EXPERIMENT-2

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

1: Install the Java JDK

- If you haven't installed the **Java JDK**

Overview of the Project

2: Creating a Maven Project

There are a few ways to create a Maven project, such as using the command line, IDEs like IntelliJ IDEA or Eclipse, or generating it via an archetype.

1. **Using Command Line:**
   - To create a basic Maven project using the command line, you can use the following command:

```
mvn archetype:generate -DgroupId=com.example -DartifactId=myapp -DarchetypeArtifactId=maven-archetype-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.

3: Understanding the POM File

The **POM (Project Object Model)** file is the heart of a Maven project. It is an XML file that contains all the configuration details about the project. Below is an example of a simple POM file:

A basic **`pom.xml`** structure looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<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.example</groupId>
    <artifactId>my-project</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <dependencies>
        <!-- Dependencies go here -->
    </dependencies>

    <build>
        <plugins>
            <!-- Plugins go here -->
        </plugins>
    </build>
</project>
```

Key element in `pom.xml`:

- **`<groupId>`:** The group or organization that the project belongs to.
- **`<artifactId>`:** The name of the project or artifact.

- **`<version>`:** The version of the project (often follows a format like `1.0-SNAPSHOT`).
- **`<packaging>`:** Type of artifact, e.g., `jar`, `war`, `pom`, etc.
- **`<dependencies>`:** A list of dependencies the project requires.
- **`<build>`:** Specifies the build settings, such as plugins to use.

4: Dependency Management

Maven uses the `<dependencies>` tag in the `pom.xml` to manage external libraries or dependencies that your project needs. When Maven builds the project, it will automatically download these dependencies from a repository (like Maven Central).

Example of adding a dependency:

```
<dependencies>
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-lang3</artifactId>
        <version>3.12.0</version>
    </dependency>
</dependencies>
```

- **Transitive Dependencies**
  - Maven automatically resolves transitive dependencies. For example, if you add a library that depends on other libraries, Maven will also download those.

- **Scopes**
  - Dependencies can have different scopes that determine when they are available:
    - **compile** (default): Available in all build phases.
    - **provided**: Available during compilation but not at runtime (e.g., a web server container).
    - **runtime**: Needed only at runtime, not during compilation.
    - **test**: Required only for testing.

5: Using Plugins

Maven plugins are used to perform tasks during the build lifecycle, such as compiling code, running tests, packaging, and deploying. You can specify plugins within the `<build>` section of your **`pom.xml`**.

- **Adding Plugins**
  - You can add a plugin to your `pom.xml` like so:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-
plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```

In this example, the **`maven-compiler-plugin`** is used to compile Java code and specify the source and target JDK versions.

1. **Common Plugins**
   - **maven-compiler-plugin**: Compiles Java code.
   - **maven-surefire-plugin**: Runs unit tests.
   - **maven-jar-plugin**: Packages the project as a JAR file.
   - **maven-clean-plugin**: Cleans up the `target/` directory.

2. **Plugin Goals** Each plugin consists of goals, which are specific tasks to be executed. For example:
   - **`mvn clean install`:** This will clean the target directory and then install the package in the local repository.
   - **`mvn compile`:** This will compile the source code.
   - **`mvn test`:** This will run unit tests.

Working with Maven Project

Note: Always create separate folder to do any program.

- Open command prompt.
  - C:\Users\ItiShree>cd desktop
  -
  - C:\Users\ItiShree\Desktop>mkdir MAVENPROJECT2
  -
  - C:\Users\ItiShree\Desktop>cd MAVENPROJECT2
  - After then follow the below step to working 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**.

  - **C:\Users\ItiShree\Desktop\MAVENPROJECT2>mvn archetype:generate -DgroupId=com.example -DartifactId=myapp -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false**

```
C:\Users\ItiShree\Desktop\MAVENPROJECT2>mvn archetype:generate -DgroupId=com.example -DartifactId=myapp -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/maven-metadata.xml
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-metadata.xml
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-metadata.xml (14 kB at 2.8 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/maven-metadata.xml (21 kB at 4.1 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-archetype-plugin/maven-metadata.xml
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-archetype-plugin/maven-metadata.xml (1.0 kB at 9.6 kB/s)
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> archetype:3.3.1:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< archetype:3.3.1:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- archetype:3.3.1:generate (default-cli) @ standalone-pom ---
[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 43 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\ItiShree\Desktop\MAVENPROJECT2
[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\ItiShree\Desktop\MAVENPROJECT2\myapp
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  06:27 min
[INFO] Finished at: 2025-03-04T10:33:49-03:00
```

Note: See in your terminal below the project folder path showing after executing the cmd manually navigate the path and see the project folder name called myapp.

