maven-notes

September 3, 2018

Contents

Part I

POM

Chapter 1

Overview, Content

Project Object Model, an XML representing of project resources. Shoud be **in the root directory of the project** it belongs to.

1.1 Build Life-Cycles, Phases, Goals

It's a tree:

- **build life-cycles** contain:
- ...**phases**, they contain:
- **goals**

The interaction with a Maven is by sending a life-cycle, phase or goal name as a command. The command executes also **all predecessors**.

1.2 Dependencies and Repositories

JARs, libraries, local and remote repositories - It is one of the first Maven responsiblity to check them.

1.3 Build Plugins

Can be custom or predefined. Set of actions not covered by the standard Maven, but can be added as plugins.

1.4 Build Profiles

They are used when a program need to be build in different ways. For instance testing and deployment version.

Chapter 2

POM Syntax

2.1 Elements

Here are all elements of POM:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<!-- The Basics -->
<groupId>...</groupId>
<artifactId>...</artifactId>
<version>...</version>
<packaging>...</packaging>

<dependencies>...</dependencies>

<parent>...</parent>
<dependencyManagement>...</dependencyManagement>
<modules>...</modules>
<properties>...</properties>


<!-- Build Settings -->
<build>...</build>
<reporting>...</reporting>


<!-- More Project Information -->
```

```
<name>...</name>
<description>...</description>
<url>...</url>
<inceptionYear>...</inceptionYear>
<licenses>...</licenses>
<organization>...</organization>
<developers>...</developers>
<contributors>...</contributors>


<!-- Environment Settings -->
<issueManagement>...</issueManagement>
<ciManagement>...</ciManagement>
<mailingLists>...</mailingLists>
<scm>...</scm>
<prerequisites>...</prerequisites>
<repositories>...</repositories>
<pluginRepositories>...</pluginRepositories>
<distributionManagement>...</distributionManagement>
<profiles>...</profiles>
</project>
```

- modelVersion defines the version of POM. Version 4.0.0 is the only version supported by Maven 2 and 3 and is required.
- the minimum POM must contain:
  - preamble - opennimg project tag with the atributes set as above
  - groupId:artifactId:versionId fields, however **groupId and versionID can be inherited**

2.2 Example

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.jenkov</groupId>
    <artifactId>java-web-crawler</artifactId>
    <version>1.0.0</version>
</project>content...
```

groupId, **artefactId**, **version** tags, as well as **dots** are used to create directory structure for a project. For instance a JAR file created using this POM would be placed in

```
MAVEN_REPO_dir/com/jenkov/java-web-crawler/1.0.0/java-web-crawler-1.0.0.jar
```

2.2.1 modelVersion

Use **4.0.0** for Maven version 2 and 3.

2.2.2 groupId

Usage:

- **project** name, usually *the root java package* of the project, or
- **organisation** name

Maven repository matches its directory structure with the groupID. Each dot represents a subdirectory. In this example it is ***MAVEN_REPO****/com/jenkov*. The MAVEN_REPO is a path variable which is replaced by actuall path to Maven repository.

2.2.3 artifactId

It contains the name of the project. This name is also used as the name of JAR file produced when a project is built.

2.2.4 version

Nothing unusual.

2.2.5 packaging

jar, **maven-plugin**, also *war*, *ear*, *rar*.When packaging is not defined then **jar is assumed** by Maven.

Chapter 3

Inheritance

3.1 Overview

POMs can inherit from parent POM.

3.2 Effective POM

Show the Maven file which will be executed after **all inheritance is aplied**.

```
mvn help:effective-pom
```

Part II

Setting Files

Chapter 4

Overview

`settings.xml`

They are used to define repository locations, profile files. Two standard locations (both otpional, however):

- the **Maven installation directory**: *$M2_HOME/conf/settings.xml*
- the **user's home directory**: *${user.home}/.m2/settings.xml*

Part III

Executing Maven

Chapter 5

phase

`mvn phase1 phase2...`

...executes phases and **all their predecessors**. For instance:

`mvn clean install`

Chapter 6

goal

```
mvn phase:goal
```

Part IV

Maven Directory Structure

Chapter 7

Standard Directory Structure

```
- src
    - main
        - java
        - resources
        - webapp
    - test
        - java
        - resources

- target
```

If this structure is followed then there is no need to specify above directories. ***src*** is the root directory of **source code** (in ***main***) and **test files** (in ***test***).

target is created by Maven. It contains all output files (binaries, Jars, etc.). ***mvn clean*** removes all files from this dear

Part V

Dependency Management

Chapter 8

Overview

It is Maven built-in tool, which download external dependencies and also recursively all **transitive** dependencies in a depndance tree. They are downloaded from the central Maven repository, however, only when not present in the local repository. If a dependency is missing in the central repository (**external**), it can be downloaded and added anually. **The directory structure must much POM!** See syntax below.

