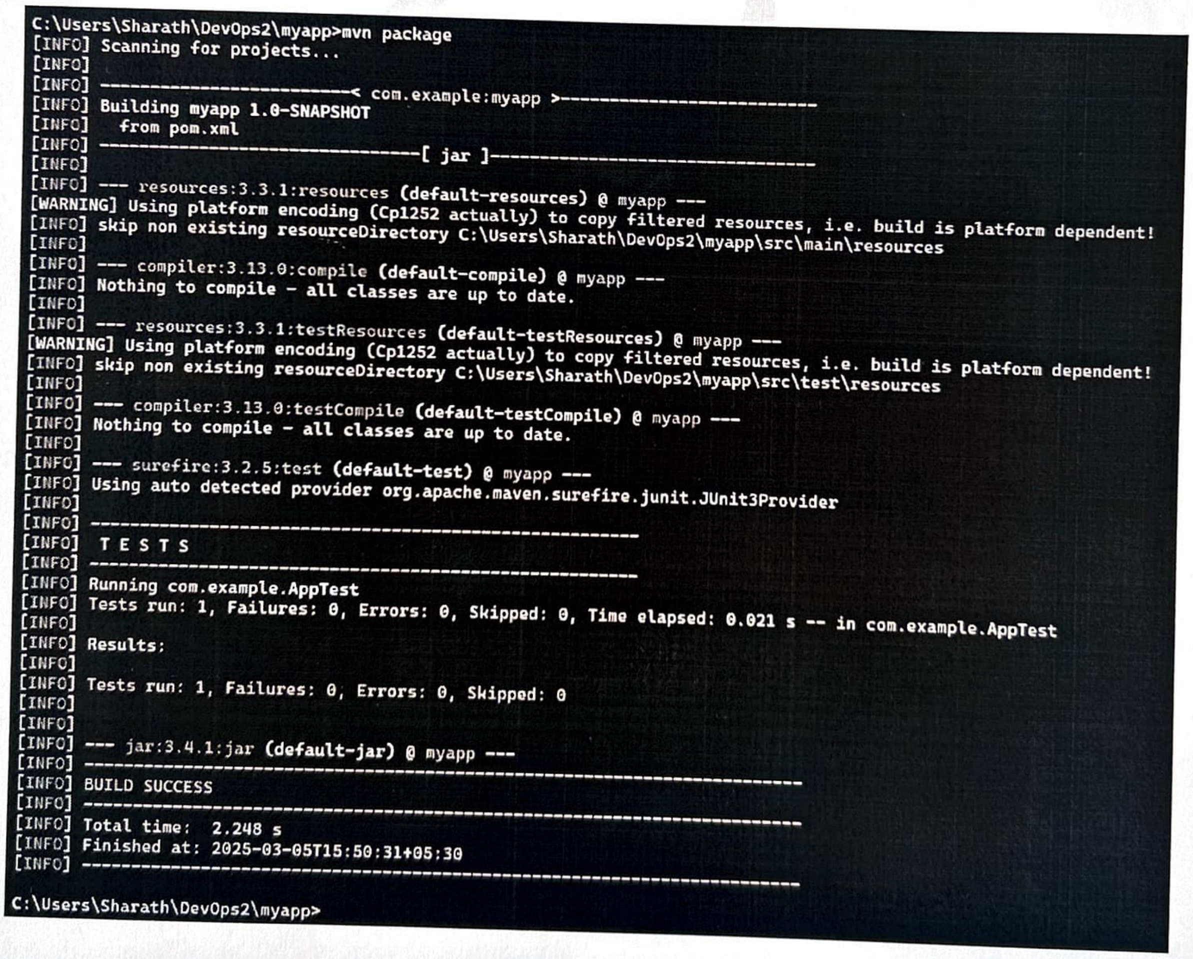
Exp-2

2. **Run the Unit Tests**



```
C:\Users\Sharath\DevOps2\myapp>mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------------< com.example:myapp >--------------------------
[INFO] Building myapp 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ myapp ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\Sharath\DevOps2\myapp\src\main\resources
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ myapp ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ myapp ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\Sharath\DevOps2\myapp\src\test\resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ myapp ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ myapp ---
[INFO] Using auto detected provider org.apache.maven.surefire.junit.JUnit3Provider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.022 s -- in com.example.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.811 s
[INFO] Finished at: 2025-03-05T15:48:18+05:30
[INFO] ------------------------------------------------------------------------

C:\Users\Sharath\DevOps2\myapp>
```

3. **Package the project into a JAR**



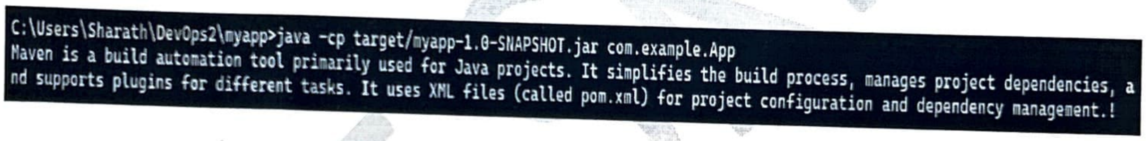
```
C:\Users\Sharath\DevOps2\myapp>mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------------< com.example:myapp >--------------------------
[INFO] Building myapp 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ myapp ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\Sharath\DevOps2\myapp\src\main\resources
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ myapp ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ myapp ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\Sharath\DevOps2\myapp\src\test\resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ myapp ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ myapp ---
[INFO] Using auto detected provider org.apache.maven.surefire.junit.JUnit3Provider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.021 s -- in com.example.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.4.1:jar (default-jar) @ myapp ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.248 s
[INFO] Finished at: 2025-03-05T15:50:31+05:30
[INFO] ------------------------------------------------------------------------

C:\Users\Sharath\DevOps2\myapp>
```

4. **Run the application (using JAR)**

```
java -cp target/myapp-1.0-SNAPSHOT.jar com.example.App
```

The above command is used to **run a Java application** from the command line. Here's a breakdown of each part:

- **java**: This is the Java runtime command used to run Java applications.
- **-cp**: This stands for **classpath**, and it specifies the location of the classes and resources that the JVM needs to run the application. In this case, it's pointing to the JAR file where your compiled classes are stored.
- **target/myapp-1.0-SNAPSHOT.jar**: This is the **JAR file** (Java ARchive) that contains the compiled Java classes and resources. It's located in the target directory, which Maven creates after you run mvn package.
- **com.example.App**: This is the **main class** that contains the main() method. When you run this command, Java looks for the main() method inside the App class located in the com.example package and executes it.



```
C:\Users\Sharath\DevOps2\myapp>java -cp target/myapp-1.0-SNAPSHOT.jar com.example.App
Maven is a build automation tool primarily used for Java projects. It simplifies the build process, manages project dependencies, a
nd supports plugins for different tasks. It uses XML files (called pom.xml) for project configuration and dependency management.!
```