# dependencyManagement使用

## 作版本号管理使用

1、dependencyManagement是声明而不引入依赖--------因此子项目需要显式的声明需要用的依赖（不需要版本号）

2、子项目中引用一个依赖而无版本号时，会沿着父子层次向上走，直到找到一个拥有dependencyManagement元素的项目，使用其版本号

## 用来实现多继承

1、maven只能单继承，即一个项目只能使用parent标签定一个父级项目

2、使用dependencyManagement并设置其scope=import，即表示继承此项目为父项目

# maven版本

## SNAPSHOT

SNAPSHOT---快照版本：以-SNAPSHOT为结尾的版本号（是一系列的版本的集合），随时更新不稳定的---每个版本都只是特定时间点的快照

例如：A-->B-1.3.8-SNAPSHOT（理解为A依赖了B的1.3.8-SNAPSHOT版本），那么B-1.3.8-SNAPSHOT更新之后重新deploy到仓库之后，A只 需要重新构建就可以拿到最新的代码，不用修改依赖B的版本。这样达到了变更传达的透明性。

SNAPSHOT的不稳定性，带来风险----本地仓库中快照版本的依赖的目录下会看到带有时间戳的jar包

## RELEASE

RELEASE---发布（正式）版本：所有非-SNAPSHOT结尾的版本号，是稳定的版本号。---应该一旦发布永远不变。-----有的仓库会配置成redeploy覆盖（挖坑专家）

# maven仓库

---------maven仓库，就是放置所有的jar文件的地方

## 本地仓库

本地环境安装maven后，会生成一个本地仓库位置（.m2/respository/ ），可在maven的配置文件中更改此位置（建议更改）

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0

http://maven.apache.org/xsd/settings-1.0.0.xsd">

<localRepository>D:/mvn</localRepository>

</settings>

## 远程仓库和中央仓库

当maven要查找构件时，发现本地仓库中没有，则需要从远程仓库下载构件到本地仓库，供项目使用。

如下面，就使用阿里云仓库：

<mirrors>

<mirror> <id>alimaven</id>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

<mirrorOf>central</mirrorOf>

</mirror>

</mirrors>

当我们不配置远程仓库时，maven会默认一个仓库，即maven官方的仓库，称之中央仓库

 URL：<http://search.maven.org/#browse>

## Maven 依赖搜索顺序

当我们执行 Maven 构建命令时，Maven 开始按照以下顺序查找依赖的库：

### 依赖是具体的发布版本

* **步骤 1**－ 在本地仓库中搜索，找到则成功。
* **步骤 2** － 在远程仓库中搜索，找到则下载。
* **步骤 3** － 如果没有设置远程仓库，Maven 默认去中央仓库搜索，找到则下载。
* **步骤 4** － 在一个或多个远程仓库中搜索依赖的文件，如果找到则下载到本地仓库以备将来引用，否则 Maven 将停止处理并抛出错误（无法找到依赖的文件）。

### 依赖是快照版本（RELEASE或LATEST）

* **步骤 1  -**基于更新策略更新（updatePolicy= always），则总是尝试去远程仓库拉取最新版本
* **步骤 2 -** 强制快照更新------mvn clean install-U

## 私服nexus的使用

### 源码安装方式搭建

因nexus比较消耗资源，linux环境需要提供至少2G以上内存

1、下载源码包

https://www.sonatype.com/download-oss-sonatype下选择unix版本

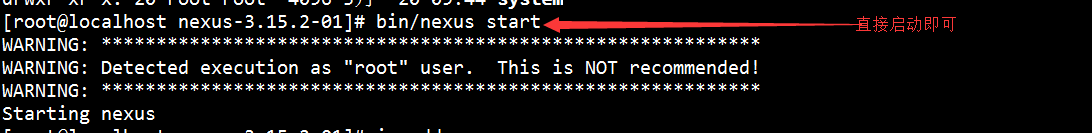
或者到peter的云盘（https://pan.baidu.com/s/1OjuyfgdIgR49rwnrzC-J5g 提取码：ax7t ）

