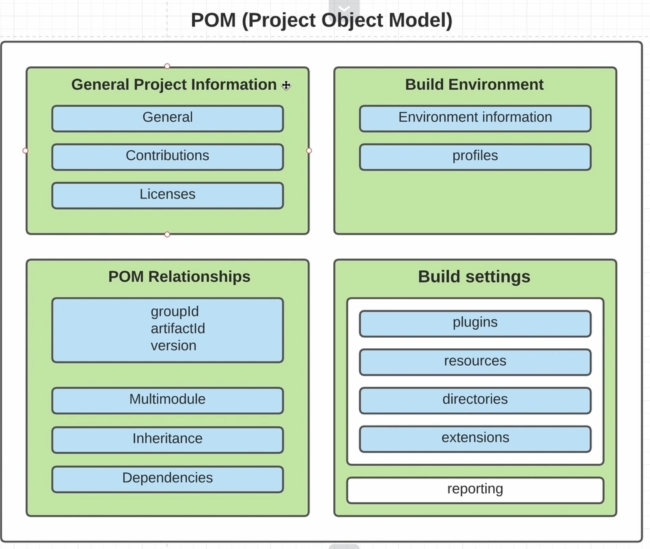
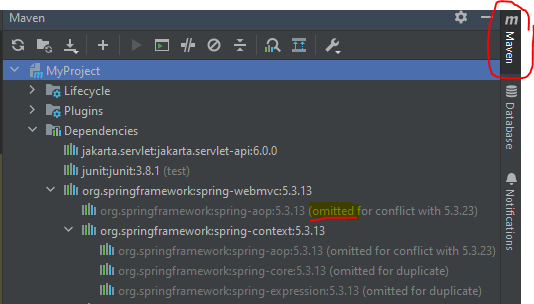
Структура pom.xml –конфигурационного файла



**Snapshot -** это специальная версия артефакта, которая указывает текущую разрабатываемую копию проекта. Обозначаются промежуточные версии, которые находятся в разработке. (**«1.2.3-SNAPSHOT»** здесь «1» – мажорная версия, «2» – минорная версия, «3» – инкремент).

**Release –** финальная версия.

При запуске maven из среды разработки используется тот maven, который прописан в settings/build, execution, deployment/maven:

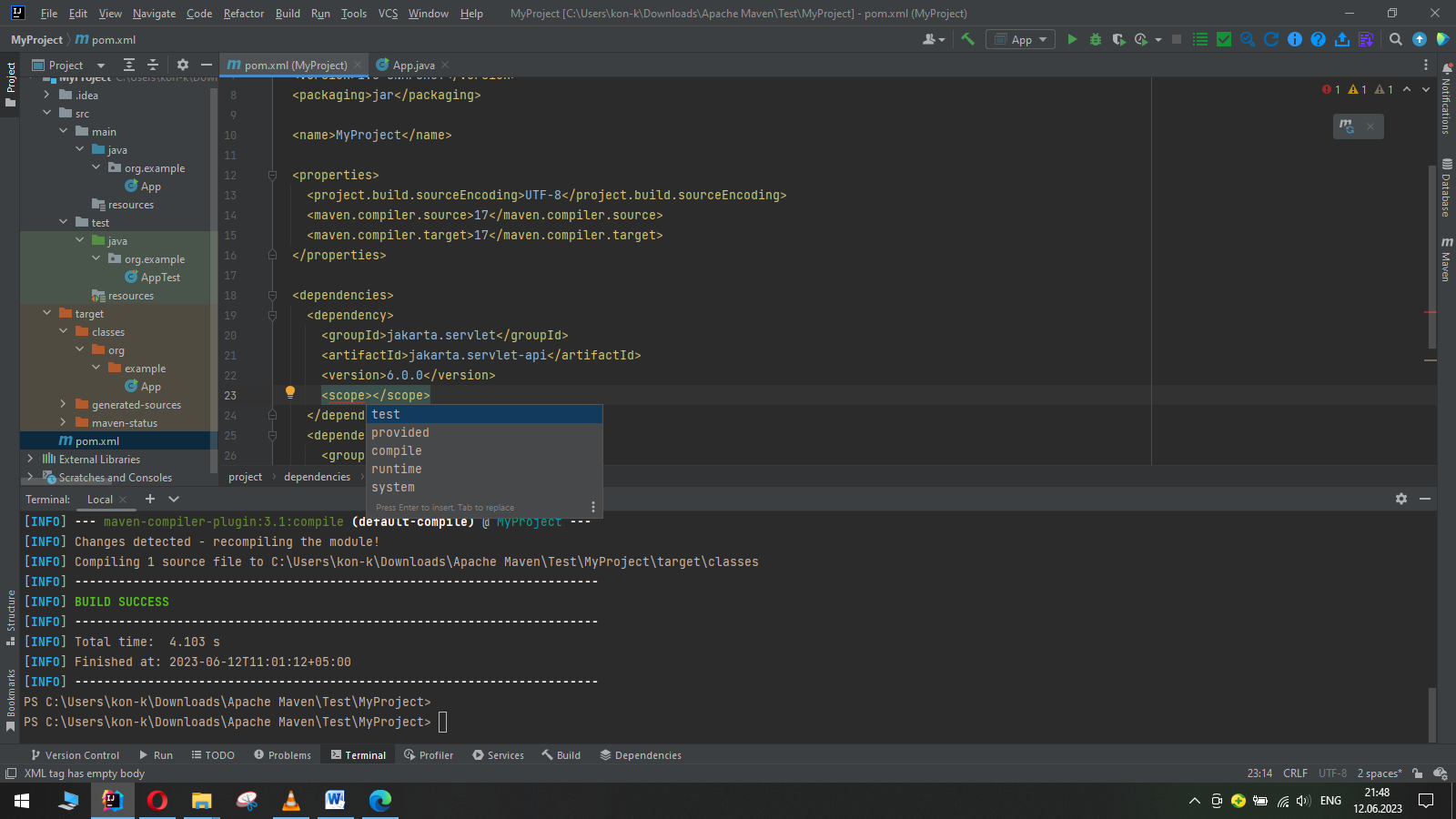


При запуске через консоль используется maven, который прописан в PATH переменных среды.

В обоих случах settings.xml будет браться из \.m2\settings.xml.

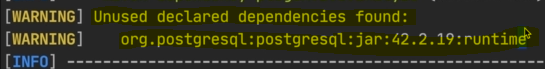
**scope:**

* **compile** (по умолчанию) – зависимость для компиляции проекта, jar-файл зависит от этой зависимости и не будет работать без нее;
* **provided** - артефакт используется на этапе компиляции и тестирования, а в сборку (jar-файл) *не включается*. Предполагается, что среда исполнения (JDK или WEB-контейнер) предоставят данный артефакт во время выполнения программы. Нужна для возможности импортов и обращений в коде к методам из классов зависимости.
* **runtime** - не нужна для компиляции проекта и используется только на стадии выполнения приложения (например jdbc драйвер для подключения к БД).
* **system** - аналогична provided за исключением того, что содержащий зависимость артефакт указывается явно в виде абсолютного пути к файлу, определенному в теге systemPath. Не рекомендуется использовать.
* **test** – зависимость для выполнения тестов, в сборку (jar-файл) не включается.

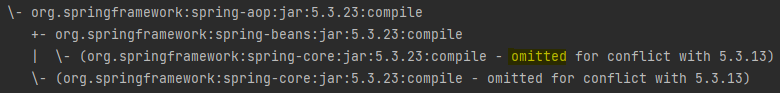


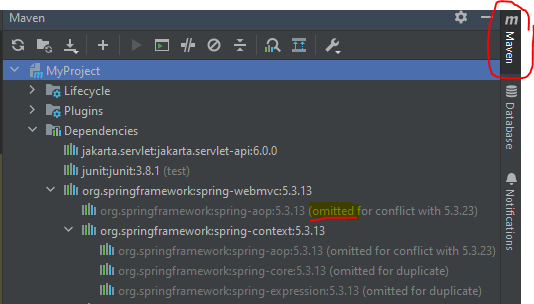
Команда **mvn help:effective-pom** – выведет финальный pom.xml как совокупность super-pom и текущего pom.

