APACHE MAVEN

INTRODUCTION

INTRODUCTION

MAVEN ?

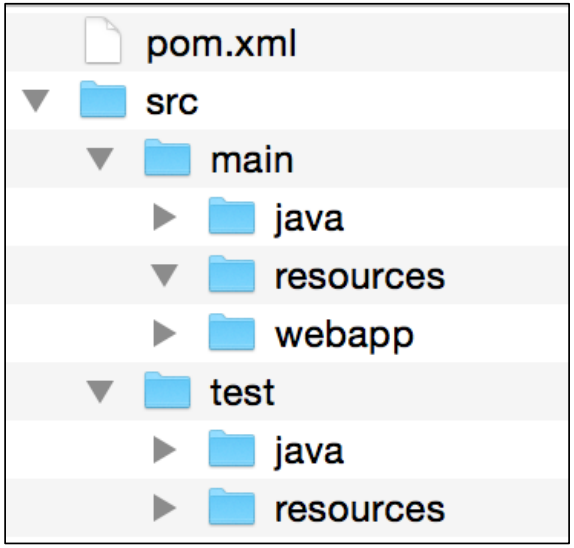
MODÈLE OBJET DE PROJET

- Définit par un fichier de configuration XML appelé **pom.xml**
  - POM = **P**roject **O**bjet **M**odel
- Configure des informations concernant l'identité du projet (groupId, artifactId, version, packaging)
- Définit de façon déclarative les différentes tâches à exécuter
- etc…

```
<?xml version="1.0" encoding="UTF-8"?>
<project
xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLS
chema-instance"
    xsi:schemaLocation="http://maven.apach
e.org/POM/4.0.0
http://maven.apache.org/xsd/maven-
4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>dta</groupId>
    <artifactId>mywebapp</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <name>mywebapp</name>
    <properties>
```

UN ENSEMBLE DE CONVENTIONS

- Maven privilégie une approche "convention over configuration"
  - Les sources java sont supposés être dans le répertoire **src/main/java**
  - Les autres fichiers nécessaire au programme (.properties, .xml) sont supposés être dans le répertoire **src/main/resources**
  - Les sources java des tests sont sont supposés être dans le répertoire **src/test/java**
  - Les autres fichiers nécessaire aux tests (.properties, .xml) sont supposés être dans le répertoire **src/test/resources**
  - Les fichiers générés par Maven sont dans le répertoire **target**
  - Les fichiers web (html, css, jsp, …) sont supposés être dans le répertoire **src/main/webapp**



GESTION DE DÉPENDANCES

- Maven permet de gérer les dépendances d'un projet en configurant le fichier pom.xml
- Les dépendances sont identifiées grâce aux informations : groupId, artifactId, version
- Les dépendances peuvent être de plusieurs types : jar, zip, pom, …

```
<dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-web-api</artifactId>
            <version>7.0</version>
            <scope>provided</scope>
        </dependency>
          <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <scope>test</scope>
        </dependency>
</dependencies>
```

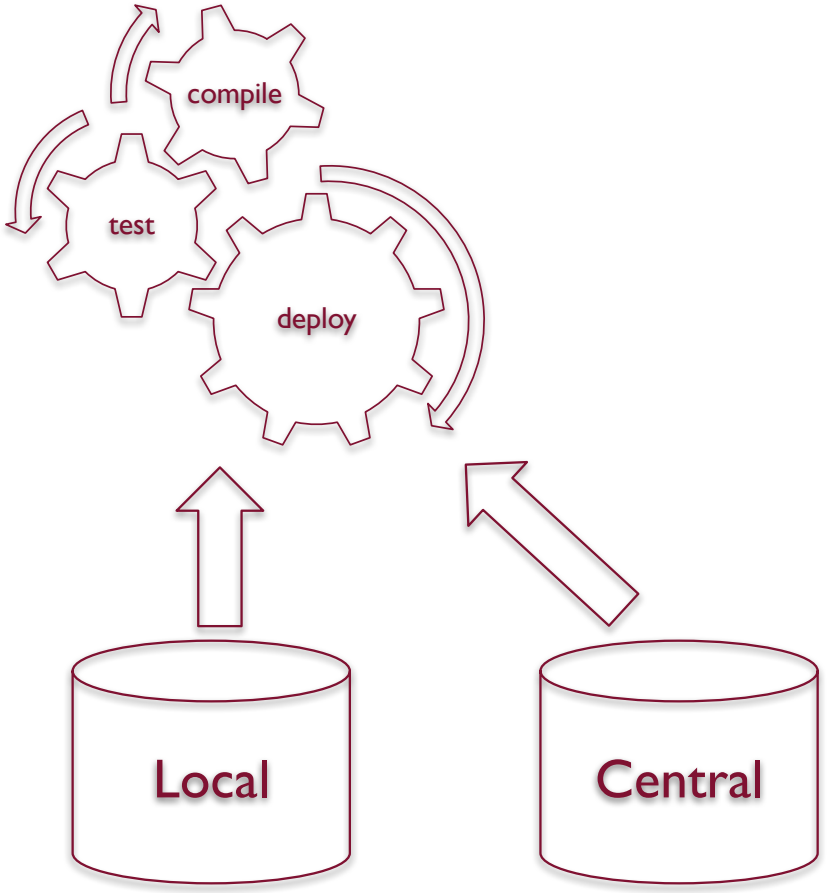
UN CYCLE DE VIE DE PROJET

- Maven propose une construction de projet en plusieurs phases :
  - …
  - generate-sources
  - process-resources
  - compile
  - process test resources
  - test
  - package
  - install
  - deploy
  - …

