# 尚马教育 JAVA 高级课程

# Maven项目管理

文档编号：C11

创建日期： 2017-07-07

最后修改日期：2019-09-09

版 本 号：V3.0

电子版文件名：尚马教育-第三阶段-11.maven项目管理工具.docx

**文档修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新作者 | 更新说明 | 版本号 |
| 2017-07-30 | 张元林 | 初始版本 | V1.0 |
| 2018-08-11 | 王绍成 | Mybatis版本更新 | V2.0 |
| 2019-10-25 | 冯勇涛 | 课件格式以及课程深度加深 | V3.0 |

目录

[1. Maven介绍](#_Toc308_WPSOffice_Level1) [3](#_Toc308_WPSOffice_Level1)

[1.1. 认识maven](#_Toc32058_WPSOffice_Level2) [3](#_Toc32058_WPSOffice_Level2)

[1.2. Maven作用](#_Toc9841_WPSOffice_Level2) [3](#_Toc9841_WPSOffice_Level2)

[1.3. maven仓库说明](#_Toc7994_WPSOffice_Level2) [3](#_Toc7994_WPSOffice_Level2)

[2. Maven客户端安装：](#_Toc32058_WPSOffice_Level1) [3](#_Toc32058_WPSOffice_Level1)

[2.1. 下载客户端](#_Toc2631_WPSOffice_Level2) [3](#_Toc2631_WPSOffice_Level2)

[2.2. Settings.xml配置](#_Toc28884_WPSOffice_Level2) [3](#_Toc28884_WPSOffice_Level2)

[2.2.1. 本地仓库](#_Toc32058_WPSOffice_Level3) [3](#_Toc32058_WPSOffice_Level3)

[2.2.2. 镜像地址](#_Toc9841_WPSOffice_Level3) [3](#_Toc9841_WPSOffice_Level3)

[2.3. 环境变量配置](#_Toc3647_WPSOffice_Level2) [3](#_Toc3647_WPSOffice_Level2)

[2.4. eclipse配置maven客户端](#_Toc10806_WPSOffice_Level2) [3](#_Toc10806_WPSOffice_Level2)

[3. Eclipse创建java工程](#_Toc9841_WPSOffice_Level1) [4](#_Toc9841_WPSOffice_Level1)

[3.1. Java工程目录认识](#_Toc860_WPSOffice_Level2) [4](#_Toc860_WPSOffice_Level2)

[3.1.1. Src目录](#_Toc7994_WPSOffice_Level3) [4](#_Toc7994_WPSOffice_Level3)

[3.1.2. Target目录](#_Toc2631_WPSOffice_Level3) [4](#_Toc2631_WPSOffice_Level3)

[3.1.3. 核心Pom.xml](#_Toc28884_WPSOffice_Level3) [5](#_Toc28884_WPSOffice_Level3)

[3.2. Maven常用命令](#_Toc5047_WPSOffice_Level2) [6](#_Toc5047_WPSOffice_Level2)

[4. eclipse创建web工程](#_Toc7994_WPSOffice_Level1) [7](#_Toc7994_WPSOffice_Level1)

[4.1. Web工程目录认识](#_Toc26225_WPSOffice_Level2) [7](#_Toc26225_WPSOffice_Level2)

[4.2. Web工程骨架错误修改](#_Toc7504_WPSOffice_Level2) [7](#_Toc7504_WPSOffice_Level2)

[4.3. Web工程运行](#_Toc31805_WPSOffice_Level2) [8](#_Toc31805_WPSOffice_Level2)

[4.3.1. 插件运行](#_Toc3647_WPSOffice_Level3) [8](#_Toc3647_WPSOffice_Level3)

[4.3.2. 本地tomcat运行](#_Toc10806_WPSOffice_Level3) [8](#_Toc10806_WPSOffice_Level3)

[4.3.3. 远程tomcat运行](#_Toc860_WPSOffice_Level3) [8](#_Toc860_WPSOffice_Level3)

