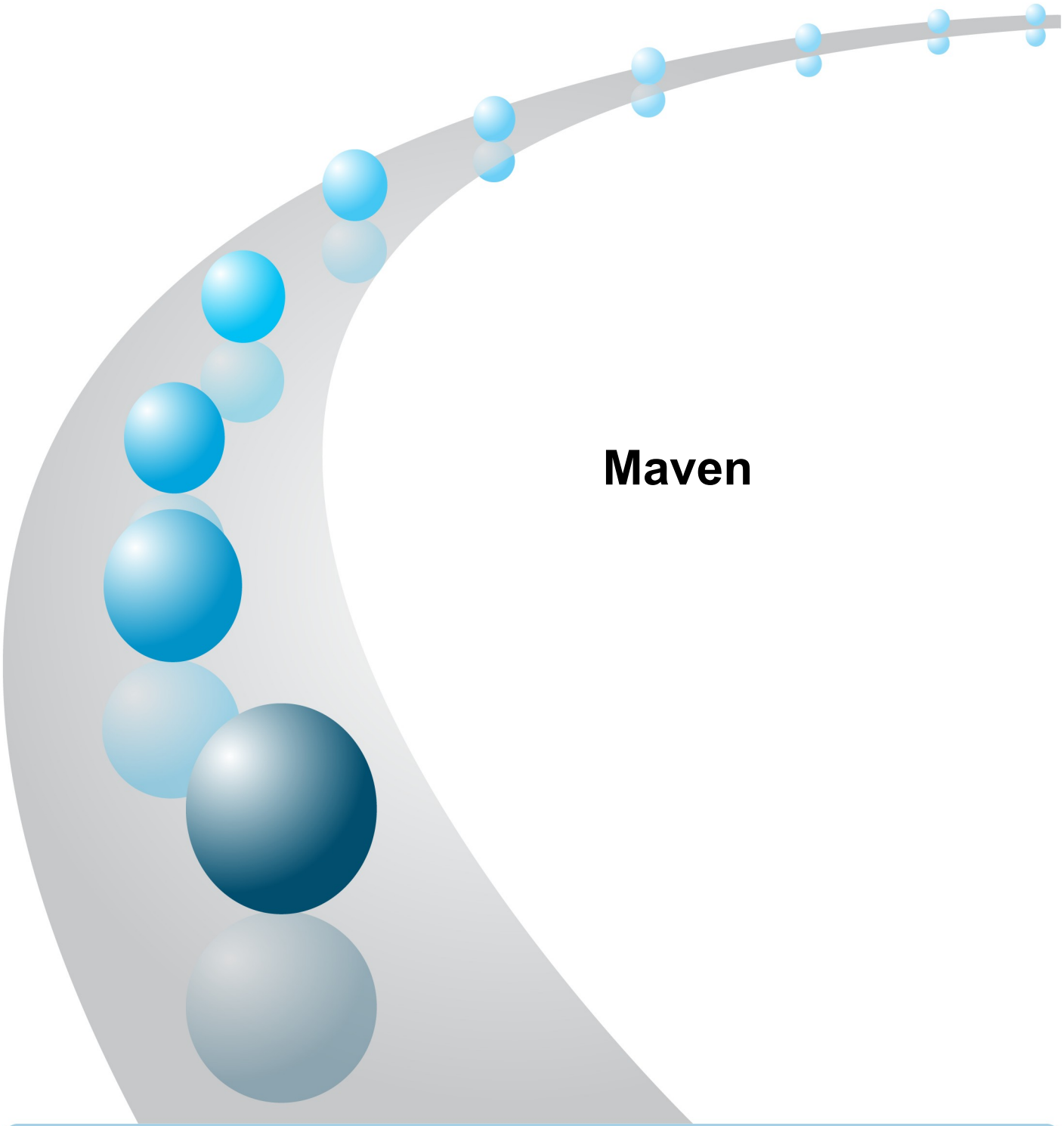
Noël Vaes

Java Trainer & Consultant



# Maven

Roode Roosstraat 5
3500 Hasselt
België

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

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



04/04/2018

# Inhoudsopgave

# 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 k*an 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"
  ```

