EXERCISE -1

Use of maven-jar-plugin to package the project into a jar file. | | Showing how to run a main class and show simple output:

Summary of What We've Done So Far: main class

1. Website Deployment: You've deployed your simple HTML, CSS, and assets (like logo.png) to GitHub Pages using Maven. You've created a Selenium Test to validate the website title using TestNG and ran the test to ensure the website is functioning as expected.

2. Next Steps: We can show how to use the maven-jar-plugin to create a runnable JAR file. Demonstrate running a main class inside this JAR to produce a simple output.

Summary:

1. Add maven-jar-plugin: Configure the plugin in pom.xml to specify the

2. Create Main Class: Write a simple MainClass with a main class . main method that outputs a message.

3. Package with Maven: Run mvn clean package to package the project into a JAR file.

4. Run the JAR: Use java -jar target/your-project-name.jar to run the packaged JAR and print the output.

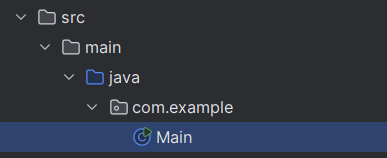
Steps to Package the Project as a JAR and Run a Main Class

1. Add maven-jar-plugin to pom.xml : To package your Maven project as a JAR file and specify the pom.xml

Add the following configuration to your POM.XML

<plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-jar-plugin</artifactId>  
 <version>3.1.0</version>  
 <configuration>  
 <!-- Specify the main class to be executed -->  
 <archive>  
 <manifestEntries>  
 <Main-Class>org.example.Main</Main-Class> <!-- Replace with your actual main class -->  
 </manifestEntries>  
 </archive>  
 </configuration>  
</plugin>

This will tell Maven to include the Main-Class in the JAR manifest and specify the main class that should be executed when the JAR is run.



2. Create a Main Class: In your src/main/java directory, create a class with a main method. For example, create a com.example :

package com.example;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Hello, World!");  
 }  
}

3. Package the Project into a JAR: After configuring the plugin and creating the MainClass , run the following Maven command to build the project and package it into a JAR file

**Run the below command in IntelliJ Git Terminal**

**mvn clean package**

4. Run the JAR File: Once the JAR is created, you can run it with the following command:

**java -jar target/your-project-name.jar**

This will execute the main method from your MainClass and print the message:

Hello, this is a simple output from the main class!

1. Clean Lifecycle Used to remove old build files before a new build.

mvn clean

1. Site Lifecycle This lifecycle generates project documentation

Step 1: Add Site Plugin in pom.xml Before running the site command, you need to add the Maven Site Plugin inside the pom.xml :

**<plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-site-plugin</artifactId>  
 <version>3.12.1</version> <!-- Use the latest version -->  
</plugin>**

Step 2: Run the Site Command Once the plugin is added, execute:

mvn site

What Happens? Maven scans your project for available reports. Generates an HTML-based website inside target/site/ . Includes various reports like dependencies, plugin management, and test results.

Step 3: Open the Generated Site After successful execution, open the following file in a browser:

After successful execution, open the following file in a browser

Below folder will be in users specific /home/ewitise/username/username\_maven\_repository/target/site/index.html

D:\Idea Projects\CA-MVN\target\site\index.html

You’ll See Reports Like:

✔ Project Summary

✔ Dependencies Report

✔ Plugin Management

✔ Unit Test Results (if configured)

✔ Code Coverage (if applicable)

**EXERCISE -2**

mvn deploy Command

The mvn deploy command is used to upload the built artifact (JAR, POM, etc.) to a repository for distribution and sharing

Step 1:

Create a Local Repository

mkdir D:\my-local-maven-repo

Step 2: Configure pom.xml for Local Deployment Add the following inside in pom.xml :

<distributionManagement>  
 <repository>  
 <id>local-repo</id>  
 <url>[file:///D:/Users/Vinayak/EWIT/Sixth\_Semester/DevOps/MavenTest</url](file:///D:/Users/Vinayak/EWIT/Sixth_Semester/DevOps/MavenTest%3c/url)>

<! …Please note that this is my specific folder you need to create your own local directory and specify here ….>  
 </repository>  
</distributionManagement>

Step 3:

Run the Deploy Command

mvn deploy

What Happens?

Maven builds the project.

Stores the artifact (JAR, POM, etc.) in D:/my-local-maven-repo.

Step 4:

Verify Deployment Navigate to D:/my-local-maven-repo/ and check if the project is stored correctly.

GRADLE

Exercise -1

**Step 1**: Modify build.gradle (Groovy DSL)

plugins **{** id 'application'  
**}**repositories **{** mavenCentral()  
**}**dependencies  
 **{** testImplementation 'org.junit.jupiter:junit-jupiter:5.8.1'  
 **}**application **{**mainClass = 'com.example.Main'  
**}**

**Step 2:** Create Main.java in src/main/java/com/example

package com.example;  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Hello, World!");  
 }  
}

**Step 3:** Build and Run the Project

In IntelliJ IDEA, open the Gradle tool window (View → Tool Windows → Gradle).

Click Tasks > application > run .

Or run from terminal:

**gradle run**