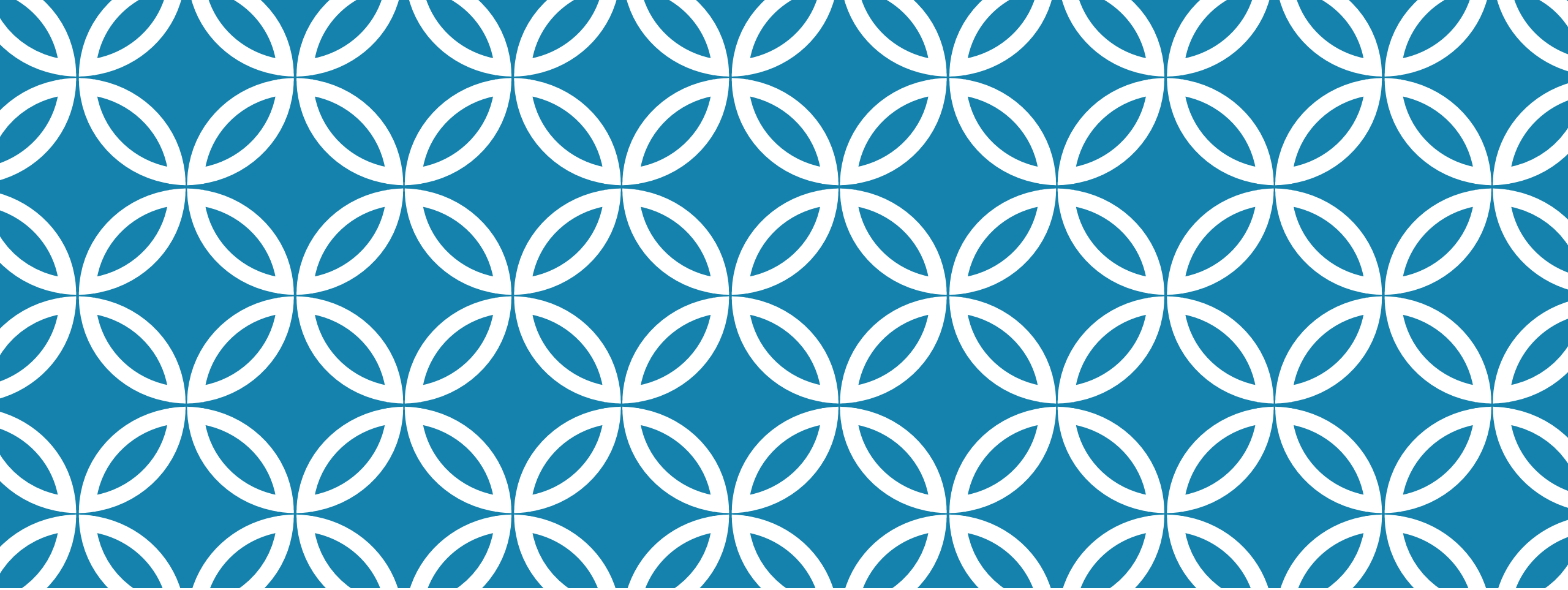
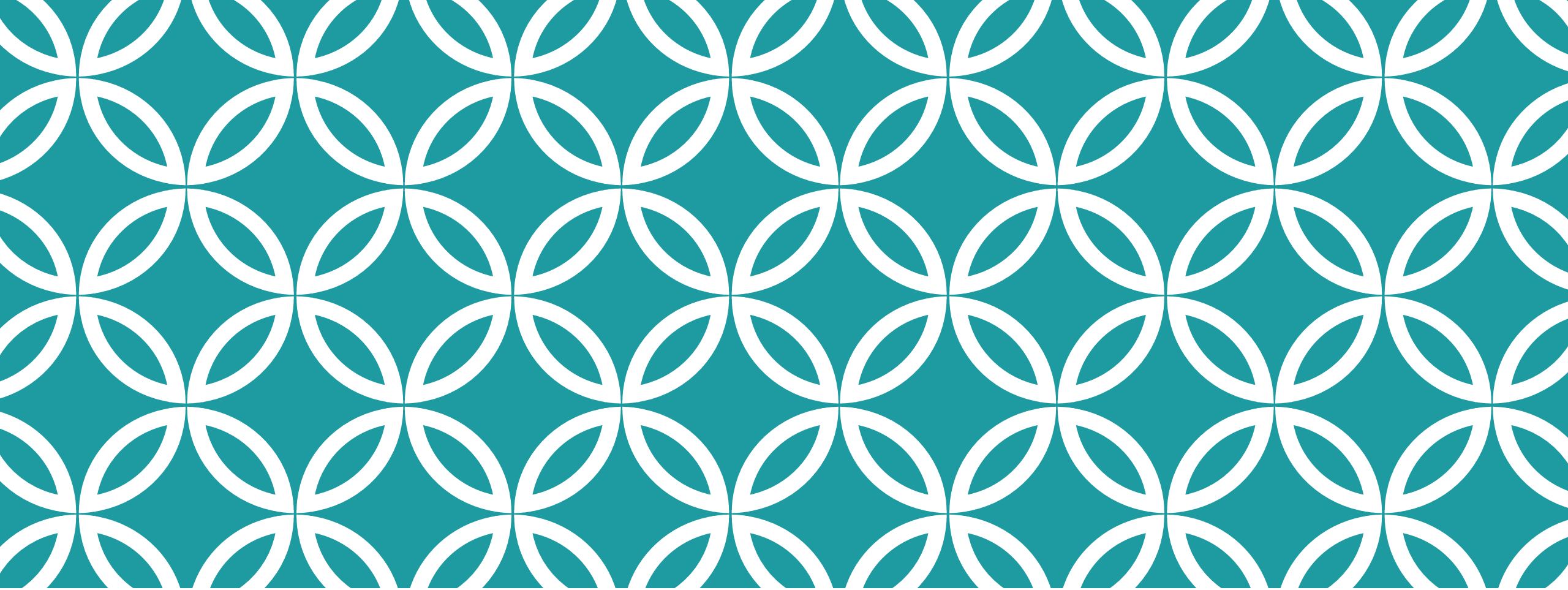


