**Assignment 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.**

**Answer: -**

Let’s create a simple Maven project to execute the Maven lifecycle phase, explaining their purpose and actions in detail. Below is a sample Maven project to illustrate this:

sample-project

├── pom.xml

├── src

│ ├── main

│ │ └── java

│ │ └── com

│ │ └── example

│ │ └── App.java

│ └── test

│ └── java

│ └── com

│ └── example

│ └── AppTest.java

**In this structure:**

- pom.xml is the Maven Project Object Model file containing configuration details.

- App.java is a simple Java application.

- AppTest.java is a test class for App.java.

Let’s see the Maven Lifecycle Phases: -

**1. Clean Phase**

Command:

mvn clean

**Purpose:**

The clean phase is responsible for cleaning up the project directory by removing files generated in the previous builds. This includes compiled classes, packaged files, and other artifacts.

**Actions:**

- Deletes the target directory.

- Ensures a fresh start for the next build process.

**Detailed Execution:**

When you run mvn clean, Maven reads the pom.xml to understand the project structure. It then locates the target directory, where all the build outputs are stored, and deletes it. This step is crucial to avoid conflicts or inconsistencies caused by leftover files from previous builds.

**2. Compile Phase**

**Command:**

mvn compile

**Purpose:**

The compile phase compiles the source code of the project. It translates .java files in the src/main/java directory into .class files.

**Actions:**

- Compiles source code.

- Stores compiled classes in target/classes.

**Detailed Execution:**

Running mvn compile triggers the Java compiler to process all .java files in src/main/java. The resulting .class files are placed in the target/classes directory. This phase ensures that the project's source code is syntactically correct and can be executed.

**3. Test Phase**

**Command:**

mvn test

**Purpose:**

The test phase executes the unit tests using a testing framework like JUnit. This phase ensures the correctness of the code by running tests defined in src/test/java.

**Actions:**

- Compiles test code.

- Executes tests.

- Generates test reports.

**Detailed Execution:**

When you run mvn test, Maven compiles the test classes in src/test/java and then runs them. Test results, including successes and failures, are reported in the console and saved in target/surefire-reports. This phase is critical for verifying that the application behaves as expected under different scenarios.

**4. Package Phase**

**Command:**

mvn package

**Purpose:**

The package phase takes the compiled code and packages it into a distributable format, such as a JAR or WAR file.

**Actions:**

- Assembles all compiled classes and resources.

- Packages them into a JAR (or WAR) file.

- Places the package in the target directory.

**Detailed Execution:**

Running mvn package bundles the application into a single archive. Maven reads the configuration in pom.xml to determine the packaging type (JAR by default). It includes compiled classes, resources (like src/main/resources), and dependencies (if configured). The resulting artifact, e.g., target/sample-project-1.0-SNAPSHOT.jar, is ready for distribution or deployment.

**5. Install Phase**

**Command:**

mvn install

**Purpose:**

The install phase installs the packaged artifact into the local Maven repository. This makes the artifact available as a dependency for other local projects.

**Actions:**

- Copies the JAR file to the local repository (~/.m2/repository).

**Detailed Execution:**

When you run mvn install, Maven places the JAR file into your local repository directory. This local repository acts as a cache of dependencies. By installing the artifact, you allow other projects on your machine to use it as a dependency, facilitating modular development and reuse.

**6. Deploy Phase**

**Command:**

mvn deploy

**Purpose:**

The deploy phase copies the final package to a remote repository for sharing with other developers and projects. This phase is typically used in a CI/CD pipeline.

**Actions:**

- Uploads the artifact to a remote repository.

- Uses repository details specified in pom.xml.

**Detailed Execution:**

Running mvn deploy sends the artifact to a remote repository specified in the pom.xml. This remote repository is often a shared server accessible by multiple developers or CI/CD systems. By deploying the artifact, you make it available for consumption by other projects and teams, supporting collaborative development and consistent dependency management.

**Summary**

By running each phase individually or in sequence, Maven ensures that your project builds correctly and consistently every time. Here is a quick overview of each phase:

Clean: Deletes the previous build artifacts.

Compile: Compiles the source code.

Test: Runs the unit tests.

Package: Creates the JAR/WAR file.

Install: Installs the package in the local repository.

Deploy: Deploys the package to a remote repository.

These phases help manage the lifecycle of your project, from development to deployment, ensuring a smooth and repeatable build process.