maven 项目管理工具（用于项目构建）

# 教学计划

1、maven介绍

重点理解maven仓库工具原理（掌握）

2、搭建maven环境

重点配置maven仓库（掌握）

3、基础，maven对项目生命周期的管理

重点在eclipse中如何maven工程（掌握）

记住maven工程目录结构（掌握）

4、基础，maven的依赖管理（掌握）

依赖管理是maven的核心。

maven通过标准管理项目工程所依赖的jar包（构件）。

构件的坐标（掌握）

处理依赖冲突（掌握）

重要案例：通过maven构建 ssh整合（掌握）

5、使用maven对工程分模块构建（掌握）

实际企业开发，对工程进行分模块构建。

重要案例：通过maven ssh整合工程分模块构建（掌握）

6、maven私服的使用（了解）

实际企业开发都会使用maven私服。

# 开发中一些问题

maven解决企业开发项目的问题！

1、maven提供一套对项目生命周期管理的标准，开发人员、和测试人员统一使用maven进行项目构建。

项目生命周期管理：编译、测试、打包、部署、运行、。

2、maven对项目所原来的第三方构件（jar包）进行统一管理。

向工程中加入jar包不要手工从其它地方拷贝，通过maven定义jar包的坐标，自动从maven仓库中去下载到工程中。解决Jar的冲突

3、maven对工程分模块构建，提高开发效率。

4、在实际开发中，在公司架设一台maven私服（maven仓库），存储了各各项目组所共享的jar包，还存储了从互联网下载的常用的jar（struts2、spring、hibenrate、 springmvc、log4...）

# maven概念模型

maven是apache下的开源项目，纯JAVA开发，项目管理工具，管理java项目。

1、目对象模型 (Project Object Model)

POM对象模型，每个maven工程中都有一个pom.xml文件，定义工程所依赖的jar包、本工程的坐标、打包运行方式。。

2、依赖管理系统（基础核心 ）

maven通过坐标对项目工程所依赖的jar包统一规范管理。

3、maven定义一套项目生命周期

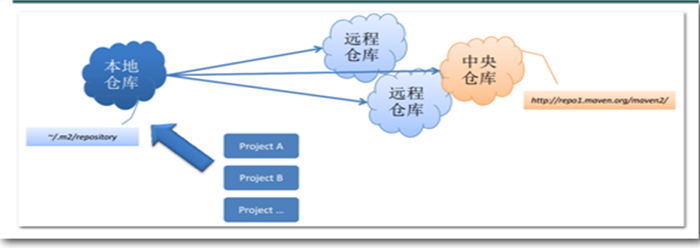
清理、初始化、编译、测试、报告 、打包、部署、站点生成

4、一组标准集合

强调：maven工程有自己标准的工程目录结构、定义坐标有标准。

5、maven 管理项目生命周期过程都是基于插件完成的开发中使用tomcat的插件。

# maven的仓库（掌握）



1、本地仓库

相当于缓存，工程第一次会从远程仓库（互联网）去下载jar 包，将jar包存在本地仓库（在程序员的电脑上）。

第二次不需要从远程仓库去下载。先从本地仓库找，如果找不到才会去远程仓库找。

2、中央仓库

就是远程仓库，仓库中jar由专业团队（maven团队）统一维护。

中央仓库的地址：http://repo1.maven.org/maven2/

3、远程仓库

在公司内部架设一台私服，其它公司架设一台仓库，对外公开。

# maven安装

## 下载maven



解压到磁盘，不要有中文或空格。

## 配置环境变量

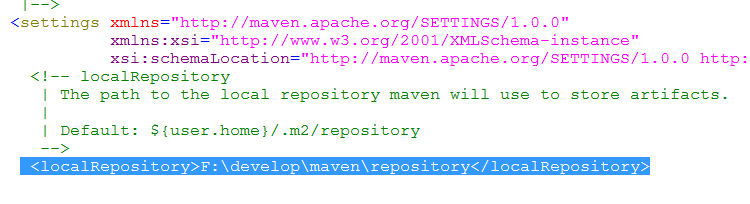


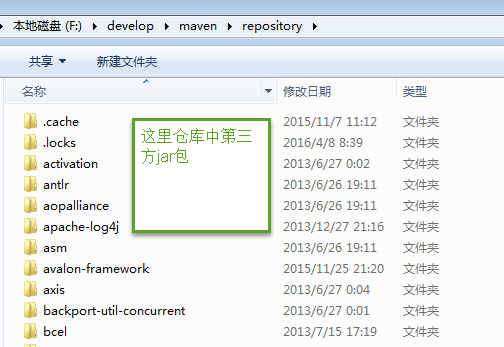




## 本地仓库配置（重点）

在maven的安装目录中conf/ settings.xml文件，在这里配置本地仓库。





如果本地仓库不配置，默认地址：${user.dir}/.m2，user.dir(windows用户目录)。

理解全局setting和用户setting。

全局setting：在maven的安装目录中conf/ settings.xml文件.

所有配置应用于所有项目。

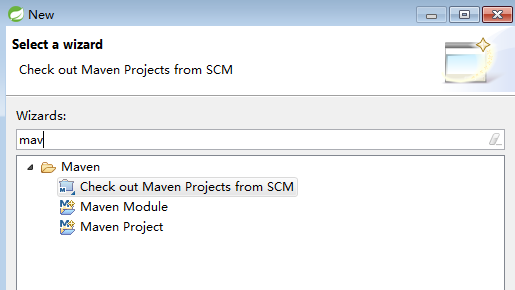
用户setting：${user.dir}/.m2。

某些用户需要对maven特殊设置，用户可以自定义setting.