2、解压

tar -zxvf nexus-3.15.2-01-unix.tar.gz -C nexus

3、设置用户并启动





### docker方式安装

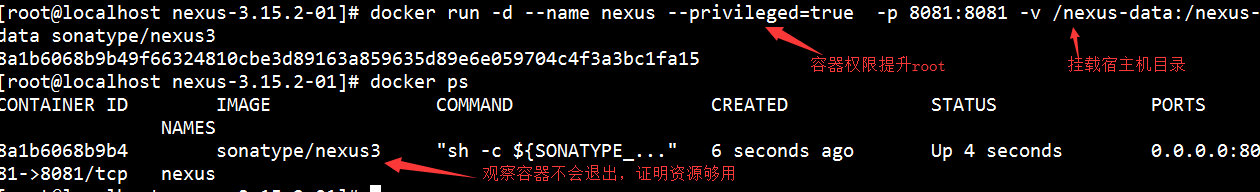
1、创建一个持久目录，存放nexus数据

mkdir /nexus-data

chown 200 /nexus-data ##此处200对应容器内nexus用户的id

2、docker命令直接启动（网速慢的建议先docker pull sonatype/nexus3）

docker run -d --name nexus --privileged=true -p 8081:8081 -v /nexus-data:/nexus-data sonatype/nexus3