[4.4. Maven继承](#_Toc25288_WPSOffice_Level2) [8](#_Toc25288_WPSOffice_Level2)

[4.4.1. 使用场景](#_Toc5047_WPSOffice_Level3) [8](#_Toc5047_WPSOffice_Level3)

[4.5. Maven工程聚合](#_Toc15206_WPSOffice_Level2) [9](#_Toc15206_WPSOffice_Level2)

[4.5.1. 使用场景](#_Toc26225_WPSOffice_Level3) [10](#_Toc26225_WPSOffice_Level3)

## Maven介绍

### 认识maven

Maven是一个采用纯Java编写的开源项目管理工具, Maven采用了一种被称之为Project Object Model (POM)概念来管理项目，所有的项目配置信息都被定义在一个叫做POM.xml的文件中, 通过该文件Maven可以管理项目的整个生命周期，包括清除、编译，测试，报告、打包、部署等等。目前Apache下绝大多数项目都已经采用Maven进行管理. 而Maven本身还支持多种插件, 可以方便更灵活的控制项目, 开发人员的主要任务应该是关注商业逻辑并去实现它, 而不是把时间浪费在学习如何在不同的环境中去依赖jar包,项目部署等。Maven正是为了将开发人员从这些任务中解脱出来而诞生的。

### Maven作用

Maven 简化了工程的构建过程，并对其标准化。它无缝衔接了编译、发布、文档生成、团队合作和其他任务。Maven 提高了重用性，负责了大部分构建相关的任务。

- 构建应用程序：maven支持许多种的应用程序类型，对于每一种支持的应用程序类型都定义好了一组构建规则和工具集。

- 输出物管理：maven可以管理项目构建的产物，并将其加入到用户库中。这个功能可以用于项目组和其他部门之间的交付行为。

- 依赖关系：maven对依赖关系的特性进行细致的分析和划分，避免开发过程中的依赖混乱和相互污染行为。

- 文档和构建结果：maven的site命令支持各种文档信息的发布，包括构建过程的各种输出，javadoc，产品文档等。

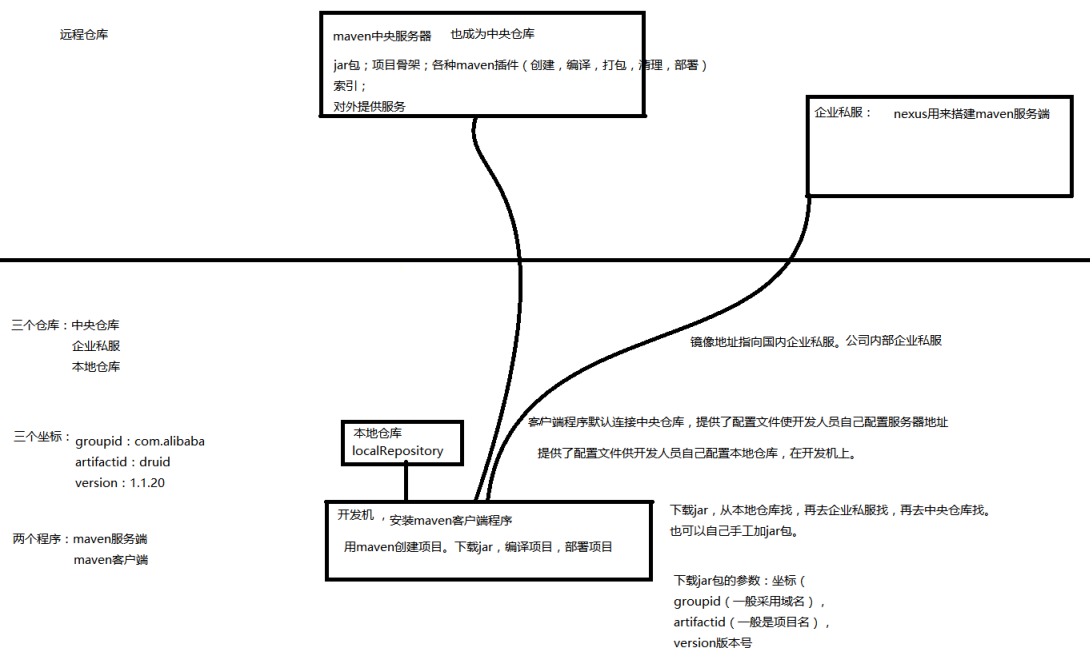
- 项目关系：一个大型的项目通常有几个小项目或者模块组成，用maven可以很方便地管理

