（项目存在空间以及依赖关系）mavent-pom-dependency

Maven的一个哲学是惯例优于配置(Convention Over Configuration), Maven默认的依赖配置项中，scope的默认值是compile，项目中经常傻傻的分不清，直接默认了。今天梳理一下maven的scope。

scope的分类（及使用例子）

compile（默认）

ps-》编译器必须存在该jar

默认就是compile，什么都不配置也就是意味着compile。compile表示被依赖项目需要参与当前项目的编译，当然后续的测试，运行周期也参与其中，是一个比较强的依赖。打包的时候通常需要包含进去。

runntime

ps-》编译器不检查，但是运行的时候需要。

runntime表示被依赖项目无需参与项目的编译，不过后期的测试和运行周期需要其参与。与compile相比，**跳过编译**而已，说实话在终端的项目（非开源，企业内部系统）中，和compile区别不是很大。比较常见的如JSR×××的实现，对应的API jar是compile的，具体实现是runtime的，compile只需要知道接口就足够了。Oracle jdbc驱动架包就是一个很好的例子，一般scope为runntime。另外runntime的依赖通常和optional搭配使用，optional为true。我可以用A实现，也可以用B实现。

test

ps-》测试才需要，正式发布不需要

scope为test表示依赖项目仅仅参与测试相关的工作，包括测试代码的编译，执行。比较典型的如junit。

provided

ps-》打包的时候不用打包-web容器已经提供。

provided意味着打包的时候可以不用包进去，别的设施(Web Container)会提供。事实上该依赖理论上可以参与编译，测试，运行等周期。相当于compile，但是在打包阶段做了exclude的动作。

system

ps-》该jar不被maven管理，从系统指定路径获取。

从参与度来说，也provided相同，不过被依赖项不会从maven仓库抓，而是从本地文件系统拿，一定需**要配合systemPath属性使用**。

scope的依赖传递

java项目一般需要依赖其他项目（使用其功能），crm项目-使用spring-jdbc，-spring-jdbc依赖，spring-bean，mysql驱动，oralce驱动。。。

例如：项目A–>项目B–>项目C ....

当前【开发项目为A】，A依赖于B，B依赖于C。

那么开发【项目A】是否需要下载C的项目jar呢。

a）B里配置的C的scope**【test或者provided时】**

A不需要C直接丢弃

b）B里配置的C的scope**【其他**compile，runtime等】

A需要C直接下载

ps-》A方案和B方案的scope怎么配置比较好，如果依赖的功能是核心功能，属于项目的基本功能（选择B方案），如果是非核心功能属于扩展功能（选择A方案）

依赖的传递

A->B(compile)　第一关系: a依赖b　compile
B->C(compile)　第二关系: b依赖c　compile

当在A中配置

```
<dependency>
        <groupId>com.B</groupId>
        <artifactId>B</artifactId>
        <version>1.0</version>
</dependency>
```

则会自动导入c包。关系传递如下表：

第一　第二	compile	test	provided	runtime
compile	compile	-	-	runtime
test	test	-	-	test
provided	provided	-	provided	provided
runtime	runtime	-	-	runtime

3.依赖冲突的调节（jar版本冲突）

A->B->C->X(1.0)
A->D->X(2.0)
由于只能引入一个版本的包，此时Maven按照最短路径选择导入x(2.0)

A->B->X(1.0)
A->D->X(2.0)
路径长度一致，则优先选择第一个，此时导入x(1.0)

4.排除依赖（手动把不满足要求的jar去除）

A->B->C(1.0)
此时在A项目中，不想使用C(1.0)，而使用C(2.0)
则需要使用exclusion排除B对C(1.0)的依赖。并在A中引入C(2.0).

pom.xml中配置

```
<!--排除B对C的依赖-->
<dependency>
        <groupId>B</groupId>
        <artifactId>B</artifactId>
        <version>0.1</version>
        <exclusions>
            <exclusion>
              <groupId>C</groupId>
              <artifactId>C</artifactId><!--无需指定要排除项目的版本号-->
            </exclusion>
        </exclusions>
</dependency>
<!---在A中引入C(2.0)-->
<dependency>
        <groupId>C</groupId>
        <artifactId>C</artifactId>
        <version>2.0</version>
</dependency>
```

5.依赖关系的查看

cmd进入工程根目录，执行 mvn dependency:tree
会列出依赖关系树及各依赖关系
mvn dependency:analyze　分析依赖关系