# Maven Java build tool

## CONCEPTS

Name a phase and Maven runs every earlier phase too: mvn package also compiles and tests.

**validate** check POM
↓
**compile** main sources
↓
**test** unit tests
↓
**package** jar / war
↓
**verify** integration tests
↓
**install** to ~/.m2
↓
**deploy** to remote repo

Default lifecycle: asking for a phase runs all phases before it

Goals are plugin:goal (dependency:tree). Phases are lifecycle steps (package) that run the goals bound to them.

## BUILD

**mvn compile**
Compile main sources

**mvn test**
Run unit tests

**mvn package**
Build the artifact into target/

**mvn verify**
Run integration tests and checks

**mvn install**
Copy artifact to local repo ~/.m2

**mvn clean**
Delete the target/ directory

**mvn clean install**
Fresh build, then install locally

**mvn deploy**
Publish to the remote repository

**mvn test-compile**
Compile test sources only

## OPTIONS

**-DskipTests**
Compile tests but do not run them

**-Dmaven.test.skip=true**
Skip compiling and running tests

## OPTIONS (cont.)

**-o / --offline**
Use only the local repository

**-q / --quiet**
Print errors only

**-e / --errors**
Show full stack traces

**-U / --update-snapshots**
Force re-check of SNAPSHOT deps

**-T** *<n>*
Build modules in parallel with n threads

**-B / --batch-mode**
Non-interactive, no colours (CI)

**-D***<key>*=*<value>*
Set a system property

## MULTI-MODULE

**mvn -pl** *<module>*
Build only this module

**mvn -pl** *<module>* **-am**
Also build what it depends on

**mvn -pl** *<module>* **-amd**
Also build modules depending on it

**mvn -pl '!***<module>***'**
Exclude a module

**mvn -rf :***<module>*
Resume the reactor from a module

## DEPENDENCIES

**mvn dependency:tree**
Print the resolved dependency tree

**mvn dependency:tree -Dincludes=***<g>***:***<a>*
Find who pulls in a library

**mvn dependency:analyze**
Find unused or undeclared deps

**mvn versions:display-dependency-updates**
List newer dependency versions

## INSPECT AND DEBUG

**mvn help:effective-pom**
Show the POM after inheritance

**mvn help:effective-settings**
Show merged settings.xml

## INSPECT AND DEBUG (cont.)

**mvn help:active-profiles**
List active profiles

**mvn help:evaluate -Dexpression=***<expr>* **-q -DforceStdout**
Print one POM value

**mvn -v**
Show Maven, Java and OS versions

**mvn -Dtest=***<Class>*#*<method>*
Run one test method

## PROFILES AND PROPERTIES

**mvn -P***<id>*
Activate a profile (-Pa,b for several)

## WRAPPER AND PROJECT

**mvn wrapper:wrapper**
Add mvnw and mvnw.cmd to the project

**./mvnw** *<args>*
Run with the pinned Maven version

### Where things live

| Path | Holds |
|---|---|
| `pom.xml` | project model |
| `~/.m2/settings.xml` | user settings, mirrors, servers |
| `~/.m2/repository` | local artifact cache |
| `target/` | build output |
| `.mvn/` | wrapper and jvm.config |

## RECIPES

### Fast CI build

```
mvn -B -ntp -T 1C clean verify
```

### Rebuild one module and its dependencies

```
mvn -pl services/api -am clean install -DskipTests
```

### Find and read a version conflict

```
mvn dependency:tree -Dverbose \
  -Dincludes=com.google.guava
mvn help:effective-pom | less
```

### Run a single test in a module

```
mvn -pl app -
Dtest=UserServiceTest#login \
  -Dsurefire.
failIfNoSpecifiedTests=false test
```