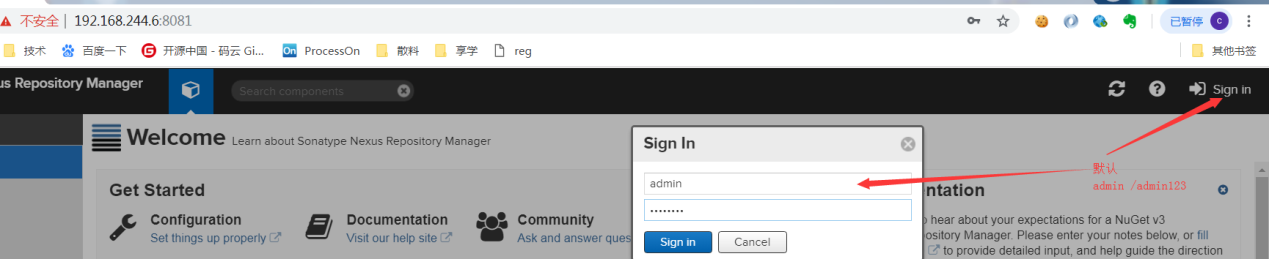


### nexus概念与使用

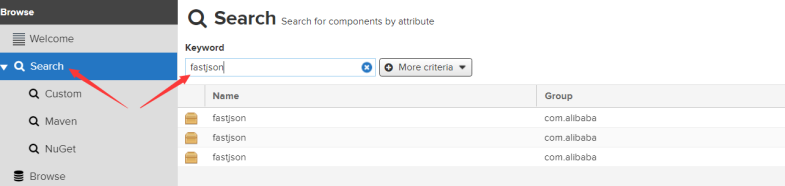
#### 登陆

打开nexus地址如：http://192.168.244.6:8081/

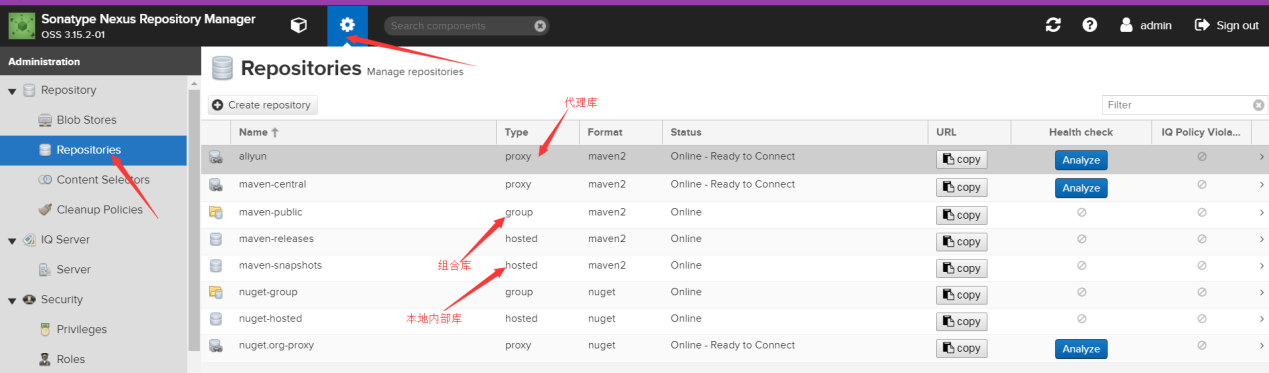
使用admin/admin123登陆



#### 搜索



#### Nexus的仓库类型



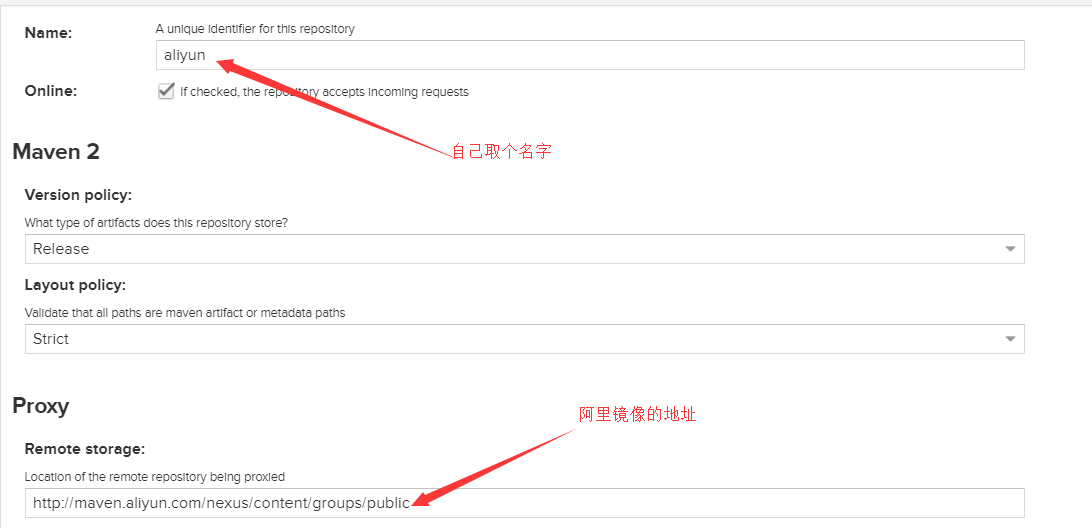
group(仓库组类型--门面)：组合自己的多个库，成为一个路径对外提供服务；  
hosted(宿主类型)：内部项目的发布仓库，内部jar存在在这；  
proxy(代理类型)：代理一个远程仓库url，缓存一份打此经过的jar

~~nuget库是 NET库~~

#### 配置国内代理加速

--------因maven中央仓库速度慢，我们一般优先使用国内远程仓库，这里配一个aliyun代理

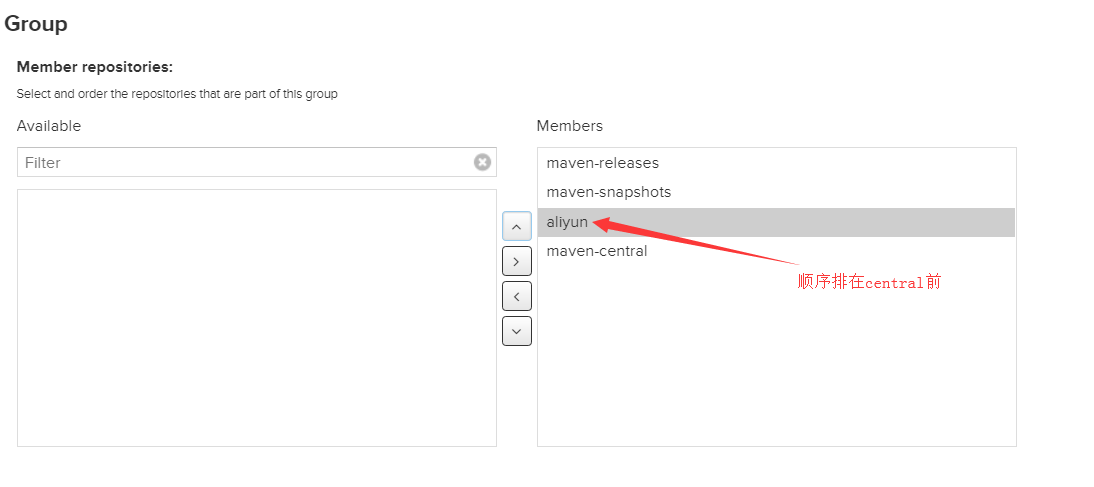
点击create repository按钮，填写name与remote storage值



