**MAVEN**

Table des matières

[Chapitre 1 MAVEN DEFINITION : 2](#_Toc102840208)

[Chapitre 2 : LE FICHIER POM (Project Object Model) 2](#_Toc102840209)

[1-Le Bloc « En-Tête » du fichier POM : 3](#_Toc102840210)

[2-Deuxième Bloc du fichier POM : 4](#_Toc102840211)

[3-3ème Bloc (bloc dependencies) du fichier POM : 4](#_Toc102840212)

[Chapitre 3 : CENTRAL REPOSITORY : 5](#_Toc102840213)

[Chapitre 4 DEFINTIONS DIVERSES: 5](#_Toc102840214)

# 

# MAVEN DEFINITION :

Maven : Maven est un outil qui facilite la vie aux développeurs Java il permet de faire ce que faire un IDE comme Eclipse en local et bien d’autre choses :

* Compilation et déploiement des applications Java (JAR, WAR)
* Gestion des librairies requises par l'application
* Exécution des tests unitaires
* Génération des documentations du projet (site web, pdf, Latex)
* Intégration dans différents IDE (Eclipse, JBulder)

En gros permet la gestion de projet comme Ant mais d’une façon beaucoup plus parfaite (déclaration des métadonnées, le build, construction du projet, etc…)

**Build (MAVEN)**

**Code source**

**Livrable**

**(Artefacts)**

* Générer les sources
* Traiter et valider les sources
* Test
* Test compile
* Compile
* Packaging
* Integration-test
* Process ressources

Les principales **phases** dans Maven sont :

* **Compile** : compilation du code source du projet
* **Test** : compilation du code source des tests et exécution des tests
* **Package** : construction du livrable (pour une application Web, il s’agit de l’archive WAR)

# : LE FICHIER POM (Project Object Model)

Connaitre maven 🡺 connaitre les différentes sections du fichier POM

Le fichier Pom est un fichier XML, qui se trouve dans la racine du projet .

A la création d’un projet maven ce fichier POM est vide.



Figure : Exemple de fichier POM

## 1-Le Bloc « En-Tête » du fichier POM :

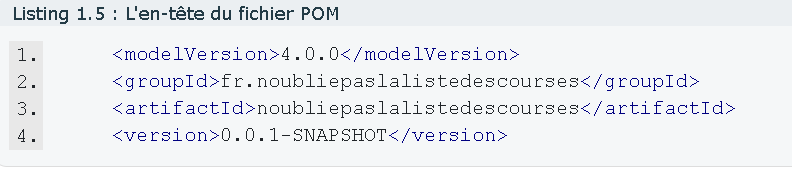
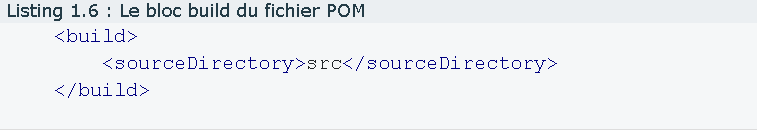


Figure : L'entête d'un fichier POM

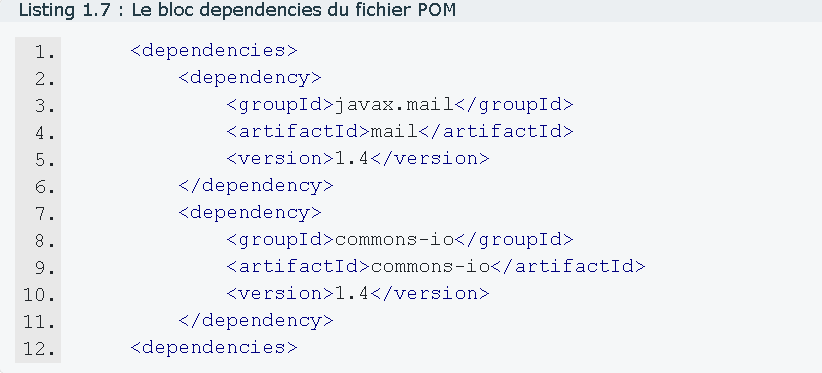
* L'élément **modelVersion** permet de savoir quelle version de la structure de données « modèle de projet » est représentée dans le fichier XML. « 4.0.0 ».
* L'identifiant de groupe **(groupId)** permet de connaître l'organisation, l'entreprise, l'entité ou la communauté qui gère le projet. Par convention, on utilise le nom de domaine Internet inversé, selon la même logique que celle généralement recommandée pour les noms de package Java.
* Enfin, on précise quelle version du projet est considérée. La plupart des projets utilisent la formule <**Version** Majeure>.<Version Mineure>.<Correctif>. Le groupId comme le package sont faits pour identifier le projet d’une façon unique. (éviter les ambiguités)

## 2-Deuxième Bloc du fichier POM :



L'approche déclarative utilisée par Maven permet de définir l'emplacement de nos fichiers sources. Le projet étant à la fois très simple et très banal, aucune autre déclaration n'est nécessaire. Si nous avions utilisé le répertoire conventionnel de Maven pour les fichiers sources Java, nous n'aurions même pas eu besoin de ce bloc <build> !

## 3-3ème Bloc (bloc dependencies) du fichier POM :



Une nouvelle fois, l'approche déclarative prend le dessus : nous n'indiquons pas l'emplacement physique de ces bibliothèques, à savoir /lib pour notre projet, mais des identifiants groupId + artifactId version. Il s'agit des mêmes identifiants de groupe, de composant et de version, que nous venons de rencontrer, appliqués à une bibliothèque. Nous indiquons, par exemple, que nous utilisons l'API standard JavaMail en version 1.4.

Nous avons ici une réponse partielle à notre question sur la nécessité d'un accès Internet : Maven va télécharger les bibliothèques indiquées, à partir d'une source fiable, plutôt que de se contenter des fichiers JAR présents dans le répertoire /lib et dont la version et l'origine sont incertaines. L'espace contenant l'ensemble des bibliothèques téléchargées est un dépôt d'archives local (local repository) et respecte une convention

# : CENTRAL REPOSITORY :

<https://mvnrepository.com/repos/central>

Le central repository de maven est un site internet où se trouve le dépôt de binaires Open source le plus connu, également connu sous le nom de Maven Central. En effet, lors de la construction d'un projet Java, Apache Maven essaie par défaut de localiser ses dépendances depuis Maven Central

***NB :***

***le classePath*** est le fichier d’un projet java contenant tous les chemins de toutes les classes .java par package.

***Artéfact : Dans*** Maven, un artefact est un élément spécifique issu de la construction du logiciel. Dans Java, les artefacts les plus communs sont des JARs, mais ce peut être aussi un fichier WAR, un EAR, un ZIP, etc

# DEFINTIONS DIVERSES:

***Un archétype*** est un Template de projet. Le fait d'utiliser des archétypes pour initialiser un projet permet de gagner du temps et de respecter une certaine convention.

***Dépendance*** : Une dépendance est une référence vers un artefact spécifique contenu dans un repository. Cet artefact est nécessaire pour une ou plusieurs phases du cycle de vie du projet. L'exemple le plus simple est une dépendance sur une bibliothèque jar qui permet d'en utiliser le contenu dans le projet.

***Le repository local, distant*** : Un repository local est un répertoire sur le poste du développeur permettant de stocker, suivant la même arborescence, tous les artefacts téléchargés depuis le(s) repository distant(s). Un projet ayant pour POM :

sera stocké, dans le repository, suivant cette arborescence : ${repository\_home}/fr/masociete/monprojet/1.0/