DES RÉFÉRENTIELS

- Les artefacts Maven peuvent être stockés dans des référentiels locaux ou distants.

- Le référentiel central Maven (http://search.maven.org/) héberge une grande base d'artefacts réutilisables dans des projets.



DES PLUGINS

- Les plugins permettent d'étendre la construction de projet Maven pour ajouter et configurer des tâches spécifiques

```
<plugins>
      <plugin>
          <artifactId>maven-war-plugin</artifactId>
          <version>2.6</version>
          <configuration>
              <failOnMissingWebXml>false</failOnMissingWebXml>
          </configuration>
       </plugin>
       <plugin>
          <groupId>org.wildfly.plugins</groupId>
          <artifactId>wildfly-maven-plugin</artifactId>
          <version>3.0</version>
       </plugin>
 </plugins>
```

QUE FAIRE AVEC MAVEN ?

- Créer un projet à l'aide de modèles
- Compiler un projet avec ses dépendances
- Exécuter des tests unitaires, d'intégrations, …
- Gérer les versions d'un projet
- Assembler un projet en livrable (JAR, ZIP, WAR, EAR, …)
- Archiver l'artefact d'un projet dans un référentiel
- Générer un rapport de qualité du code
- Déployer un artefact sur un serveur
- …

INSTALLATION

INSTALLER MAVEN

- Prérequis :
  - Avoir un JDK 1.7+ installé
- Télécharger Maven
  - https://maven.apache.org/download.cgi
- Décompresser l'archive
- Mettre à jour la variable PATH
- Créer la variable JAVA_HOME
- Tester l'installation de Maven

```
mvn -v
```

POM

POM

Relations du POM

Coordonnées d'un projet
Sous-modules
Héritage
Dépendances

Configuration de construction

Répertoires de sources
Plugins
Reporting

Informations générales

Généralités
Contributeurs
Licences

Configuration d'environnement

Informations sur l'environnement
Les profils

COORDONNÉES D'UN PROJET

- Un projet est identifié à l'aide des informations suivantes :
  - groupId
  - artefactId
  - version
  - packaging (par défaut "jar")

SUPER POM

- Tous les projets Maven héritent d'un super POM

- Pour visualiser le super POM
  - Désarchiver le fichier **M2_HOME/lib/maven-model-builder-3.3.3.jar**
  - Le super POM se trouve dans le fichier **org/apache/maven/model/pom-4.0.0.xml**

- Il contient la configuration Maven par défaut => la convention Maven

POM EFFECTIF

- Les POMs héritent leur configuration d'autres POMs (à minima du super POM).
- Le modèle objet du projet va au final être la combinaison des configurations (via le lien d'héritage)
- Pour visualiser toute la configuration d'un POM

  mvn help:effective-pom

VERSIONS D'UN PROJET

- Maven propose une utilisation des versions d'un projet avec le format suivant :

<version majeure>.<version mineure>.<version incrémentale>-<qualifieur>

- Quelques exemples :
  - 3.4.5 → version majeure 3, mineure 4 et incrémentale 5
  - 12 → version majeure 12
  - 2.5-beta-01 → version majeure 2, mineure 5 et qualifieur "beta-01"

PROPRIÉTÉS (1)

- Un POM peut contenir des références à des propriétés via
  - **${propriété}**

- Maven initialise 3 variables :
  - **env**
    - permet d'accéder aux variables d'environnement du système d'exploitation. Exemple ${env.PATH}
  - **project**
    - permet d'accéder au POM. Exemple ${project.version}
  - **settings**
    - permet d'accéder aux informations de configuration Maven (fichier settings.xml). Exemple : ${settings.offline}

PROPRIÉTÉS

- Il est possible d'ajouter des propriétés dans un fichier pom.xml via la balise <properties>

- Dans l'exemple, ci-contre, les propriétés sont accessibles via :
  - ${junit.version}
  - ${source.version}

```
<properties>
  <junit.version>4.2</junit.version>
  <source.version>1.8</source.version>
</properties>
```

DÉPENDANCES D'UN PROJET (1)