移植性管理。

### 约定优于配置

* Maven 使用约定而不是配置，意味着开发者不需要再自己创建构建过程开发者不需要再关心每一个配置细节。
* Maven 为工程提供了合理的默认行为。
* 当创建 Maven 工程时，Maven 会创建默认的工程结构。
* 开发者只需要合理的放置文件，而在pom.xml中不再需要定义任何配置为了构建工程，Maven 为开发者提供了选项来配置生命周期目标和工程依赖（依赖于 Maven 的插件扩展功能和默认的约定）。
* 大部分的工程管理和构建相关的任务是由 Maven 插件完成的。
* 开发人员不需要了解每个插件是如何工作的，就能够构建任何给定的 Maven 工程。

### maven仓库说明



## Maven客户端安装：

### 下载客户端

官网地址：<https://maven.apache.org/>

|  |
| --- |
|  |
|  |

### Settings.xml配置

#### 本地仓库

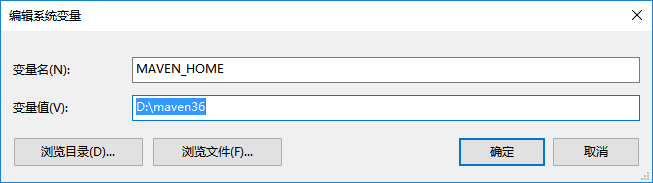
|  |
| --- |
|  |

#### 镜像地址

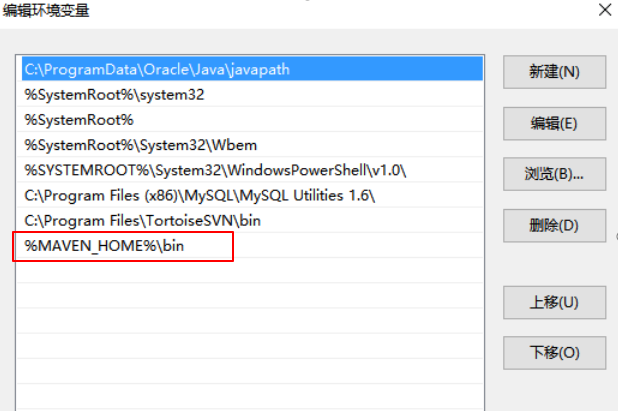
|  |
| --- |
| <mirror>  <id>nexus-javasm</id>  <mirrorOf>central</mirrorOf>  <name>Nexus javasm</name>  <url>http://192.168.20.252:8081/repository/maven-public/</url>  </mirror> |

### 环境变量配置

配置MAVEN\_HOME:



配置Path变量：



检查是否安装成功：

mvn -version

### Idea配置maven客户端

#### Setting配置

|  |
| --- |
|  |

#### Other settings配置

|  |
| --- |
|  |

## 创建java工程

### Java工程目录认识

#### Src目录

* src/main/java：源代码目录
* Src/main/resources：资源文件目录
* Src/test/java：测试用例目录
* Src/test/resources：测试用例资源文件目录

