

MAVEN



P.Mathieu

LP DA2I Lille

<http://www.iut-a.univ-lille.fr>

prenom.nom@univ-lille.fr

8 novembre 2019

1 MAVEN

- Outil de gestion et d'automatisation de production de projets Java en général et Java EE en particulier.
- fourni par l'Apache Software Foundation.
- Objectifs
 - ▶ Fournir une arborescence standardisée
 - ▶ Assurer les tâches de fabrication
 - ▶ Gérer automatiquement les dépendances nécessaires
- Version actuelle 3.5
- Dans la lignée de Make, Ant, Gradle (pour Java) npm (Javascript), Composer (php)

- **Installation** : <http://maven.apache.org/>
- **Test** : `mvn -v`
- **Aide en ligne** : dans <http://maven.apache.org/> aller sur `use` puis `user center` puis “Maven in 5 minutes” et “getting started”

```
mvn archetype:generate
```

4 informations importantes :

- l'archetype à utiliser
défaut : `maven-archetype-quickstart`
pour le web : `maven-archetype-webapp`
- le `groupId`, nom du package principal, `fr.da2i`
- l'`artifactId`, nom du projet : `tp07`
- le package principal
par défaut, le `groupId`

```
mvn archetype:generate -DarchetypeArtifactId=maven-archetype-quickstart  
-DgroupId=fr.da2i -DartifactId=tp07 -DinteractiveMode=false
```

L'arborescence dépend de l'archétype choisi

tp07

```
|-- pom.xml
|-- src
    |-- main
        |-- java
            |-- fr
                |-- da2i
                    |-- App.java
    |-- test
        |-- java
            |-- fr
                |-- da2i
                    |-- AppTest.java
```

Chaque projet est configuré dans un fichier `pom.xml` placé à la racine du projet.

`pom.xml` fournit des informations sur la version, la gestion des configurations, les dépendances, les ressources de l'application, les tests, les membres de l'équipe

```
<project xmlns="http://maven.apache.org/POM/4.0.0" ...>
  <modelVersion>4.0.0</modelVersion>
  <groupId>fr.da2i</groupId>
  <artifactId>tp07</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>monProj</name>
  <url>http://maven.apache.org</url>
  <properties>
    <maven.compiler.source>1.9</maven.compiler.source>
    <maven.compiler.target>1.9</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
```

Pour n'importe quel but, tous les buts en amont doivent être exécutés

- compile
- test
- package
- install
- deploy

D'autres buts sont exécutables en dehors du cycle de vie : `clean`, `site`, ...


```
mvn archetype:generate
    -DarchetypeArtifactId=maven-archetype-quickstart
    -DgroupId=fr.da2i -DartifactId=tp07 -DinteractiveMode=false

cd tp07

mvn compile
mvn test
mvn package

java -cp target/classes/:.... fr.da2i.App
java -jar target/tp07-1.0-SNAPSHOT.jar fr.da2i.App
mvn exec:java -Dexec.mainClass=fr.da2i.App


mvn clean package
```

trois manières d'exécuter un projet

- `mvn exec:java -Dexec.mainClass=fr.da2i.App`
Maven s'occupe du classpath en pointant sur son `.m2`
- `java -cp target/classes/:.... fr.da2i.App`
On exécute un `.class` Il faut gérer sois-même son classpath, et donc avoir ses dépendances quelque part (voir le `maven-dependency-plugin`)
- `java -jar target/tp07.jar fr.da2i.App`
On exécute l'archive `jar`. Les dépendances doivent être indiquées dans le `MANIFEST.MF` du jar (voir le `maven-jar-plugin`)

Maven calcule et télécharge automatiquement les dépendances dans les référenciels déclarés (dépôts distants) et les place dans un répertoire local

- Local : `/.m2`
- Maven : `https://www.mvnrepository.com/`



Repository

- Central 2.3k
- Sonatype 1.3k
- Spring Plugins 647
- Spring Lib M 587
- JCenter 171
- Clojars 165
- Spring Lib Release 138
- IBiblio 102

Group





- com.github 296
- org.apache 192
- org.onosproject 66
- org.eclipse 59
- com.google 43
- org.glassfish 43
- org.mobincents 43
- org.springframework 39

Category

- Android Package 103
- Eclipse Plugin 29
- JDBC Extension 21
- Web App 17
- JDBC Pool 14

