Maven

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

Maven

Maven

- **Outil de gestion de projet et de compréhension** : **Maven** facilite la gestion des dépendances et des versions des bibliothèques.
- **Automatisation de la construction de projet** : **Maven** utilise un modèle de cycle de vie de construction pour compiler, tester et déployer le projet.
- **Convention over Configuration** : **Maven** adopte des conventions standard pour les structures de projet afin de minimiser la configuration.
- **Fichier POM (Project Object Model)** : Le fichier `pom.xml` contient les informations sur le projet et les configurations de construction.
- **Référentiel centralisé** : **Maven** utilise un référentiel centralisé pour télécharger automatiquement les dépendances nécessaires.
- **Plugins extensibles** : **Maven** supporte une large gamme de plugins pour différentes tâches de construction et de gestion de projet.
- **Gestion des dépendances transitive** : **Maven** résout automatiquement les dépendances transitives, téléchargeant les bibliothèques nécessaires.

Maven

Autres gestionnaires de paquets (dépendances) pour **Java**

- **Gradle**
- **Ivy**
- **SBT (Simple Build Tool)**
- **Bazel (Google)**
- ...

Maven

pom.xml

- Cœur du projet **Maven**
- Fichier **XML** décrivant le projet
- Contenant toutes les configurations : les informations nécessaires à **Maven** pour construire et gérer le projet

Maven

pom.xml : éléments clés

- project : c'est la balise racine de ce fichier
- modelVersion : c'est la version actuelle d'**Apache Maven** 4.0.0.
- groupId : Il permet d'identifier le groupe créateur du projet. Cet identifiant doit permettre de retrouver plus facilement et rapidement le projet.
- artifactId : il contient le nom du projet.
- version : il précise le numéro de la version du projet plus soit SNAPSHOT (pour les versions encours) soit RELEASE (pour les versions terminées)
- description : elle contient une description du projet.
- ...

Maven

Autres éléments de pom.xml

- Propriétés du projet :
  - Version du compilateur
  - Encodage
  - ...
- <dependencies> : Liste des dépendances.
- <build> : Configuration de la construction du projet.
- <plugins> : Plugins Maven utilisés.
- <profiles> : Profils de construction pour différentes configurations.

Maven

Comment créer un **Java Project** avec **Maven** ?

- Aller dans `File > New > Other`
- Chercher puis sélectionner `Maven Project`
- Cliquer sur `Next`
- Choisir `maven-archetype-quickstart`
- Remplir les champs
  - `Group Id` avec `org.eclipse`
  - `Artifact Id` avec `cours-maven`
  - `Package` avec `org.eclipse.main`
- Valider et attendre la fin des téléchargements (vérifier la présence du fichier `pom.xml`)

Maven

Arborescence d'un projet **Maven** (quickstart) ≡ Java Project

- **Maven** utilise une structure standardisée pour les projets.
- **Répertoires importants** :
  - `src/main/java` : Code source principal.
  - `src/test/java` : Tests unitaires.
  - `src/main/resources` : Ressources de l'application.
  - `src/test/resources` : Ressources pour les tests.
  - `target` : Répertoire de sortie pour les builds.

Maven

Comment créer un projet **JEE** avec **Maven**

- Aller dans `File > New > Maven Project`
- Cliquer sur `Next`
- Choisir `maven-archetype-webapp`
- Remplir les champs
  - `Group Id` avec `org.eclipse`
  - `Artifact Id` avec `first-jee-maven`
  - `Package` avec `org.eclipse.controller`
- Valider et attendre la fin de téléchargements (vérifier la présence du fichier `pom.xml`)

Maven

Si le projet est signalé en rouge

- Aller `Project > Properties > Targeted Runtimes`
- Sélectionner un serveur de la liste ou ajouter un nouveau en cliquant sur `New`
- Ensuite valider en cliquant sur `Apply and Close`

Maven

Si le projet est signalé en rouge

- Aller `Project > Properties > Targeted Runtimes`
- Sélectionner un serveur de la liste ou ajouter un nouveau en cliquant sur `New`
- Ensuite valider en cliquant sur `Apply and Close`

Pour exécuter

Lancer le serveur puis aller sur `http://localhost:8080/first-jee-maven/`

Maven

Arborescence d'un projet**Maven** (webapp) ≡ **JEE** Application

- `/src`
- `/src/main`
- `/src/main/java`
- `/src/main/resources`
- `/src/main/webapp` : webapp du projet
- `/src/test`
- `/src/test/java`
- `/src/test/resources`
- ...

