Développement des systèmes d'information

Services Web

01- Outil Maven

Mohamed ZAYANI

ISET-SFAX - 2024/2025

C’est quoi Maven?

- Maven est un outil de gestion de projet logiciel pour Java maintenu par « l'Apache Software Foundation ».
- Maven est un outil qui permet de gérer les dépendances d’un projet JAVA et automatiser sa construction :
  - compilation, test,
  - packaging,
  - déploiement,
  - production de livrable
  - gestion de sites web

- Maven permet aussi de générer des documentations (sous forme de rapports) concernant le projet.

Caractéristiques Maven

Principe de Convention plutôt que configuration

- Maven établit un certain nombre de conventions afin d'automatiser certaines tâches et rendre la procédure de configuration plus facile.
- Une de ces conventions est de fixer l'arborescence d'un projet. Ainsi, Maven permet de générer une squelette du code du projet en utilisant la notion des « archetypes ».

Approche déclarative

- Maven utilise une approche déclarative où la structure du projet et son contenu sont décrits dans un document XML nommé POM.xml
  (Project Object Model)

Aspect extensible

- Maven est extensible grâce à un mécanisme de plugins qui permettent d'ajouter des fonctionnalités.

Notion d'artéfact

- Un artéfact est un composant packagé possédant un identifiant unique composé de trois éléments : un groupId, un artifactId et un numéro de version.

1. groupId : définit l'organisation ou groupe qui est à l'origine du projet. Il est formulé sous la forme d'un package Java

   (Exemple : org.jee.maven)

2. artifactId : définit le nom du projet (nom unique dans le groupe)

   (Exemple: premierProjet)

3. version : définit la version du projet. Les numéros de version sont souvent utilisés pour des comparaisons et des mises à jour.

NB: La gestion des versions est importante pour identifier quel artefact doit être utilisé: la version est utilisée comme une partie de l'identifiant d'un artéfact.

Exemple

- La déclaration d'une dépendance est spécifiée dans le fichier « pom.xml » (cœur de Maven)
- Exemple:

```
<dependency>
   <groupId>org.hibernate</groupId>
   <artifactId>hibernate</artifactId>
   <version>3.3.2</version>
   <scope>compile</scope>
</dependency>
```

- La notion de «scope» définit la portée d'une dépendance: permet de préciser dans quel contexte une dépendance est utilisée
- La portée « compile » indique que la dépendance est utilisable par toutes les phases et à l'exécution. C'est le scope par défaut

Fichier « pom.xml »

- Le fichier POM (Project Object Model) contient la description du projet Maven Il contient les informations nécessaires à la génération du projet : (identification de l'artéfact, déclaration des dépendances, définition d'informations relatives au projet..). Voici un exemple:

```
<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>com.jee.test</groupId>
    <artifactId>MaWebApp</artifactId>
    <packaging>war</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>Mon application web</name>
    <dependencies>
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>4.7</version>
                <scope>test</scope>
            </dependency>
    </dependencies>
</project>
```

Balises du fichier « pom.xml »

dossier	Description
<modelVersion>	Préciser la version du modèle de POM utilisée
<groupId>	Préciser le groupe ou l'organisation qui développe le projet. C'est une des clés utilisée pour identifier de manière unique le projet et ainsi éviter les conflits de noms
<artifactId>	Préciser la base du nom de l'artéfact du projet
<packaging>	Préciser le type d'artéfact généré par le projet (jar, war, ear, pom, ...). La valeur par défaut est jar
<version>	Préciser la version de l'artéfact généré par le projet. Le suffixe -SNAPSHOT indique une version en cours de développement
<name>	Préciser le nom du projet utilisé pour l'affichage

Balises du fichier « pom.xml »

dossier	Description
<description>	Préciser une description du projet
<url>	Préciser une url qui permet d'obtenir des informations sur le projet
<dependencies>	Définir l'ensemble des dépendances du projet
<dependency>	Déclarer une dépendance en utilisant plusieurs tags fils : <groupId>, <artifactId>, <version> et <scope>

- Le fichier POM doit être à la racine du répertoire du projet.
- La balise racine du fichier « pom.xml » est la balise <project>.