Found 3206 results

Sort: **relevance** | popular | newest

- 
1. Spring JDBC 2,503 usages
[org.springframework » spring-jdbc](#) Apache
 Spring JDBC
 Last Release on Oct 29, 2018
- 
2. MongoDB Java Driver 911 usages
[org.mongodb » mongo-java-driver](#) Apache
 The MongoDB Java Driver uber-artifact, containing the legacy driver, the mongodb-driver, mongodb-driver-core, and bson
 Last Release on Nov 7, 2018
- 
3. SQLite JDBC 472 usages
[org.xerial » sqlite-jdbc](#) Apache
 SQLite JDBC library
 Last Release on Oct 1, 2018
- 
4. DataStax Java Driver For Apache Cassandra Core 542 usages
[com.datastax.cassandra » cassandra-driver-core](#) Apache
 A driver for Apache Cassandra 1.2+ that works exclusively with the Cassandra Query Language version 3 (CQL3) and Cassandra's binary protocol.
 Last Release on Aug 30, 2018

Déclaration des dépendances

```
<dependencies>
  ...
  <dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.2.4</version>
  </dependency>
  ...
</dependencies>
```

Dépendances les plus courantes : Log4J, JUnit, Apache commons,
...

- `scope: compile` (default)
bibliothèque nécessaire à la compilation à l'exécution et au test
- `scope: runtime`
bibliothèque est nécessaire à l'exécution et au test mais pas à la compilation (ex driver SGBD)
- `scope: test`
bibliothèque requise uniquement en phase de test (ex JUnit)
- `scope: provided`
bibliothèques qui seront fournies par l'utilisateur final (ex servlet api, driver jdbc), ne sont donc pas dans le war
- ...

- Lister les dépendances `mvn dependency:list`
- Exporter les dépendances `mvn dependency:copy-dependencies`
- elles sont exportées dans `/target/dependency`
- Pour exporter ailleurs `-DoutputDirectory="/tomcat/lib"`

- à mettre dans `main/resources`
- Maven les recopie automatiquement dans `target/classes`
- On accède impérativement aux ressources via le `ClassLoader`

```
InputStream is = App.class.getClassLoader().getResourceAsStream("fich.txt");  
BufferedReader br = new BufferedReader(new InputStreamReader(is));
```


Les plugins permettent l'ajout de nouvelles fonctionnalités à Maven.

- `maven-dependency-plugin` (installé par défaut) (Avec but `"dependency:copy-dependencies"` pour exporter les dépendances du projet)
- `maven-jar-plugin` (pour définir un jar exécutable)
- `maven-assembly-plugin` (avec but `"assembly-single"` , Pas recommandé de mettre des jars dans des jars)

Chaque plugin vient avec de nouveaux buts : `mvn plugin:but`

```
<plugin>
  <groupId>org.apache.tomcat.maven</groupId>
  <artifactId>tomcat7-maven-plugin</artifactId>
  <version>2.2</version>
  <configuration>
    <path>/${project.build.finalName}</path>
    <contextReloadable>true</contextReloadable>
  </configuration>
</plugin>
```

Utiliser maven-archetype-webapp

```
mvn archetype:generate -DarchetypeArtifactId=maven-archetype-webapp  
-DgroupId=fr.da2i -DartifactId=tp07 -DinteractiveMode=false
```

tp07

```
|-- pom.xml  
|-- src  
    |-- main  
        |-- resources  
        |-- webapp  
            |-- WEB-INF  
                |-- web.xml  
            |-- index.jsp
```

- Créer `java/fr/da2i` sous `main`
- Ajouter dans le `pom` la dépendance `servlet-api.jar` pour compiler les servlets
- Vérifiez que le `web.xml` utilise bien l'api servlet 4.0 (voir `conf/web.xml` pour un exemple)
- `mvn compile` compile simplement les classes et les range dans `target/classes`
- `mvn package` crée une structure de webapp sous le rep `target` ainsi qu'un war

Parmi les plugins possibles d'un projet Maven, il y a tomcat lui-même. On peut donc au choix

- deployer le war obtenu dans un tomcat externe
- pointer sur le repertoire `target/tp07` (structure de webapp)
- intégrer tomcat (plugin `tomcat7-maven-plugin`) et le lancer via `mvn tomcat7:run`
- ou intégrer jetty (plugin `jetty-maven-plugin` et le but `mvn jetty:run`

De très nombreuses autre possibilités,

- Héritage entre POM
- Gestion multi-modules
- Gestion de profils
- ...

... plein d'autres choses encore !

Voir aussi Graddle

MAVEN

Spécificité Univ Lille : définir le Proxy dans `.m2/settings.xml`

```
<settings>
  <proxies>
    <proxy>
      <active>true</active>
      <protocol>https</protocol>
      <host>cache-etu.univ-lille1.fr</host>
      <port>3128</port>
      <username></username>
      <password></password>
      <nonProxyHosts></nonProxyHosts>
    </proxy>
  </proxies>
</settings>
```