#### 调配nexus对外的门面

------------当外部请求nexus下载jar时，nexus统一对外部提供一个服务窗口。我们配置它对远程仓库的代理顺序。

编辑maven-public库，加入aliyum远程仓库

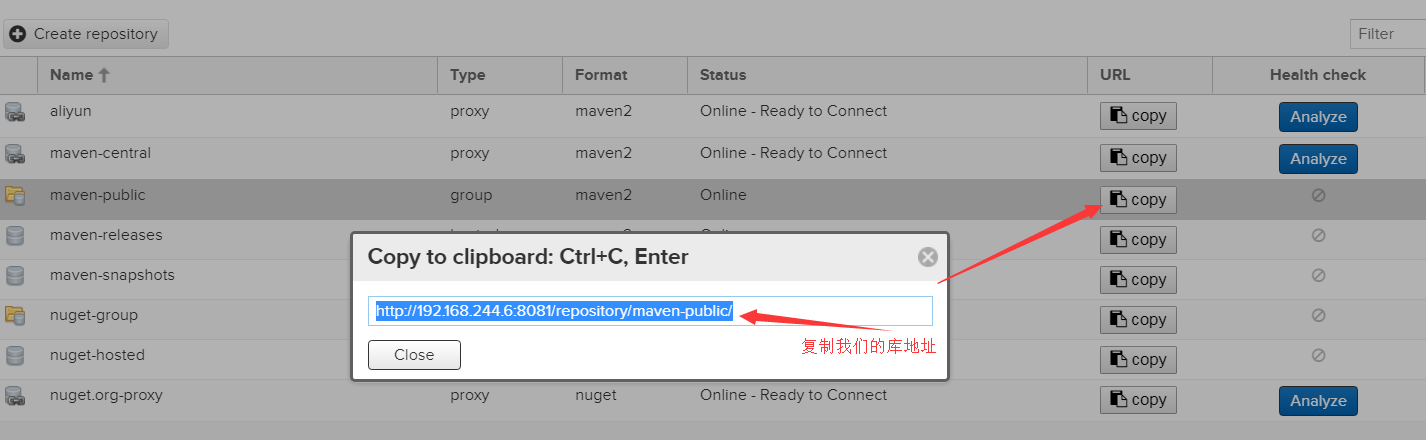


当查询构件时，会依次查releases/napshots/aliyun/central这四个库：

优先级：本地最优先，aliyun次之，central最末（因其速度最慢）

#### 使用nexus拉取jar包

我们的nexus对外服务地址url，如下图



配置此url有两种方式：

##### 1、全局配置方式

在maven的setting文件里，通过mirror指定私服地址，此方式对全局生效



##### 2、局部配置方式

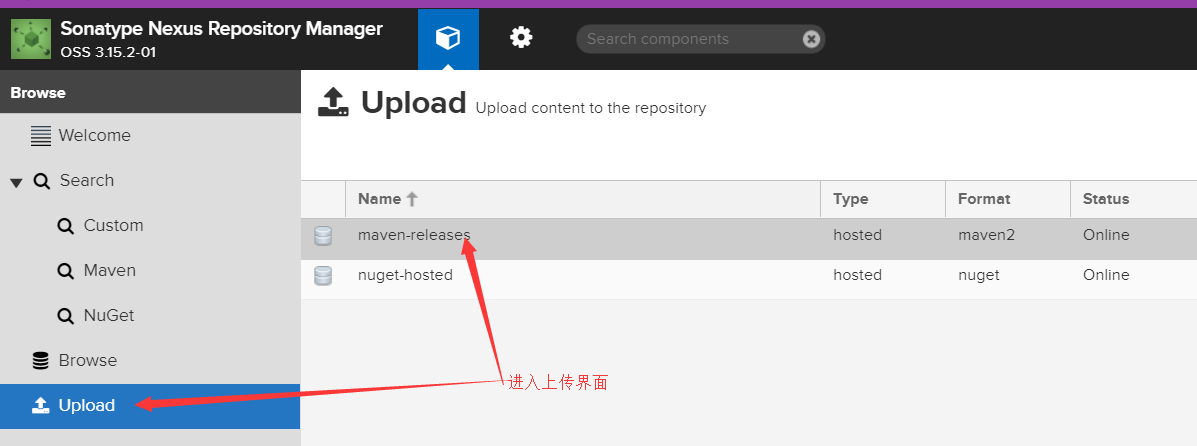
在本项目的pom里，指定repository配置，此方式只对本项目生效



