



Inleiding

Wat is Maven ?

= Project management tool

- Java project vormgeven en beheren
- Genereren van rapporten, documentatie ...

TESTS

```
-----  
Running be.inventory.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] -----
```

Wat is Maven ?

- Een tool waarmee, onafhankelijk van de gebruikte IDE, software gebouwd, gepackaged en gedeployed kan worden.
- Een variant van Ant (volgt een andere filosofie)
- Volgt het principe van “**convention over configuration**”

Convention over configuration ?

- Veel zaken zijn op voorhand afgesproken. Projectstructuur is vast bepaald
- Configuratie is beperkt tot één enkele bestand (POM)
- Maven projecten lijken daardoor op elkaar.
- Maven projecten kan je openen in om het even welke IDE die Maven ondersteunt (zoals intelliJ & eclipse)

Dependencies

jar's
libraries
softwarebibliotheeken

De jar's, libraries, softwarebibliotheeken,... noem je in een Maven project een **dependency**

maven

Dependencies

Wanneer je een project hebt dat gebruikt maakt van Hibernate of Spring zullen deze dependencies terug afhangen van andere dependencies. Maven zal bij het downloaden van de dependency nagaan welke dependencies er nog nodig zijn en ook deze downloaden.

- Het internet - - - - -

Dependencies
Dependencies

Dependencies

Dependencies

Installatie & configuratie

Maven downloaden

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

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

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: [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

Configuratie van de werkomgeving

JAVA_HOME en MAVEN_HOME omgevingsvariabelen toevoegen.

JAVA_HOME

C:\Program Files\Java\jdk1.8.0_144

MAVEN_HOME

C:\Program Files\Apache Maven\apache-maven-3.5.0

Nadien toevoegen in de PATH variabele:

%JAVA_HOME%/bin en %MAVEN_HOME%/bin

Testen van 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"
```

Open een command window en type mvn -version. Je ziet dan welke versie van Maven je hebt.

Meer over Maven

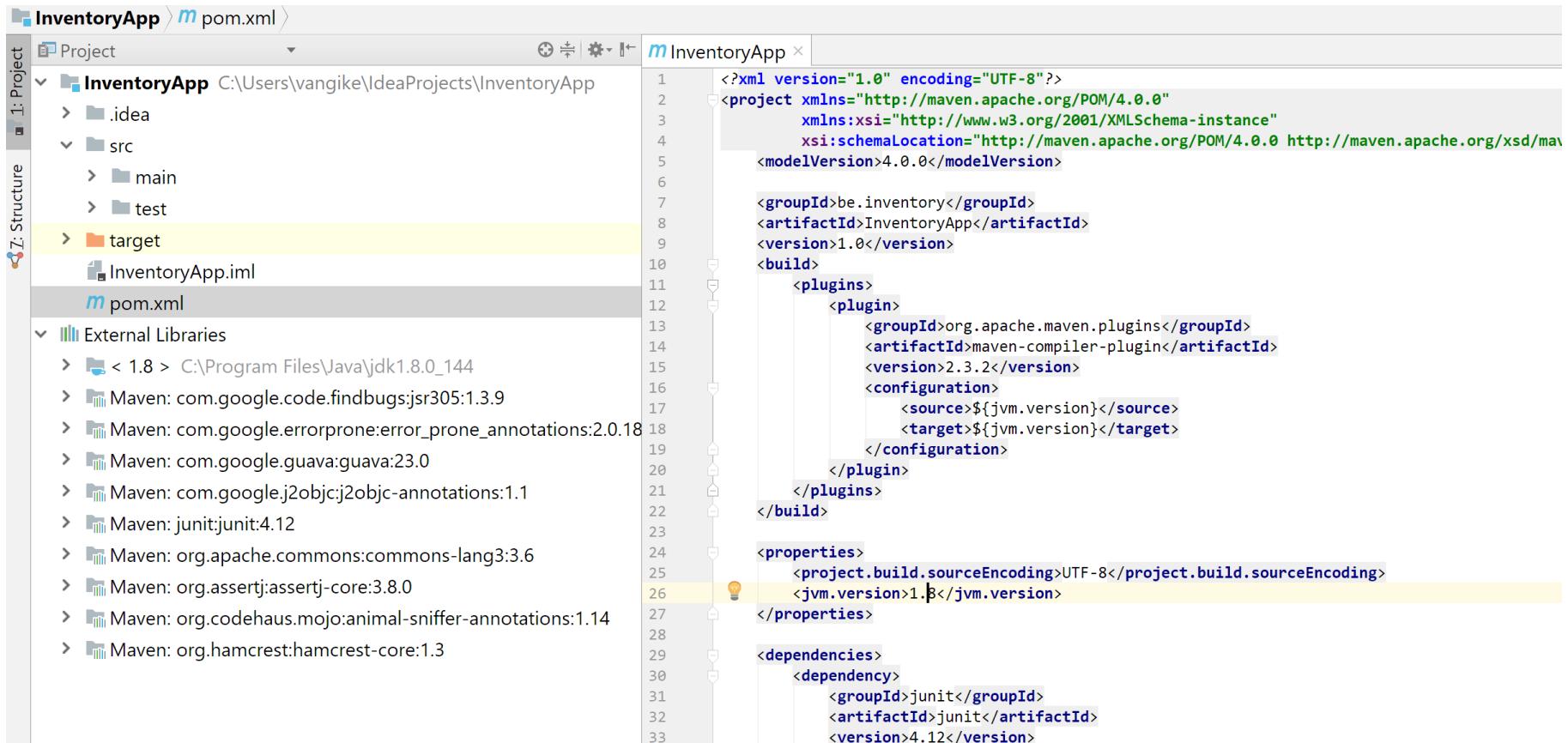
De directorystructuur

In Maven ligt de directory structuur vast.

voordeel: kan geopend worden in elke IDE die Maven ondersteunt

src/main/java	Java broncode
src/main/resources	Property bestanden, afbeeldingen, geluidsfragmenten,...
src/test/java	Java broncode van de testklassen
src/test/resources	Bestanden die we enkel nodig hebben voor testen
target	Gegenereerde artifacts: JAR – WAR
target/classes	Gecompileerde klassen
target/test-classes	Gecompileerde test-klassen
pom.xml	Project Object Model

De POM = Project Object Model



The screenshot shows the IntelliJ IDEA interface with the project 'InventoryApp' open. On the left, the Project and Structure toolbars are visible, along with a list of external libraries. The main window displays the contents of the `pom.xml` file, which defines the Maven project structure, build configuration, properties, and dependencies.