Maven

Remarque (pour les projets **JEE**)

À la création d'une **Servlet** dans un projet **JEE** avec **Maven**, ce dernier enregistre automatiquement la **Servlet** dans le `web.xml`.

Maven

Pour gérer les dépendances

- La section (une balise) `<dependencies> </dependencies>` contenant des balises `<dependency> </dependency>`
- Dans une balise `<dependency> </dependency>`, on spécifie les détails d'une dépendance
  - `groupId`
  - `artifactId`
  - `version`
  - ...

Maven

Exemple de dépendance

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>5.3.9</version>
</dependency>
```

Maven

Où chercher les dépendances ?

- Central Repository :
  - Référentiel par défaut
  - `https://mvnrepository.com/`
- **Nexus**, **Artifactory** : Solutions pour héberger des référentiels privés.

Maven

Où chercher les dépendances ?

- Central Repository :
  - Référentiel par défaut
  - `https://mvnrepository.com/`
- **Nexus**, **Artifactory** : Solutions pour héberger des référentiels privés.

Maven

- résout automatiquement les dépendances des dépendances.
- gère automatiquement le téléchargement et la résolution des conflits.

Maven

Comment ajouter des dépendances ?

- Soit en cherchant dans le repository et en copiant la dépendance dans `pom.xml` (source)
- Soit en allant sur l'onglet `Dependencies` de `pom.xml`, de cliquer sur `Add`, chercher et sélectionner une dépendance puis valider en enregistrer et attendre le téléchargement.

Maven

`pom.xml` : **exemple**

```
<?xml version="1.0" encoding="UTF-8"?>
<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>org.eclipse</groupId>
  <artifactId>cours-maven</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>cours-maven</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.33</version>
    </dependency>
  </dependencies>
</project>
```

Maven

Cycle de vie d'un projet **Maven**

- Séquence de phases qui définissent l'ordre dans lequel les tâches de build sont exécutées.
- **Maven** propose plusieurs phases du cycle de vie.

Maven

Phases du cycle de vie

- `validate` : Valide que le projet est correct et toutes les informations nécessaires sont disponibles.
- `compile` : Compilation du code source principal.
- `test` : Exécution des tests unitaires.
- `package` : Emballage du code compilé dans son format distribué (**JAR**, **WAR**...).
- `install` : Installe le package dans le dépôt local.
- `deploy` : Copie le package final dans un dépôt distant pour le partager avec d'autres développeurs ou projets.
- ...

Maven

Liens entre les phases et les goals

- Chaque phase peut exécuter un ou plusieurs goals associés.
- Exemple : La phase `compile` exécute le goal `maven-compiler-plugin:compile`.

Maven

Liens entre les phases et les goals

- Chaque phase peut exécuter un ou plusieurs goals associés.
- Exemple : La phase `compile` exécute le goal `maven-compiler-plugin:compile`.

Exécution d'un cycle de vie

- Exécuter la commande `mvn <phase>` pour lancer Maven jusqu'à la phase spécifiée et toutes les phases précédentes.
- Exemple : `mvn package` exécute les phases de `validate` à `package`.

Maven

Goal

- Une tâche précise que l'on peut exécuter.
- Une action bien définie au sein du cycle de vie de **Maven**.
- Par exemple, le goal `compile` du plugin `maven-compiler-plugin` permet de compiler le code source **Java** du projet.

Maven

Plugin

- Un ensemble de goals.
- Un module qui étend les fonctionnalités de base de **Maven**.
- Par exemple, le plugin `maven-compiler-plugin` contient des goals pour compiler du code **Java**.

Maven

Plugin

- Un ensemble de goals.
- Un module qui étend les fonctionnalités de base de **Maven**.
- Par exemple, le plugin `maven-compiler-plugin` contient des goals pour compiler du code **Java**.

Plugins : pourquoi ?

- permettent d'organiser les fonctionnalités de Maven de manière modulaire.
- peuvent être réutilisés dans différents projets.

Plugin-goals

- Le plugin **maven-compiler-plugin** contient les goals suivants :
  - `compiler:compile` : Compile le code source
  - `compiler:testCompile` : Compile les tests
- Le plugin **maven-site-plugin** contient les goals suivants :
  - `site:site` : Génère la documentation du site pour le projet
  - `site:deploy` : Déploie la documentation du site générée dans un serveur web
- Le plugin **maven-jar-plugin** contient le goal `jar:jar` : il emballe le code compilé dans un fichier **JAR**
- Le plugin **maven-war-plugin** contient le goal `war:war` : il emballe le code compilé et les ressources dans un fichier **WAR** pour les applications **web**.
- Le plugin **maven-eclipse-plugin** contient le goal `eclipse:eclipse` : il génère les fichiers de configuration du projet pour **Eclipse IDE**.
- Le plugin **maven-clean-plugin** contient le goal `clean:clean` : il nettoie le répertoire `target` en supprimant tous les fichiers générés par le build précédent.
- Le plugin **maven-javadoc-plugin** contient le goal `javadoc:javadoc` : il génère la documentation en format **Javadoc**.
- ....

Maven

Exécution d'un cycle de vie

- Exécuter la commande `mvn <phase>` pour lancer **Maven** jusqu'à la phase spécifiée et toutes les phases précédentes.
- Exemple : `mvn package` exécute les phases de `validate` à `package`.

Maven

En exécutant `mvn compile`, les phases suivantes (avec leurs goals typiques) sont exécutées dans cet ordre

1. `validate`
2. `initialize`
3. `generate-sources`
4. `process-sources`
5. `generate-resources`
6. `process-resources` (`resources:resources` du `maven-resources-plugin`)
7. `compile` (`compiler:compile` du `maven-compiler-plugin`)

Maven

CLI : **C**ommand **L**ine **I**nterface (interface en ligne de commande)

- Une interface permettant à l'utilisateur de communiquer avec la machine en utilisant des lignes de commande
- Les commandes
  - peuvent être exécutées à partir d'une console, IDE...
  - ont la forme `command options dataOrFiles`

Maven

Pour vérifier la version de maven