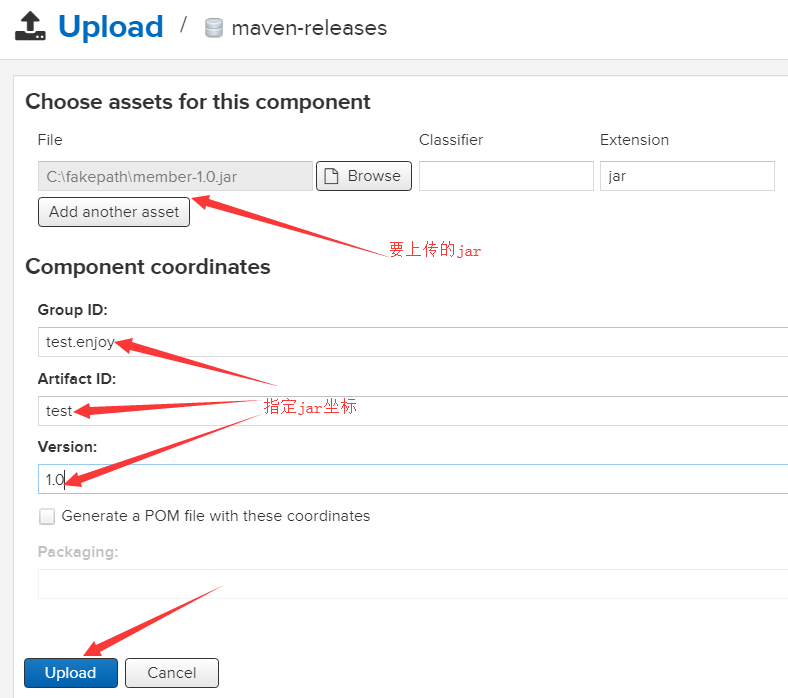
#### 上传jar到nexus

##### 1、手动上传

直接在nexus的界面上，使用upload上传



填写jar与坐标



在browse界面里，可查到上传的jar



##### 2、命令上传

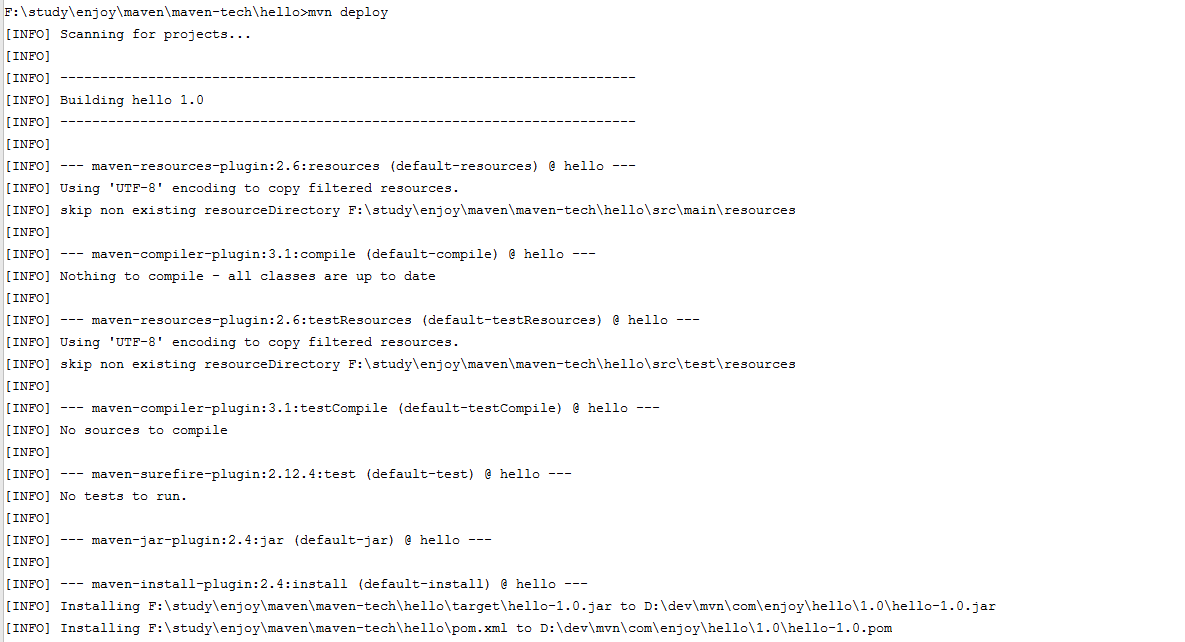
命令方式上传，需要权限认证，在setting文件中配置



在项目pom中，配置上传url

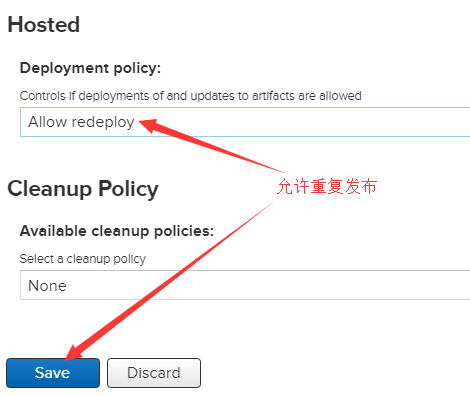


mvn deploy命令直接上传





设置允许覆盖发布



# maven自定义项目骨架

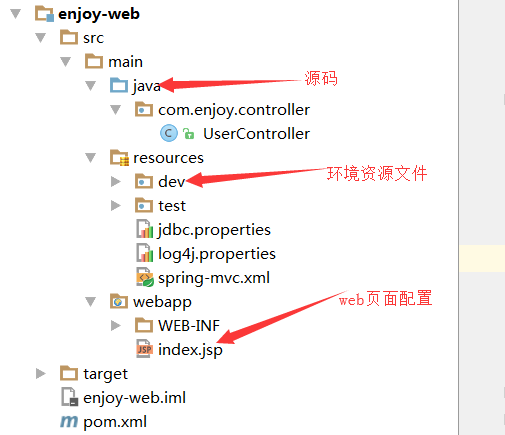
--------自定义符合我们自己需求的archetype

前面讲过通过Maven原型maven-archetype-archetype来创建自定义Archetype的方法。此插件是maven官方的项目结构，不一定符合我们的企业本义