教学中统一使用全局setting.xml

## 在eclipse安装maven插件

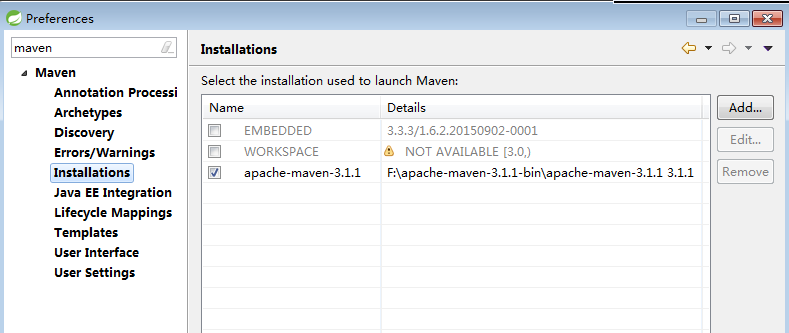
一些高版本的eclipse，maven插件自动有了



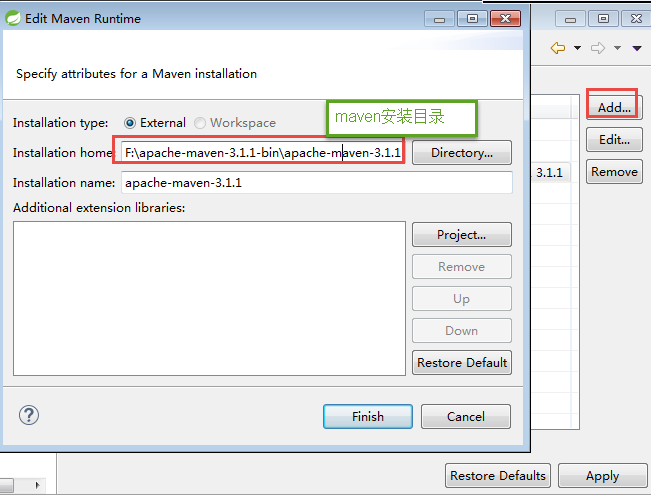
## 在eclipse中配置maven

### 配置maven安装程序

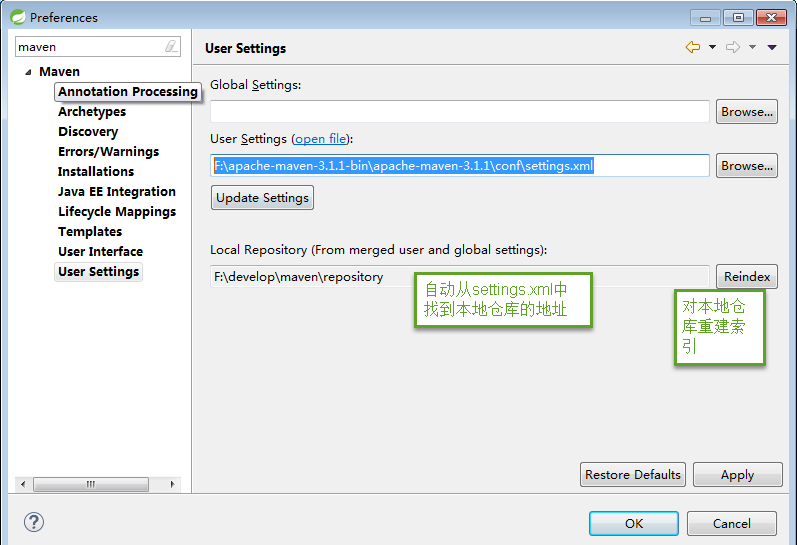
一些高版本的eclipse，内置了maven程序。



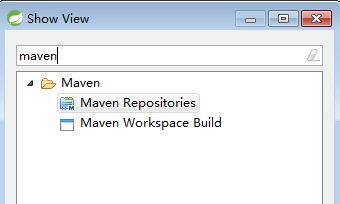
一般不用内置的maven程序，为了在项目中统一maven版本。

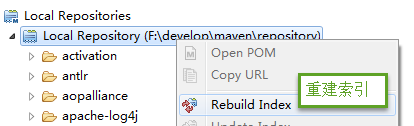


### 指定user setting



### 通过eclipse浏览本地仓库



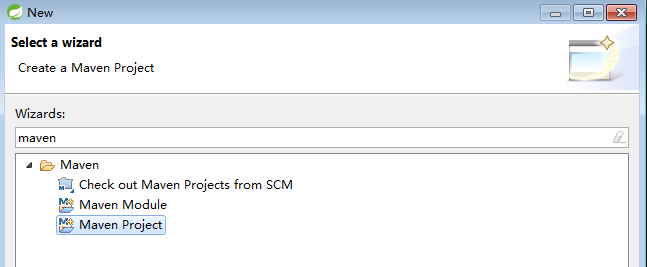


# 管理项目生命周期

maven对项目生命周期进行定义，规范，开发人员和测试人员使用maven软件完成构建。

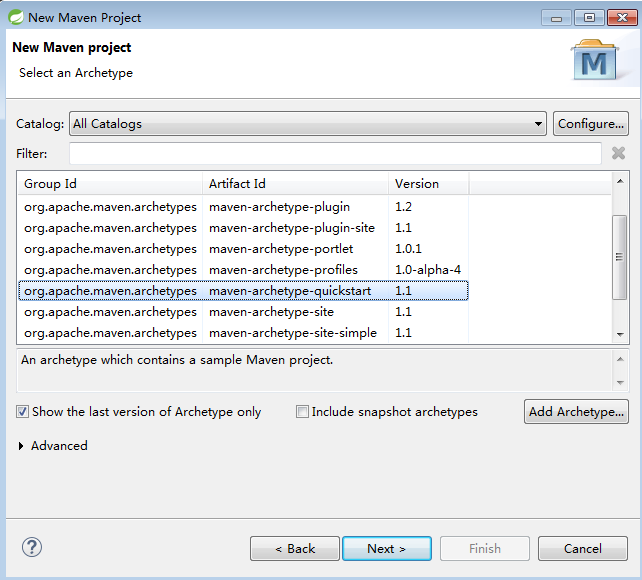
## 创建java工程（maven）(掌握)

