Assignment Day-24

Core Java with DS and Algorithms

Name : Joshnitha Rangolu

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 Project Setup:

1. Create a directory for your project (e.g., maven-demo).
2. Inside the directory, create a file named pom.xml
3. Create a source code file named src/main/java/com/example/App.java with some basic code (e.g., a class with a main method that prints a message).
4. Create a test file named src/test/java/com/example/AppTest.java with a unit test for your App class (using JUnit).

Running Maven Lifecycle Phases:

Use the Maven command-line tool (mvn) to execute each phase:

1. clean:
    - Command: mvn clean
    - Explanation: This phase removes all files generated during previous builds from target and project directories. It cleans up the project for a fresh build.
2. compile:
    - Command: mvn compile
    - Explanation: This phase compiles your source code (.java files) into bytecode (.class files).
3. test:
    - Command: mvn test
    - Explanation: This phase executes your unit tests located in the src/test/java directory. It verifies if your code functions as expected.
4. package:
    - Command: mvn package
    - Explanation: This phase packages your compiled code and dependencies into a distributable artifact (typically a JAR file). This creates a ready-to-deploy package.
5. install:
    - Command: mvn install
    - Explanation: This phase installs the packaged artifact into your local Maven repository. This makes the artifact available for other projects that depend on it.
6. deploy:
    - Command: mvn deploy (Note: Usually requires configuration)
    - Explanation: This phase deploys the packaged artifact to a remote repository (e.g., company Nexus repository). This makes it accessible to other developers or environments.

Execution:

Step1: we have to add dependencies in project

```
<properties>
<maven.compiler.source>1.8</maven.compiler.source>
<maven.compiler.target>1.8</maven.compiler.target>
</properties>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.12</version>
<scope>test</scope>
</dependency>
</dependencies>
public class App {
   public static void main(String[] args) {
      System.out.println("Hello, Maven!");
   }
   public int add(int a, int b) {
      return a + b;
   }
}
public class AppTest {
   public void testAdd() {
      App app = new App();
      assertEquals(9, app.add(6, 3));
   }
}
mvn clean
mvn compile
```

```
mvn test
mvn package
mvn install
mvn deploy
```

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

```
package com.wipro;
import org.junit.Test;
import static org.junit.Assert.*;

public class AppTest {
    public void testApp() {
        assertEquals("Hello, Maven!", "Hello, Maven!");
    }
}
```

Step 2: Execute Maven Lifecycle Phases

clean

The clean phase deletes the target directory, which contains all the build artifacts.

```
mvn clean
```

Output:

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
```

[INFO] BUILD SUCCESS

[INFO] --

This removes any previous build artifacts, ensuring a fresh start.

compile

The compile phase compiles the source code of the project.

mvn compile

Output:

Copy code

[INFO] Compiling 1 source file to /path/to/maven-sample/target/classes

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

The source code (App.java) is compiled to the target/classes directory.

test

The test phase runs the unit tests using a suitable testing framework (JUnit in this case).

mvn test

Output:

[INFO] Surefire report directory: /path/to/maven-sample/target/surefire-reports

[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.013 s - in com.example.AppTest

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

The unit test (AppTest.java) is executed, and the results are reported.

package

The package phase packages the compiled code into a distributable format, such as a JAR.

mvn package

Output:

Copy code

[INFO] Building jar: /path/to/maven-sample/target/maven-sample-1.0-SNAPSHOT.jar

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

The compiled code and resources are packaged into a JAR file (maven-sample-1.0-SNAPSHOT.jar).

install

The install phase installs the package into the local repository, which is used as a cache for dependencies and to share artifacts among projects locally.

mvn install

Output:

[INFO] Installing /path/to/maven-sample/target/maven-sample-1.0-SNAPSHOT.jar to /path/to/.m2/repository/com/example/maven-sample/1.0-SNAPSHOT/maven-sample-1.0-SNAPSHOT.jar

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

The JAR is installed into the local Maven repository (~/.m2/repository).

deploy

The deploy phase copies the final package to the remote repository for sharing with other developers and projects. This requires configuration of the remote repository in pom.xml and appropriate credentials.

mvn deploy

Output:

[INFO] Deploying /path/to/maven-sample/target/maven-sample-1.0-SNAPSHOT.jar to https://your-repo-url/com/example/maven-sample/1.0-SNAPSHOT/maven-sample-1.0-SNAPSHOT.jar

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

7.