下面，我们自已定义一种项目结构模式，并将其打成标准包，供企业整体员工，后续使用

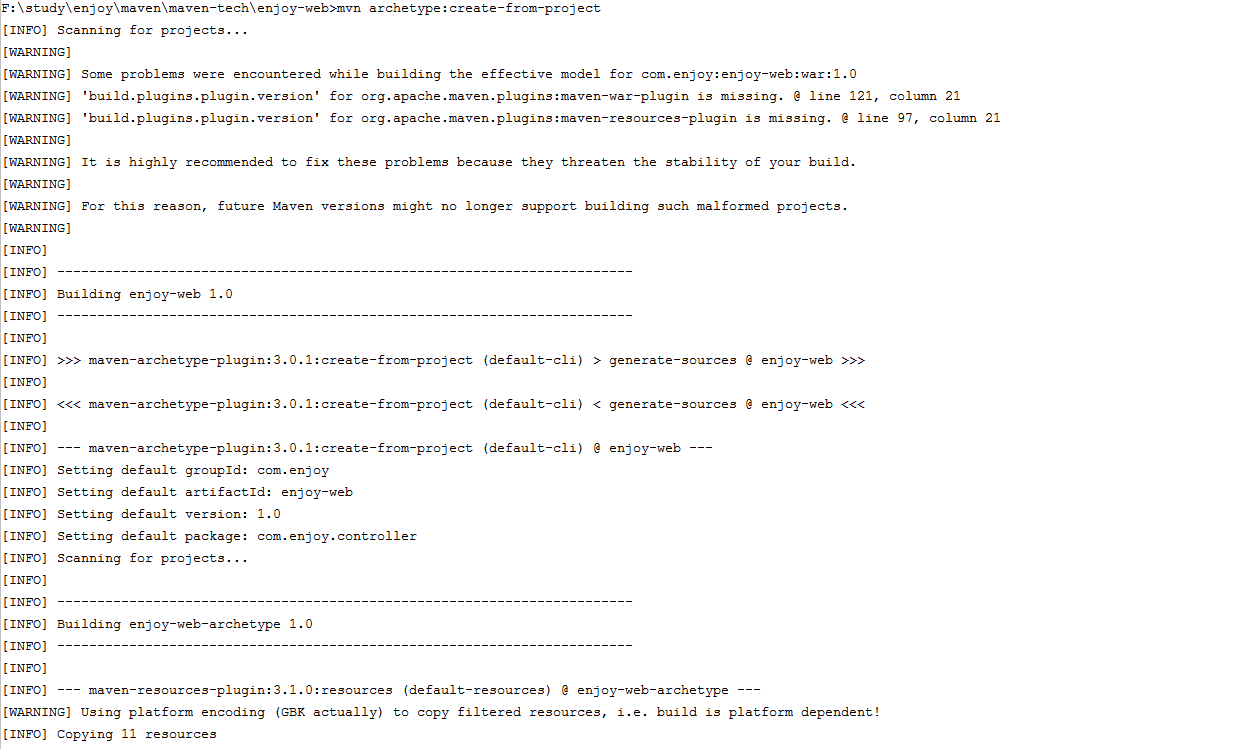
## 选一个标准项目结构

我们选用一个标准的项目结构框架---如，我们前面的enjoy-web项目，结构如下：

