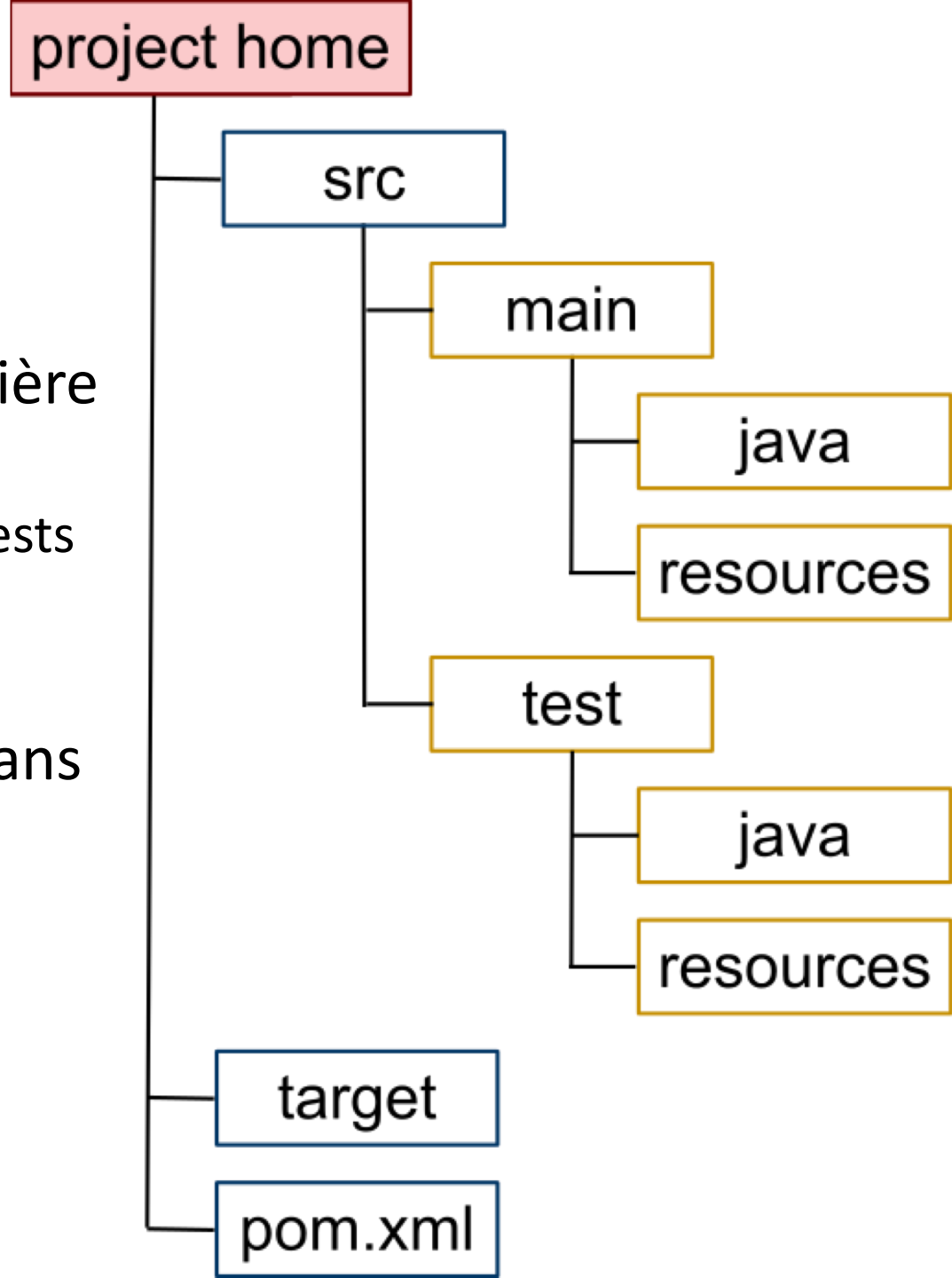
Maven

Introduction

- Maven est un système de *build* pour projets Java
  - Automatise toutes les étapes de développement
- Il intègre dans un projet
  - Les dépendances du logiciel
  - Les étapes à effectuer pour le construire (et leur ordre)
- Tout est spécifié dans un ou plusieurs fichiers XML
- Philosophie générale :
  - Pré-configuré pour le cas de base
  - Modifiable pour les autres