JAVA EE

Jérémy PERROUAULT



MAVEN

Introduction à Maven

INTRODUCTION

Ensemble de standards

Fournit un cycle de vie standard

- Compiler (build)
- Tester – Intégrer
- Déployer

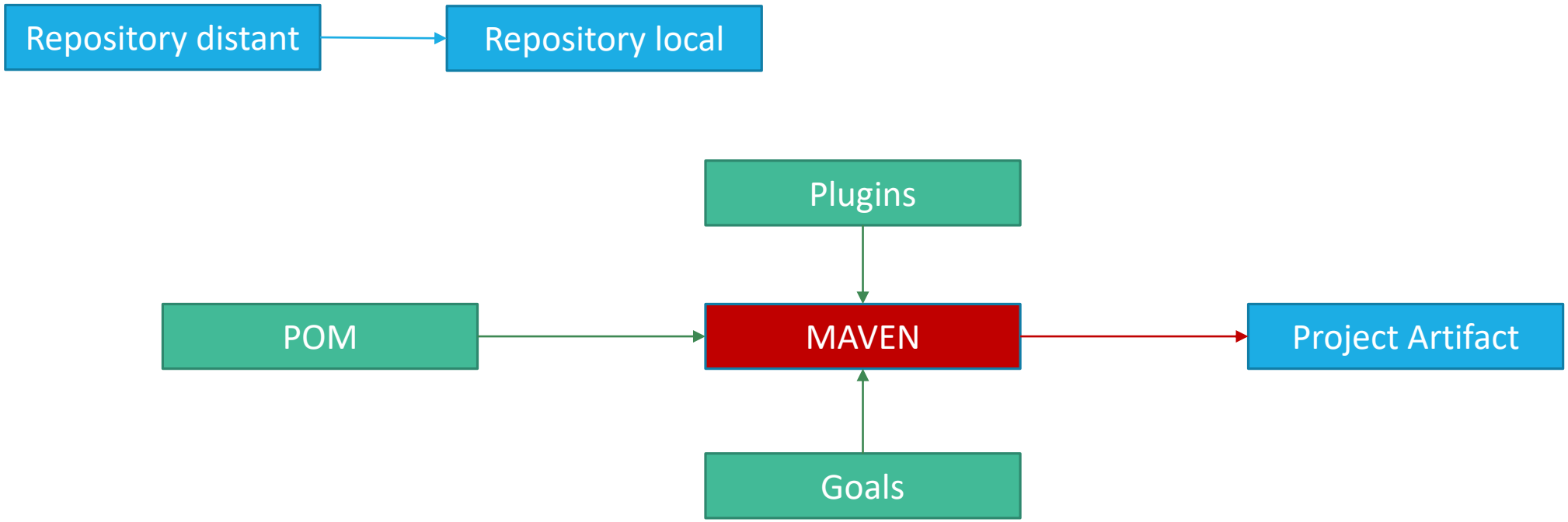
S’articule autour d’une description de projet (POM – Project Object Model)