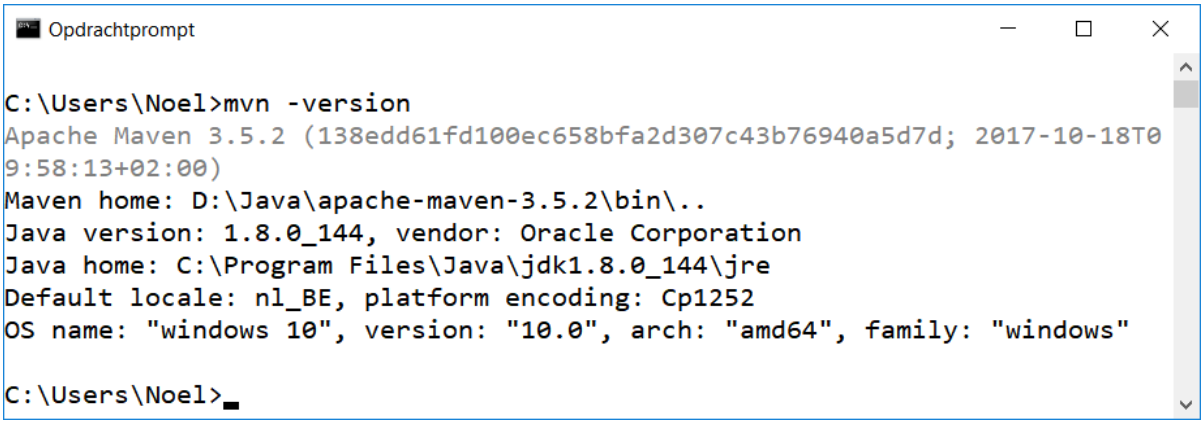
- Voeg tevens de plaats van *Maven* toe aan de variabele `PATH` zodat we *Maven* kunnen uitvoeren vanop de commandolijn:

  ```
  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
  ```



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

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 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());
   }
}
```