#### Target目录

Maven输出物目录，保存编译输出物，打包输出物等。

#### 核心Pom.xml

##### 项目坐标

|  |
| --- |
|  |

不写packageing的话，默认packageing为jar打包方式。

##### 修改maven依赖jdk版本

|  |
| --- |
| 方法1： |
| 方法2： |
| 方法3：全局配置，在maven客户端的settings.xml中修改 |

##### Dependencies依赖项

|  |
| --- |
|  |

##### **scope定义jar包使用阶段**

|  |
| --- |
|  |

1. compile

默认scope为compile，表示为当前依赖参与项目的编译、测试和运行阶段，属于强依赖。打包之时，会达到包里去。

1. test

该依赖仅仅参与测试相关的内容，包括测试用例的编译和执行，比如定性的Junit。

1. runtime

依赖仅参与运行周期中的使用，编译不参与。一般这种类库都是接口与实现相分离的类库，比如JDBC类库，在编译之时仅依赖相关的接口，在具体的运行之时，才需要具体的mysql、oracle等等数据的驱动程序。 此类的驱动都是为runtime的类库。

1. provided

该依赖在打包过程中，不需要打进去，这个由运行的环境来提供，比如tomcat或者基础类库等等，事实上，该依赖可以参与编译、测试和运行等周期，与compile等同。区别在于打包阶段进行了exclude操作。

##### 依赖包版本统一管理

|  |
| --- |
|  |

##### 排除依赖项

|  |
| --- |
|  |

##### 查看jar依赖关系

|  |
| --- |
| 方法1：右侧边栏 |
| 方法2：Diagrams查看 |
|  |

##### 打包时的资源配置

为避免mybatis使用时，映射文件丢失的问题，必须使用该配置：

|  |
| --- |
|  |

### Maven常用命令

clean：清理target目录；  
compile：编译命令；  
test：执行测试命令；  
package：打包命令；一般打包的时候都跳过测试，

eclipse：package -Dmaven.test.skip=true

idea：闪电符号  
install：安装命令：把打包的jar或war上传到本地仓库  
deploy：部署命令：把打包的jar或war上传到本地仓库，并上传到远程仓库（私服）

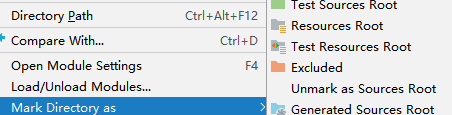
## web工程

### Web工程骨架错误修改

src/main目录下手工创建java，resources目录，并指定为SourceRoot，ResourceRoot；

src目录下手工创建test，test目录下创建java目录，并指定test目录为test Source Root；

右键目录名：



### Web工程运行

#### 本地tomcat运行

|  |
| --- |
|  |

#### 插件运行

Maven插件内嵌应用服务器：tomcat7，jetty

|  |
| --- |
| Tomcat7：一般不用，版本过低 |
| Jetty：经常使用 |

#### 远程tomcat运行

要求远程服务器的tomcat下的webapps目录下有manager，host-manager项目，并tomcat处于运行状态；

配置用户名密码，角色：

|  |
| --- |
| 位置：Tomcat>>conf>>tomcat-users.xml  <role rolename="admin-gui"/>  <role rolename="admin-script"/>  <role rolename="manager-gui"/>  <role rolename="manager-script"/>  <role rolename="manager-jmx"/>  <role rolename="manager-status"/>  <user username="admin" password="admin" roles="manager-gui,manager-script,manager-jmx,manager-status,admin-script,admin-gui"/> |
| 位置：Pom.xml  <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**version**>2.2</**version**>  <**configuration**>  <**url**>http://localhost:8080/manager/text</**url**>*<!-- tomcat7部署管理路径 -->* <**username**>admin</**username**>*<!-- tomcat的管理员账号 -->* <**password**>admin</**password**>  <**port**>8080</**port**>  <**path**>/1128ssm2</**path**>*<!-- 部署路径 -->* <**charset**>UTF-8</**charset**>  *<!-- 运行redeploy命令前，要能正常访问http://localhost:8080/manager-->* </**configuration**> </**plugin**> |