Step 2: Open The `pom.xml` File

- You can manually navigate the **project folder** named call **myapp** and open the file pom.xml and copy the below code and paste it then save it.
- In case if you not getting project folder then type command in your cmd.
  - **cd myapp** – is use to navigate the project folder.
  - **notepad pom.xml** – is use to open pom file in notepad.

```
<?xml version="1.0" encoding="UTF-8"?>

<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.example</groupId>

    <artifactId>myapp</artifactId>

    <version>1.0-SNAPSHOT</version>

    <dependencies>

        <!-- JUnit Dependency for Testing -->

        <dependency>

            <groupId>junit</groupId>

            <artifactId>junit</artifactId>

            <version>4.13.2</version>

            <scope>test</scope>

        </dependency>

    </dependencies>
```

```
  <build>
    <plugins>
      <!-- Maven Surefire Plugin for running tests -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.22.2</version>
        <configuration>
          <redirectTestOutputToFile>false</redirectTestOutputToFile>
          <useSystemOut>true</useSystemOut>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Step 3: Open Java Code (`App.java`) File

- Open a file **`App.java`** inside the **`src/main/java/com/example/`** directory.
- After opening the **App.java** copy the below code and paste it in that file then save it.

```
package com.example;

public class App {
    public int add(int a, int b) {
        return a + b;
    }
    public static void main(String[] args) {
        App app = new App();
        int result = app.add(2, 3);
        System.out.println("2 + 3 = " + result);
        System.out.println("Application executed
successfully!");
    }
}
```

Step 4: Open Java Code (`AppTest.java`) File

- Open a file **`AppTest.java`** inside the **`src/test/java/com/example/`** directory.
- After opening the **AppTest.java** copy the below code and paste it in that file then save it.

```
package com.example;

import org.junit.Assert;

import org.junit.Test;

public class AppTest {
```

```
    @Test
    public void testAdd() {
        App app = new App();
        int result = app.add(2, 3);
        System.out.println("Running test: 2 + 3 = " + result);

        Assert.assertEquals(5, result);
    }
}
```

Note: before building the project make sure you are in the project folder if not navigate the project folder type command in your command prompt `cd myapp`

Step 4: Building the Project

To build and run this project, follow these steps:

1. Compile the Project

```
mvn compile
```

```
C:\Users\ItiShree\Desktop\MAVENPROJECT2\myapp>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< com.example:myapp >-------------------------
[INFO] Building myapp 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/junit/junit/4.13.2/junit-4.13.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/junit/junit/4.13.2/junit-4.13.2.pom (27 kB at 33 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/hamcrest/hamcrest-core/1.3/hamcrest-core-1.3.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/hamcrest/hamcrest-core/1.3/hamcrest-core-1.3.pom (766 B at 4.9 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/hamcrest/hamcrest-parent/1.3/hamcrest-parent-1.3.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/hamcrest/hamcrest-parent/1.3/hamcrest-parent-1.3.pom (2.0 kB at 70 kB/s)
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ myapp ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\ItiShree\Desktop\MAVENPROJECT2\myapp\src\main\resources
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ myapp ---
[INFO] Recompiling the module because of changed source code.
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file with javac [debug target 1.8] to target\classes
[WARNING] bootstrap class path not set in conjunction with -source 8
[WARNING] source value 8 is obsolete and will be removed in a future release
[WARNING] target value 8 is obsolete and will be removed in a future release
[WARNING] To suppress warnings about obsolete options, use -Xlint:-options.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.394 s
[INFO] Finished at: 2025-03-04T10:39:34-03:00
[INFO] ------------------------------------------------------------------------
```

2. Run the Unit Tests

```
mvn test
```

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit4/2.22.2/surefire-junit4-2.22.2.jar (85 kB at 97 kB/s)
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.AppTest
Running test: 2 + 3 = 5
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.044 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:  06:36 min
[INFO] Finished at: 2025-03-04T10:46:32-03:00
[INFO] ------------------------------------------------------------------------
```

3. Package the project into a JAR

```
mvn package
```

```
C:\Users\ItiShree\Desktop\MAVENPROJECT2\myapp>mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< com.example:myapp >-------------------------
[INFO] Building myapp 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[WARNING] Parameter 'useSystemOut' is unknown for plugin 'maven-surefire-plugin:2.22.2:test (default-test)'
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ myapp ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\ItiShree\Desktop\MAVENPROJECT2\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 (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\ItiShree\Desktop\MAVENPROJECT2\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:2.22.2:test (default-test) @ myapp ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.AppTest
Running test: 2 + 3 = 5
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.146 s - in com.example.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
```

4. Run the application (using JAR)

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

```
C:\Users\ItiShree\Desktop\MAVENPROJECT2\myapp>java -cp target/myapp-1.0-SNAPSHOT.jar com.example.App
2 + 3 = 5
Application executed successfully!
```

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.