创建一个maven工程



### 选择原型（骨架）

maven内部提供很多项目原型，方便去创建项目。



### 定义maven工程的坐标

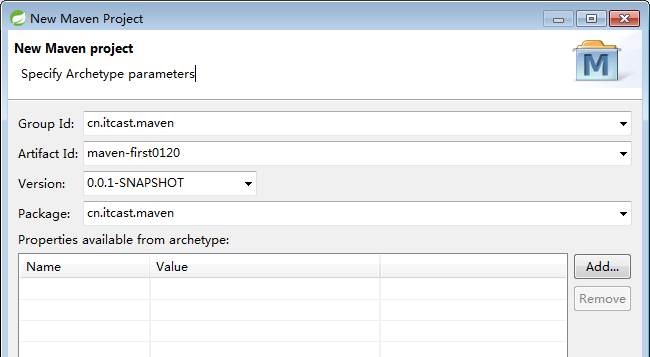
maven对所有工程管理基于坐标进行管理。

坐标包括：

group id：项目的名称，项目名称以域名的倒序，比如：cn.itcast.crm

artifactid：模块名称（子项目名称）

version：模块的版本，snapshot（快照版，没有正式发行）、release（正式发行版本）



### maven工程目录结构

新创建的maven工程一个完整的目录结构

project

/src/main/java 主体程序 java源文件（不要放配置文件）

/src/main/resources 主体程序所需要的配置文件（不要放java文件）

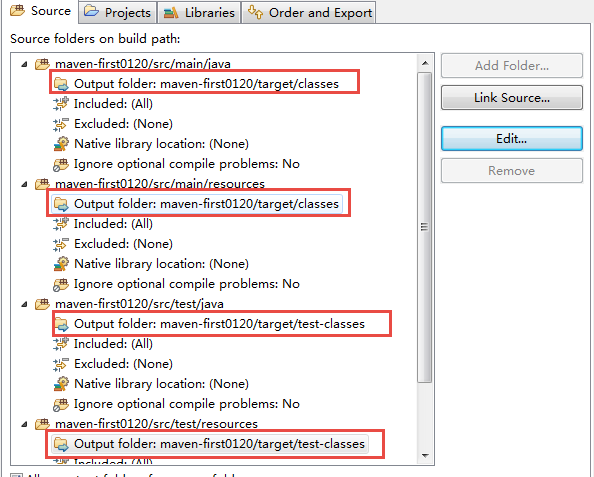
/src/test/java 单元测试程序 java源文件

/src/test/resources单元测试程序所用的配置文件

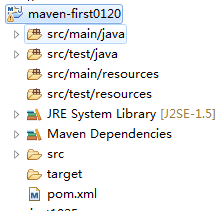
/target 编译输出目录

/pom.xml

手动指定输出目录：



完整的工程目录：



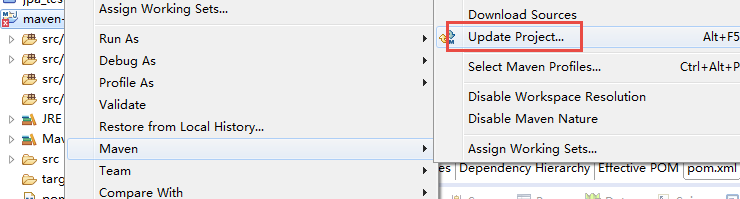
### 设置编译版本

使用jdk1.7

在pom.xml设置：



Update project：



## 测试maven命令

maven通过命令对工程进行编译、测试、打包、发布、运行。。

mvn：运行不调试

mvnDebug：调试运行



mvn compile：编译

mvn clean：清理，将target下的class文件清理

mvn test：执行单元测试类，执行src/test/java下的类

mvn package ：将java工程打成jar、war。

mvn install：安装命令，将将程的jar发布到本地仓库

发布到仓库，目录按照本工程定义的坐标生成目录



如果在cmd命令行下测试maven命令：

1、前边加mvn

2、进入工程当前目录要有pom.xml

## 项目生命周期（了解）

maven包括三套项目生命周期：

Clean Lifecycle 在进行真正的构建之前进行一些清理工作。

Default Lifecycle 构建的核心部分，编译，测试，打包，部署等等。

Site Lifecycle 生成项目报告，站点，发布站点。

注意：

执行一个生命周期阶段，会自动前台的阶段执行完成。

比如：执行package，先test测试。

## pom.xml（重点）

基本配置：

每个maven都有一个pom.xml，pom.xml在工程根目录。

gropuid：项目名称

artifactid：模块名称(子项目名称)

version：版本

packaging：打包方式，jar(java工程)、war(web工程)、pom(作为父工程)

dependencies：依赖管理，配置本工程所依赖jar包（构件）

build：工程构建配置，配置插件

# Maven依赖管理（掌握）

## 坐标定义

maven对每个工程通过坐标管理。

groupId：定义当前Maven项目名称

artifactId：定义项目模块

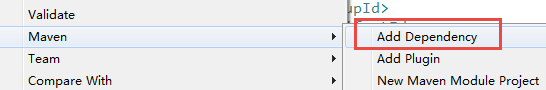
version：定义当前项目的当前版本

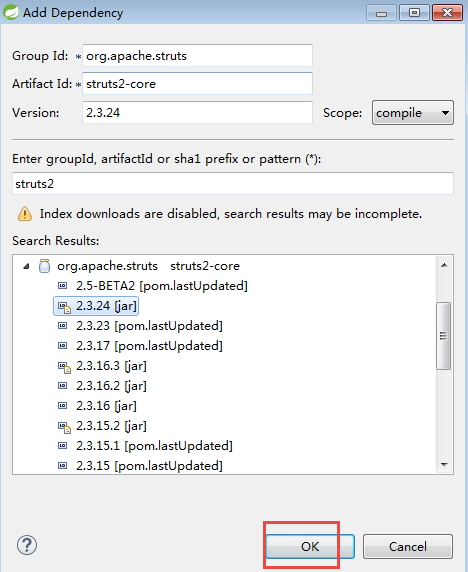
## 查询其它构件的坐标

1、通过互联网搜索

2、通过本地仓库搜索（学习）

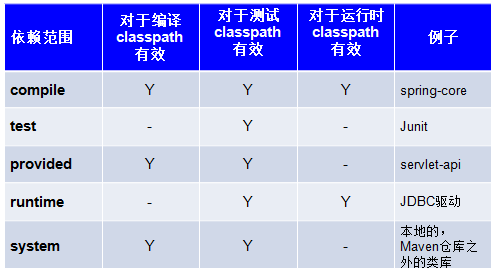
## 从本地仓库添加依赖





## 依赖范围scope（重点理解）

A依赖 B，依赖B的范围。



1、compile 编译、测试、运行，A在编译时依赖B，并且在测试和运行时也依赖。

strus-core、spring-beans

打到war包或jar包。

2、provided 编译、和测试有效，A在编译和测试时需要B。

比如：servlet-api就是编译和测试有用，在运行时不用（tomcat容器已提供）。

不会打到war。

3、runtime：测试、运行有效。

比如：jdbc驱动包 ，在开发代码中针对java的jdbc接口开发，编译不用。

在运行和测试时需要通过jdbc驱动包（mysql驱动）连接数据库，需要的！！

会打到war。

4、test：只是测试有效，只在单元测试类中用。

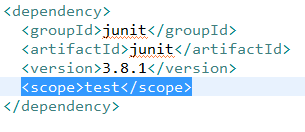
比如：junit

不会打到war。

按照依赖强度，由强到弱来排序：(理解)