Structuration d'un projet

- Maven impose une structuration particulière
  - Séparation des sources et du code compilé
  - Séparation du source entre code (*main*) et tests
  - Séparation du code source et des fichiers ressources (images…)
- La configuration du projet est spécifiée dans *pom.xml* (Project Object Model)



https://en.wikipedia.org/wiki/Apache_Maven

POM

- Décrit la configuration d'un projet au format XML
- À minima les coordonnées Maven (*maven coordinates*)
  - **groupId** : identifiant unique à l'échelle d'une organisation. Peut être le nom de package de base des sources (*fr.unice.miage…)*
  - **artifactId** : nom du projet
  - **version** : version du projet

```
1.  <project xmlns="http://maven.apache.org/POM/4.0.0"
2.    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3.    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4.                        http://maven.apache.org/xsd/maven-4.0.0.xsd">
5.    <modelVersion>4.0.0</modelVersion>
6.
7.    <groupId>org.codehaus.mojo</groupId>
8.    <artifactId>my-project</artifactId>
9.    <version>1.0</version>
10. </project>
```

http://maven.apache.org/pom.html#Maven_Coordinates

POM

- Spécification des dépendances du projet
- Permet de modifier le comportement de Maven par projet
- Notion de hiérarchie entre pom pour projet complexe
- Peut contenir des *properties* référencées ensuite par *$name*

```
<junit.jupiter.version>5.3.1</junit.jupiter.version>
```

```
<version>${junit.jupiter.version}</version>
```

Plugins

- Maven de base ne sait (presque) rien faire
- Comportement modifiable par plugins
- Maven télécharge les plugins quand il en a besoin
  - Nécessite une connexion internet les première fois (https://mvnrepository.com/)
  - Tout est dans ~/.m2 *(local maven repository)*
- Invocation
  - *mvn pluginName:goal*
  - Ex : *mvn compiler:compile, mvn surefire:test*

Lifecycle

- Invoquer spécifiquement un plugin est compliqué
- Abstraction dans lifecycle
  - Ensemble ordonné de phases pour construire un logiciel
- À chaque phase est associé des *goals* de plugins
  - Modifiable suivant le projet
- Existe un lifecycle par défaut
- Possible d'utiliser la phase directement
  - mvn compile <=> mvn compiler:compile

```
1  validate
2  generate-sources
3  process-sources
4  generate-resources
5  process-resources
6  compile
7  process-test-sources
8  process-test-resources
9  test-compile
10 test
11 package
12 install
13 deploy
```

https://en.wikipedia.org/wiki/Apache_Maven#Project_Object_Model

Lifecycle

- Exécution d’une phase implique exécution des précédentes
- Phases
  - Compile : compilation du source
  - Test : exécution des tests
  - Package : création de l’archive (jar, war…)
  - Install : copie du jar généré dans le répo local (~/.m2)
  - Deploy : installation du jar dans un répo **distant**

Dépendances

- Maven gère les dépendances du projet
  - Indiqué dans le pom avec les coordonnées de la dépendance
  - Gestion transitive
- Le scope permet de préciser la phase où elle est nécessaire
  - compile : défaut, accessible depuis tout le projet
  - provided : dépendance fournit à l'exécution par un tiers
  - runtime : par nécessaire pour la compilation, juste exécution
  - test : seulement pour la phase de test
  - system : jar fournit dans le projet, ne pas télécharger

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>
```

Commencer avec Maven

- Plus simple de partir d'un projet vide
- Utilisation du plugin *archetype*
  - Fournit un modèle de base d'un projet
- Invocation de *mvn archetype:generate*
  - Liste tous les archetypes officiels (plus de 2000 ☹)
  - Possible de filtrer par nom (*maven-archetype-quickstart, maven-archetype-simple,…)*
- Création de la structure du projet et du pom minimal.

https://maven.apache.org/guides/introduction/introduction-to-archetypes.html

Maven

Plugins

Configuration

- Tout se fait dans le pom.xml
  - Ajouter plugins
  - Modifier comportement
- Zone *<build> </build>*
- Pour chaque plugin
  - Indiquer ses coordonnées
  - Spécifier la configuration dans *<configuration>*

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <excludes>
                    <exclude>**/edu/**</exclude>
                </excludes>
            </configuration>
        </plugin>
```

Compiler

- *maven-compiler-plugin*
- Gère la compilation du projet
  - Par défaut source Java 1.6 et target 1.6
- 2 goals
  - compile : compile le source
  - testCompile : compile les tests

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.8.0</version>
  <configuration>
    <source>1.8</source>
    <target>1.8</target>
  </configuration>
```

Assembly

- Permet de construire des archives (jar, war…)
  - Implicite, type spécifié par *<packaging>jar</packaging>*
- Invocation
  - *mvn assembly:assembly* (appelé manuellement)
  - *mvn assembly:single* (appelé automatique dans phase *package*)
- *Descriptors* prédéfinis
  - bin
  - jar-with-dependencies
  - projet (tout sauf target)
  - src
- Doit être configuré dans pom.xml
  - Dans <build> <plugins> <plugin> …

Assembly : jar exécutable avec dépendances

```
<plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <version>2.2-beta-2</version>
    <executions>
        <execution>
            <id>create-executable-jar</id>
            <phase>package</phase>
            <goals>
                <goal>single</goal>
            </goals>
            <configuration>
                <descriptorRefs>
                    <descriptorRef>
                        jar-with-dependencies
                    </descriptorRef>
                </descriptorRefs>
                <archive>
                    <manifest>
                        <mainClass>fr.unice.miage.MaClasseMain</mainClass>
                    </manifest>
                </archive>
            </configuration>
        </execution>
    </executions>
</plugin>
```