```
mvn --version
```

Maven

Pour vérifier la version de maven

```
mvn --version
```

Ou le raccourci

```
mvn -v
```

Maven

Pour vérifier la version de maven

```
mvn --version
```

Ou le raccourci

```
mvn -v
```

Le résultat

```
Apache Maven 3.5.3 (...; 2018-02-24T20:49:05+01:00)
Maven home: C:\apache-maven-3.5.3\bin\..
Java version: 1.8.0_191, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jre1.8.0_191
Default locale: fr_FR, platform encoding: Cp1252
OS name: "windows10", version: "10.0", arch: "amd64", family: "windows"
```

Maven

Pour créer un projet Maven (copier la en une seule ligne)

```
mvn archetype:generate
    -DgroupId={project-packaging}
    -DartifactId={project-name}
    -DarchetypeGroupId={archetype-groupId}
    -DarchetypeArtifactId={archetype-name}
    -DinteractiveMode=false
```

Maven

Pour créer un projet Maven (copier la en une seule ligne)

```
mvn archetype:generate
    -DgroupId={project-packaging}
    -DartifactId={project-name}
    -DarchetypeGroupId={archetype-groupId}
    -DarchetypeArtifactId={archetype-name}
    -DinteractiveMode=false
```

Explication

- `mvn archetype:generate` la commande permettant de générer un projet **Maven**
- Ce qui vient après : la liste des options
  - `-DgroupId={project-packaging}` : structure des packages du projet
  - `-DartifactId={project-name}` : nom du projet
  - `-DarchetypeArtifactId={archetype-name}` : nom du archetype ou le type du projet
  - `-DarchetypeGroupId={archetype-groupId}` : l'identifiant du groupe de l'archétype (valeur par défaut `org.apache.maven.archetypes`)
  - `-DinteractiveMode=false` (facultatif) : pour éviter de demander des confirmations et d'afficher des messages détaillant les différentes phase du génération du projet

Maven

Pour créer un `Java Project` avec maven (copier la en une seule ligne)

```
mvn archetype:generate
    -DarchetypeArtifactId=maven-archetype-quickstart
    -DgroupId=org.eclipse.classes
    -DartifactId=MavenJavaProject
    -DinteractiveMode=false
```

Maven

Pour créer un `Java Project` avec maven (copier la en une seule ligne)

```
mvn archetype:generate
    -DarchetypeArtifactId=maven-archetype-quickstart
    -DgroupId=org.eclipse.classes
    -DartifactId=MavenJavaProject
    -DinteractiveMode=false
```

Remarque

- L'`artifactId` de l'archétype `quickstart` est `maven-archetype-quickstart`
- Le `groupId` de l'archétype `quickstart` est `org.apache.maven.archetypes`
- On n'a pas précisé le `groupId` ici car sa valeur correspond à la vleur par défaut considérée par Maven

Maven

Pour créer un `Java Project` avec maven (copier la en une seule ligne)