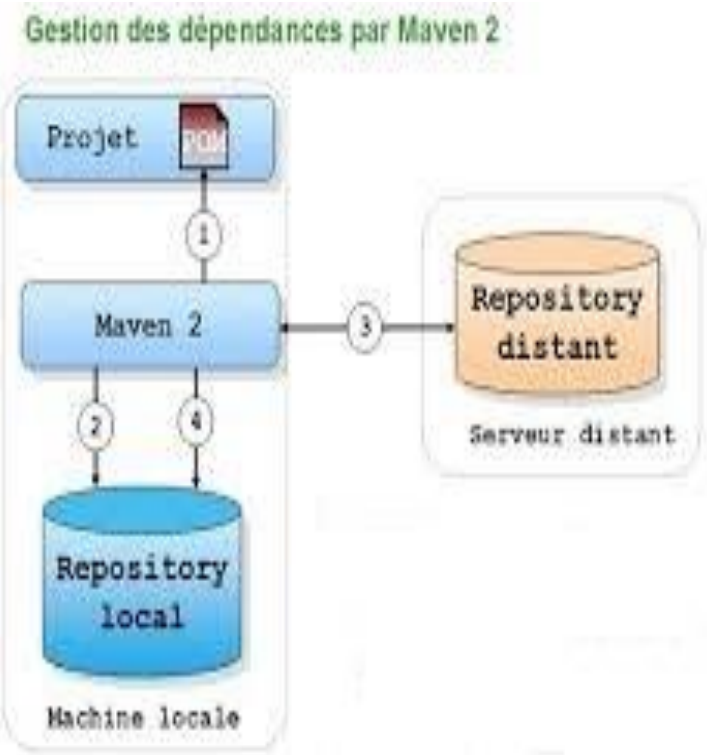
Gestion de dépendances

- Maven utilise la notion de référentiel ou dépôt (repository) pour stocker les dépendances et les plugins requis pour générer les projets.
- Un dépôt contient un ensemble d'artéfacts qui peuvent être des livrables, des dépendances, des plugins, ...
- Ceci permet de centraliser ces éléments qui sont généralement utilisés dans plusieurs projets : c'est notamment le cas pour les plugins et les dépendances.
- Maven distingue deux types de dépôts : local et distant (remote):

1. **dépôt central (repository central):** il stocke des dépendances et les plugins utilisables par tout le monde car disponible sur le web; ce sont généralement des artéfacts open source
2. **dépôt local (repository local)** : il stocke une copie des dépendances et plugins requis par les projets à générer en local. Ces artéfacts sont téléchargés des dépôts centraux

Gestion de dépendances

- Maven utilise un ou plusieurs dépôts (repository) qui peuvent être locaux ou distants
- Si un élément n'est pas trouvé dans le répertoire local, il sera téléchargé dans ce dernier à partir d'un dépôt distant



1. Lecture du fichier « pom.xml » et la liste des dépendances
2. Vérification de l'existence des dépendances dans le repository local (dépôt local)
3. Téléchargement des dépendances non trouvées en accédant au repository central (via Internet)
4. Copies de dépendances dans le repository local

Repository Maven

- La première exécution d'une commande « Maven », un dossier nommé « .m2 » est créé dans le répertoire « HOME » de l'ordinateur

  (Exemple: c:\Utilisateurs\Ma_Machine\ )
- Le dossier «.m2» ,ainsi créé, comporte un sous-dossier « repository » constituant le dépôt local de Maven
- Il est possible de personnaliser l'emplacement du repository local en spécifiant le chemin dans un fichier « settings.xml » à placer dans le dossier « .m2 ».

Fichier « settings.xml »

Nouveau dossier du dépôt local

```
<settings>
  ...
<localRepository>D:\Maven\repository</localRepository>
  ...
</settings>
```

Archetypes Maven

- Afin de générer la squelette d'un projet, Maven s'appuie sur des archétypes (ou modèles).
- Un Archetype est un outil pour faire des templates de projet.
- Un projet généré via un Archetype est dit projet Maven.
- Un Archetype est un modèle de projet à partir duquel d'autres projets sont créés.
- L'utilisation d'archétypes a pour principal avantage de normaliser le développement de projets et de permettre aux développeurs de suivre facilement les meilleures pratiques tout en débutant leurs projets plus rapidement.
- Maven fournit, aux utilisateurs, une très grande liste de différents types de modèles de projet (ENVIRON 614) en utilisant le concept d'Archetype.

Exemples d'Archetypes Maven

archetype	Description
quickstart	Contient un exemple projet Maven standard
simple	Contient un simple projet Maven
webapp	Contient un exemple projet Maven d'une application web
j2ee-simple	Contient un exemple projet Maven d'une application JEE

- Maven permet de créer la structure d'un projet selon un modèle donné (archetype) en utilisant la commande suivante:

```
mvn archetype:generate
```

Structure d'un projet Maven

dossier	Description
/src	les sources du projet
/src/main	les fichiers sources principaux
/src/main/java	le code source (sera compilé dans /target/classses)
/src/main/resources	les fichiers de ressources (fichiers de configuration, images, ...). Le contenu de ce répertoire est copié dans target/classes pour être inclus dans l'artéfact généré
/src/main/webapp	les fichiers de l'application web
/src/test	les fichiers pour les tests
/src/test/java	le code source des tests (sera compilé dans /target/test-classses)

Structure d'un projet Maven

dossier	Description
/src/test/resources	les fichiers de ressources pour les tests
/target	les fichiers générés pour les artéfacts et les tests
/target/classes	les classes compilées
/target/test-classes	les classes compilées des tests unitaires
/target/site	site web contenant les rapports générés et des informations sur le projet
/pom.xml	le fichier POM de description du projet

NB: l'arborescence d'un projet Maven est par défaut imposée par l'outil Maven

(par convention)