compile> provided> runtime> test

测试依赖范围：



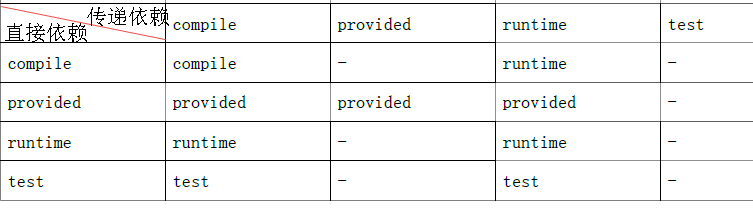
没有调试scope默认为compile。

## 传递依赖（理解）

### 什么传递依赖

A依赖B，B依赖C，在A中加入B的坐标，自动导入C的jar包，称C为A传递依赖。

### 传递依赖与范围（理解）



1、纵坐标：直接依赖

A依赖B，B是A的直接依赖。

在A的pom.xml中添加B的坐标。

2、横坐标：传递依赖

B依赖C，C是A的传递依赖。

3、中间部分：传递依赖的范围，A依赖C的范围。



## 小结

maven是项目管理工具，使用maven进行java项目构建！

每个maven工程包括一个pom.xml文件。

pom.xml基本配置：

1、坐标

groupid 项目名称

artifactid：模块名称

version：模块版本

2、依赖管理

dependency

groupid 项目名称

artifactid：模块名称

version：模块版本

3、build，构建配置

主要配置插件，编译、tomcat插件。

maven的运行依赖仓库包括：

1、本地仓库，缓存

2、远程仓库

3、中央仓库，是特殊的远程仓库，是maven软件默认远程仓库的地址

当向工程加入一个依赖，先从本地仓库找，如果找不到再从远程仓库找。

依赖管理：

1、依赖范围

由强到弱：compile（编译、测试、运行有效）》provided（编译、测试）》runtime（测试、运行有效）》test（测试）

2、传递依赖

A依赖B，B依赖C，C就是A的传递依赖。

3、传递依赖范围

## 依赖版本冲突解决

### 依赖调解原则（了解）

maven自己提供依赖调解原则：

1、第一声明者优先原则

2、路径近者优先原则

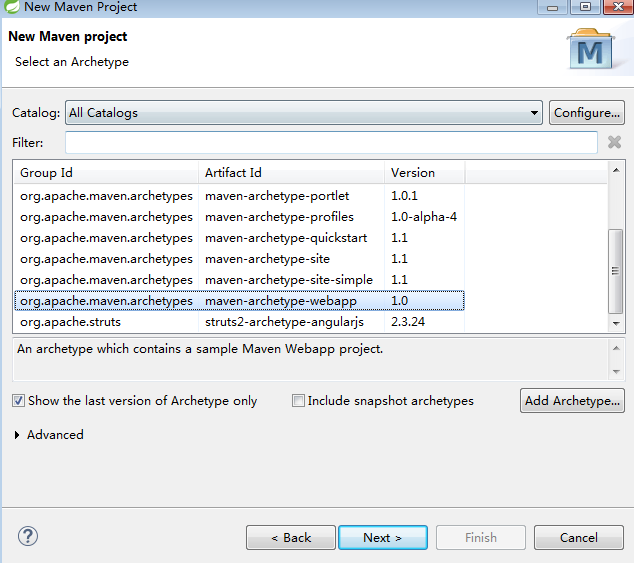
### 锁定版本（推荐使用）



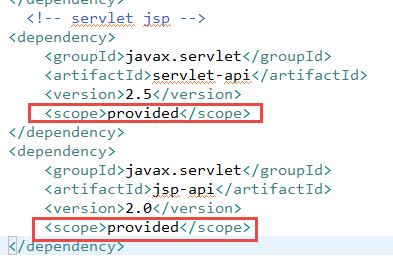
# 使用maven构建ssh整合工程（掌握）

## 创建maven（web工程）

### 选择原型，web工程要选择webapp.



### 引入servlet-api的jar



### 修改web.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

id=*"WebApp\_ID"* version=*"2.5"*>

</web-app>

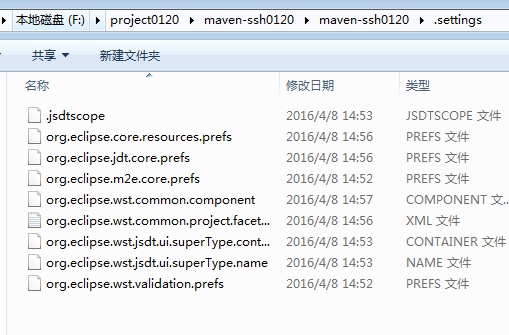
### 问题

Description Resource Path Location Type

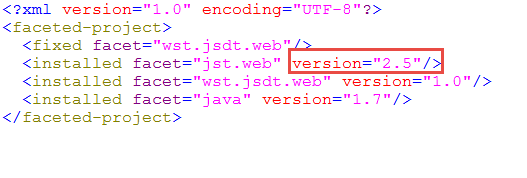
Cannot change version of project facet Dynamic Web Module to 2.5. maven-ssh0120 line 1 Maven Java EE Configuration Problem

