# Multi-modules

## Introduction

On a déjà vu depuis un certain temps qu’une application monolithique classique était composée de 3 couches :

* Persistence (accès aux données)
* Business (services)
* Application (interface)

On a donc 3 projets Maven, avec Application qui dépend de Business qui dépend de Persistence.

On voit aussi que si on veut générer tous les war/jar correspondants, il faut lancer Maven sur chacun des projets (et dans le bon ordre en plus).

Le mieux est de créer un projet multimodule.

## Dans Eclipse

Dans Eclipse, Maven est directement intégré et les projets multimodules ont la forme d’une arborescence :

* Au niveau le plus haut, le projet **parent**
* Les 3 projets sont des sous-répertoires du parent. Ces sous-projets s’appellent des *modules*. **ATTENTION** : dans Eclipse, ces 3 projets apparaissent au même niveau que le parent, mais c’est juste une astuce visuelle de Eclipse.

A ce moment-là, pour générer tous les jar/war ; il suffit de lancer Maven sur le parent. Ou pour tout mettre à jour, de mettre à jour au niveau du parent.

## C’est l’horreur

En pratique, créer un projet multimodules Maven depuis zéro est une vraie abomination. Le mieux est de partir d’un projet squelette qu’on réutilisera et modifiera pour différents besoins.

Un projet squelette est fourni et on va l’analyser pour voir comment tout fonctionne. Le projet a comme fonctionnalités :

* Persistence : on a une Entity Specie et le Repository associé. Ainsi que des tests unitaires
* Business : un service qui utilise le Repository avec juste une fonction list()
* Application : une simple application ligne de commande qui affiche le résultat de list()

**ATTENTION** : il s’agit d’un *squelette*. Il faudra le modifier pour votre développement et en particulier enlever les classes d’exemples. Et surtout modifier les pom (artifactId, groupId, etc) ainsi que les packages

## Analyse

Dans un premier temps on va importer le projet dans Eclipse. Ce projet s’appelle *multi-module.zip*.

* Faire *Import > Projects from folder or archive*
* Cliquer sur *Archive* et sélectionner le zip
* Désélectionner xxxx-expanded (le premier normalement)
* Faire *Ok*

Les 4 projets Maven apparaissent (le parent et les 3 couches).

Voyons le pom.xml du parent :

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

**Ici on dit que ce projet est un projet spring boot (et a donc comme parent spring-boot-starter)**

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.5.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

**Ici, on donne les identifiants classiques pour un projet Maven (groupId, etc). On remarque qu’il est de packaging *pom* ce qui signifie que c’est un projet parent.**

<groupId>fr.afpa.formation</groupId>

<artifactId>squelette-multi-module</artifactId>

<version>0.0.1</version>

<packaging>pom</packaging>

**Ces properties définies au niveau du parent seront partagées par les 3 modules. A noter que d’autres choses peuvent être partagées par les modules : des dépendances, des plugins, etc**

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<java.version>1.8</java.version>

</properties>

**Ici on définit quels sont les modules (les fils) du parent.**

<modules>

<module>squelette-persistence</module>

<module>squelette-business</module>

<module>squelette-application</module>

</modules>

</project>

Voyons le pom.xml de Application :

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

**Ici on signale quel est le parent. Le *..* signale que le parent est au niveau du dessus.**

<parent>

<groupId>fr.afpa.formation</groupId>

<artifactId>squelette-multi-module</artifactId>

<version>0.0.1</version>

<relativePath>..</relativePath>

</parent>

**On remarque qu’on ne signale que le artifactId. Dans un projet multimodules, le groupId et la version sont censés être les mêmes pour tout le monde.**

<packaging>jar</packaging>

<artifactId>squelette-application</artifactId>

**La seule dépendances est le module business mais il peut y avoir d’autres choses**

<dependencies>

<dependency>

<groupId>fr.afpa.formation</groupId>

<artifactId>squelette-business</artifactId>

<version>0.0.1</version>

</dependency>

</dependencies>

</project>

Et celui de Persistence :

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>fr.afpa.formation</groupId>

<artifactId>squelette-multi-module</artifactId>

<version>0.0.1</version>

<relativePath>..</relativePath>

</parent>

<artifactId>squelette-persistence</artifactId>

<packaging>jar</packaging>

**Ici, toutes les dépendances utiles (JPA, MySql, H2, Junit)**

<dependencies>

<!-- JPA Data (We are going to use Repositories, Entities, Hibernate, etc...) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- Use MySQL Connector-J -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

</project>

## Modifier

Pour modifier un squelette il y a 3 étapes préliminaires à faire

* Modifier les noms des 4 projets
* Modifier les pom.xml
  + On commence par le parent : changer le groupId, l’artifactId, la version et aussi les noms des modules.
  + Pour chaque module : changer le parent, puis l’artifactId
* Modifier les package en commençant par le parent. *Ne pas oublier* : Spring Boot scanne les *@Component* récursivement. Donc : faire en sorte que le package de plus haut niveau se trouve dans Application et que les autres couches ont des package au même niveau ou en dessous.   
  Aussi par convention d’écriture : le package de plus haut niveau est la même chose que le groupId+artifactId (si groupId est *a.b.c* et l’artifactId du parent *d*, le package de plus haut niveau est : *a.b.c.d*)

Ensuite on fait *Maven > Update project* pour tout mettre d’équerre.