**Introduction**

Apache Maven est un outil "Open Source" utilisé pour gérer la production de projets logiciels Java de manière général. Maven est un projet de l'organisation Apache Software Foundation et était historiquement une branche de l'organisation Jakarta Project. L'objectif de cet outil peut être comparé au système Make sous Unix, à savoir : produire un logiciel à partir de ses sources, en optimisant les tâches réalisées à cette fin et en garantissant le bon ordre de fabrication. Il est également comparable à l'outil Ant, mais fournit des moyens de configuration plus simples, eux aussi basés sur le format XML.

Maven choisit une approche différente, fondée sur le constat suivant : tous les projets Java vont suivre à peu près le même schéma. Ainsi, les développeurs de Maven considèrent qu'il est plus simple de décrire en quoi un projet est différent de ce "scénario type" que de répéter des commandes très comparables d'un projet à l'autre.

## Conventions

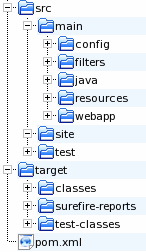
* **Structure des répertoires**

Sous le répertoire des sources, Maven effectue un découpage explicite entre ce qui fait partie du projet et ce qui sert d'outillage de test. Deux sous-répertoires, *main* et *test*, marquent cette distinction.

Enfin, dans chacune de ces branches, un dernier niveau de répertoires sépare les fichiers sources par language :

* *"java"* pour le code source des classes Java.
* *"resources"* pour les fichiers de ressources (configuration XML, fichiers de propriétés...).
* *"webapp"* pour les fichiers statiques d'une application web.

Structure des répertoires d'un projet Maven :



La force des conventions Maven n'est pas dans le nom des répertoires qui ont été choisis, mais dans le fait qu'il offre à la communauté des développeurs Java une base commune.

* **Cycle de vie**

Le cycle de vie est une suite de phases prédéfinies qui doit couvrir les besoins de tout projet. Ces phases sont symboliques et ne sont associées à aucun traitement particulier, mais elles permettent de définir les étapes clés de la construction du projet. Quels que soient le projet et ses particularités, tout traitement réalisé pour le "construire" fera partie de l'une de ces étapes.

Cycle de vie défini par Maven :

|  |  |
| --- | --- |
| **Phase** | **Description** |
| validate | Validation du projet Maven. |
| initialize | Initialisation. |
| generate-sources | Génération de code source. |
| process-resources | Traitement des fichiers de ressources. |
| compile | Compilation des fichiers sources. |
| process-classes | Post-traitement des fichiers binaires compilés. |
| test-compile | Compilation des tests. |
| test | Execution des tests. |
| package | Assemblage du projet sous forme d'archive Java. |
| install | Mise à disposition de l'archive sur la machine locale pour d'autres projets. |
| deploy | Mise à disposition publique de l'archive Java. |

* **Project Object Model (POM)**

Chaque projet ou sous-projet est configuré par un POM qui contient les informations nécessaires à Maven pour traiter le projet (nom du projet, numéro de version, dépendances vers d'autres projets, bibliothèques nécessaires à la compilation, noms des contributeurs, etc.). Ce POM se matérialise par un fichier pom.xml à la racine du projet. Cette approche permet l'héritage des propriétés du projet parent. Si une propriété est redéfinie dans le POM du projet, elle recouvre celle qui est définie dans le projet parent. Ceci introduit le concept de réutilisation de configuration. Le fichier pom.xml du projet principal est nommé *pom parent*. Il contient une description détaillée de votre projet, avec en particulier des informations concernant le visionnage et la gestion des configurations, les dépendances, les ressources de l'application, les tests, les membres de l'équipe, la structure et bien plus.

**Rapports**

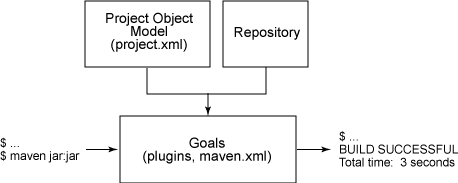
Les rapports générés par Maven portent notamment sur :

* Les dépendances du projet
* Les résultats des tests lancés, incluant:
  + Des statistiques sur les résultats
  + Le pourcentage d’efficacité de ces tests
* Le pourcentage de code testé
* La complexité du code
* Le respect des normes de codage

[**Les plugins**](http://www-igm.univ-mlv.fr/~dr/XPOSE2003/site_maven/html/plugins.html)

Les tâches exécutées par Maven reposent sur des plugins qui sont des fichiers jar utilisés par le moteur de l'application. Chaque plugin permet d'effectuer une tâche particulière. Pour cela, on peut leur spécifier un certain nombre de paramètres, appelés goals. Les goals sont configurables dans le fichier de configuration de maven : **maven.xml**

La figure suivante récapitule l'architecture utilisée :



Voici une liste non exhaustive de divers plugins et de leurs goals :

* **jar** : deploy, install
* **war** : clean, deploy, init, install
* **j2ee** : validate-war
* **javadoc** : deploy, install
* **pdf**
* **latex** : prepare-filesystem, generate
* **eclipse** : add-maven-repo, generate-classpath, generate-project
* **jbulder** : generate-library, generate-project

La commande *maven -g* permet d'avoir la liste de tous les plugins disponibles avec leurs goals.

Il existe 3 niveaux de personnalisations des plugins :

* maven.xml : ajout d'extensions <postgoal> et <pregoal>
* maven.xml : création de goals
* Développement de plugins en Jelly

### Le Jelly

C'est le langage utilisé pour le développement simple et rapide de plugins. Il s'appuie sur le XML et rajoute la possibilité d'effectuer des boucles et des tests dans le code, comme le montre l'exemple suivant :

<?xml version="1.0" encoding="ISO-8859-1"?>  
<project xmlns:j="jelly:core">  
  <goal name="Echo Dependencies">  
    <j:forEach var="dep” items="${pom.dependencies}">  
      <echo>${pom.getDependencyPath(dep.getId())}</echo>  
    </j:forEach>  
  </goal>  
</project>

**Conclusion**

Le projet Maven est un projet open source en pleine effervescence. Une preuve en est l'ajout régulier de nouveaux plugins par des développeurs externes au projet (maven-plugins.sourceforge.net). Aujourd'hui, ces plugins sont crits en jelly mais ils pourront être prochainement développés en Java ou Python.

Par ailleurs, les créateurs de Maven travaillent actuellement sur l'amélioration des performances d'exécution qui sont parfois un peu limitées.

Enfin, il est prévu pour les versions futures de supporter d'autres types de projets que les projets Java, notamment des projets .Net développés en C#.