在企业工作中，有这样的一种场景853769620

比如说，你开发了一个第三方的一个组件，就是一个不属于任何一个项目的基础的一个组件，比如基于activiti开发了一个流程审批的封装以后的一个组件

然后呢，其他人如果要用你的这个组件，肯定是在depedencies里面去定义和声明，然后看去依赖你的组件，然后才可以基于你的组件去开发么

但是有一个问题，假设说他呢依赖了你的组件，然后呢你的组件依赖了activiti 1.3版本，但是有个问题，结果依赖方自己没头没脑的又声明了一个队activiti 1.2的依赖，此时根据依赖调解的原则，肯定是用他自己的旧版本

比较容易出现依赖的冲突

所以说，对于公司里一些重量级的组件，一般来说会采取下面的这种方式

首先自己开发一个工程，那个工程是对外提供服务的；接着再开发一个pom包，这个pom包专门用dependencyManagement来约束依赖方的各个版本号，不要跟自己出现冲突

然后依赖方，不是直接依赖那个工程自己的包，而是依赖于那个工程的pom类型的包，然后再在dependency里面去声明自己要的依赖，此时版本就被约束了

<dependencyManagement>

<dependencies>

<dependency>

<groupId>com.zhss.commons</groupId>

<artifactId>commons-flow-bom</artifactId>

<version>1.2.9</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

可能很多同学都不知道是怎么回事，所以这块儿是这样子，我再把思路给总结一遍

（1）如果你是公司基础组件的开发人，你的组件依赖了很多重要的第三方的开源框架，然后你为了保证依赖你的组件的人，不会自行导入一些开源框架的过旧的版本，导致跟你的组件出现依赖冲突

（2）你需要为你的组件开发一个类型为pom的工程，后缀名为bom，这个工程，要声明一套dependencyManagement，里面声明好对你的组件的依赖的版本号，还有你的组件使用的重要的第三方开源框架的版本号

（3）然后依赖方在引用你的组件的时候，需要在自己的dependencyManagement中，加一个scope范围为import，类型为pom的这么依赖导入，此时就可以将你的bom包声明的那一套依赖的版本号导入他那里，作为版本的约束

（4）然后依赖方接着在dependencies里面可以声明对你的组件的依赖，此时版本号都不用写，因为已经被你约束了

（5）同时，假设依赖方要自己依赖一个过旧的开源框架的版本，会有提示报警，不让他自行定义过旧版本的框架