Команда **mvn dependency:analyze** – показывает неиспользуемые («Unused»), устаревшие зависимости



**mvn dependency:tree (mvn dependency:tree -Dverbose)** – строит иерархию зависимостей проекта, включая транзитивные зависимости. В maven если есть несколько зависимостей, в которых транзитивно подтягиваются другие зависимости, то приоритет будет у тех зависимостей, которые в pom.xml расположены выше. Опущенные зависимости (после вызова «mvn dependency:tree -Dverbose» либо в модуле maven) будут помечены «omitted»





**Exclusions** – исключение транзитивных зависимостей, если возможен конфликт версий.

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.13</version>  
 **<exclusions>  
 <exclusion>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-beans</artifactId>  
 </exclusion>  
 </exclusions>**  
</dependency>

<dependency>  
 <groupId>org.springframework</groupId> // spring-beans вместо 5.3.13 подтянется   
 <artifactId>spring-aop</artifactId> // версия свежее 5.3.23   
 <version>5.3.23</version>  
</dependency>

**${project.basedir}** – путь к корневой директории проекта

**<executions>…</executions> -** настройка вызова определенных goal в manen-плагинах (16 урок).

**Maven-surefire-plugin** – вызов unit тестов (вызывается на стадии **test** )

**Maven-failsafe-plugin** – вызов интеграционных тестов (вызывается на стадии **verify** )

Стадия **install** собирает артифакт и размещае его в локальном репозитории (папка .m2/repository )

Стадия **site** формирует отчеты, документацию (jacoco)

<dependency>  
 <groupId>net.sf.json-lib</groupId>  
 <artifactId>json-lib</artifactId>  
 <version>2.4</version>  
 <classifier>jdk15</classifier>  
</dependency>

**параметры GAV (groupId, artifactId, version):**

* **groupId** - идентификатор производителя объекта. Часто используется схема принятая в обозначении пакетов Java. Например, если производитель имеет домен ***domain.com***, то в качестве значения groupId удобно использовать значение ***com.domain***. То есть, groupId это по сути имя пакета.
* **artifactId** - идентификатор объекта. Обычно это имя создаваемого модуля или приложения.
* **version** - версия описываемого объекта. Для незавершенных проектов принято добавлять суффикс SNAPSHOT. Например 1.0.0-SNAPSHOT.
* **сlassifier -** «необязательный» классификатор используется в тех случаях, когда деление артефакта по версиям является недостаточным. К примеру, определенная библиотека (артефакт) может быть использована только с определенной JDK (VM), либо разработана под windows или linux.

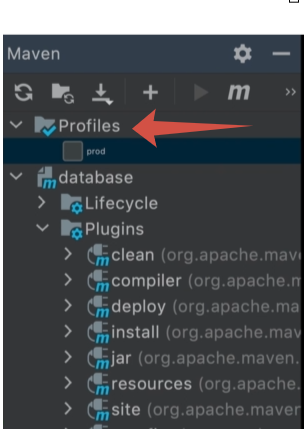
Наследование проектов/модулей – у родительского pom будет тип <packaging>**pom**</packaging> и блок modules, где перечислены унаследованные модули. У дочерних модулей в pom будет модуль <parent>.

В родительском pom указываются зависимости и их версии, которые используются во всех дочерних модулях. Делается это не в блоке <**dependencies**>, а в блоке <**dependencyManagement**>. Тогда унаследованные модули **не будут тянуть** **все** зависимости из родительского pom (их не будет в артифакте), а требуемые зависимости для конкретного модуля нужно указывать в pom в блоке <**dependencies**> используя <groupId> и <artifactId> без указания версии и скоупа, т.к. версия и скоуп <**version**> и <**scope**> подтянутся из родителькой pom (только указанные зависимости попадут в артифакт). **Если в модуле требуется версия, отличная от указанной в родительмком pom**, то указать необходимую версию <**version**> в pom дочернего модуля. С плагинами все также, только через блок <**pluginManagement**> родительского pom.

Подстановка в статические файлы директории **resources** (напр. application.properties) значений из pom через ${…} – нужно использовать **filtering** (см. урок 23).

Секция <**profiles**> - можно переопределить для разных профилей (prod, dev) различные параметры (properties, dependencies, buld, plugins и т.д.) (см.урок 24). Выбор нужного профиля в секции profiles maven плагина:

либо через командную строку через параметр «**-P**» (-Pdev).



В секции <profiles>**<activation>** можно настраивать условия активации нужных профилей.

**Толстая и тонкая jar** – плагины **maven-jar-plugin** и **maven-assembly-plugin** (уроки 25, 26).