


## [maven](#)传递依赖规则

用**maven**很长时间了(2年)，下面把一些需要注意的细节加以总结：


情景一：

learn-1 pom.xml

Xml代码 


```
1.  <dependency>
2.      <groupId>org.springframework</groupId>
3.      <artifactId>spring-core</artifactId>
4.      <version>4.1.4.RELEASE</version>
5.      <!-- 依赖commons-logging 1.2的版本-->
6.  </dependency>
7.  <dependency>
8.      <groupId>com.learn</groupId>
9.      <artifactId>learn2</artifactId>
10.     <version>0.0.1-SNAPSHOT</version>
11.     <!-- 依赖commons-logging 1.1.3的版本-->
12.     <optional>true</optional>
13. </dependency>
```

learn-2 pom.xml

Xml代码 

```
1.  <dependency>
2.      <groupId>commons-logging</groupId>
3.      <artifactId>commons-logging</artifactId>
4.      <version>1.1.3</version>
5.  </dependency>
6.  <dependency>
7.      <groupId>junit</groupId>
8.      <artifactId>junit</artifactId>
9.      <version>3.8.1</version>
10.     <scope>test</scope>
11. </dependency>
```

**mvn dependency:tree**分别查看依赖情况：

Java代码 

```
1.  [INFO] com.learn:learn2:jar:0.0.1-SNAPSHOT
2.  [INFO] +- commons-logging:commons-logging:jar:1.1.3:compile
3.  [INFO] \- junit:junit:jar:3.8.1:test
4.  [INFO] -----
   [INFO] -----
```

Java代码 

```
1.  [INFO] com.learn:learn1:jar:0.0.1-SNAPSHOT
2.  [INFO] +- org.springframework:spring-core:jar:4.1.4.RELEASE:compile
3.  [INFO] | \- commons-logging:commons-logging:jar:1.2:compile
```

```


4. [INFO] +- com.learn:learn2:jar:0.0.1-SNAPSHOT:compile
5. [INFO] \- junit:junit:jar:3.8.1:test
6. [INFO] -----
-----

```

结论:

learn1模块里的spring-core依赖common-logging的1.2版本, learn1依赖learn2,并且learn2也依赖common-logging, 版本号是1.1.3。那么最后learn1将会依赖commons-logging的1.2版本, 对于maven中的间接依赖, 哪个依赖在pom文件中定义的位置在前面, 就采用在前面定义的那个依赖。spring-core定义在learn2的前面, 那么此时learn1就会依赖common-logging1.2的版本, 如果spring-core和learn2的顺序调换一下, 就会用commons-logging1.1.3的版本。

情景二:


Xml代码 

```

1. <dependency>
2.     <groupId>org.springframework</groupId>
3.     <artifactId>spring-core</artifactId>
4.     <!-- 版本一 -->
5.     <version>2.0.6</version>
6. </dependency>
7. <dependency>
8.     <groupId>org.springframework</groupId>
9.     <artifactId>spring-core</artifactId>
10.    <!-- 版本二 -->
11.    <version>4.1.4.RELEASE</version>
12. </dependency>

```

分析结果:

Java代码 

```

1. [INFO] com.learn:learn1:jar:0.0.1-SNAPSHOT
2. [INFO] +- org.springframework:spring-core:jar:4.1.4.RELEASE:compile
3. [INFO] | \- commons-logging:commons-logging:jar:1.2:compile

```

结论:

learn1将会依赖后声明的版本为4.1.4.RELEASE的spring-core依赖, 一个Pom文件中声明了对一个项目的高低版本的依赖, 使用最后声明者。

情景三:

learn-1依赖项目learn-2,learn-3,learn-2依赖commons-logging:1.1.3版本, learn-3依赖learn-4, learn-4依赖commons-logging:1.2版本。那么最终learn-1依赖的commons-logging版本将会是1.1.3 版本的。这是因为maven采用最短路径优先原则


情景四(optional):

learn5 pom.xml

Java代码 

```
1. <dependency>
2.     <groupId>commons-logging</groupId>
3.     <artifactId>commons-logging</artifactId>
4.     <version>1.2</version>
5.     <optional>true</optional>
6. </dependency>
```

learn6 pom.xml

Java代码 

```
1. <dependency>
2.     <groupId>com.learn</groupId>
3.     <artifactId>learn5</artifactId>
4.     <version>0.0.1-SNAPSHOT</version>
5. </
    来源: http://zxy-920823.iteye.com/blog/2277646
    dependency>
```

分析结果:

Java代码 

```
1. [INFO] com.learn:learn6:jar:0.0.1-SNAPSHOT
2. [INFO] \- com.learn:learn5:jar:0.0.1-SNAPSHOT:compile
```

如果learn5把commons-logging的optional标签注释掉, 分析的结果将不一样。

分析结果:

Java代码 

```
1. [INFO]
2. com.learn:learn2:jar:0.0.1-SNAPSHOT
3. [INFO] \- com.learn:learn1:jar:0.0.1-SNAPSHOT:compile
4. [INFO]     \- commons-logging:commons-logging:jar:1.2:compile
```

结论:

optional标签如果设置为true,意味着子模块将不依赖此模块, 如果想依赖这个jar,必须在自己的pom.xml文件中在声明一遍。官方文档说: 这样做为了避免错误jar违反license出现问题或者classpath发生问题。