```
<?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
1      <modelVersion>4.0.0</modelVersion>
2
3      <groupId>be.inventory</groupId>
4      <artifactId>InventoryApp</artifactId>
5      <version>1.0</version>
6
7      <build>
8          <plugins>
9              <plugin>
10                 <groupId>org.apache.maven.plugins</groupId>
11                 <artifactId>maven-compiler-plugin</artifactId>
12                 <version>2.3.2</version>
13                 <configuration>
14                     <source>${jvm.version}</source>
15                     <target>${jvm.version}</target>
16                 </configuration>
17             </plugin>
18         </plugins>
19     </build>
20
21     <properties>
22         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
23         <jvm.version>1.8</jvm.version>
24     </properties>
25
26
27     <dependencies>
28         <dependency>
29             <groupId>junit</groupId>
30             <artifactId>junit</artifactId>
31             <version>4.12</version>
32         </dependency>
33     </dependencies>

```

De **pom.xml** file is het centrale configuratiebestand van een Maven project. Daarin bevindt zich onder andere de beschrijving van de **properties** en de **dependencies**.

Maven commando's: inleiding

`mvn archetype:generate` → Aanmaken mappenstructuur

`mvn package` → Compileren en inpakken in JAR (in de target map)

`mvn clean` → Gecompileerde klassen en JAR verwijderen

Samen: `mvn clean package`

Opdracht

Maak opdracht 20 in de cursus (p.53)

De POM = Project Object Model

Eenvoudige POM:

<modelVersion>: Versienummer van de pom

<groupId>: Eigenaar vh project (omgekeerde domeinnaam)

<artifactId>: Projectnaam

<version>: Versie van het project

Dependencies

= Afhankelijk van ander project → Zoveel mogelijk beroep doen op reeds bestaande code

Dependencies toevoegen (in POM.xml)

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

<https://mvnrepository.com/>

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

Opdracht

Opdracht 21 p. 57-58

Dependencies scope

De mogelijke waarden voor scope zijn:

- **compile:** Het jar bestand wordt opgenomen in het uiteindelijke jar/war bestand waarin de applicatie zit. = **DEFAULT**
- **test:** Het jar bestand is alleen nodig tijdens de testfase. Het typische voorbeeld hiervan is het JUnit artifact.
- **runtime:** het jar bestand wordt opgenomen in het uiteindelijke jar/war bestand maar is overbodig tijdens het compileren.
Voorbeeld: mysql-connector-java.
- **provided:** het jar bestand wordt NIET opgenomen in het uiteindelijke jar/war bestand, want dit wordt geleverd door de runtime omgeving.
Voorbeeld: javax.servlet-api en javax.servlet.jsp-api artefacten.
- **system:** is identiek aan runtime maar het pad moet expliciet worden opgegeven.
- **import:** Dit wordt gebruikt wanneer het artefact een andere POM is en dependencies moeten worden vervangen.

Artifacts

- Engelse term voor artefact
= een speciaal stukje software dat nodig is om de uiteindelijke software te kunnen maken.
- Een **artifact** heeft onder andere een
 - groepsId
 - artifactId
 - versienummer

Uitleg versie notatie:

Wanneer het belangrijk is dat een bepaalde versie van de dependency of plugin gebruikt wordt, noteer je die als volgt: `<version>2.2.1</version>`

andere mogelijkheden: <https://maven.apache.org/enforcer/enforcer-rules/versionRanges.html>

