

# JAVAE- MAVEN



# PLAN DU COURS

- C'est Quoi Maven?
- But de la Formation
- Création d'un Projet Maven
- Balises POM.XML
- Arborescence Standard
- Buts (Goals)
- Tests
- Annexe
- Place à la Pratique

# C'EST QUOI MAVEN?

- Maven est un outil de construction de projets (build) open source développé par la fondation Apache.
- Il permet de faciliter et d'automatiser certaines tâches de la gestion d'un projet Java JEE.

# FONCTIONNALITÉS

- Fonctionnalités
  - Automatisation de tâches récurrentes
  - Construction, compilation des projets Gestion des dépendances
  - Gestion du code source
  - Génération de la documentation et de rapports
  - Déploiement d'applications
- Modèle de projet basé sur des conventions (POM)
  - Configuration basée sur le format XML

# INSTALLATION DE MAVEN

- Maven entant que Plugin : Un plugin Maven est installé par défaut avec Eclipse.
- Maven en stand-alone :
  - Télécharger Maven : Site <http://maven.apache.org> ,  
apache-maven-xxx.zip
  - Décompresser l'archive dans le dossier de votre choix, par exemple : C:\Products\apache-maven-3.5.0

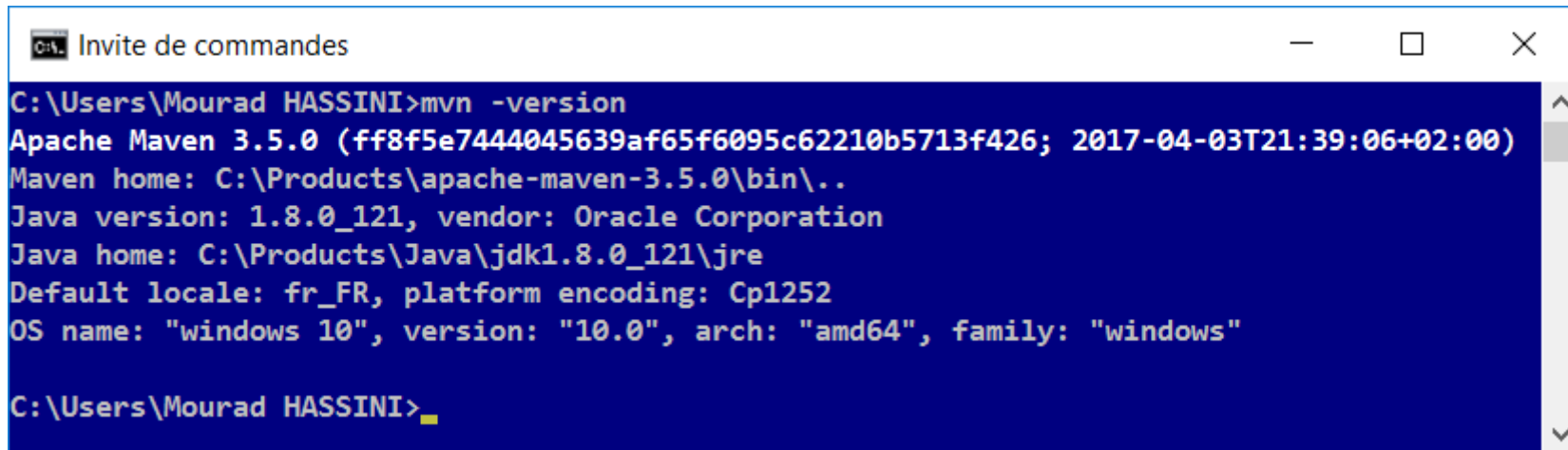
# INSTALLATION DE MAVEN

- Variables d'environnement :

Nom	Description	Exemples
JAVA_HOME	Répertoire racine du JDK	C:\Products\Java\jdk1.8.0_121
M2_HOME	Répertoire racine de Maven	C:\Products\apache-maven-3.5.0
PATH	Chemin d'accès vers les principaux exécutables du système	PATH=%PATH%;%JAVA_HOME%\bin;%M2_HOME%\bin

