
Apache
maven

Introduction

What is Maven?

= Project management tool

- Manage Java Projects
- Generate documentation

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running be.invetory.service.impl.ReadingServiceTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.18 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ InventoryApp ---
[INFO] Building jar: C:\Users\vangike\IdeaProjects\InventoryApp\target\InventoryApp-1.0.jar
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ InventoryApp ---
[INFO] Installing C:\Users\vangike\IdeaProjects\InventoryApp\target\InventoryApp-1.0.jar to C:\Users\vangike\.m2\repository\be\inventory\InventoryApp\1.0\InventoryApp-1.0.jar
[INFO] Installing C:\Users\vangike\IdeaProjects\InventoryApp\pom.xml to C:\Users\vangike\.m2\repository\be\inventory\InventoryApp\1.0\InventoryApp-1.0.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.403 s
[INFO] Finished at: 2017-09-04T10:57:19+02:00
[INFO] Final Memory: 20M/186M
[INFO] ------------------------------------------------------------------------
```

What is Maven?

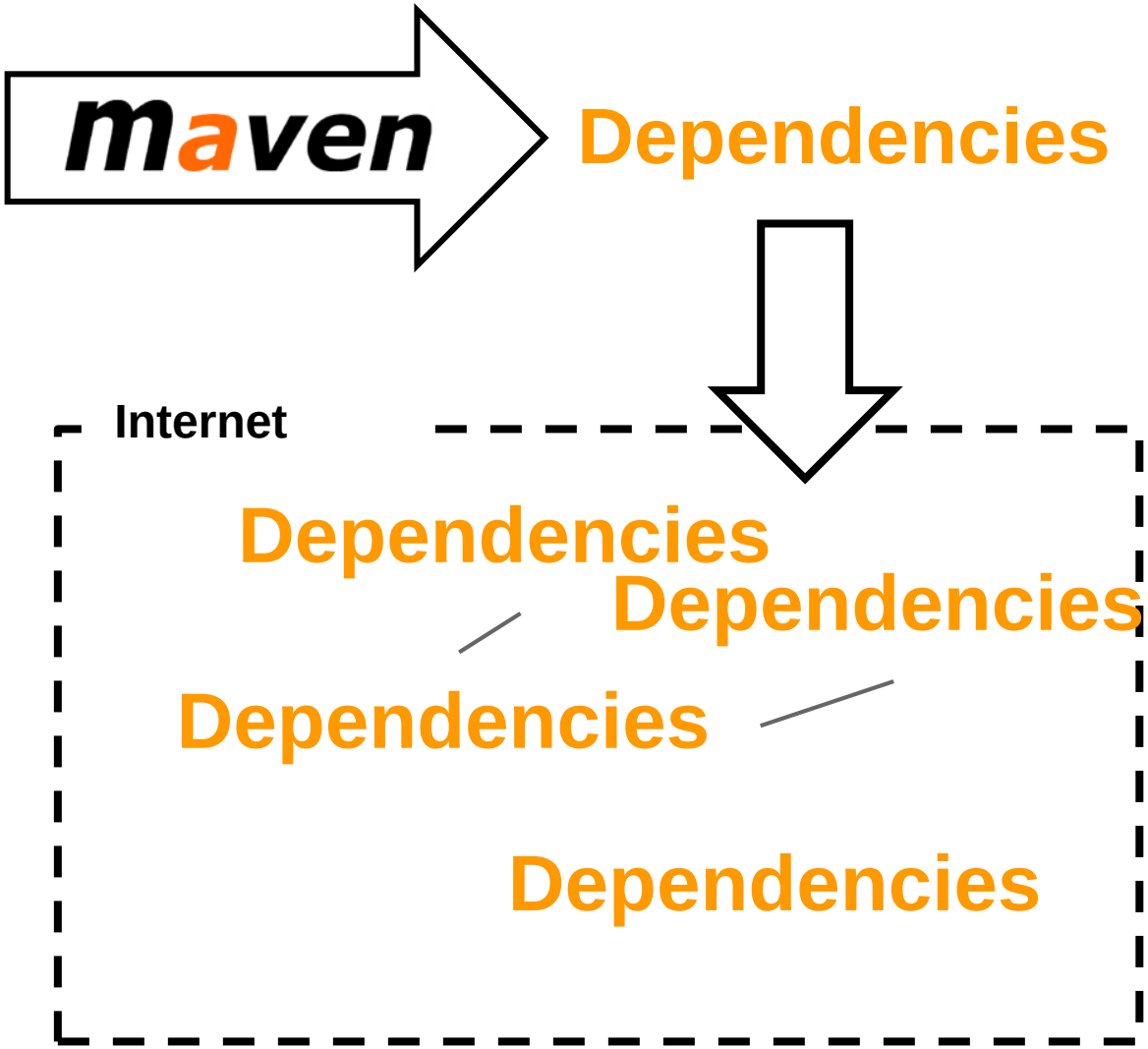
- A tool to built, package and deploy software.
- Based on Ant
- "**Convention over configuration**"

Convention over configuration ?

- Project Structure is fixed
- Project Object Model (POM): configuration file
- All Maven projects look alike
- IDE independent

Dependencies

jar's
Libraries



Installation & Configuration

Download Maven

Veilig | https://maven.apache.org/download.cgi

Apps | Mobile | Android | Somedi | @venture | Activiteitengids | PXL_Hasselt | Tutorials | Oak3 | Argenta Digital | Pivotal (Spring) | Ninite | Maven | Git course

Apache Maven Project
http://maven.apache.org/

Maven™

Apache / Maven / Download Apache Maven

Last Published: 2017-09-02

MAIN
- Welcome
- License
- Download
- Install
- Configure
- Run
- IDE Integration

ABOUT MAVEN
- What is Maven?
- Features
- FAQ
- Support and Training

DOCUMENTATION
- Maven Plugins
- Index (category)
- Running Maven
- User Centre
- Plugin Developer Centre