Organise les projets en composants interdépendants

- Permet une gestion fine des dépendances

Apporte cohérence, réutilisabilité, agilité et maintenabilité

INTRODUCTION



INTRODUCTION

Le repository local est situé dans ce répertoire

- %USER_HOME%/.m2/repository

Vous pouvez la modifier dans la configuration de Maven, le fichier settings.xml

- maven/conf/settings.xml

CONFIGURATION

Chaque projet Maven est défini dans un fichier *pom.xml*

- Format XML

Contient l'identité du projet, le *GAV*

- GroupId
- ArtifactId
- Version

La liste des dépendances

La liste des tâches (au travers de plugin) à réaliser pendant les phases

- De compilation
- De test
- D'assemblage
- D'installation

CONFIGURATION

```xml
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>fr.formation</groupId>
  <artifactId>tp-maven</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <properties>...</properties>
  <dependencies>...</dependencies>
  <build>...</build>
</project>
```

project

- Racine du fichier *pom.xml*

modelVersion

- Version de POM utilisée

groupId

- Identifiant du groupe du projet
- Suit les mêmes règles de nommage que les packages

artifactId

- Identifiant du projet dans le groupe
- Utilisé par défaut pour construire l'artefact final
  - artefictid-version

version

- La version du projet
- SNAPSHOT
  - Version-SNAPSHOT est la Version en cours de développement

packaging

- Type de packaging du projet (JAR, WAR ou POM)

dependencies

- Liste des dépendances

build

- Liste des plugins pour le processus de construction

properties

- Propriétés, paramètres Maven

CONFIGURATION – PACKAGING

JAR

- Projet de bibliothèque Java
- Projet d'éxécutable Java

WAR

- Projet WEB

POM

- Projet Maven générique
- Regroupe des dépendances et propriétés
- Pourra être utilisé par d'autres projet Maven (Héritage)
  - Les projets hériteront des dépendances et des propriétés !

```
<parent>
  <groupId>fr.formation</groupId>
  <artifactId>maven-parent</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</parent>
```

CONFIGURATION – PROPERTIES

Propriétés, paramètres

```
<properties>
  <spring.version>5.0.2.RELEASE</spring.version>
</properties>
```

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>${spring.version}</version>
</dependency>
```

Modifier des propriétés préexistantes

- Indiquer à Maven quelle version de Java utiliser

```
<properties>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

CONFIGURATION – DEPENDENCIES

Dépendances

- Référence vers un artefact spécifique contenu dans un repository
- On y fait référence avec son *GAV*
- On peut préciser sa portée (scope)
  - compile ➔ disponible dans toutes les phases, valeur par défaut
  - provided ➔ utilisée lors de la compilation, mais pas déployée
  - runtime ➔ déployée, mais pas nécessaire à la compilation
  - test ➔ compiler et exécuter des tests

```
<dependencies>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```

Pour rechercher une dépendance, vous pouvez utiliser http://mvnrepository.com/

CONFIGURATION – DEPENDENCIES

Scope	Compilation	Exécution	Distribution	Déploiement
Compile	X	X	X	X
Provided	X	X		
Runtime		X	X	X
Test				

CONFIGURATION – DEPENDENCYMANAGEMENT

Gestionnaire de dépendances

- Confier la gestion des versions au projet MAVEN parent

Projet parent

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Projet enfant

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
  </dependency>
</dependencies>
```

CONFIGURATION – BUILD

Processus build

- Indique comment compiler le projet

```xml
<build>
  <plugins>
    <plugin>
      <!-- ... -->
    </plugin>

    <plugin>
      <!-- ... -->
    </plugin>
  </plugins>
</build>
```

```xml
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.5.1</version>
  <configuration>
    <source>1.8</source>
    <target>1.8</target>
  </configuration>
</plugin>
```

```xml
<plugin>
  <artifactId>maven-war-plugin</artifactId>
  <version>2.6</version>
  <configuration>
    <warSourceDirectory>WebContent</warSourceDirectory>
    <archiveClasses>false</archiveClasses>
  </configuration>
</plugin>
```