```
mvn archetype:generate
    -DarchetypeArtifactId=maven-archetype-quickstart
    -DgroupId=org.eclipse.classes
    -DartifactId=MavenJavaProject
    -DinteractiveMode=false
```

Remarque

- L'`artifactId` de l'archétype `quickstart` est `maven-archetype-quickstart`
- Le `groupId` de l'archétype `quickstart` est `org.apache.maven.archetypes`
- On n'a pas précisé le `groupId` ici car sa valeur correspond à la vleur par défaut considérée par Maven

Ensuite, se déplacer dans le projet

```
cd MavenJavaProject
```

Maven

Arborescence générée (contenant 3 fichiers sources `App.java`, `AppTest.java` et `pom.xml`)

```
install-maven-tutorial
|--pom.xml
--src
----main
------java
--------org
----------eclipse
------------classes
|--------------App.java

----test
------java
--------org
----------eclipse
------------classes
|--------------AppTest.java
```

Maven

Contenu de `App.java`

```java
package org.eclipse.classes;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

Pour compiler le projet et générer le `.jar` dans le dossier target `target`

```
mvn compile package
```

Pour compiler le projet et générer le `.jar` dans le dossier target `target`

```
mvn compile package
```

Un extrait du résultat affiché

```
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ MavenCli ---
[INFO] Building jar: C:\Users\elmou\eclipse\java-oxygen\MavenCli\target
  \MavenCli-1.0-SNAPSHOT.jar
[INFO]
  ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO]
  ------------------------------------------------------------------------
[INFO] Total time: 5.017 s
[INFO] Finished at: 2019-03-30T11:00:34+01:00
[INFO]
  ------------------------------------------------------------------------
```

Pour compiler le projet et générer le `.jar` dans le dossier target `target`

```
mvn compile package
```

Un extrait du résultat affiché

```
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ MavenCli ---
[INFO] Building jar: C:\Users\elmou\eclipse\java-oxygen\MavenCli\target
  \MavenCli-1.0-SNAPSHOT.jar
[INFO]
  ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO]
  ------------------------------------------------------------------------
[INFO] Total time: 5.017 s
[INFO] Finished at: 2019-03-30T11:00:34+01:00
[INFO]
  ------------------------------------------------------------------------
```

Pour lancer le `.jar`

```
java -cp target/MavenCli-1.0-SNAPSHOT.jar  org.eclipse.classes.App
```

Pour compiler le projet et générer le `.jar` dans le dossier target `target`

```
mvn compile package
```

Un extrait du résultat affiché

```
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ MavenCli ---
[INFO] Building jar: C:\Users\elmou\eclipse\java-oxygen\MavenCli\target
  \MavenCli-1.0-SNAPSHOT.jar
[INFO]
  ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO]
  ------------------------------------------------------------------------
[INFO] Total time: 5.017 s
[INFO] Finished at: 2019-03-30T11:00:34+01:00
[INFO]
  ------------------------------------------------------------------------
```

Pour lancer le `.jar`

```
java -cp target/MavenCli-1.0-SNAPSHOT.jar  org.eclipse.classes.App
```

Le résultat

```
Hello World!
```

Maven

Pour créer un `Web Project` avec maven (copier la en une seule ligne)

```
mvn archetype:generate
    -DgroupId=org.eclipse
    -DartifactId=MavenWebProject
    -DarchetypeArtifactId=maven-archetype-webapp
    -DinteractiveMode=false
```

Maven

Autres commandes **Maven**

- `mvn javadoc:javadoc` pour générer la Javadoc
- `mvn test` pour exécuter les tests unitaires
- `mvn clean` pour supprimer les fichiers générés
- `mvn -Dmaven.test.skip=true clean package` pour supprimer les fichiers générés reconstruire le projet sans exécuter les tests unitaires
- ...

Maven

Autres commandes **Maven**

- `mvn javadoc:javadoc` pour générer la Javadoc
- `mvn test` pour exécuter les tests unitaires
- `mvn clean` pour supprimer les fichiers générés
- `mvn -Dmaven.test.skip=true clean package` pour supprimer les fichiers générés reconstruire le projet sans exécuter les tests unitaires
- ...

Liste des options

`http://maven.apache.org/ref/3.1.0/maven-embedder/cli.html`

Maven

Création et utilisation de profils

- Profils pour gérer différentes configurations d'environnement (dev, test, prod).
- Activation des profils
  - via la ligne de commande : `mvn clean install -Pdev`
  - ou dans `pom.xml`

Maven

Exemple avec activation du profil développement dans `pom.xml`

```
<profiles>
    <profile>
        <id>dev</id>
         <activation>
            <properties>
                <env>development</env>
            </properties>
        </activation>
    </profile>
    <profile>
        <id>prod</id>
        <properties>
            <env>production</env>
        </properties>
    </profile>
</profiles>
```