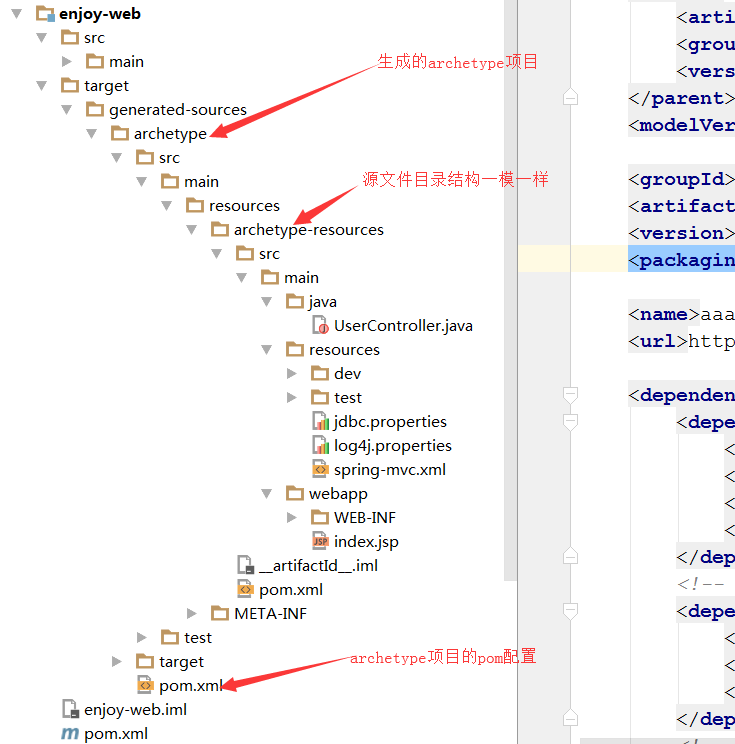


## 生成archetype项目

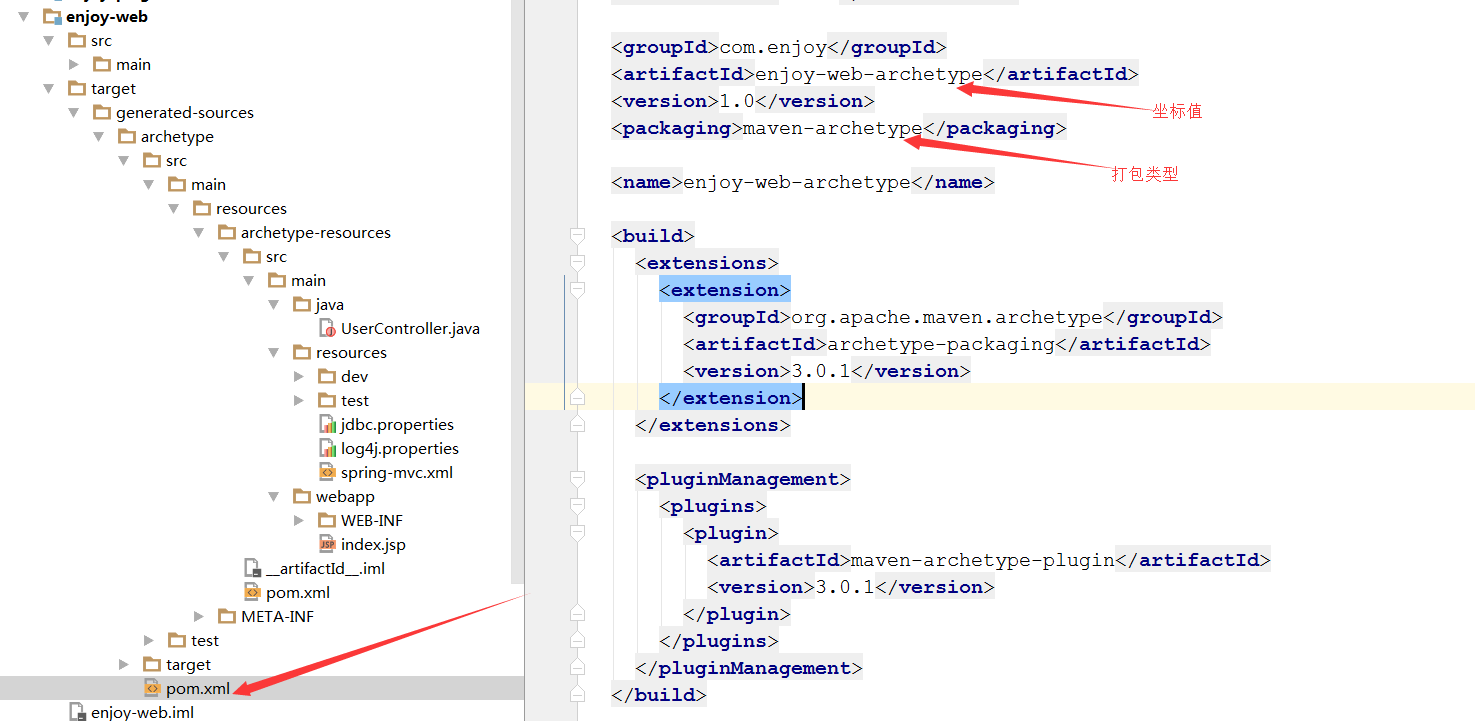
在选定的项目根目录下，执行命令：mvn archetype:create-from-project