Lifecycle

Wat is een lifecycle?

- Het **traject** die Maven doorloopt tijdens het opbouwen van een project tot het gevraagde eindresultaat: bijvoorbeeld het compileren van het project.
- Wordt beschreven in de POM en bestaat uit meerdere **fases**. Zo'n fase bestaat dan weer uit een of meerdere acties die tot een **goal** leiden.
- Het is mogelijk om in te grijpen in elke fase door er **extra plug-ins** te laten uitvoeren.

Soorten lifecycles

- Clean lifecycle
- Default lifecycle
- Site lifecycle

Clean & Site Lifecycle

Clean Lifecycle

In deze lifecycle worden alle bestanden, die gemaakt werden tijdens de vorige build, verwijderd.

Fase	Goal	Omschrijving
pre-clean		Voorbereiding op de opkuis
clean	clean:clean	Aangemaakte bestanden worden verwijderd
post-clean		Activiteiten na de opkuis

Site Lifecycle

Wanneer je niet alleen werkt zijn afspraken over codering een absolute noodzaak. Maven kan ook webprojecten met daarvoor nuttige informatie genereren

Fase	Goal	Omschrijving
pre-site		Voorbereiding op het maken van de site
site	site:site	Het genereren van de site
post-site		Activiteiten na het genereren
site-deploy	site:deploy	Het in werking stellen van de site

Default Lifecycle

De **default lifecycle** heeft tot doel

- het project te compileren
- het project te testen
- een JAR/WAR-bestand te maken
- deze eventueel te deployen op een server

Clean Lifecycle	Default Lifecycle	Site Lifecycle
pre-clean	validate	test-compile
clean	initialize	process-test-classes
post-clean	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
	process-test-resources	

Fase uitvoeren

Specifieke fase uitvoeren: mvn naamVanDeFase

Bv: mvn package → packaging type?

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

Welke goals worden er uitgevoerd?
(Zie schema p.65)

Opdracht

Opdracht 25 p. 69

→ Voor de specifieke commando's kijk even kort in de voorbeelden erboven.

Repositories

Globale Repository

<http://repo1.maven.org/maven2/> : Globale repository van maven

Specifiek eentje zoeken:

<http://mvnrepository.com/>

Als deze niet in de globale van Maven staat moeten we de repository zelf toevoegen:

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

Lokale Repository

Bestanden uit globale repo worden lokaal opgeslagen!
= lokale cache

C:\Users\vangike\.m2\repository

mvn install → Deze module is nu ook bruikbaar in andere projecten

Eigen Repository

Modules delen met anderen

Closed-source libraries

Eigen ontwikkelingen

→ Maven repository Manager (Bv: Nexus)

`mvn deploy`: module in eigen repo installeren

Opdracht

Maak opdracht 26 in de cursus (p.72-73)

Properties

Properties

- Waarden gedefineerd in POM of ergens anders
- **\${propertyName}** : opvragen van de property in de POM

MAVEN Properties

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

OmgevingsVariabelen (ingesteld in OS):

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

Properties

Systeemproperties (java.lang.System) :

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

Zelf gedefinieerde properties:

→ In pom.xml:

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

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

Resource filtering

= Properties gebruiken in resource folder.

Bv: Log4j.properties file

pom.xml:

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

log4j.properties:

```
log4j.rootLogger=${debug.level}, A1
```

Resource filtering

Moet geactiveerd worden in pom.xml:

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

Opdrachten

Maak opdracht 27 in de cursus (p.76-77)

Uitvoerbare JAR: kijk eens naar het voorbeeld op p. 78 en maak de bijhorende oefening.

JAR-bestanden: Maak op basis van het voorbeeld in de cursus opdracht 29 op p.80

Profiles

Inleiding

Development omgeving – Test omgeving –
Productie omgeving

Development omgeving: test database, extra
logging ...

Productie omgeving: productie database,
geoptimaliseerde logging ...

→ Maven profiles biedt de oplossing!

Profiel aanmaken

PROD: logging instelling wijzigen

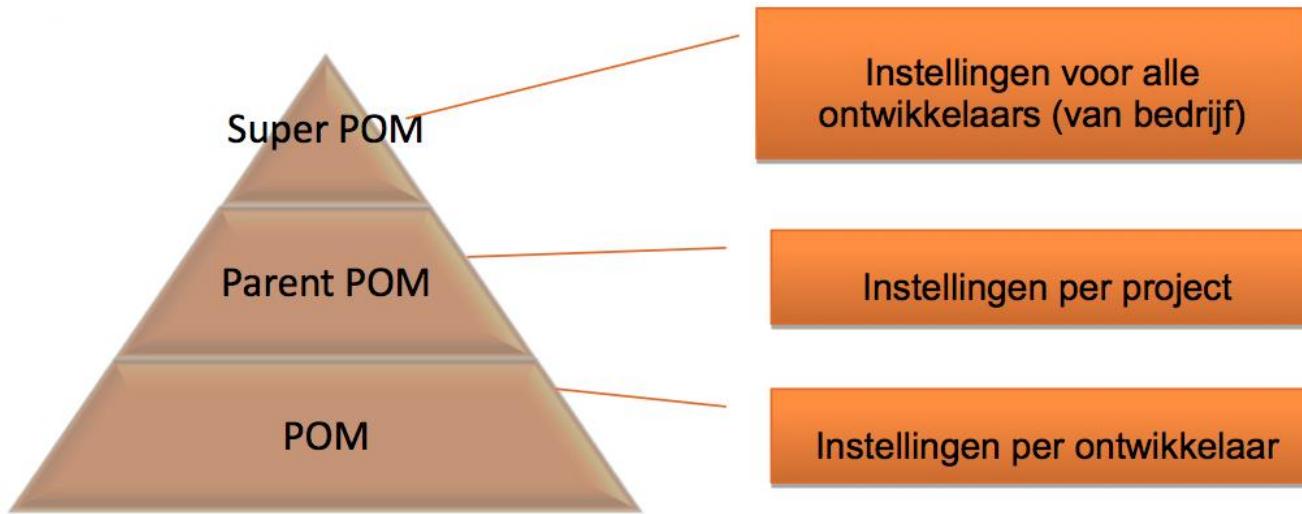
```
<profiles>
  <profile>
    <id>PROD</id>
    <properties>
      <log.level>WARN</log.level>
    </properties>
  </profile>
</profiles>
```

Profiel activeren: mvn –P PROD package

Super POM

Overerving

Een POM-bestand kan informatie overerven van een ander POM-bestand. Deze laatste wordt dan de **Parent POM** genoemd. Een hele hiërarchie is mogelijk. Bovenaan staat de **Super POM** die alle standaard-instellingen bevat.



Je laat een POM bestand als volgt erven van een parent POM:

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

Dependency/ Plugin Management

Toevoegen van veel gebruikte dependencies/plugins in parent pom.

In child pom volstaat dan om te verwijzen naar de dependency (dus geen versienummer)

Begeleide demo

Samen opdracht 31 en 32 maken p.83 en p.86

Meervoudige modules

Modules toevoegen

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

```
Cursus_Maven
    Module1
        pom.xml
    Module2
        pom.xml
pom.xml (meestal ook parent pom)
```

Opdracht

Maak de opdracht FRUIT. Deze staat in de repository vdab_genk in de map oefeningen. Doe een git pull als je deze nog niet ontvangen hebt.