## Ssm整合注意点

Mybatis映射文件打包：

|  |
| --- |
| <**resources**>  <**resource**>  <**directory**>src/main/java</**directory**>  <**includes**>  <**include**>\*\*/\*.xml</**include**>  </**includes**>  </**resource**>  <**resource**>  <**directory**>src/main/resources</**directory**>  <**includes**>  <**include**>\*\*/\*</**include**>  </**includes**>  </**resource**> </**resources**> |

Mybatis逆向工程插件：

|  |
| --- |
| <**plugin**>  <**groupId**>org.mybatis.generator</**groupId**>  <**artifactId**>mybatis-generator-maven-plugin</**artifactId**>  <**version**>1.3.2</**version**>  <**dependencies**>  <**dependency**>  <**groupId**>mysql</**groupId**>  <**artifactId**>mysql-connector-java</**artifactId**>  <**version**>5.1.34</**version**>  </**dependency**>  </**dependencies**>  <**configuration**>  <**overwrite**>true</**overwrite**>  <**configurationFile**>src/main/resources/generator.xml</**configurationFile**>  </**configuration**> </**plugin**> |
| 运行插件： |
| Resources目录下必须有generator.xml |

## Maven继承

#### 使用场景

|  |
| --- |
| 多个项目都需要某些依赖, 就可以把这些项目共同的依赖抽取到父项目中, 这些项目作为子项目通过继承父项目得到这些依赖, 这样也更好的来管理(比如升级, 删除等) **步骤:** |
| 1. 父项目的打包方式修改为 pom   <packaging>pom</packaging> |
| 1. 父项目使用 dependencyManagement 标签来管理依赖, 表示子项目默认不继承, 可以配置继承, optional 表示子 pom 无论如何都不能继承   <dependencyManagement>  <dependencies>  <!-- 子项目可以继承 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>1.2.47</version>  </dependency>  <!-- 子项目不可以继承 -->  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.17</version>  <optional>true</optional>  </dependency>  </dependencies>  </dependencyManagement> |
| 1. 子项目配置parent   <parent>  <artifactId>parent</artifactId>  <groupId>com.ictpaas</groupId>  <version>1.0-SNAPSHOT</version>  </parent> |
| 1. 子项目依赖配置   <dependencies>  <!-- 不需要版本, 会从父项目继承, 如果指定版本就是代表不是来自父 pom 而是子 pom 自己的. 父项目的 log4j 是不能继承的 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  </dependency>  </dependencies> |

