



Noël Vaes

Java Trainer & Consultant



**Maven**

Roode Roosstraat 5  
3500 Hasselt  
België

+32 474 38 23 94  
noel@noelvaes.eu  
www.noelvaes.eu

Vrijwel alle namen van software- en hardwareproducten die in deze cursus worden genoemd, zijn tegelijkertijd ook handelsmerken en dienen dienovereenkomstig te worden behandeld.

Alle rechten voorbehouden. Niets uit deze uitgave mag worden verveelvoudigd, opgeslagen in een geautomatiseerd gegevensbestand of openbaar worden gemaakt in enige vorm of op enige wijze, hetzij elektronisch, mechanisch, door fotokopieën, opnamen of op enige andere manier, zonder voorafgaande schriftelijke toestemming van de auteur. De enige uitzondering die hierop bestaat, is dat eventuele programma's en door de gebruiker te typen voorbeelden mogen worden ingevoerd opgeslagen en uitgevoerd op een computersysteem, zolang deze voor privédoeleinden worden gebruikt, en niet bestemd zijn voor reproductie of publicatie.

Correspondentie inzake overnemen of reproductie kunt u richten aan:

Noël Vaes  
Roode Roosstraat 5  
3500 Hasselt  
België

Tel: +32 474 38 23 94

noel@noelvaes.eu  
www.noelvaes.eu

Ondanks alle aan de samenstelling van deze tekst bestede zorg, kan de auteur geen aansprakelijkheid aanvaarden voor eventuele schade die zou kunnen voortvloeien uit enige fout, die in deze uitgave zou kunnen voorkomen.

04/04/2018

Copyright© 2018 Noël Vaes



# Inhoudsopgave

<b>Hoofdstuk 1: Maven</b> .....	<b>4</b>
1.1. Inleiding.....	4
1.2. <i>Maven</i> installeren en configureren.....	4
1.3. Mijn eerste <i>Maven</i> -project.....	5
1.4. Project Object Model.....	10
1.5. Dependencies.....	11
1.5.1. Scope.....	14
1.5.2. Transitiviteit van afhankelijkheden.....	16
1.5.3. Versiereeksen en conflictoplossing.....	17
1.6. <i>Phases</i> en <i>Goals</i> .....	18
1.6.1. Default lifecycle.....	19
1.6.2. Clean lifecycle.....	23
1.6.3. Site lifecycle.....	23
1.7. Repositories.....	24
1.7.1. Globale Repository.....	24
1.7.2. Lokale Repository.....	25
1.7.3. Eigen Repository.....	25
1.8. <i>Properties</i> en <i>resource filtering</i> .....	28
1.8.1. <i>Maven properties</i> .....	28
1.8.2. Omgevingsvariabelen.....	29
1.8.3. <i>System-properties</i> .....	29
1.8.4. Zelf gedefinieerde <i>properties</i> .....	29
1.8.5. Resource filtering.....	30
1.9. Uitvoerbare JAR-bestanden.....	32
1.10. JAR-bestanden voor broncode en documentatie.....	34
1.11. Profiles.....	36
1.12. Overerving.....	37
1.12.1. Parent-POM.....	37
1.12.2. <i>Dependency</i> en <i>Plugin Management</i> .....	41
1.13. Meervoudige modules.....	42
1.14. Webapplicaties.....	42
1.15. Integratie in Eclipse.....	45



# Hoofdstuk 1: Maven

## 1.1. Inleiding

*Maven* is een projectmanagement-*tool*. Het is een *tool* waarmee men een Java-project op een gestandaardiseerde wijze kan vormgeven en beheren. Dit behelst onder andere het *builden* (compileren, JAR maken enzovoort) van een project maar ook het genereren van rapporten en documentatie, het maken van een website enzovoort. *Maven* is dus meer dan een *build tool* zoals *ANT*, maar alle mogelijkheden van een *build tool* zijn wel voorzien.

In de volgende paragrafen zullen we stap voor stap de mogelijkheden van *Maven* verkennen aan de hand van praktische voorbeelden.

## 1.2. Maven installeren en configureren

*Maven* is een *open-source*-project van *Apache* en is te vinden op volgende website: <http://maven.apache.org>. *Maven* kan geïnstalleerd worden door het bestand **apache-maven-3.x.y-bin.zip** af te halen en uit te pakken op het lokale systeem.