参考：Dynamic web module to 2.5.docx

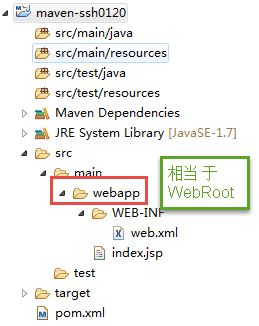
找到工程目录：



修改：



### 工程目录



## 准备环境

数据库环境：

创建maven数据库：

导入脚本



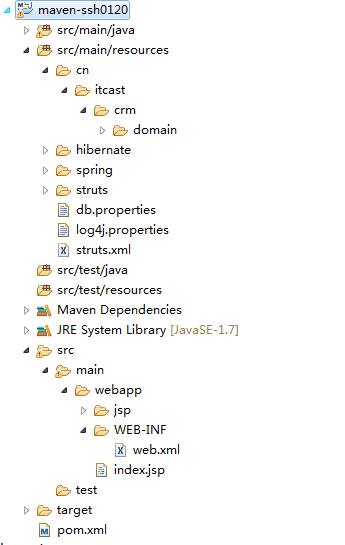
## pom.xml（重点）

加入各层依赖jar包：

1）锁定依赖构件的版本

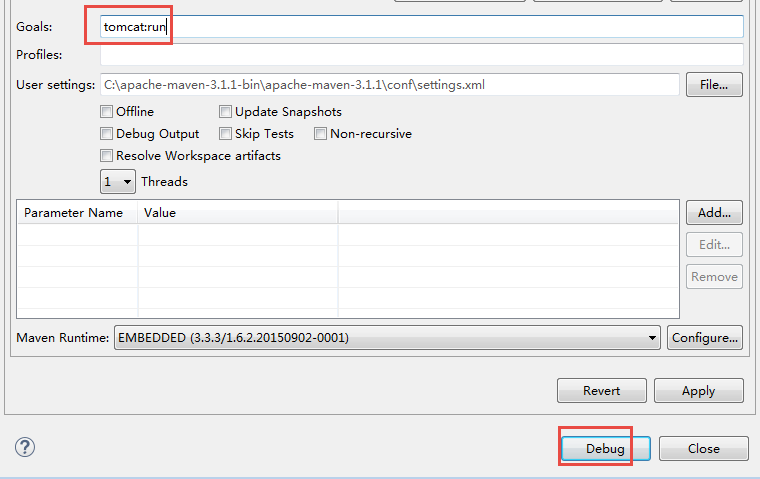
2）添加依赖

## 工程结构

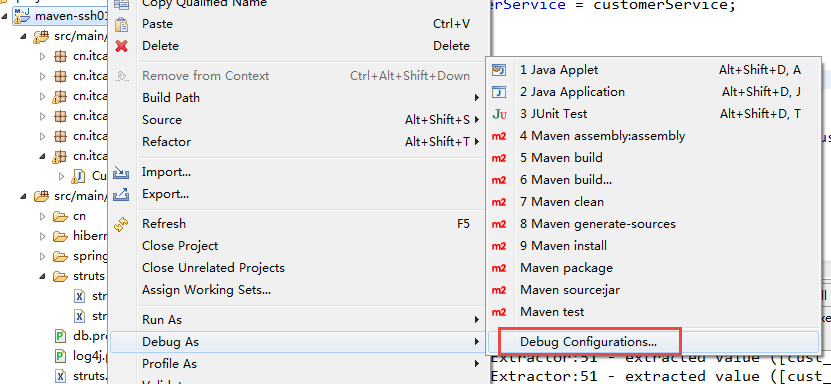


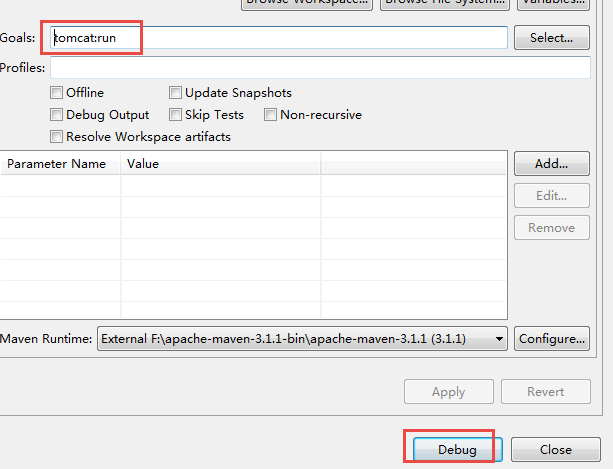
## 运行web工程

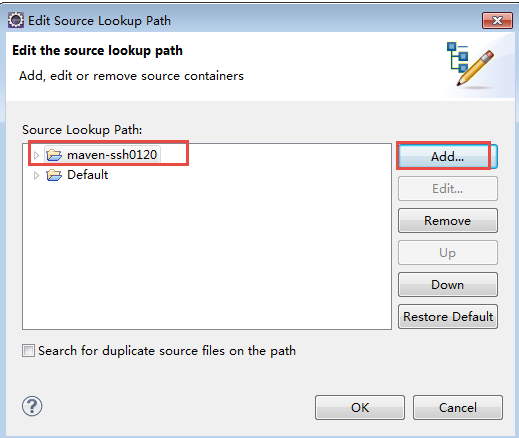
tomcat:run：使用tomcat6和maven集成插件运行。



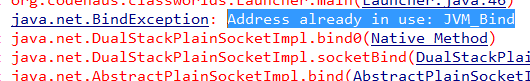
## 断点调试







## 端口占用



需要将tomcat全部关闭。

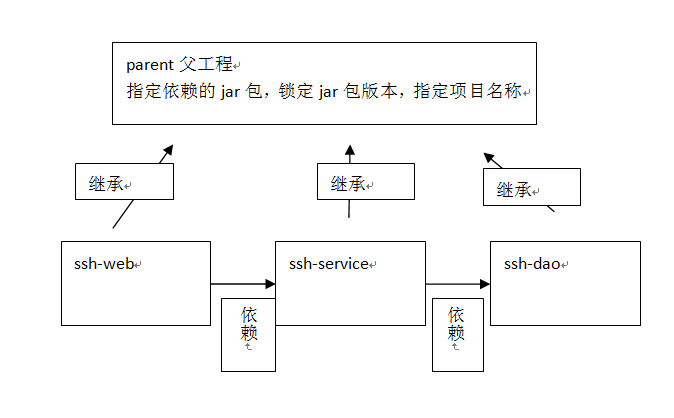
# 分模块构建工程（掌握）

## 需求

一个工程非常大的时候，多人协助开发。

为了提高开发效率，将整个工程拆分成多个模块工程，每个团队开发，开发完成将工程打成jar包发布仓库中，其它团队成员连接仓库下载其它模块的jar（class），只需要编译本模块的clas即可。

