Maven Cheat-Sheet (One Page)

1. Compile / Debug

```
mvn -X clean compile
```

Configure compiler in pom.xml:

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.10.1</version>
<configuration>
<source>17</source>
<target>17</target>
</configuration>
</plugin>
```

2. Check Dependencies

```
mvn dependency:tree -Dverbose
ls ~/.m2/repository/<group>/<artifact>/<version>/
mvn -U dependency:resolve
```

3. Apply Patch

```
git apply fix.patch
git add .
git commit -m "Apply patch"
mvn clean test
```

4. Rerun Failed Tests

```
mvn -Dsurefire.rerunFailingTestsCount=2 test
mvn -Dtest=MyTestClass#testMethod test
```

5. Unsupported Class Version Error - Check `maven-compiler-plugin` source/target - Ensure JAVA_HOME and mvn -version match

6. WAR → JAR - Change `<packaging>jar</packaging>` - Use maven-jar-plugin or maven-shade-plugin for executable JAR

7. Change Build Directory

```
<build>
<directory>${project.basedir}/build_output</directory>
</build>
```

8. Skip Tests

```
mvn clean install -DskipTests  # compile but skip run
mvn clean install -Dmaven.test.skip=true  # skip compile & run
```

9. Generate Site / Coverage

```
mvn clean verify site
# reports in target/site
mvn dependency:analyze
mvn dependency:tree -Dverbose
```

10. Build Project

```
mvn clean install
```

- target/classes/, target/test-classes/, artifact.jar/war, surefire-reports, site/

11. Dependency Conflicts - Maven uses nearest-wins - Force version via <dependencyManagement>

```
mvn dependency:tree -Dverbose
mvn dependency:analyze
```

12. JUnit Test

```
// src/test/java/...
@Test void add() { assertEquals(5, calc.add(2,3)); }
```

- Run: `mvn test` - Compiled tests: target/test-classes/ - Reports: target/surefire-reports/

13. Executable JAR - maven-jar-plugin (manifest) or maven-shade-plugin (fat JAR)

```
mvn clean package
java -jar target/app-1.0-shaded.jar
```

14. Custom JAR

```
mvn install:install-file -Dfile=lib.jar -DgroupId=com.vendor -
DartifactId=vendor-lib -Dversion=1.0.0 -Dpackaging=jar
```

- Include in pom and confirm with `mvn dependency:tree`

15. Maven Web Project

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

- Structure: src/main/java, src/main/resources, src/main/webapp, src/test/java

16. Build & Deploy WAR

```
mvn clean package
cp target/myapp.war /path/to/tomcat/webapps/
```

17. JSTL & Servlet Dependencies

```
<dependency>
<groupId>jakarta.servlet.jsp.jstl</groupId>
<artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
<version>3.0.0</version>
</dependency>
<dependency>
<groupId>jakarta.servlet</groupId>
<artifactId>jakarta.servlet-api</artifactId>
<version>5.0.0</version>
<scope>provided</scope>
</dependency>
```

- Servlet API provided by container

18. Multi-Module Project - Parent POM `<packaging>pom</packaging>` - Modules: core (jar), web (war) - Web depends on core - Build all: `mvn clean install`

19. Maven Web Project vs Standalone - WAR: deployed to servlet container, uses provided scope - Standalone JAR: embedded server (Spring Boot), run `java -jar`

```
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
```