INITIALISATION D'UN PROJET

Maven peut utiliser des archetypes (un pattern – un modèle), constitué

- D'un descripteur (archetype.xml)
- Des templates (fichiers et répertoires) qui seront copiés pour le nouveau projet
- D'un fichier POM

Il en existe déjà

- maven-archetype-archetype
- maven-archetype-quickstart
- maven-archetype-j2ee-simple
- maven-archetype-plugin
- maven-archetype-simple
- maven-archetype-site
- maven-archetype-webapp

Paramètre DarchetypeArtifactId à passer lors de la commande d'initialisation

INITIALISATION D'UN PROJET

Pour initialiser un nouveau projet

```
mvn archetype:generate
  "-DarchetypeArtifactId=maven-archetype-quickstart"
  "-DgroupId=fr.formation"
  "-DartifactId=my-app"
```

```
mvn archetype:generate
  "-DarchetypeArtifactId=maven-archetype-webapp"
  "-DgroupId=fr.formation"
  "-DartifactId=my-webapp"
```

COMMANDES UTILES

Compile le projet

```
mvn compile
```

Teste la compilation du projet

```
mvn test-compile
```

Exécute les tests unitaires du projet

```
mvn test
```

Purger les dépendances (repo local)

```
mvn dependency:purge-local-repository
```

Package le projet (JAR ou WAR)

```
mvn package
```

Déployer le projet

```
mvn deploy
```

Supprime le répertoire « target »

```
mvn clean
```

Générer la documentation Java

```
mvn javadoc:javadoc
```

EXERCICE

Télécharger Maven

- Dézipper l'archive dans C:\Maven (par exemple)
- Ajouter « C:\Maven\bin » dans votre variable d'environnement Path
- Vérifier que la variable système JAVA_HOME existe et qu'elle pointe sur la JDK
  - `mvn -version` donne un indice !

Démarrer une console dans un répertoire donné (Shift + Clique droit)

Initialiser un nouveau projet « my-app »

Compiler (avec Maven) et exécuter le projet !

- Utiliser la commande `java -classpath fichier.jar package.ClassePrincipale`
- *NOTE : La compilation crée un répertoire **target** : les sources compilées et le jar créé s'y trouvent*

JAR EXÉCUTABLE

Utilisation du plugin « maven-jar-plugin »

- On précise quelle est la classe principal

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <version>3.0.2</version>
  <configuration>
    <archive>
      <manifest>
        <mainClass>fr.formation.App</mainClass>
      </manifest>
    </archive>
  </configuration>
</plugin>
```

EXERCICE

Repackager le projet en JAR exécutable

- Penser à nettoyer avant de packager de nouveau

```
mvn clean package
```

- Exécuter le JAR avec la commande

```
java –jar fichier.jar
```

EXERCICE – ALLER PLUS LOIN

Packager le JAR dans un EXE avec le plugin launch4j-maven-plugin

- Configurer le plugin directement avec Maven (pom)
- L'application doit s'exécuter en ligne de commande et imprimer « Hello World »

ECLIPE

Il est possible de générer tout ce qu'il faut pour créer un projet Eclipse

