EPAM

Java Build Tool – Maven

Summary

1) Maven Concepts

Glossary: https://maven.apache.org/glossary.html

Artifact: read in glossary.

Maven **Coordinates**: identifies an artifact with groupId, artifactId and version.

Maven **version** types: release and snapshot: https://maven.apache.org/guides/getting-started/index.html#what-is-a-snapshot-version

Principle: convention over configuration.

Artifact Repository: store build artifacts and dependencies.

2) Install maven

M2_HOME environment variable should be created.

System settings file: <maven install folder>/conf/settings.xml

User settings file: maven <user-home>/.m2/**settings.xml**

Local repository folder: <user-home>/.m2/**repository**

3) Maven Project

3.a) Project structure

pom.xml file is located in the project folder that defines build configuration for maven.

Source files (in case of java project) are organized in following structure:

- production source code: **src/main/java** (and resources)
- test source: **src/test/java** (and resources)

In case of multi-module build, see Parent pom and Multi-module project

maven writes build result files in the **target** folder.

3.b) pom.xml

Reference: https://maven.apache.org/pom.html

pom.xml key sections

- project: project identification, parent, packaging type
- properties: named values for common reference, example: version numbers
- dependencies: project dependencies, see **dependencies**

- build: defines the project plugins, see **plugins**
- More project information: licenses, organization, developers, contributors, etc
- repositories: see **repositories**

super pom: defined by maven itself, each project inherits configuration from it.

Packaging type: jar by default

4) Dependencies

Read: https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html

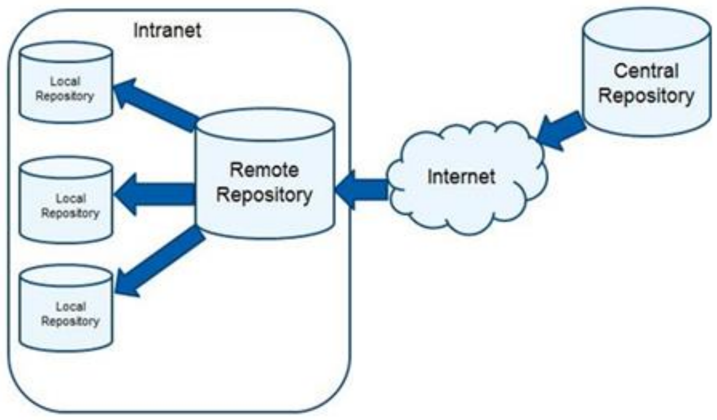
Transitive: maven will automatically include transitive dependencies. (If the project depends on A that depends on B, then B will be used as project dependency.)

Dependency scopes:

- **compile**: this is the default scope; dependency is required for both compilation and execution
- **provided**: dependency is required for compilation; at runtime it will be provided by container; example: servlet API or JDBC jar.
- **runtime**: dependency is not required for compilation, only at runtime; example: logback logging library
- **test**: dependency that is reuired for test compilation and execution; example: junit test framework
- system

5) Repositories

https://maven.apache.org/guides/introduction/introduction-to-repositories.html



Repository types:

- **Local**: developer machine (user home .m2 folder)
- **Remote**: accessed via some protocol, usually http or https; organizations usually operates their own repository in their internal network.

Central Repository is a special remote repository that is available to everyone on the internet: http://repo1.maven.org/maven2/

6) Plugins

Reading:

- https://maven.apache.org/guides/getting-started/index.html#how-do-i-use-plugins
- https://maven.apache.org/plugins/index.html

Maven is a plugin framework, all job is done by plugins.

Plugins are also identified by maven coordinate similar to artifacts coordinates (groupId, artifactId, version).

Plugins define **goals**, which are the tasks that the plugin offer for execution.

Plugin goal can be executed by maven in the following way: mvn <plugin>:<goal>

Example: `mvn compiler:compile`

Core pluings:

- **compiler**: java source code compilation
- **surefire**: junit test execution
- **failsafe**: junit integration test execution
- **install**: installs build artifact to local repository
- **deploy**: build artifact deployment to remote repository

Packaging plugins

- **jar**: creates jar file
- **war**: web application

Report plugins

- checkstyle: generates checkstyle report

7) Lifecycle and phases

Reference: https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

Build lifecycles

- default
- clean
- site

Lifecycle phases:

- validate
- compile
- test
- package
- verify
- install
- deploy

Plugin and phase relation: plugin goals are **bound to lifecycle phases**. Plugin goals are automatically executed by maven based on the binding.

Related commands:

- mvn clean – cleans the project
- mvn install – executes lifecycle phases until install

8) Parent pom and Multi-module project

There are two concepts in maven:

- **Parent pom** – inheritance, configurtion reuse
- **Aggregator pom** – multi-module project definition

Usually the multi-module project defines a single pom.xml file that plays both roles: parent pom and aggregator pom.

Multi-module maven project defines pom.xml file at the root level, and one sub-folder for each module.

- The packaging type of the main pom.xml file is **pom**
- The main pom file lists the modules in its **modules** tag.
- The main pom also serve as common configuration, so each module refer the parent pom in the **parent** tag.
- One model can refer another in dependencies section (dependency should be non-recursive)
- Best practice: modules inherit groupId and versionId from the parent.

See details: https://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Project_Inheritance_vs_Project_Aggregation

9) More topics

profile

site generation

10) Useful Maven commands

https://www.digitalocean.com/community/tutorials/maven-commands-options-cheat-sheet