Principe de fonctionnement du Maven 3

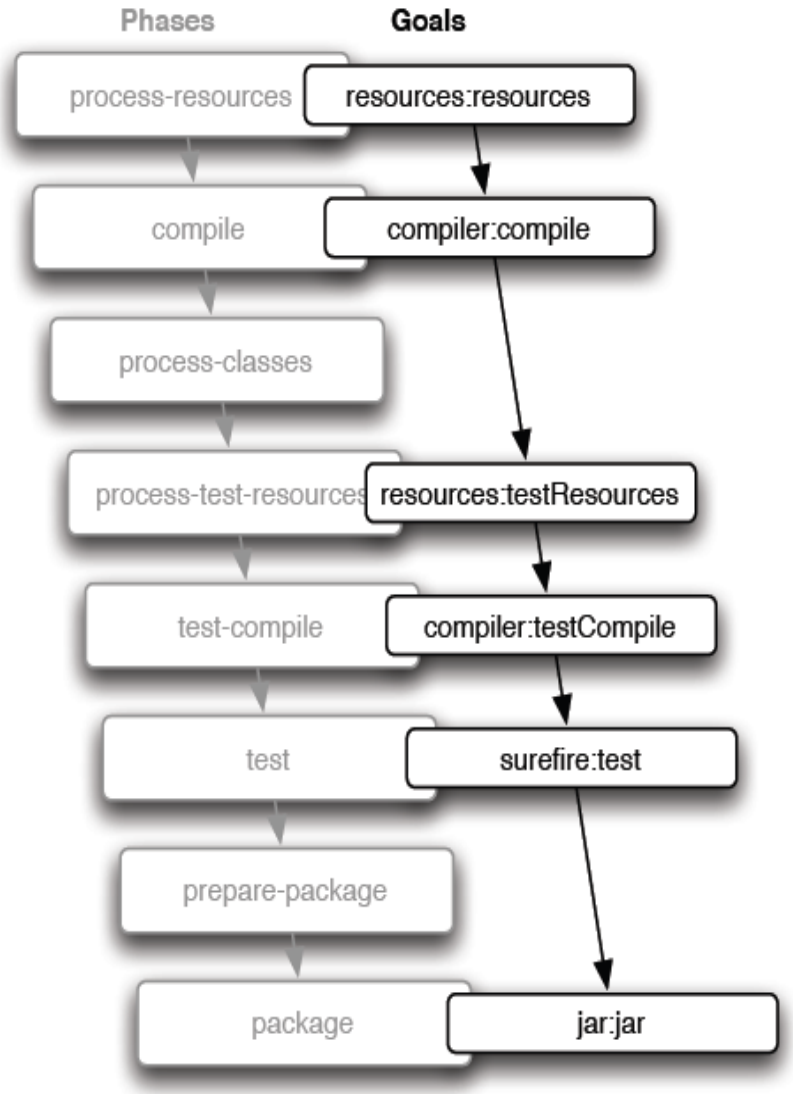
- Toutes les fonctionnalités décrites ici font partie de la version 3 de Maven.
- Une connexion à internet est nécessaire pour permettre le téléchargement des plugins requis et des dépendances.
- Pour installer Maven:
  - Télécharger l'archive sur le site:

    http://maven.apache.org/download.html
  - Décompresser l'archive dans un répertoire du système
  - Créer la variable d'environnement M2_HOME qui pointe sur le répertoire contenant Maven
  - Ajouter le chemin M2_HOME/bin à la variable PATH du système
  - Pour vérifier l'installation, il suffit de lancer la commande:

    mvn -version

Cycle de vie d'un projet Maven

- Dans le cycle de vie 'par défaut' d'un projet Maven, les phases les plus utilisées sont:
  - validate :vérifie les prérequis d'un projet maven
  - compile : compilation du code source
  - test : lancement des tests unitaires
  - package : assemble le code compilé en un livrable
  - install : partage le livrable pour d'autres projets sur le même ordinateur
  - deploy : publie le livrable pour d'autres projets dans le « repository » distant
- Les phases s'exécutent de façon séquentielle de façon à ce qu'une phase dépende de la phase précédente.
- Par exemple, le lancement par l'utilisateur de la phase test (mvn test) impliquera le lancement préalable par maven des phases « validate » et « compile ».



Commandes Maven 3

- Toutes les fonctionnalités décrites font partie de la version 3 de Maven.
- Une commande Maven3 s'utilise en ligne de commande sous la forme suivante :

mvn plugin:goal ou **mvn plugin**

- Exemple:

```
mvn archetype:generate
```

- Il est possile d'utiliser des options précédées par « - »
- Exemple:

```
mvn -version
```

Exemples de commandes Maven 3

commande	Description
mvn package	Construire le projet pour générer l'artéfact
mvn site	Générer le site de documentation dans le répertoire target/site
mvn clean	Supprimer les fichiers générés par les précédentes générations
mvn install	Générer l'artéfact et le déployer dans le dépôt local
mvn eclipse:eclipse	Générer des fichiers de configuration Eclipse à partir du POM (notamment les dépendances)
mvn javadoc:javadoc	Générer la Javadoc
mvn test	Exécuter les tests unitaires