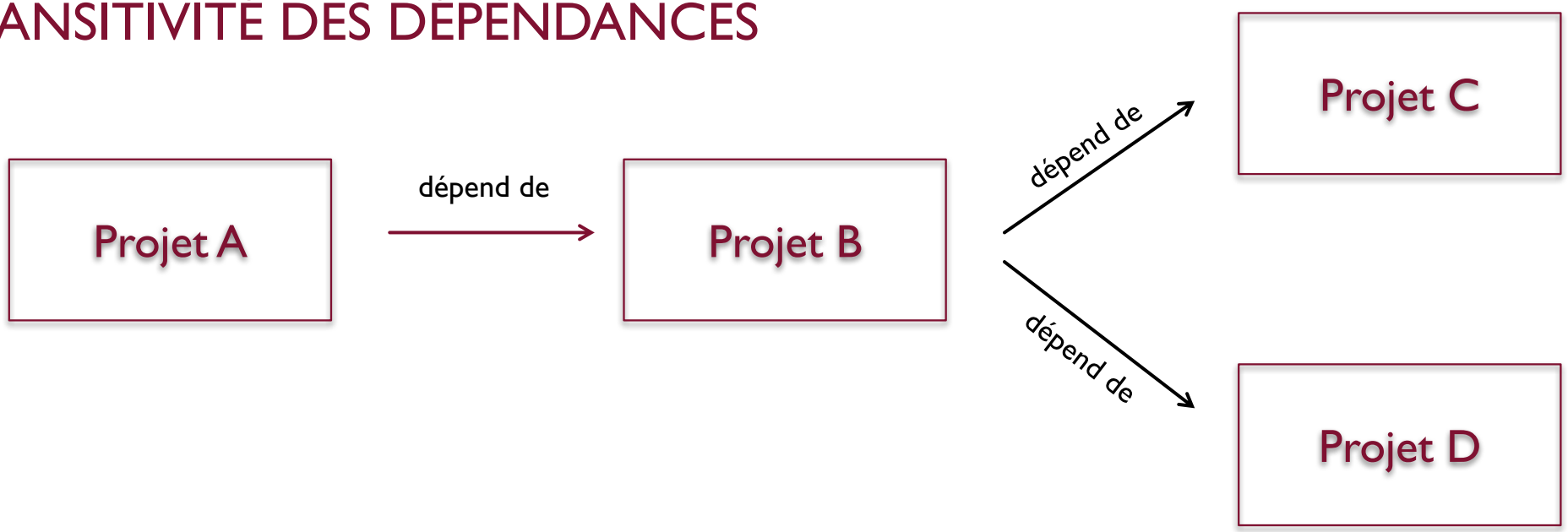
- Les dépendances d'un projet se déclarent dans la section **<dependencies>**
- La balise **<scope>** définit la portée de la dépendance

```
<dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-web-api</artifactId>
            <version>7.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <scope>test</scope>
        </dependency>
</dependencies>
```

PORTÉE D'UNE DÉPENDANCE

- **compile (par défaut)**
  - indispensable à la compilation et à l'exécution
- **runtime**
  - inutile à la compilation mais indispensable à l'exécution
- **test**
  - utile uniquement à la compilation et l'exécution des tests
- **provided**
  - indispensable à la compilation, dépendance fournie par l'environnement d'exécution (par exemple, un serveur d'application)
- ~~**system**~~
  - dépendance à récupérer en local, hors dépôt Maven => à ne pas utiliser dans la mesure du possible
- **import** (Maven version >=2.0.9)
  - Applicable uniquement à la section <dependencyManagement> à une dépendance de type pom. Il permet de mutualiser une liste de dépendances sans recourir à l'héritage.

TRANSITIVITÉ DES DÉPENDANCES



=> Le projet A peut utiliser les classes des projets B, C et D

EXCLURE UNE DÉPENDANCE

- Il est possible d'exclure une dépendance transitive via la balise **<exclusions>**

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>4.1.2</version>
    <exclusions>
        <exclusion>
            <groupId>dom4j</groupId>
            <artifactId>dom4j</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

INTERVALLES DE VERSIONS

- Il est possible de définir un ensemble de versions acceptables pour une dépendance donnée.

  (,) → définir un intervalle ouvert. Exemples :

  - (3.8,4.2)
  - (,4.2)
  - (3.8,)

  [,] → définir un intervalle fermé. Exemples :

  - [3.8, 4.2]
  - [,4.2]
  - [3.8,]

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>[4.0,4.2]</version>
  <scope>test</scope>
</dependency>
```

DÉPENDANCE OPTIONNELLE

- Les dépendances marquées comme optionnelles ne sont pas transitives.
  - Si un projet A dépend d'un projet B qui dépend d'un projet C. Si C est marqué comme optionnel dans B, alors **A ne dépendra pas automatiquement de C**.

```
<dependency>
  <groupId>net.sf.ehcache</groupId>
  <artifactId>ehcache</artifactId>
  <version>1.4.1</version>
  <optional>true</optional>
</dependency>

<dependency>
  <groupId>swarmcache</groupId>
  <artifactId>swarmcache</artifactId>
  <version>1.0RC2</version>
  <optional>true</optional>
</dependency>
```

TP 01

- Prise en main
- Travailler avec les dépendances