## 分模块构建结构



1、创建父工程

不需要编写java代码，只需要在pom.xml中定义依赖（锁定版本），指定坐标（项目名称、版本、模块名称）

作用：

1）将子工程中共同的内容统一定义

2）运行时候自动识别模块之间的依赖，聚合成一个统一工程去运行。

2、子工程包括：

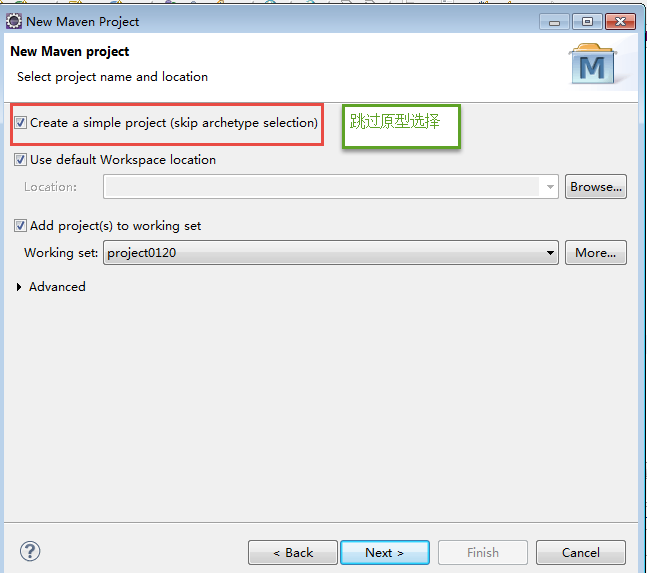
ssh-dao：dao接口、dao层所用到的配置文件

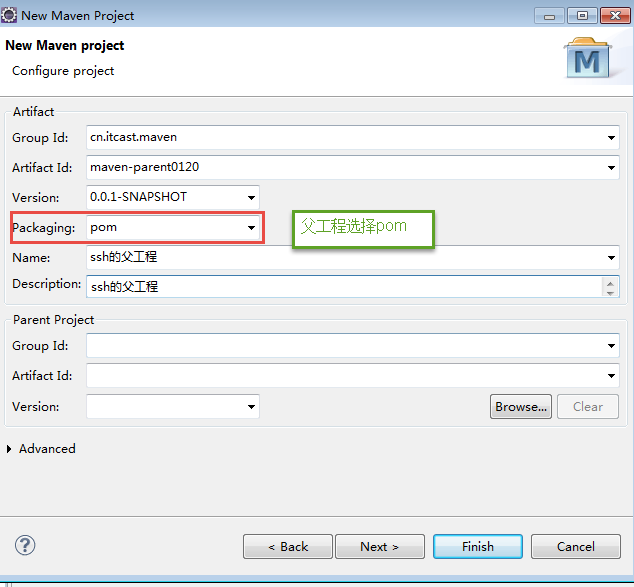
ssh-service：service接口，service层所用到的配置文件

ssh-web：action，struts2相关配置。

## 创建父工程

### 创建父工程





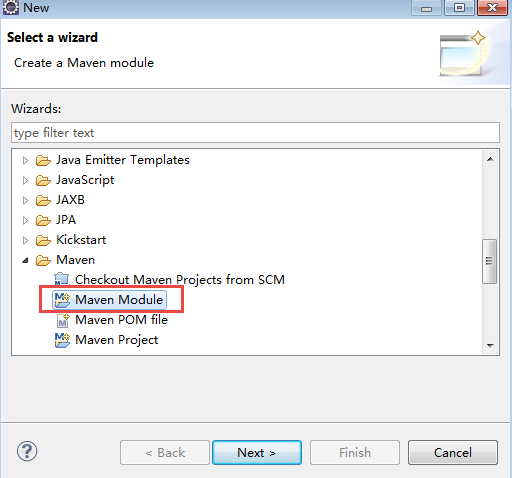
### 定义pom

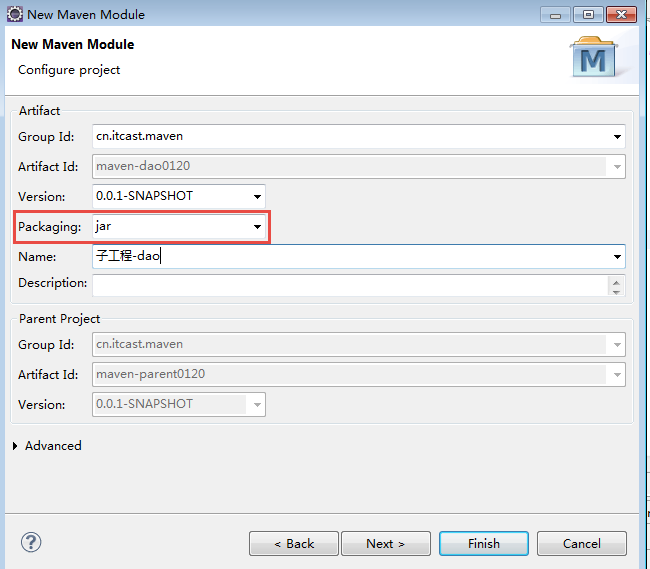
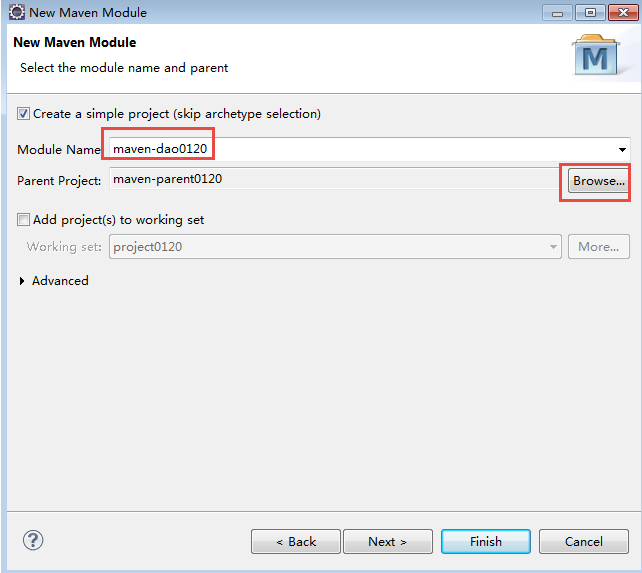
锁定依赖版本

设置编译版本

## 创建子工程-dao

### 创建maven模块 dao





### pom.xml

在子工程中继承父工程：

<parent>

<artifactId>maven-parent0120</artifactId>

<groupId>cn.itcast.maven</groupId>

<version>0.0.1-SNAPSHOT</version>

</parent>

在父工程中包括：modules，通过modules实现聚合!

<modules>

<module>maven-dao0120</module>

</modules>

在子工程的pom.xml中添加dao所依赖的jar包。

### 单元测试

@RunWith(SpringJUnit4ClassRunner.**class**)

@ContextConfiguration(locations={"classpath:spring/applicationContext.xml","classpath:spring/applicationContext-\*.xml"})

**public** **class** CustomerDaoImplTest {

@Autowired

**private** CustomerDao customerDao;

@Before

**public** **void** setUp() **throws** Exception {

}

@Test

**public** **void** testFindCustomerById() {

CstCustomer customer = customerDao.findCustomerById(1l);

System.*out*.println(customer);

}

}

### 将dao发布到仓库

mvn install

## 创建子工程-service

### 创建子工程

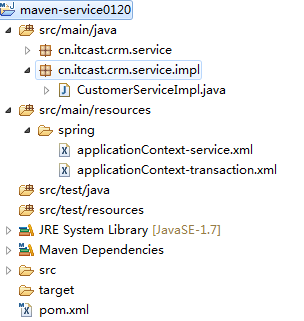
选择model module，同dao子工程。

### pom.xml

依赖dao。



### service接口



### 单元测试

@RunWith(SpringJUnit4ClassRunner.**class**)

//classpath:后边加\*表示 加载jar中的配置文件及本工程classpath下的配置文件

@ContextConfiguration(locations={"classpath\*:spring/applicationContext.xml","classpath\*:spring/applicationContext-\*.xml"})

**public** **class** CustomerServiceImplTest {

@Autowired

**private** CustomerService customerService;

@Before

**public** **void** setUp() **throws** Exception {

}

@Test

**public** **void** testFindCustomerById() {

CstCustomer customer = customerService.findCustomerById(1l);

System.*out*.println(customer);