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>
```

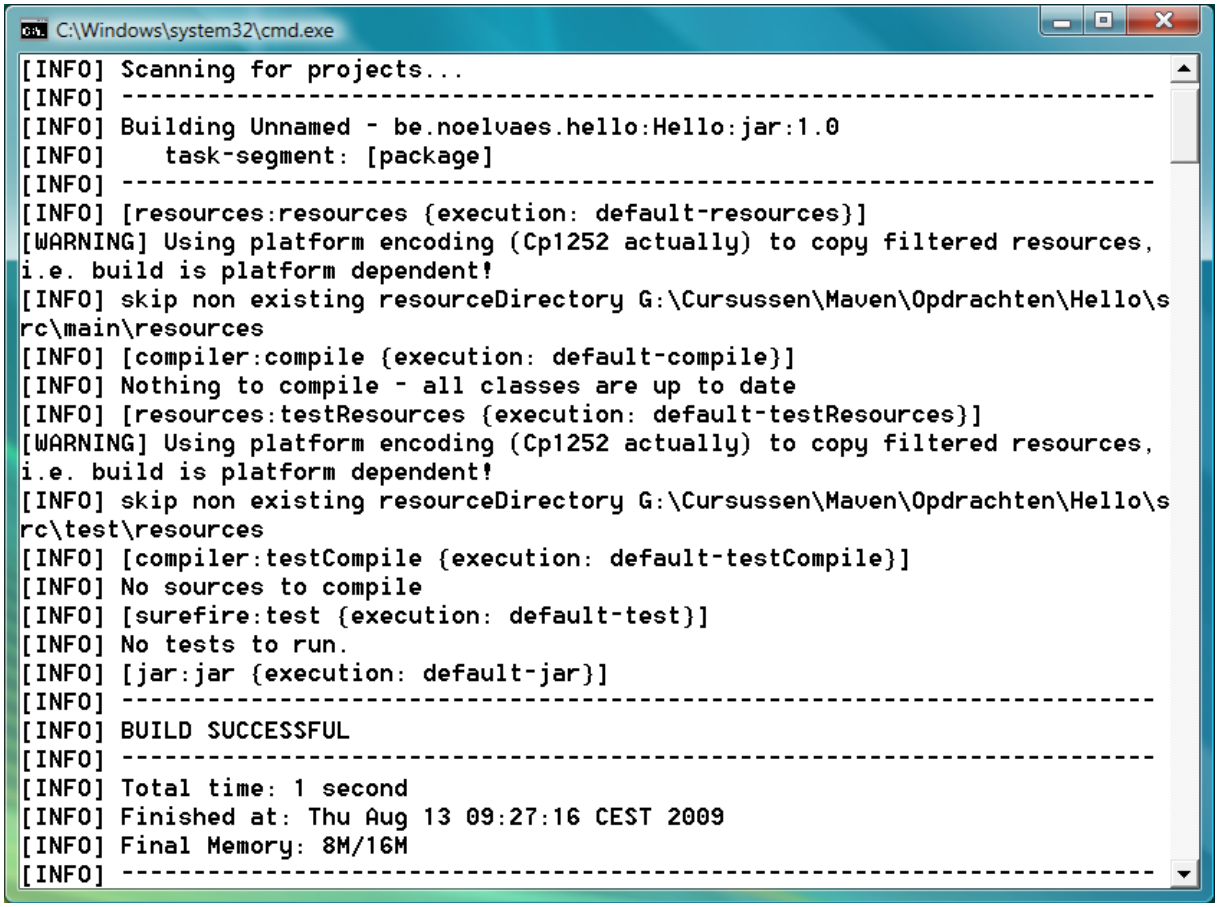
Het aanmaken van deze mappenstructuur kunnen we ook eenvoudig doen via een ingebouwd *archetype*. We voeren dan volgend commando uit:

```
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
```




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:

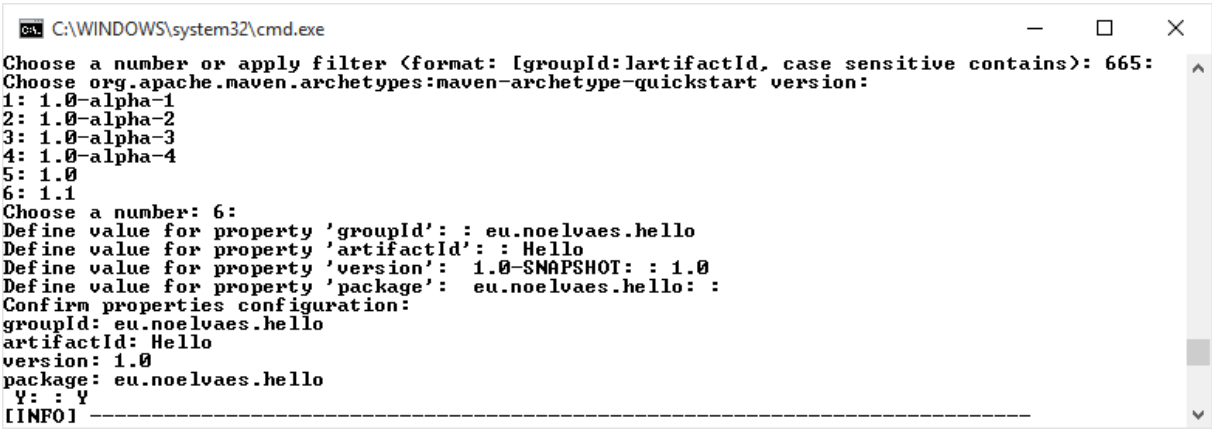
```
mvn clean package
```

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`




- 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());
```