- Maven Central Repository
- Maven Developer Centre
- Books and Resources
- Security

COMMUNITY
- Community Overview
- How to Contribute

Downloading Apache Maven 3.5.0

Apache Maven 3.5.0 is the latest release and recommended version for all users.

The currently selected download mirror is **http://apache.belnet.be/**. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are *backup* mirrors (at the end of the mirrors list) that should be available. You may also consult the complete list of mirrors.

Other mirrors: http://apache.belnet.be/ Change

System Requirements

Java Development Kit (JDK)	Maven 3.3+ require JDK 1.7 or above to execute - they still allows you to build against 1.3 and other JDK versions by Using Toolchains
Memory	No minimum requirement
Disk	Approximately 10MB is required for the Maven installation itself. In addition to that, additional disk space will be used for your local Maven repository. The size of your local repository will vary depending on usage but expect at least 500MB.
Operating System	No minimum requirement. Start up scripts are included as shell scripts and Windows batch files.

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the installation instructions. Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public KEYS used by the Apache Maven developers.

	Link	Checksum	Signature
Binary tar.gz archive	apache-maven-3.5.0-bin.tar.gz	apache-maven-3.5.0-bin.tar.gz.md5	apache-maven-3.5.0-bin.tar.gz.asc
Binary zip archive	apache-maven-3.5.0-bin.zip	apache-maven-3.5.0-bin.zip.md5	apache-maven-3.5.0-bin.zip.asc
Source tar.gz archive	apache-maven-3.5.0-src.tar.gz	apache-maven-3.5.0-src.tar.gz.md5	apache-maven-3.5.0-src.tar.gz.asc
Source zip archive	apache-maven-3.5.0-src.zip	apache-maven-3.5.0-src.zip.md5	apache-maven-3.5.0-src.zip.asc

Configuration

JAVA_HOME and MAVEN_HOME: add environment variables

JAVA_HOME	C:\Program Files\Java\jdk1.8.0_144
MAVEN_HOME	C:\Program Files\Apache Maven\apache-maven-3.5.0

Add them to PATH variable:

%JAVA_HOME%/bin en %MAVEN_HOME%/bin

Test Maven