### Opdracht 1: Maven installeren

- Haal het bestand **apache-maven-3.x.y-bin.zip** van de website <http://maven.apache.org>.
- Pak dit bestand uit op je lokale systeem, bijvoorbeeld in **C:\Program Files\**
- Voeg de volgende omgevingsvariabele toe aan het besturingssysteem:

```
MAVEN_HOME="C:\Program Files\apache-maven-3.x.y"
```

- Voeg tevens de plaats van *Maven* toe aan de variabele `PATH` zodat we *Maven* kunnen uitvoeren vanop de commandolijn:

```
PATH=...;%MAVEN_HOME%\bin
```

- Zorg er tevens voor dat de omgevingsvariabele `JAVA_HOME` verwijst naar de installatie van de JDK (en niet naar de JRE).
- Open een commandovenster en voer het volgende commando uit:

```
mvn -version
```

```
Opdrachtprompt
C:\Users\Noel>mvn -version
Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T09:58:13+02:00)
Maven home: D:\Java\apache-maven-3.5.2\bin\..
Java version: 1.8.0_144, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_144\jre
Default locale: nl_BE, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
C:\Users\Noel>
```



### 1.3. Mijn eerste *Maven*-project

Na een succesvolle installatie kunnen we *Maven* beginnen te gebruiken voor het beheer van Java-projecten.

Een project wordt beschreven in een *Project Object Model*. Dit is een XML-bestand met de naam **pom.xml** dat zich in de hoofdmap van het project dient te bevinden.

Java-projecten bevatten doorgaans afzonderlijke mappen voor de broncode, bibliotheken (JAR-bestanden), gecompileerde klassen enzovoort. *Maven* maakt het beheer van projecten eenvoudiger door deze mappenstructuur te standaardiseren. Het is mogelijk, maar niet aangewezen om hiervan af te wijken. Het gebruik van de *Maven*-standaardinstellingen heeft als voordeel dat de configuratie minimaal is en dat men makkelijk inzicht krijgt in nieuwe projecten die ook deze standaard volgen.

Voor een eenvoudig Java-project ziet deze mappenstructuur er als volgt uit:

```
project
  +--src
    +--main
      +--java
      +--resources
    +--test
      +--java
      +--resources
  +-- target
    +--classes
    +--test-classes
pom.xml
```

Map/bestand	Inhoud
src/main/java	De Java-broncode van het project.
src/main/resources	Andere bestanden die we nodig hebben, zoals <i>property</i> -bestanden, configuratiebestanden, afbeeldingen enzovoort.
src/test/java	De Java-broncode van de testklassen ( <i>JUnit</i> of <i>TestNG</i> )
src/test/resources	Andere bestanden die we enkel nodig hebben in de testklassen.
target	Gegenereerde <i>artifacts</i> : JAR - WAR - EAR - ZIP.
target/classes	De gecompileerde klassen.
target/test-classes	De gecompileerde testklassen.
pom.xml	Dit bestand bevat het <i>Project Object Model</i> .

Terwijl Java-broncodebestanden uit de map **src/main/java** gecompileerd worden naar de map **target/classes** worden de *resource*-bestanden uit de map **src/main/resource** gekopieerd naar de map **target/classes**. Dit impliceert dat deze bestanden in het finale *classpath* beschikbaar worden gesteld.

We zullen dit alles illustreren met een voorbeeld. Stel dat we een project willen maken voor de alom bekende "Hello World". We voorzien hiervoor volgende mappenstructuur:

```
project
  +--src
    +--main
```



```
+---java
  +---eu
    +---noelvaes
      +---hello
        +---App.java
```

pom.xml

De broncode:

### **App.java**

```
package eu.noelvaes.hello;

/**My own application class.
 *
 * @author Noël Vaes
 *
 */
public class App {
    /**This method says hello to the world.
     * @return "Hello World"
     */
    public String sayHello() {
        return "Hello World";
    }

    public static void main(String[] args) {
        App app = new App();
        System.out.println(app.sayHello());
    }
}
```

Het POM-bestand:

### **pom.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<project
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
  xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>
  <groupId>eu.noelvaes.hello</groupId>
  <artifactId>Hello</artifactId>
  <version>1.0</version>
</project>
```

Het aanmaken van deze mappenstructuur kunnen we ook eenvoudig doen via een ingebouwd *archetype*. We voeren dan volgend commando uit:

```
mvn archetype:generate
```

We moeten hier dan een aantal vragen beantwoorden over het project. Nadien worden de nodige folders en eventueel broncodebestanden automatisch aangemaakt.



Het compileren en het inpakken in een JAR-bestand kan met het volgende commando gebeuren:

```
mvn package
```

```
C:\Windows\system32\cmd.exe
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building Unnamed - be.noelvaes.hello:Hello:jar:1.0
[INFO]   task-segment: [package]
[INFO] -----
[INFO] [resources:resources {execution: default-resources}]
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,
i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory G:\Cursussen\Maven\Opdrachten\Hello\src\main\resources
[INFO] [compiler:compile {execution: default-compile}]
[INFO] Nothing to compile - all classes are up to date
[INFO] [resources:testResources {execution: default-testResources}]
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,
i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory G:\Cursussen\Maven\Opdrachten\Hello\src\test\resources
[INFO] [compiler:testCompile {execution: default-testCompile}]
[INFO] No sources to compile
[INFO] [surefire:test {execution: default-test}]
[INFO] No tests to run.
[INFO] [jar:jar {execution: default-jar}]
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 1 second
[INFO] Finished at: Thu Aug 13 09:27:16 CEST 2009
[INFO] Final Memory: 8M/16M
[INFO] -----
```

Bij de uitvoering van dit commando wordt de broncode gecompileerd en ingepakt in het volgende JAR-bestand: **Hello-1.0.jar** in de map **target**.

Er is weinig configuratie nodig omdat we hier gebruikmaken van de *Maven*-projectstructuur. *Maven* zoekt de broncode in de map **src/main/java**, compileert de klassen naar de map **target/classes** en maakt een JAR-bestand in de map **target**.

Om de gecompileerde klassen en het JAR-bestand te verwijderen en weer van een schone lei te beginnen, gebruiken we het volgende commando:

```
mvn clean
```

Men kan de commando's ook na elkaar plaatsen zodat ze in de aangegeven volgorde worden uitgevoerd. Heel gebruikelijk is eerst alles wissen en vervolgens alles opnieuw bouwen:

```
mvn clean package
```

Merk op dat bij het eerste gebruik van *Maven* allerlei modules van het internet gehaald worden. Deze worden opgeslagen in de lokale *repository* zodat ze voortaan onmiddellijk beschikbaar zijn. Hierover later meer.



## Opdracht 2: Mijn eerste project

In deze opdracht gaan we ons eerste *Maven*-project maken. We kunnen de benodigde mappenstructuur handmatig aanmaken maar we kunnen dat ook doen via een ingebouwd *archetype*.

- Maak een map met de naam **CursusMaven**. Hierin zullen we allerlei projecten toevoegen.
- Open in deze map een commandovenster en voer het volgende commando uit:  
mvn archetype:generate

```
C:\WINDOWS\system32\cmd.exe
Choose a number or apply filter <format: lgroupId:artifactId, case sensitive contains>: 665:
Choose org.apache.maven.archetypes:maven-archetype-quickstart version:
1: 1.0-alpha-1
2: 1.0-alpha-2
3: 1.0-alpha-3
4: 1.0-alpha-4
5: 1.0
6: 1.1
Choose a number: 6:
Define value for property 'groupId': : eu.noelvaes.hello
Define value for property 'artifactId': : Hello
Define value for property 'version': 1.0-SNAPSHOT: : 1.0
Define value for property 'package': eu.noelvaes.hello: :
Confirm properties configuration:
groupId: eu.noelvaes.hello
artifactId: Hello
version: 1.0
package: eu.noelvaes.hello
Y: : Y
[INFO] -----
```

- Je dient hier bepaalde gegevens in te voeren:
  - filter:** druk *enter* (*QuickStart*-project)
  - version:** druk *enter* (1.1)
  - groupid:** eu.noelvaes.hello (of je eigen pakketnaam)
  - artefactId:** Hello
  - version:** 1.0.0
  - package:** druk *enter* (voorgestelde pakket)
  - Y: Y
- Verifieer de aangemaakte mappenstructuur.
- Open het bestand **App.java** en wijzig de code tot het volgende:

```
package eu.noelvaes.hello;

/**My own application class.
 *
 * @author No&euml;l Vaes
 *
 */
public class App {
    /**This method says hello to the world.
     * @return "Hello World"
     */
    public String sayHello() {
        return "Hello World";
    }

    public static void main(String[] args) {
        App app = new App();
        System.out.println(app.sayHello());
    }
}
```