1. Maven的依赖传递
2. 概述

在Maven中，依赖是可以传递的，假设存在个项目，分别是项目A，项目B以及项目C。假设C依赖于B，B依赖于A，那么可以退出项目C依赖于A。其实，Maven中的依赖传递就是jar包的依赖传递，当我们的某个项目依赖于一个jar包时，这个jar包中同时也依赖着其他的jar包，因此，这个项目也同时依赖于这些jar包。

1. 依赖冲突
2. 概述

由于依赖传递现象的存在，当我们导入不同的jar包时，若恰巧两个jar都会依赖于同一个jar包，但是它们依赖的版本不一致，这就导致了依赖冲突，我们无法分清楚使用的是哪个jar包版本。

1. 解决方法
2. 第一声明者优先原则

当我们在pom.xml文件中导入依赖时，以先声明的依赖为准。其实就是根据坐标导入的顺序来确定最终使用哪个传递来的依赖版本。这种方法在导入的坐标较少时，还是够用的，但是在坐标非常多的项目中，是无能为力的。

1. 路径近者优先原则

当两个jar包同时依赖于一个jar包，且该jar包的版本不同时，我们可以直接将该jar包导入到pom.xml文件中，那么该项目就不会使用其他依赖传递过来的该jar包了。因为直接在pom.xml中定义的坐标要比其他依赖传递过来的路径要近。

1. 排除依赖

当两个jar的依赖冲突时，我们可以在pom.xml中定义坐标依赖时就使用<exclusions>标签将不需要传递过来的依赖排除出去。其中可以包括<exclusion>子标签，每个子标签代表一个jar包的坐标。

如：

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.2.4.RELEASE</version>

<exclusions>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

</exclusion>

</exclusions>

</dependency>

1. 版本锁定

采用直接锁定版本的方法确定依赖jar包的版本，版本锁定后则不考虑依赖的声明顺序或者依赖的路径，以锁定的版本为准。这是工程中最常用的方法，也是最有效的方法。

版本锁定的使用方式：

第一步：在dependencyManagement标签中锁定依赖的版本

第二步：在dependencies标签中声明需要导入的maven坐标

如，第一步：

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>5.0.2.RELEASE</version>

</dependency>

</dependencies>

</dependencyManagement>

注意，这是在dependencyManagement标签中进行依赖jar包的版本锁定，即使有dependencies标签，但是不会真正将jar包导入到项目中。我们还需要在新的<dependencies>标签中声明坐标，且此时不需要再指定版本号了。

第二步：

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

</dependency>

</dependencies>

1. 分模块构建Maven工程
2. 分模块

在企业开发中，由于项目规模庞大，业务复杂，一般会通过合理的模块拆分将项目拆分为N多个小模块，分别进行开发。常见的拆分方式有两种：

1. 按照业务模块拆分

当业务逻辑十分复杂时，我们需要按照业务模块来拆分，将每个模块拆分为一个maven工程，例如将一个项目分为用户模块，订单模块，购物车模块等，每个模块对应就是一个maven工程。

1. 按照层进行拆分

当业务不是很复杂时，可以按照层来划分，例如持久层，业务层，表现层等，每个层就是一个maven工程。

不论是哪种拆分方式，通常都会提供一个父工程，将一些公共的代码和配置抽取到父工程中进行统一的管理和配置。其他的子工程就继承这个父工程即可，同时子工程之间还有依赖关系，如DAO层依赖于 POJO层，Service层依赖于Dao层，Web层依赖于Service层。

1. Maven工程的继承
2. 继承目的

在Java语言中，类之间是可以继承的，通过集成，子类就可以引用父类中非private的属性和方法。同样，在Maven工程中也可以继承，子工程继承了父工程后，就可以使用在父工程中引入的依赖。继承的目的是为了消除重复代码。

1. 父工程中的关键定义

父工程的打包方式必须为pom，这是我们区分Maven项目是否为父工程的重要依据。即在该Maven工程的pom.xml中，打包方式是否为pom：

<groupId>com.company</groupId>

<artifactId>company-project-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>pom</packaging>

1. 子工程的关键定义

子工程需要在pom.xml文件中通过parent标签进行父工程的继承,parent标签中表示的就是父工程的各种信息：

<parent>

<groupId>com.taobao<groupId>

<artifactId>company-project-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<artifactId>company-project-children</artifactId>

1. Maven工程的聚合

在Maven工程的pom.xml文件中可以使用<modules>标签将其他Maven工程都聚合到一起，聚合的目的是为了进行统一操作。

如 ，拆分后的Maven工程有多个，如果要进行打包，就可以使用modules标签将这些工程统一聚合到Maven工程中，需要打包的时候，只需要在此工程中执行一次打包命令，其下被聚合的工程就都会被打包了。

<modules>

<module>maven\_dao</module>

<module>maven\_service</module>

<module>maven\_web</module>

<module>maven\_pojo</module>

</modules>

1. Maven私服
2. nexus仓库类型

nexus默认内置了很多仓库，这些仓库可以划分为4种类型，每种类型的仓库用于存放特定的jar包，具体说明如下：

1. hosted

宿主仓库，部署自己的jar包到这个类型的仓库中，包括Releases和Snapshots两部分，Releases为公司内部发行版本仓库，Snapshots为公司内部测试版本仓库。

1. proxy

代理仓库，用于代理远程的公共仓库，如Maven中央仓库，用户连接私服，私服自动去中央仓库下载Jar包或者插件。

1. group

仓库组，用来合并多个hosted/proxy仓库，通常用于配置自己的Maven连接仓库组。

1. virtual

兼容Maven1版本的jar或者插件