```
C:\Users\vangike>mvn --version
Apache Maven 3.5.0 (ff8f5e7444045639af65f6095c62210b5713f426; 2017-04-03T21:39:06+02:00)
Maven home: C:\Program Files\Apache Maven\apache-maven-3.5.0\bin\..
Java version: 1.8.0_144, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_144\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

CMD: mvn --version

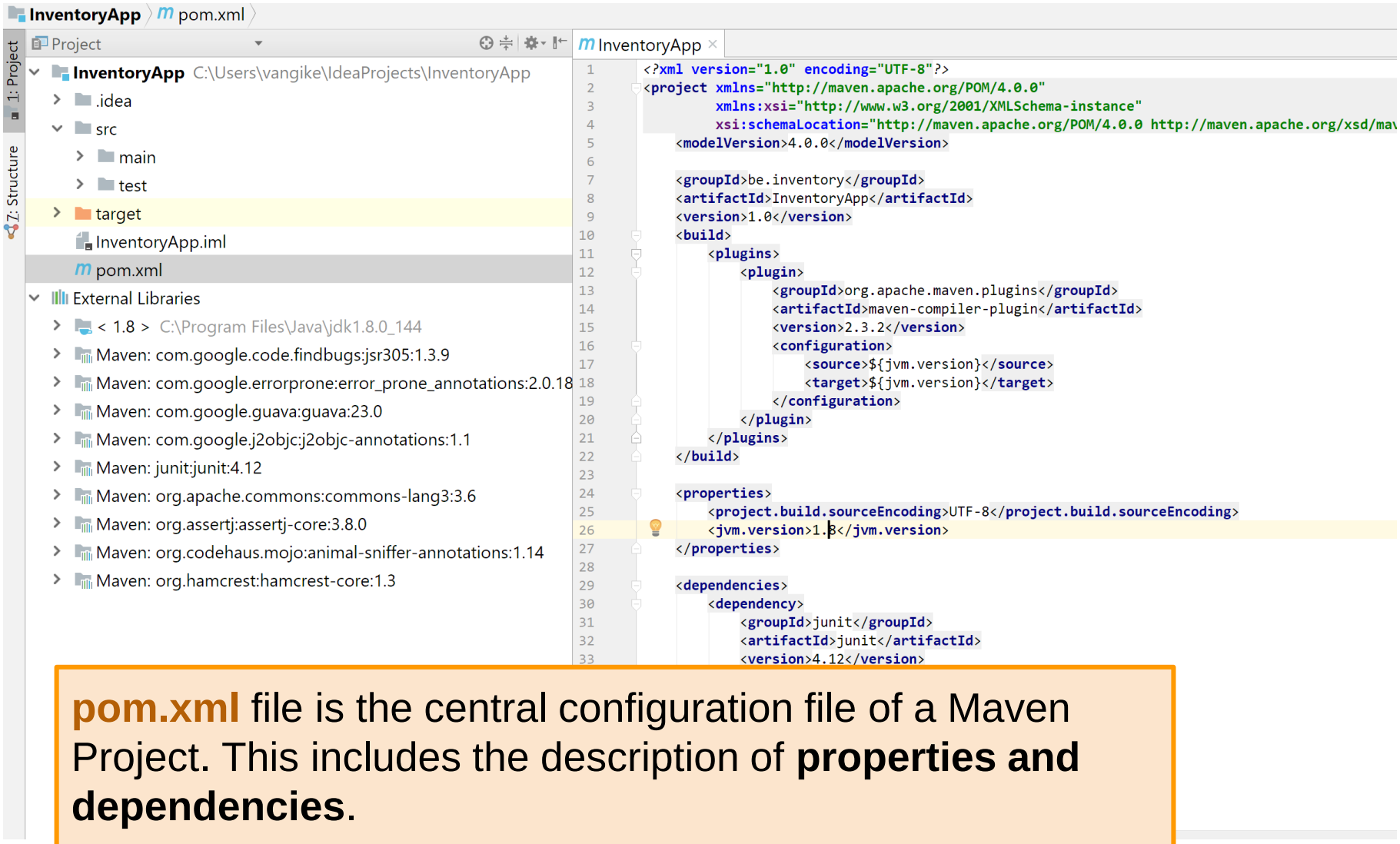
Deeper look

Project Structure

Project structure is always the same

src/main/java	Java sourcecode
src/main/resources	Property files, images ...
src/test/java	Java sourcecode for tests
src/test/resources	Resources we need in tests
target	Generated artifact: Jar, war ...
target/classes	Compiled classes
target/test-classes	Compiled test classes
pom.xml	Project Object Model

POM = Project Object Model



- InventoryApp C:\Users\vangike\IdeaProjects\InventoryApp
  - .idea
  - src
    - main
    - test
  - target
  - InventoryApp.iml
  - pom.xml
- External Libraries
  - < 1.8 > C:\Program Files\Java\jdk1.8.0_144
  - Maven: com.google.code.findbugs:jsr305:1.3.9
  - Maven: com.google.errorprone:error_prone_annotations:2.0.18
  - Maven: com.google.guava:guava:23.0
  - Maven: com.google.j2objc:j2objc-annotations:1.1
  - Maven: junit:junit:4.12
  - Maven: org.apache.commons:commons-lang3:3.6
  - Maven: org.assertj:assertj-core:3.8.0
  - Maven: org.codehaus.mojo:animal-sniffer-annotations:1.14
  - Maven: org.hamcrest:hamcrest-core:1.3

