Démo Maven

Pré requis : préparation de l'environnement de test

Maven est un outil Java, vous devez donc avoir installé Java pour continuer.

```
sudo yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel
```

Tout d'abord, téléchargez Maven et suivez les instructions d'installation .

```
1. wget https://dlcdn.apache.org/maven/maven-3/3.8.3/binaries/apache-maven-
   3.8.3-bin.tar.gz --no-check-certificate (vérifier la version!)
2. tar xzvf apache-maven-3.8.3-bin.tar.gz
3. export PATH=/chemin_vers_maven/apache-maven-3.8.3/bin:$PATH
```

Après cela, tapez ce qui suit dans un terminal ou dans une invite de commande :

```
mvn -version
```

Il devrait imprimer votre version installée de Maven, par exemple:

```
Apache Maven 3.0.5 (Red Hat 3.0.5-17)
Maven home: /usr/share/maven
Java version: 1.8.0_265, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.265.b01-1.amzn2.0.1.x86_64/jre
Default locale: en_US, platform encoding: ANSI_X3.4-1968
OS name: "linux", version: "4.14.209-160.339.amzn2.x86_64", arch: "amd64", family:
"unix"
```

Créer un projet

- Pour créer un projet Maven, vous pouvez utiliser la commande mvn en ligne de commande en spécifiant un archetype **(un modèle de projet prédéfini)** pour générer la structure initiale du projet. La syntaxe générale pour créer un projet Maven est la suivante :

```
mvn archetype:generate
    -DgroupId=
    -DartifactId=
    -DarchetypeArtifactId=
    -DarchetypeVersion=
    -DinteractiveMode=false
```

- **`mvn archetype:generate`**: C'est la commande Maven pour générer un projet à partir d'un archetype.
- **`-DgroupId=<ID_DU_GROUPE>`** : C'est l'ID du groupe auquel appartient le projet. Cela correspond généralement à un identifiant unique de votre organisation ou de votre projet.
- **`-DartifactId=<ID_DU_PROJET>`** : C'est l'ID du projet. Cela correspond généralement au nom du projet.
- **`-DarchetypeArtifactId=<ID_DE_L_ARCHETYPE>`** : C'est l'ID de l'archetype à utiliser pour générer la structure du projet. Vous pouvez spécifier l'archetype souhaité en fonction de vos besoins (par exemple, **`maven-archetype-quickstart`** pour un projet Java simple).

- **`-DarchetypeVersion=<VERSION_DE_L_ARCHETYPE>`** : C'est la version de l'archetype à utiliser.
- **`-DinteractiveMode=false`** : Cela désactive le mode interactif pour éviter les invitations à l'utilisateur lors de la génération du projet.

Vous aurez besoin d'un endroit pour que votre projet réside, créez un répertoire quelque part et démarrez un shell dans ce répertoire. Sur votre ligne de commande, exécutez l'objectif Maven suivant:

```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app-maven
-DarchetypeArtifac

lstId=maven-archetype-quickstart -DarchetypeVersion=1.4 -
DinteractiveMode=false
```

Si vous venez d'installer Maven, la première exécution peut prendre un certain temps. C'est parce que Maven télécharge les artefacts les plus récents (jars de plugins et autres fichiers) dans votre référentiel local

Vous remarquerez que l'objectif de *génération a* créé un répertoire avec le même nom que l'artefactId. Allez dans ce répertoire.chtype :

```
cd my-app-maven
```

Sous ce répertoire, vous remarquerez la structure de projet standard suivante .

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

Le répertoire `src/main/java` contient le code source du projet, le répertoire `src/test/java` contient la source du test et le fichier `pom.xml` est le modèle d'objet de projet ou POM du projet.

Le POM (Project Object Model)

`pom.xml` est le fichier de configuration principal utilisé par Maven pour gérer un projet. Il spécifie les détails du projet, les dépendances, les plugins, les configurations de construction et d'autres éléments nécessaires à la construction et à la gestion du projet. Le POM de ce projet est:

```
1.  C'est l'élément racine du fichier pom.xml
2.  <project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3.  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
4.  <modelVersion>4.0.0</modelVersion> Indique la version du modèle de projet Maven.
5.  <groupId>com.mycompany.app</groupId> spécifie l'ID du groupe du projet.
6.  <artifactId>my-app</artifactId> l'ID de l'artefact (nom du projet).
7.  <version>1.0-SNAPSHOT</version> la version du projet.
8.   <properties>
9.  <maven.compiler.source>1.7</maven.compiler.source>
10. <maven.compiler.target>1.7</maven.compiler.target>
11. </properties>
12.  <dependencies> Définit une dépendance spécifique avec un groupe, un artefact et une version.
13. <dependency>
14. <groupId>junit</groupId>
15. <artifactId>junit</artifactId>
16. <version>4.12</version>
17. <scope>test</scope>
18. </dependency>
19. </dependencies>
20. </project>
```

Builder le projet

```
mvn package
```

La ligne de commande imprimera diverses actions et se terminera par ce qui suit:

```
 ...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.953 s
[INFO] Finished at: 2019-11-24T13:05:10+01:00
[INFO] ------------------------------------------------------------------------
```

Contrairement à la première commande exécutée (*archétype: generate*), vous remarquerez peut-être que la seconde est simplement un seul mot - *package* . Plutôt qu'un *objectif* , c'est une *phase* . Une phase est une étape du cycle de vie de la construction , qui est une séquence ordonnée de phases. Lorsqu'une phase est donnée, Maven exécutera toutes les phases de la séquence jusqu'à et y compris celle définie. Par exemple, si nous exécutons la phase de *compilation*, les phases qui sont réellement exécutées sont:

1. validate
2. generate-sources
3. process-sources
4. generate-resources
5. process-resources
6. compile

Vous pouvez tester le JAR nouvellement compilé et empaqueté avec la commande suivante:

```
java -cp target/my-app-1.0-SNAPSHOT.jar com.mycompany.app.App
```

Ce qui imprimera la chaine:

```
Hello World!
```

Exécution des outils Maven

Phases Maven

Bien que cette liste ne soit pas exhaustive, ce sont les phases de cycle de vie *par défaut* les plus courantes exécutées.

- **valider** : valider que le projet est correct et que toutes les informations nécessaires sont disponibles
- **compile** : compile le code source du projet
- **test** : testez le code source compilé à l'aide d'un cadre de test unitaire approprié. Ces tests ne devraient pas exiger que le code soit empaqueté ou déployé
- **package** : prenez le code compilé et empaquetez-le dans son format distribuable, tel qu'un JAR.
- **intégration-test** : traiter et déployer le package si nécessaire dans un environnement où les tests d'intégration peuvent être exécutés
- **vérifier** : exécutez toutes les vérifications pour vérifier que le package est valide et répond aux critères de qualité
- **install** : installe le package dans le référentiel local, pour l'utiliser comme dépendance dans d'autres projets localement
- **deploy** : fait dans un environnement d'intégration ou de version, copie le package final dans le référentiel distant pour le partager avec d'autres développeurs et projets.