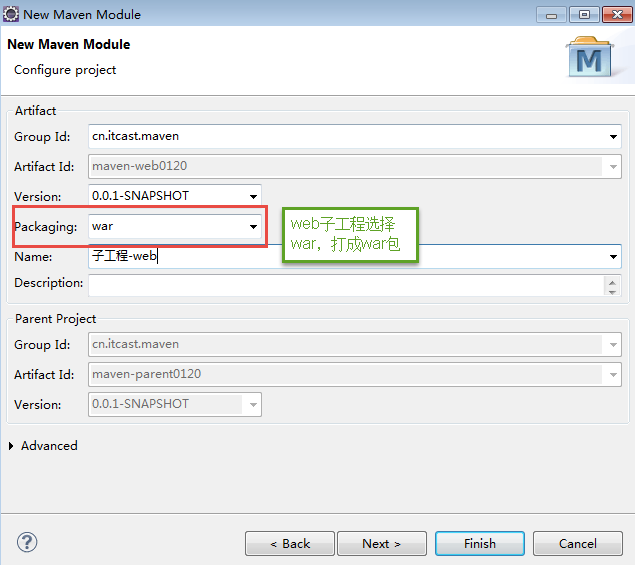
}

}

## 创建子工程-web

### 创建子工程

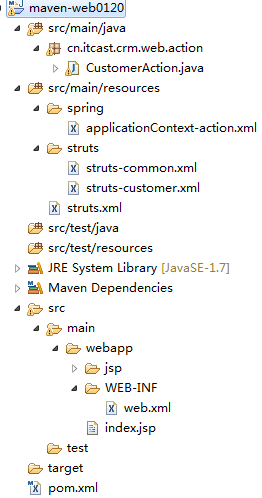
创建maven module，方法同dao和service.



### pom.xml

依赖service

### 工程构建



## 继承和聚合配置

在子工程继承父工程。

<parent>

<artifactId>maven-parent0120</artifactId>

<groupId>cn.itcast.maven</groupId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<!-- 由于子工程继承父工程，在子工程中不用定义项目名称和版本，和父工程使用相同 的项目名称和版本 -->

<!-- <groupId>cn.itcast.maven</groupId> -->

<artifactId>maven-web0120</artifactId>

<!-- <version>0.0.1-SNAPSHOT</version> -->

在父工程中定义modules：

<!-- 聚合各各子工程模块 -->

<modules>

<!--

定义的内容：相对于父工程pom.xml找子工程pom.xml文件的路径

-->

<module>maven-dao0120</module>

<module>maven-service0120</module>

<module>maven-web0120</module>

</modules>

## 运行调试

### 方法1：

通过maven-parent0120父工程运行，先将工程聚合到一起，再去运行maven-web子工程（打成war）

### 方法2：

通过maven-web0120工程运行，自动将service、dao的jar依赖到自己的工程。

### 问题

找不到service bean,需要修改web.xml中加载spring配置：classpath\*

<context-param>

<param-name>contextConfigLocation</param-name>

<!-- spring配置文件 可以采用加载多个文件，中间以半角逗号分隔，也可以使用通配符号 -->

<param-value>classpath\*:spring/applicationContext.xml,classpath\*:spring/applicationContext-\*.xml</param-value>