# PREMIÈRES COMMANDES DE MAVEN

- Maven est-il présent et quelle version :



```
C:\Users\Mourad HASSINI>mvn -version
Apache Maven 3.5.0 (ff8f5e7444045639af65f6095c62210b5713f426; 2017-04-03T21:39:06+02:00)
Maven home: C:\Products\apache-maven-3.5.0\bin\..
Java version: 1.8.0_121, vendor: Oracle Corporation
Java home: C:\Products\Java\jdk1.8.0_121\jre
Default locale: fr_FR, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

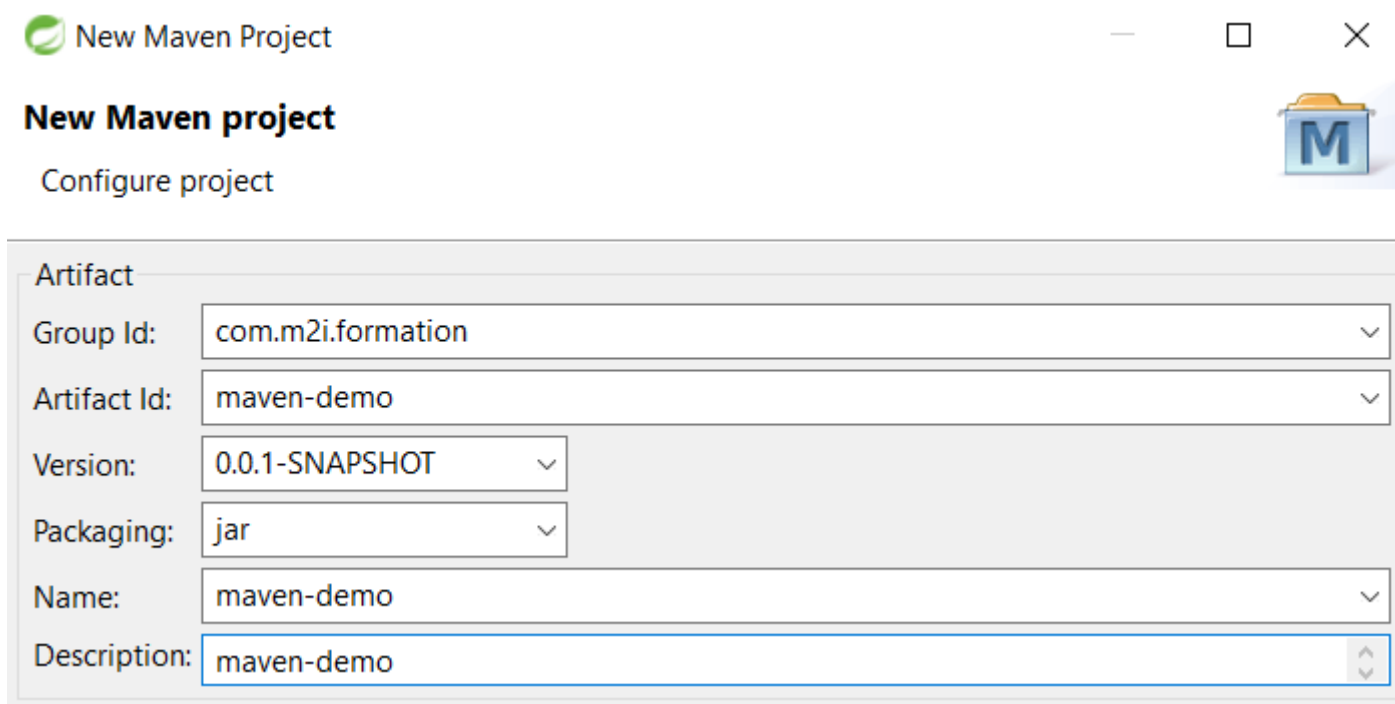
C:\Users\Mourad HASSINI>
```

# BUT DE LA FORMATION

- Créer un projet MAVEN simple via le plugin Maven
- Maîtriser l'arborescence standard du code et de ses ressources
- Maîtriser les différents buts (Goals) du cycle de vie d'un projet Maven (la compilation, le test, le packaging d'une application, ...)
- Installer une application dans un Repository local
- Gérer les dépendances (bibliothèques) d'un projet donné
- Exécuter les tests unitaires automatiquement