Chapter 9

Syntax

Each dependency is described by its:

- groupId
- artifactId
- version

9.1 Elements

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.0</version>
    <type>jar</type>
    <scope>test</scope>
    <optional>true</optional>
</dependency>}
```

- **groupID, artifactId, version** - dependency coordinates
- **type** - correspond to depndant **packaging** type. It also **defaults to jar**.

9.2 Example

```
<dependencies>

    <dependency>
        <groupId>org.jsoup</groupId>
        <artifactId>jsoup</artifactId>
        <version>1.7.1</version>
```

```
    </dependency>

    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.8.1</version>
        <scope>test</scope>
    </dependency>

</dependencies>
```

This POM will result with following directories:

```
MAVEN_REPOSITORY_ROOT/junit/junit/4.8.1

MAVEN_REPOSITORY_ROOT/org/jsoup/jsoup/1.7.1
```

Chapter 10

External Dependencies

They are the dependencies not present in Maven repositories (neither local, central or remote).

10.1 Syntax Example

```
<dependency>
    <groupId>mydependency</groupId>
    <artifactId>mydependency</artifactId>
    <scope>system</scope>
    <version>1.0</version>
    <systemPath>${basedir}\war\WEB-INF\lib\mydependency.jar</systemPath>
</dependency>
```

- **groupId** and **artifactId** are set to the name of the dependency
- standard **scope** s *system*
- **systemPath** points to the location of the JAR file.
  - **${basedir}** is a variable storing the path to the POM

Chapter 11

Snapshot Dependencies

When set then Maven always downloads the latest version of the dependency. The frequency of updates checking is configurable. Snapshot can be set:

- for **entire project** at the beginning of a POM

  ```
  <version>1.0-SNAPSHOT</version>
  ```

- for a **particular dependency**

Part VI

Maven Repositories

Chapter 12

Overview

Three types:

- local
- central
- remote

When maven looks for dependencies then looks for them in above directories in this order. Each dependency is a JAR file with associated POM, which contains the information of further dependencies. Ths allows to download recursively entire dependency tree.

Chapter 13

Local Repository

This is a directory on the developer's pc. It contains all dependencies Maven has downloaded. Each dependency is downloaded only once, then it is shared between projects, if needed. Default location of the repository can be changed by updating settings.xml, tag <*localRepository*>.

Chapter 14

Central Repository

This community maintained repository. Maven looks it up when a dependency is not present in the local repository. No configuration is required.

Chapter 15

Remote Repository

Maintained anywhere by a web server. Usually used to host projects internal to an organisation.

Part VII

Build Life-Cycles, Phases, Goals

Chapter 16

Build Life-Cycles

There are 3 built-in

1. default
2. clean
3. site

default deals with compilation and packaging. **It cannot be executed directly!** Some contained phase or goal has to be invoked instead. Examples of predefined phases:

- *validate* - checks a project integrity (all dependencies downloaded, project is correct)
- *compile*
- *test*
- *package* - generates a JAR file
- *install* - installs a project into the **local repository** for use as a **dependency** for other local projects.
- *deploy* - copies the project to **the remote repository** for sharing with other developers and projects.

clean removes all files from *target* directory

site generates a documentation.

Part VIII

IDE Support

Chapter 17

Eclipse

17.1 Creating a Simple Project

- 'new'
- 'other'
- 'Maven project'
- on "New Maven Project' screen **check 'Create simple project'** in order to skip the window with artifact selection.
- provide Maven coordinates and *finish*.

Part IX

Appendix

Chapter 18

Commands

18.1 Executing Maven

18.1.1 phase

```
mvn phase1 phase2...
```

...executes phases and **all their predecessors**. For instance:

```
mvn clean install
```

18.1.2 goal

```
mvn phase:goal
```

18.2 Version

```
mvn --version
```

or

```
mvn -v
```

18.3 Effective POM

```
mvn help:effective-pom
```

18.4 Clean Up

```
mvn clean
```

...removes all files from *target* directory.

Chapter 19

Vocabulary

19.1 P

19.1.1 POM

Project Object Model, an XML representing of project resources. Shoud be in the root directory of the project it belongs to.

Chapter 20

Resources

20.1 Web

- http://maven.apache.org/pom.html#Maven_Coordinates
- http://tutorials.jenkov.com/maven/maven-tutorial.html
- https://maven.apache.org/guides/index.html