</context-param>

# tomcat插件

提供tomcat6和tomcat7：

区别：

tomcat6和tomcat7的插件包不一样

运行方式不一样：

tomcat6：tomcat:run

tomcat7：tomcat7:run

在maven-web工程中配置tomcat插件：

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<configuration>

<!-- 指定端口 -->

<port>8081</port>

<!-- 请求路径 -->

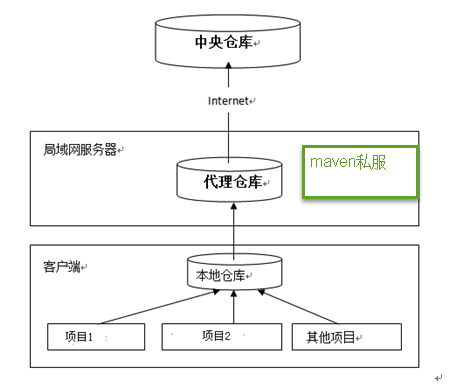
<path>/</path>

</configuration>

</plugin>

# maven私服(了解)

## 私服部署结构



在公司的局域网内部署一台服务器，服务器上安装maven私服软件。

服务器上存储了maven仓库。

在私服环境下本地项目找jar流程是：

１、先找本地仓库

2、如果本地仓库没有从私服找

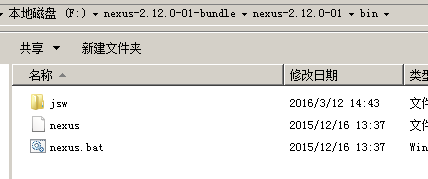
3、如果私服没有从中央仓库下载，下载成功将jar向私服和本地仓库都存一份。

## 安装私服

下载Nexus， 下载地址：http://www.sonatype.org/nexus/archived/

下载：nexus-2.12.0-01-bundle.zip

解压nexus-2.12.0-01-bundle.zip，本教程将它解压在F盘，进入bin目录：



cmd进入bin目录，执行nexus.bat install



## 卸载nexus

cmd进入nexus的bin目录，执行：nexus.bat uninstall

## 运行nexus

方法1：

cmd进入bin目录，执行nexus.bat start

方法2：

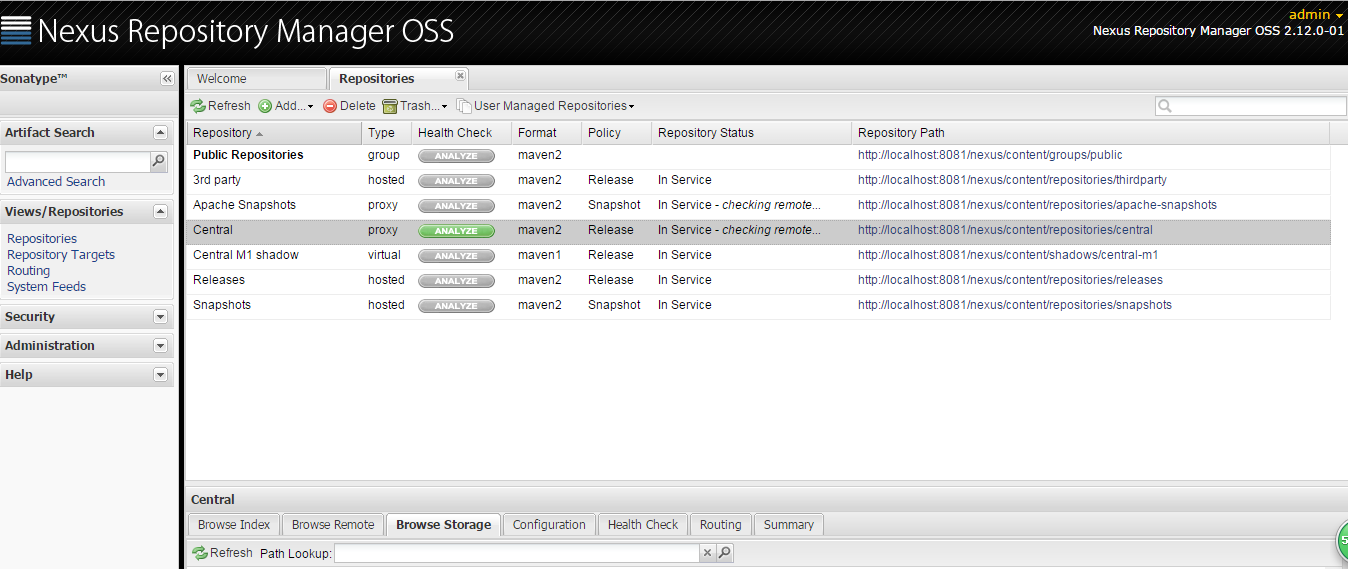
直接启动nexus服务



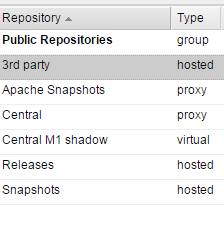
查看nexus的配置文件conf/nexus.properties

默认用户名和密码 ：

admin/admin123



## 仓库类型



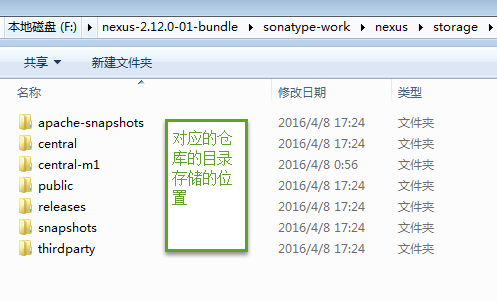
1、hosted:宿主仓库，自己项目的jar要放到hosted类型的仓库中。

2、proxy：代理仓库，代理中央仓库

3、virtual：虚拟仓库，存储了maven1的构件，一般不用。

4、group:仓库组，将一些仓库组成一个组，自己项目连接仓库组去下载jar包。

平常开发中都去连接仓库组，仓库组中包括：hosted、proxy等。。



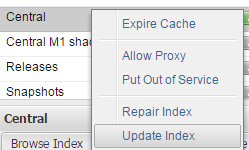
## 使用nexus

### 在nexus搜索jar包

在nexus加入jar包索引。

方法1：

连网下载

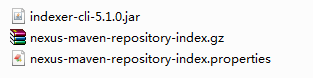


方法2：

手动下载索引



从网上下载：indexer-cli-5.1.0.jar

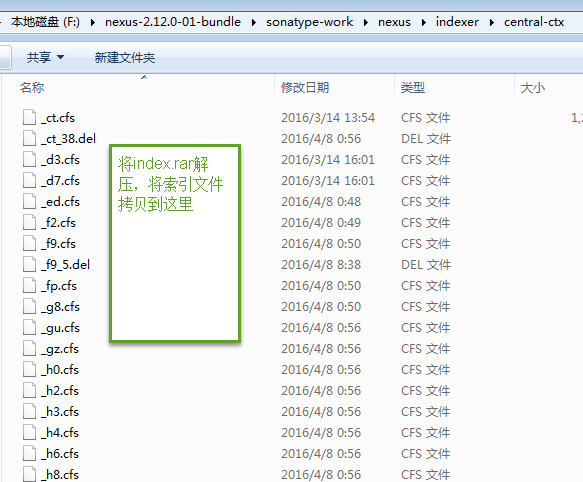


将这三者放在一个文件夹，cmd执行：

java -jar indexer-cli-5.1.0.jar -u nexus-maven-repository-index.gz -d indexer

生成索引目录：





### 将项目发布到私服（掌握）

1、在自己电脑的maven的setting.xml中配置

<server>

releases对应nexus的仓库名称

<id>releases</id>

<username>admin</username>

<password>admin123</password>

</server>

<server>

snapshots对应nexus的仓库名称

<id>snapshots</id>

<username>admin</username>

<password>admin123</password>

</server>

2、配置项目pom.xml

<distributionManagement>

<repository>

<id>releases</id>

<url>http://localhost:8081/nexus/content/repositories/releases/</url>

</repository>

<snapshotRepository>

<id>snapshots</id>

<url>http://localhost:8081/nexus/content/repositories/snapshots/</url>

</snapshotRepository>

</distributionManagement>

注意：pom.xml这里<id> 和 settings.xml 配置 <id> 对应！

3、准备将自己工程的jar发布到nexus私服中。

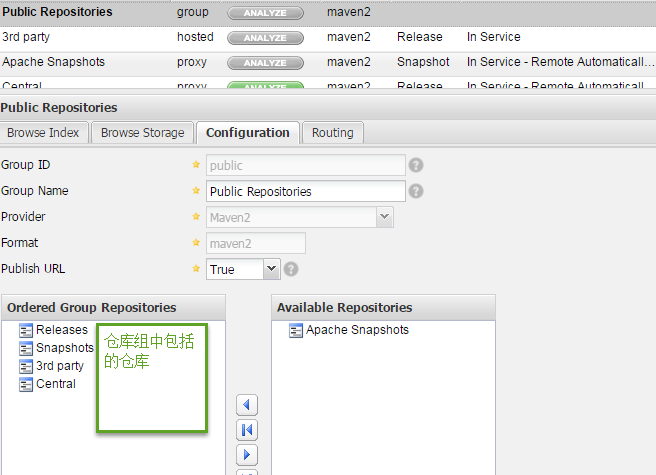
当你的工程版本是releases会发布到私服的releases仓库

当你的工程版本是snapshots会发布到私服的snapshots仓库

执行deploy

### 从私服下载jar包

#### 配置仓库组：



仓库组地址：

http://localhost:8081/nexus/content/groups/public/

#### 配置镜像

镜像地址为nexus仓库组的地址

在setting.xml配置镜像

<mirror>

<!--镜像id，mirrors中可以配置多个镜像，保证id不重复-->

<id>nexus</id>

<!--mirrorOf指定仓库id，仓库组包括了id为 releases、snapshots、thirdparty 、central 的仓库，可以配置具体的id，如果镜像所有的仓库则设置为\*-->

<mirrorOf>\*</mirrorOf>

<!--镜像地址-->

<url>http://localhost:8081/nexus/content/groups/public/</url>

</mirror>

本地项目不连中央仓库，连接镜像（相当于连接私服）