# CRÉATION D'UN PROJET MAVEN



New Maven Project

**New Maven project**

Configure project

Artifact

Group Id: com.m2i.formation

Artifact Id: maven-demo

Version: 0.0.1-SNAPSHOT

Packaging: jar

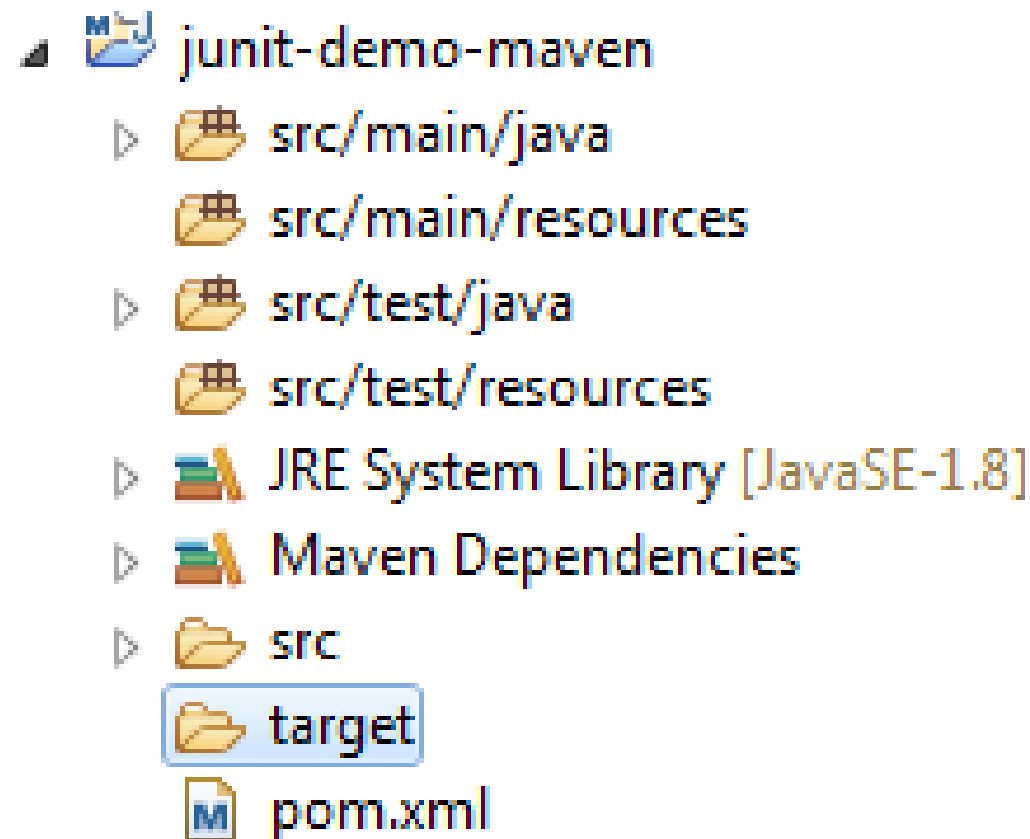
Name: maven-demo

Description: maven-demo

# BALISES DU POM.XML

- **pom.xml** : Projet Model Object
- **project** : Balise racine de tous les fichiers pom.xml.
- **modelVersion** : Version de POM utilisée.
- **groupId** : Identifier un groupe qui a créé le projet. Ex: org.apache.
- **artifactId** : Nom unique utilisé pour nommer l'artifacts à construire.
- **packaging** : Type de packaging du projet ( ex. : JAR, WAR, EAR...).
- **version** : Version de l'artifact généré par le projet.
- **name** : Nom du projet.
- **description** : Description du projet.
- **dependencies** : balise permettant de gérer les dépendances.
- **archetype** : Template de Projet.

# ARBORESCENCE STANDARD



# PREMIÈRES COMMANDES

- Mettez vous sur la racine du projet maven, en ligne de commande et lancer la commande : **mvn eclipse:eclipse**
- En ligne de commande lancer : **mvn clean install**
- Changer la version de Java utilisée dans le projet à Java 8.

# ARBORESCENCE STANDARD

- **pom.xml** : le fichier de configuration du projet
- **/src** : code source et fichiers source principaux
- **/src/main/java** : code source java
- **/src/main/resources** : fichiers de ressources (images, fichiers config...)
- **/src/main/webapp** : webapp du projet
- **/src/test** : fichiers de test
- **/src/test/java** : code source Java de test
- **/src/test/resources** : fichiers de ressources de test
- **/src/site** : informations sur le projet et/ou les rapports générés suites aux traitements effectués
- **/target** : fichiers résultat, les binaires (du code et des tests), les packages générés et les résultats des tests

# BUTS (GOALS)

- **mvn compile** : Créer les .class
- **mvn test** : Jouer les tests unitaires
- **mvn package** : Creation du livrable dans target.
- **mvn install** : Copie du livrable dans le Repository local :  
~\.m2\repository\...
- **mvn deploy** : Copie du livrable sur le repository central
- **mvn clean** : Supprime le contenu du dossier target.

# TESTS

- mvn install -**Dmaven.test.skip=true**
- mvn install -**Dmaven.test.skip=false**
- Il est possible de configurer le plugin Maven de Eclipse pour sauter les tests.