### Maven工程聚合

Maven工程聚合基于maven继承。

#### 使用方式

|  |
| --- |
| 某个子项目被多个其他子项目依赖，比如实体类单独作为一个子项目，可以被web子项目依赖，可以被andorid子项目依赖。此时使用maven工程聚合。 |
| 第一步：先建立父项目，作用是用来统一管理依赖版本。  *<!--父工程：管理子工程（子工程的依赖项版本统一管理）-->* <**groupId**>com.javasm</**groupId**>  <**artifactId**>1129ssm-parent</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  <**packaging**>pom</**packaging**>*<!--pom是父工程使用打包方式-->* <**properties**>  <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>  <**maven.compiler.source**>1.8</**maven.compiler.source**>  <**maven.compiler.target**>1.8</**maven.compiler.target**>  <**spring-version**>5.1.5.RELEASE</**spring-version**>  <**aspect-version**>1.9.2</**aspect-version**>  <**fastjson-version**>1.2.58</**fastjson-version**>  </**properties**>   *<!--在父项目中定义dependencies，子项目自动继承，其实有些依赖是子项目不需要继承的。-->  <!--父项目dependencyManagement做依赖管理，子项目不会自动继承这些依赖。-->* <**dependencyManagement**>  <**dependencies**>  *<!--1.spring：aop注解-->* <**dependency**>  <**groupId**>org.aspectj</**groupId**>  <**artifactId**>aspectjrt</**artifactId**>  <**version**>${aspect-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.aspectj</**groupId**>  <**artifactId**>aspectjweaver</**artifactId**>  <**version**>${aspect-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>aopalliance</**groupId**>  <**artifactId**>aopalliance</**artifactId**>  <**version**>1.0</**version**>  </**dependency**>  <**dependency**>  <**groupId**>cglib</**groupId**>  <**artifactId**>cglib-nodep</**artifactId**>  <**version**>2.2</**version**>  </**dependency**>   *<!--2.springMVC-->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-core</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-beans</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-aop</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-aspects</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-expression</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-webmvc</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>fastjson</**artifactId**>  <**version**>${fastjson-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>commons-fileupload</**groupId**>  <**artifactId**>commons-fileupload</**artifactId**>  <**version**>1.3.1</**version**>  </**dependency**>   *<!--3.mybatis,spring集成-->* <**dependency**>  <**groupId**>mysql</**groupId**>  <**artifactId**>mysql-connector-java</**artifactId**>  <**version**>5.1.38</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>druid</**artifactId**>  <**version**>1.1.20</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.mybatis</**groupId**>  <**artifactId**>mybatis</**artifactId**>  <**version**>3.4.6</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.mybatis</**groupId**>  <**artifactId**>mybatis-spring</**artifactId**>  <**version**>1.3.2</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.github.pagehelper</**groupId**>  <**artifactId**>pagehelper</**artifactId**>  <**version**>5.1.10</**version**>  </**dependency**>   *<!--5.redis-->* <**dependency**>  <**groupId**>redis.clients</**groupId**>  <**artifactId**>jedis</**artifactId**>  <**version**>3.1.0</**version**>  </**dependency**>   *<!--6.quartz-->* <**dependency**>  <**groupId**>org.quartz-scheduler</**groupId**>  <**artifactId**>quartz</**artifactId**>  <**version**>2.3.0</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context-support</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  *<!--7.junit测试用例-->* <**dependency**>  <**groupId**>junit</**groupId**>  <**artifactId**>junit</**artifactId**>  <**version**>4.11</**version**>  <**scope**>test</**scope**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-test</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  *<!-- 8.spring的事务管理-->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-jdbc</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-tx</**artifactId**>  <**version**>${spring-version}</**version**>  </**dependency**>  *<!--9.servlet环境-->* <**dependency**>  <**groupId**>javax.servlet</**groupId**>  <**artifactId**>javax.servlet-api</**artifactId**>  <**version**>3.1.0</**version**>  <**scope**>provided</**scope**>*<!--package命令不打包-->* </**dependency**>  <**dependency**>  <**groupId**>javax.servlet.jsp</**groupId**>  <**artifactId**>javax.servlet.jsp-api</**artifactId**>  <**version**>2.3.1</**version**>  <**scope**>provided</**scope**>  </**dependency**>   </**dependencies**>  </**dependencyManagement**> |
| 第二步：建立子项目，dao，entity，commons，service，handler等  完成后，父项目的pom：  <**modules**>  <**module**>1129ssm-entity</**module**>  <**module**>1129ssm-mapper</**module**>  <**module**>1129ssm-service</**module**>  <**module**>1129ssm-handler</**module**>  <**module**>1129ssm-commons</**module**>  <**module**>1129ssm-interface</**module**>  </**modules**>  子项目的pom：子项目有parent标签，没有groupid与version。子项目引用依赖，如果是父项目所管理的依赖不需要版本，如果不是父项目管理的依赖，需要版本。各子项目之间可以互相依赖。  <**parent**>  <**artifactId**>1129ssm-parent</**artifactId**>  <**groupId**>com.javasm</**groupId**>  <**version**>1.0-SNAPSHOT</**version**> </**parent**> <**modelVersion**>4.0.0</**modelVersion**>  <**artifactId**>1129ssm-service</**artifactId**> <**dependencies**>  <**dependency**>  <**groupId**>com.javasm</**groupId**>  <**artifactId**>1129ssm-mapper</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>   <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>fastjson</**artifactId**>  </**dependency**>  </**dependencies**> |