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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/mav
    <modelVersion>4.0.0</modelVersion>

    <groupId>be.inventory</groupId>
    <artifactId>InventoryApp</artifactId>
    <version>1.0</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.3.2</version>
                <configuration>
                    <source>${jvm.version}</source>
                    <target>${jvm.version}</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <jvm.version>1.8</jvm.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
```

pom.xml file is the central configuration file of a Maven Project. This includes the description of **properties and dependencies**.

Maven commands

mvn archetype:generate → Create project structure

mvn package → Compile and pack in JAR file (target folder)

mvn clean → Delete compiled classes and jar file

Can be used together: mvn clean package

Exercise

Page 5: ex. 2

De POM = Project Object Model

Simple POM:

<modelVersion>: Version number of the POM

<groupId>: Owner of the project (package name)

<artifactId>: Projectname

<version>: Version of the project

Dependencies

= Depend on other projects→ Use existing code as much as possible.

Add dependencies(in POM.xml)

https://mvnrepository.com/

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.6</version>
</dependency>
```

Exercise

Page 10: Ex. 3

Dependencies scope

- **compile:** Jar file is included in final jar or war of the project= **DEFAULT**
- **test:** Jar file is only necessary during tests. **Example**: JUnit
- **runtime:** Jar file is included in final Jar or War of the project but is unnecessary at comile time. **Example:** mysql-connector-java.
- **provided:** the jar file is not included in the final jar/ war, because this is provided by the runtime environment
  Example: javax.servlet-api, javax.servlet.jsp-api artefacts.
- **system:** is identical to runtime but the path must be explicitly specified.
- **import:** This is used when the artifact is another POM and dependencies need to be replaced.

Artefact

= a special piece of software that is needed to make the final software

- An **artefact** consist of a
  - groupId
  - artefactId
  - version

Lifecycle

What is a lifecycle?

- The process Maven goes through during the construction of a project. (E.G. Compiling the project)
- Is described in the POM and consists of several phases. A **phase** cosist of one or more actions that lead to a **goal**.
- It is possible to intervene in every phase by executing extra plug-ins

Lifecycles

- ❑ Clean lifecycle
- ❑ Default lifecycle
- ❑ Site lifecycle

Clean & Site Lifecycle

Clean Lifecycle

Delete all files created by a previous built

Phase	Goal	Description
pre-clean		Preparation
clean	clean:clean	Deletig created files
post-clean		Activities after cleaning

Site Lifecycle

Maven can generate files for webprojects.

Phase	Goal	Description
pre-site		Preparation
site	site:site	Generate site
post-site		Activities after generating site
site-deploy	site:deploy	Deploy site

Default Lifecycle

Purpose of the **default lifecycle:**

- Compile project
- Test project
- Create jar/war file
- Optional: deploy on server

Clean Lifecyle
pre-clean
clean
post-clean

Default Lifecyle	
validate	test-compile
initialize	process-test-classes
generate-sources	test
process-sources	prepare-package
generate-resources	**package**
process-resources	pre-integration-test
compile	integration-test
process-classes	post-integration-test
generate-test-sources	verify
process-test-sources	**install**
generate-test-resources	**deploy**
processs-test-resources	

Site Lifecyle
pre-site
site
post-site
site-deploy

Phase

Execute specific phase: mvn nameOfThePhase

E.G.: mvn package → packaging type?

```
<groupId>be.oak3.hello</groupId>
<artifactId>HelloMaven</artifactId>
<version>1.0.0</version>
<packaging>jar</packaging>
```

Exercise

Page 14 ex. 4

Page 21 ex. 6

Repositories

Global Repository

http://repo1.maven.org/maven2/ : Global Maven Repository

Look for one: http://mvnrepository.com/

If the dependency is not available in the global maven repository, we have to add the repository to the POM.

```
<repositories>
    <repository>
        <id>noelvaes</id>
        <url>https://www.noelvaes.eu/maven/repository/student</url>
    </repository>
</repositories>
```

Local Repository

All files of the global repository will be saved locally
= local cache

C:\Users\vangike\.m2\repository

mvn install → this specific module is available in other projects as well.

Own Repository

Share modules with others

Closed-source libraries

Own development

→ Maven repository Manager (Bv: Nexus)

mvn deploy: install module in own repo

Properties

Properties

- Values defined in POM or somewhere else
- **${propertyName}:** Request property in POM

MAVEN Properties

```
<build>
        <finalName>${project.artifactId}</finalName>
</build>
```

Environment variables (OS):

```
<build>
        <finalName>${project.artifactId}_${env.USERNAME}</finalName>
</build>
```

Properties

System properties (java.lang.System) :

- ${java.version}
- ${file.separator}

Self defined properties:

→ In pom.xml:

```
<properties>
    <jvm.version>1.8</jvm.version>
</properties>
```

```
<configuration>
    <source>${jvm.version}</source>
    <target>${jvm.version}</target>
</configuration>
```

Resource filtering

= Use properties in resource folder.

