



Rapport

Sujet : Atelier Maven

Réalisé par :	Encadré par :
KHADROUF Oumaima	Mr. NAFIL Khalid
LARHOUTI Meriem	

Année universitaire : 2017/2018

Table des matières

0.1	Introduction	3
0.2	Maven c'est quoi ?	3
0.3	Structure des répertoires	3
0.4	Project Object Model (POM)	3
0.5	Phases de construction	4
0.6	Comparaison avec Ant	4
0.6.1	Ant	5
0.6.2	Maven	5
0.7	Partie TP	5
0.7.1	Génération de projet	5
0.7.2	Compilation et test	7
0.7.3	Exécution du programme	7

0.1 Introduction

Au cours du temps, les développeurs ont remarqué que des tâches comme le téléchargement et l'import des bibliothèques, la compilation des classes java, la fabrication des fichiers d'archives ou autre, se faisaient sur chaque projet. Ces tâches sont répétitives d'un projet à un autre. Donc comment automatiser certaines tâches répétitives et limiter ainsi les erreurs ou les oublis ? Pour résoudre tous ces problèmes (et d'autres encore), les développeurs ont pensé à un outil tel que Maven.

0.2 Maven c'est quoi ?

Maven est un outil développé par la fondation Apache, c'est un outil de gestion de projet qui permet de gérer dans un environnement la compilation, la documentation, la génération de rapports et de site web. Il rend simples et rigoureux la distribution de projets et leur partage au moyen d'un système de gestion des dépendances et des plugins très efficace.

0.3 Structure des répertoires

Les projets Maven sont organisés selon une structure hiérarchique, appelée Standard Directory Layout. Elle permet de bien faire la différence entre les divers éléments du projet, pour une maintenance et une mise en place plus efficaces.

src/main/java : Contient les sources Java de l'application.

src/main/resources : Contient les ressources de l'application.

src/main/webapp : Contient les fichiers de l'application Web.

src/test/java : Contient les sources Java pour les tests unitaires.

src/site : Contient les fichiers pour le site.

target : Répertoire de destination de tous les traitements maven.

0.4 Project Object Model (POM)

POM est le modèle central du projet Maven. C'est un fichier au format XML qui contient toutes les informations nécessaires pour que Maven puisse procéder à la construction du projet. Il permet de préciser une série d'informations sur le projet :

project : c'est la balise racine de tous les fichiers pom.xml.

modelVersion : cette balise indique la version de POM utilisée. Bien que cette version ne change pas fréquemment, elle est obligatoire afin de garantir la stabilité d'utilisation.

groupId : cette balise permet d'identifier un groupe qui a créé le projet. Cette clé permet d'organiser et de retrouver plus facilement et rapidement le projet.

artifactId : cette balise indique un nom unique utilisé pour nommer les artifacts à construire.
packaging : type de packaging du projet (exemple : JAR, WAR, EAR, etc.).
version : version de l'artifact généré par le projet.
name : nom du projet.
url : adresse du site du projet.
description : description du projet.
dependencies : balise permettant de gérer les dépendances.

0.5 Phases de construction

Un autre élément central de Maven est son cycle de vie de construction, qui est organisé en phases. Les phases sont les étapes de ce cycle, qui lui est une séquence de différentes phases. Concrètement on peut passer le nom d'une phase en paramètre de Maven (commande mvn). Maven exécute ensuite toutes les phases qui précèdent avant d'exécuter cette phase. Voici les phases principales qui résument le cycle de vie de construction par défaut :

Validate : valide que le projet soit correct et que toute information nécessaire soit disponible.

Compile : compile le code source du projet.

Test-compile : compile les éventuels tests unitaires associés au code du projet.

Test : exécute les tests unitaires en utilisant un framework spécifique (ex : junit).

Package : place le code compilé dans un paquetage (par défaut il est de format JAR).

Integrationtest : si besoin, déploie le paquetage dans un environnement pour y faire les tests d'intégration.

Verify : lance des tests sur la qualité du paquetage, avec critères de qualité.

Install : installe le paquetage dans un dépôt local pour son utilisation dans un autre projet.

Deploy : copie le paquetage final du projet vers un dépôt distant (rien n'empêche qu'il soit local) pour son partage (en tant que dépendance) avec d'autres projets.

0.6 Comparaison avec Ant

Ant et Maven sont tous les deux des outils de construction fournis par Apache. Le but principal de ces technologies est de faciliter le processus de construction d'un projet.

Il y a beaucoup de différences entre la fourmi et le Maven qui sont données ci-dessous :

0.6.1 Ant

Ant n'a pas de conventions formelles, nous devons donc fournir des informations sur la structure du projet dans le fichier build.xml. Ant est procédural, vous devez fournir des informations sur ce qu'il faut faire et quand le faire par le code. Vous devez fournir une commande.

- Il n'y a pas de cycle de vie dans Ant.
- C'est principalement un outil de construction.
- Les scripts Ant ne sont pas réutilisables.

0.6.2 Maven

Maven a une convention pour placer le code source, le code compilé etc. Donc nous n'avons pas besoin de fournir des informations sur la structure du projet dans le fichier pom.xml.

- Maven est déclaratif, tout ce que vous définissez dans le fichier pom.xml.
- Il y a un cycle de vie à Maven.
- C'est principalement un outil de gestion de projet.
- Les plugins maven sont réutilisables.

