Day 24:.

Task 1: Build Lifecycle

Demonstrate the use of Maven lifecycle phases (clean, compile, test, package, install, deploy) by executing them on a sample project and documenting what happens in each phase.

Sample Maven Project

Project Structure

Create a basic Maven project structure with the following files and directories:

my-maven-project

```
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── com
    │           └── example
    │               └── App.java
    └── test
        └── java
            └── com
                └── example
                    └── AppTest.java
```

App.java

Create a simple Java class App.java in src/main/java/com/example/:

```
package com.example;
```

```
public class App {
    public static void main(String[] args) {
        System.out.println("Hello Maven!");
    }
}
```

AppTest.java

Create a simple JUnit test class AppTest.java in src/test/java/com/example/:

```
package com.example;

import org.junit.Test;
import static org.junit.Assert.*;

public class AppTest {
    @Test
    public void testApp() {
        assertTrue(true);
    }
}
```

pom.xml

This is the Maven configuration file (pom.xml) that defines the project structure, dependencies, and build settings:

```
<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-maven-project</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
   <maven.compiler.source>1.8</maven.compiler.source>
   <maven.compiler.target>1.8</maven.compiler.target>
</properties>

<dependencies>
   <!-- JUnit for testing -->
   <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
   </dependency>
</dependencies>

<build>
   <plugins>
      <!-- Maven Compiler Plugin -->
      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
```

```
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <source>${maven.compiler.source}</source>
                <target>${maven.compiler.target}</target>
            </configuration>
        </plugin>

        <!-- Maven Surefire Plugin for running tests -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.0.0-M5</version>
        </plugin>
    </plugins>
  </build>

</project>
```

Maven Lifecycle Phases

Clean Phase

Command: mvn clean

Purpose: Cleans the project by deleting the target/ directory.

Output: Deletes all compiled classes, resources, and other files generated during the build.

Compile Phase

Command: mvn compile

Purpose: Compiles the main source code (src/main/java) of the project.

Output: Generates .class files for all Java source files in target/classes/.

Test Phase

Command: mvn test

Purpose: Executes the tests in the src/test/java directory using a testing framework like JUnit.

Output: Runs unit tests and reports test results. Maven creates a test report in target/surefire-reports/.

Package Phase

Command: mvn package

Purpose: Packages the compiled code (classes and resources) into a distributable format, such as a JAR.

Output: Generates the artifact (e.g., target/my-maven-project-1.0-SNAPSHOT.jar).

Install Phase

Command: mvn install

Purpose: Installs the packaged artifact into the local Maven repository (~/.m2/repository/).

Output: Copies the packaged artifact (JAR) to the local repository for use as a dependency in other Maven projects.

Deploy Phase (Example Only, Requires Configuration)

Command: mvn deploy

Purpose: Copies the final packaged artifact to a remote repository for sharing with other developers or projects.

Output: Typically used in enterprise settings where artifacts need to be shared across teams or deployed to a central repository.

Execution Example

Let's execute these phases on our sample project:

Clean, Compile, and Test Phases:

mvn clean compile test

Cleans the project, compiles the main source code, and runs tests.

Package Phase:

mvn package

Packages the application into a JAR file (target/my-maven-project-1.0-SNAPSHOT.jar).

Install Phase:

mvn install

Installs the JAR file into the local Maven repository (~/.m2/repository/com/example/my-maven-project/1.0-SNAPSHOT/my-maven-project-1.0-SNAPSHOT.jar).

Deploy Phase (Not Executed in Local Environment):

mvn deploy

This phase typically requires configuration to deploy artifacts to a remote repository and is usually not demonstrated in a local environment without proper setup.