E.G.: Log4j.properties file

pom.xml:

```
<properties>
        <debug.level>DEBUG</debug.level>
</properties>
```

log4j.properties:

```
log4j.rootLogger=${debug.level}
```

Resource filtering

Should be activated in POM file

```
<build>
    <resources>
        <resource>
            <directory>src/main/resources</directory>
            <filtering>true</filtering>
        </resource>
    </resources>
</build>
```

Exercise

Page 28: ex. 8
Page 30: ex. 9

Profiles

Inleiding

DEV – TEST – PROD

Development omgeving: test db, extra logging

Productie omgeving: prod db, specific logging

→ Maven profiles

Create profile

PROD: change settings for PROD

```
<profiles>
    <profile>
        <id>PROD</id>
        <properties>
            <log.level>WARN</log.level>
        </properties>
    </profile>
</profiles>
```

Activate profile: mvn –P PROD package

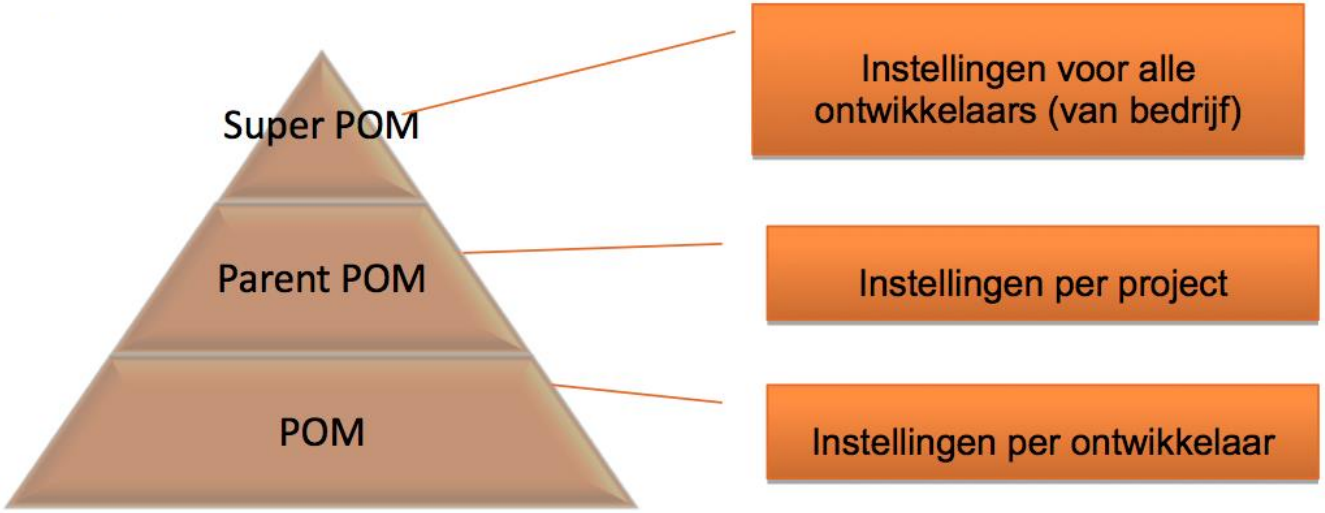
Exercise

Page 33: ex. 11

Super POM

Overerving

A POM file can inherit information from another POM file. The last one is called the Parent POM. An entire hierarchy is possible. At the top is the Super POM which contains all default settings.



Inherit from other POM:

```
<parent>
    <groupId>be.oak3.parentalpom</groupId>
    <artifactId>ParentPom</artifactId>
    <version>1.2.0</version>
</parent>
```

Dependency/ Plugin Managment

Add frequently used dependencies / plugins in parent pom.

In child pom suffices to refer to the dependency (so no version number)

Demo

Page 36: ex. 12

Multiple modules

AddModules

```
<packaging>pom</packaging>
<modules>
        <module>Module1</module>
        <module>Module2</module>
</modules>
```

```
Course_Maven
        Module1
                pom.xml
        Module2
                pom.xml
        pom.xml (most of the time this is the parent pom)
```

Demo multimodule

Sportpaleis