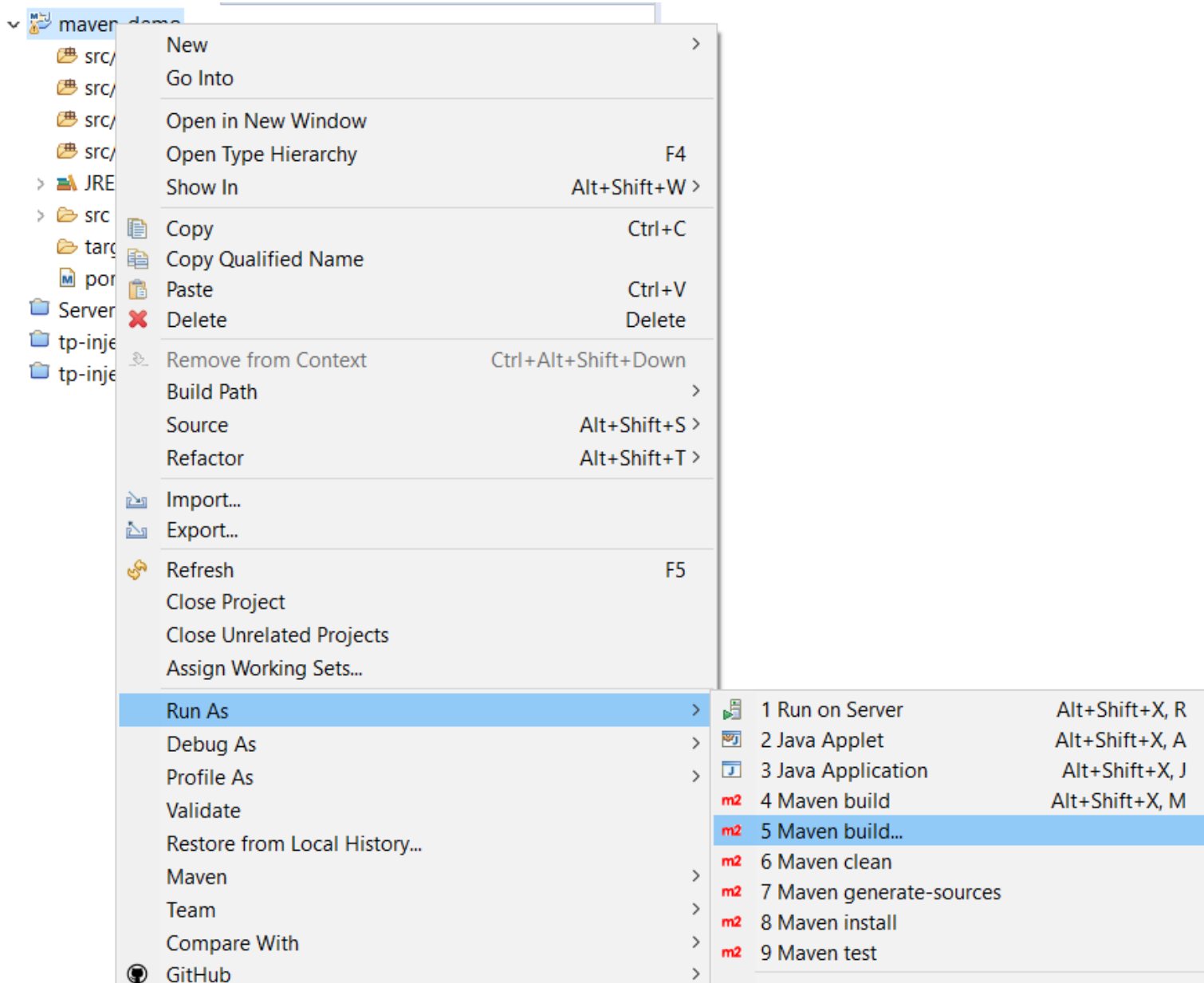
# BUTS (GOALS)

- Emplacement du livrable :  
    {emplacement Repository}/groupId/artifactId/version
- Nom du package (jar en général) : {artifactId}-{version}.{package}

```
<dependency>  
  <groupId>log4j</groupId>  
  <artifactId>log4j</artifactId>  
  <version>1.2.17</version>  
</dependency>
```



# EXÉCUTION SOUS ECLIPSE



# PLACE A LA PRATIQUE

**Développement d'un Projet qui implémente  
MAVEN**

**Ajout de MAVEN dans un Projet déjà existant**