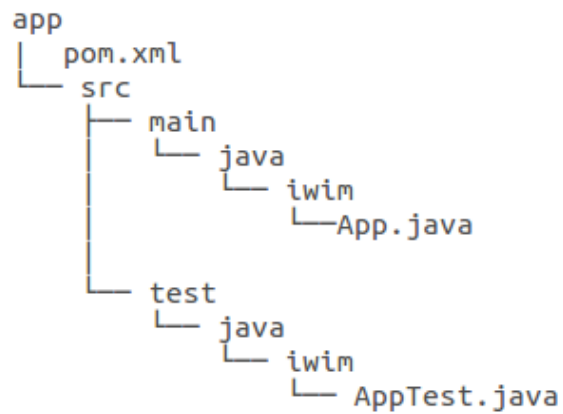
0.7 Partie TP

0.7.1 Génération de projet

Dans ce projet, nous utilisons la fonctionnalité d'échafaudage de Maven pour créer un projet Java. Pour éviter le mode interactif, toutes les propriétés requises sont transmises directement à la commande. Sinon, Maven demande tous les paramètres requis. Avec cette commande, Maven génère un projet Java.

```
mvn archetype:generate -DartifactId=app -DarchetypeArtifactId=maven-archetype-quickstart -DgroupId=iwim -DarchetypeVersion=1.1
```

Le projet généré par Maven sur le système de fichiers est de la structure suivante.



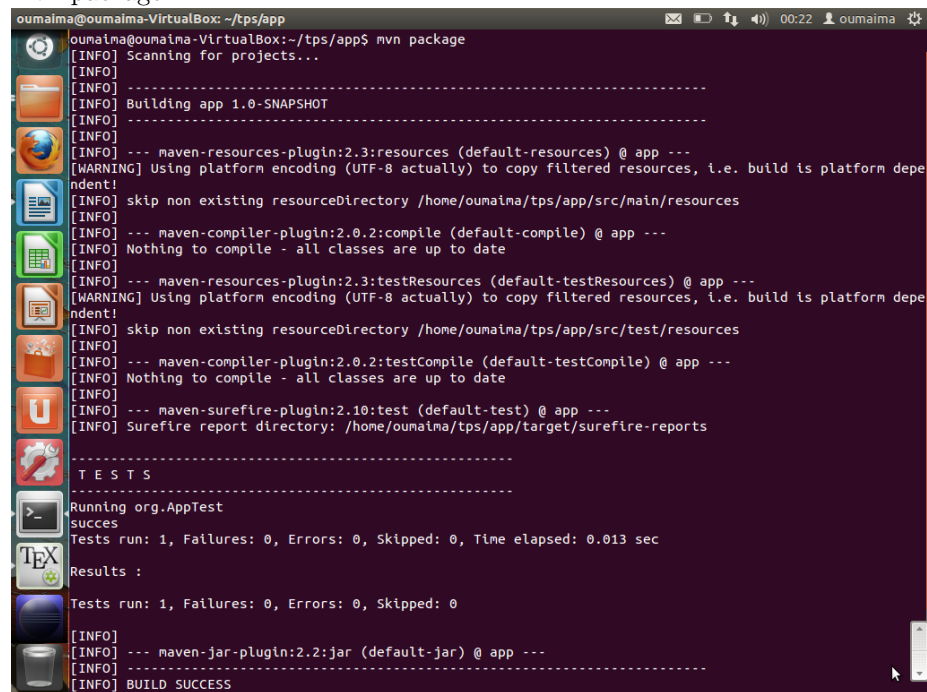
Maven a créé une classe App.java dans le dossier ./src/main/ , qui est juste un simple programme "Hello World". Il a également créé un exemple de classe de test dans ./src/test/. Dans le dossier racine, il y a un fichier pom.xml :

```
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>iwim</groupId>
  <artifactId>app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>app</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

0.7.2 Compilation et test

Pour la compilation et le test on exécute la commande suivante :

`mvn package`



```
oumalma@oumalma-VirtualBox: ~/tps/app
oumalma@oumalma-VirtualBox:~/tps/app$ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building app 1.0-SNAPSHOT
[INFO] -----
[INFO] --- maven-resources-plugin:2.3:resources (default-resources) @ app ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, t.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/oumalma/tps/app/src/main/resources
[INFO] --- maven-compiler-plugin:2.0.2:compile (default-compile) @ app ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-resources-plugin:2.3:testResources (default-testResources) @ app ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, t.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/oumalma/tps/app/src/test/resources
[INFO] --- maven-compiler-plugin:2.0.2:testCompile (default-testCompile) @ app ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-surefire-plugin:2.10:test (default-test) @ app ---
[INFO] Surefire report directory: /home/oumalma/tps/app/target/surefire-reports

-----
T E S T S
-----
Running org.AppTest
succes
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.013 sec

Results :

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

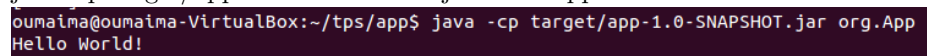
[INFO] --- maven-jar-plugin:2.2:jar (default-jar) @ app ---
[INFO] BUILD SUCCESS
```

Le package target a crée un fichier JAR déployable et exécutable.

0.7.3 Exécution du programme

Pour l'exécution on utilise la commande suivante :

`java -cp target/app-1.0-SNAPSHOT.jar iwim.App`



```
oumalma@oumalma-VirtualBox:~/tps/app$ java -cp target/app-1.0-SNAPSHOT.jar org.App
Hello World!
```