```
mvn eclipse:eclipse
```

Il suffira ensuite d'importer un projet existant dans le Workspace Eclipse

ECLIPE

Il existe un plugin Maven embarqué dans Eclipse

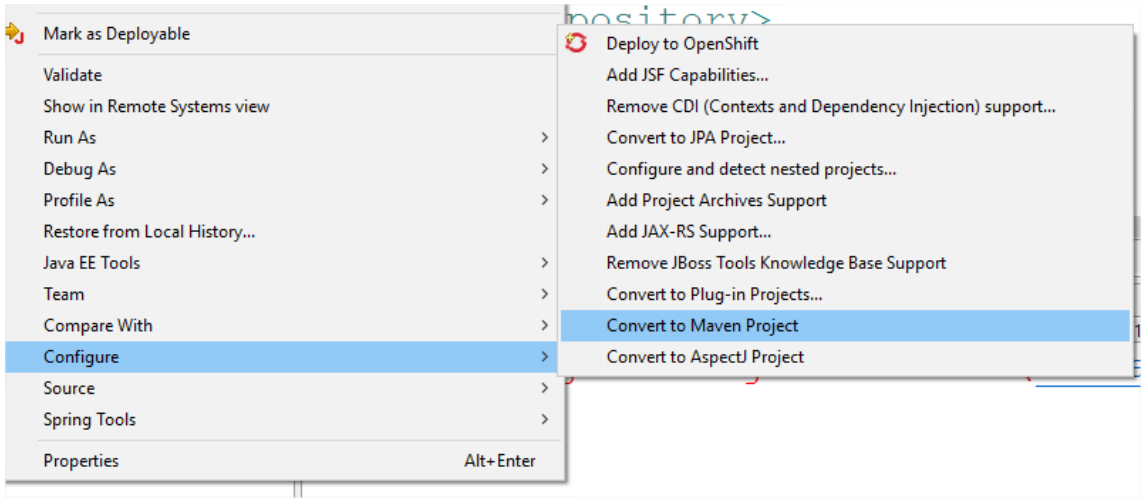
- m2eclipse

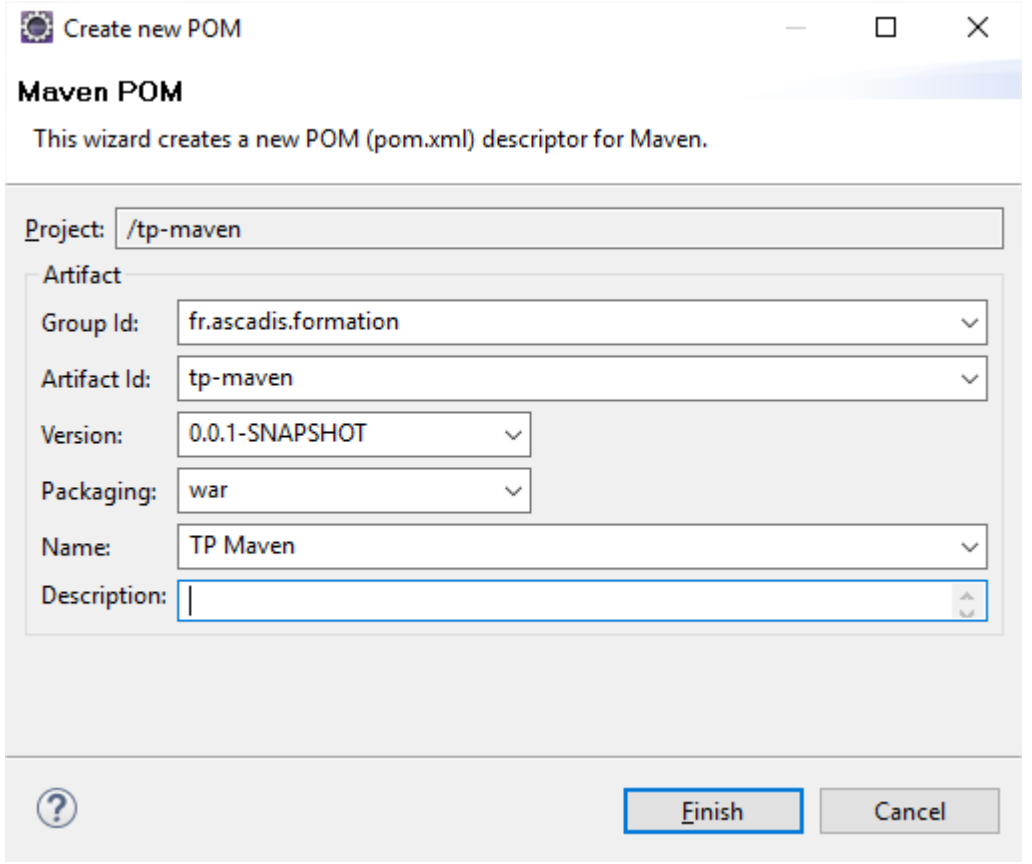
Permet de

- Créer de nouveaux projets Maven
- Convertir un projet JAR ou WAR existant en projet Maven

ECLIPE

Pour convertir un projet existant, suivre ces étapes





ECLIPE

Pour un projet Maven sous Eclipse, possible de faire un Maven-Update

- Clique droit > Maven > Update Project
- Force Maven à
  - Télécharger les dépendances et à mettre à jour les SNAPSHOTS
  - Nettoyer le projet

EXERCICE

Sous Eclipse

- « Separation of Concern » (SoC)

Discussion autour d’un projet

- Découpage du projet en dépendances réutilisables
- Création des projets Maven et interconnexion des dépendances avec Maven