执行完毕后，我target目录下，查看生成的archetype项目结构



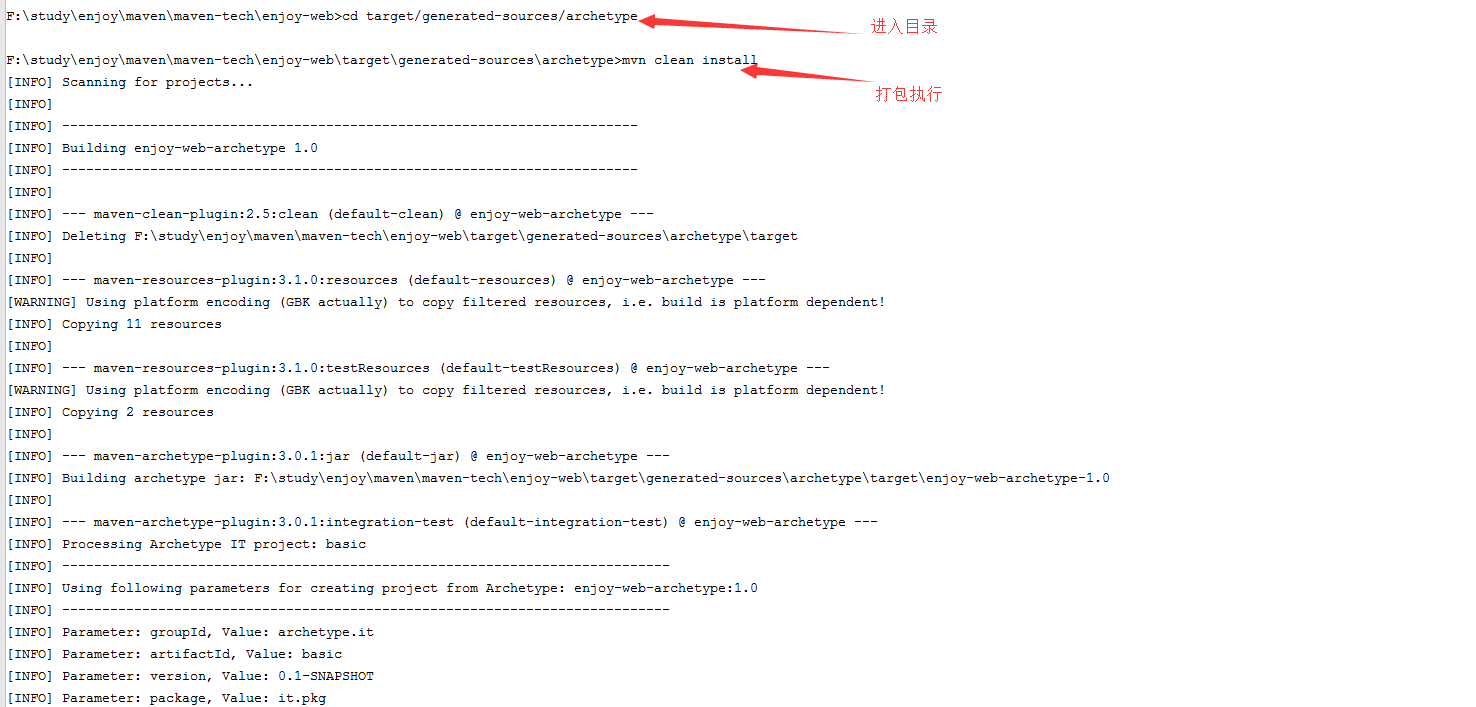
再看此项目的pom配置



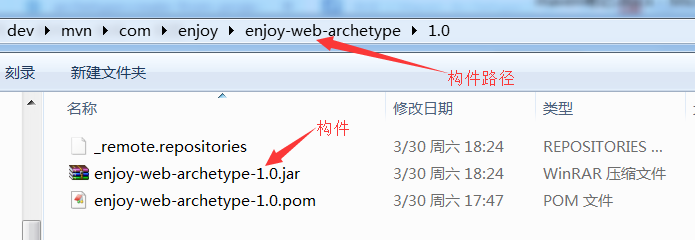
## 打包archetype插件并安装

此archetype项目也是一个maven工程，一样有构建/安装/发布等动作

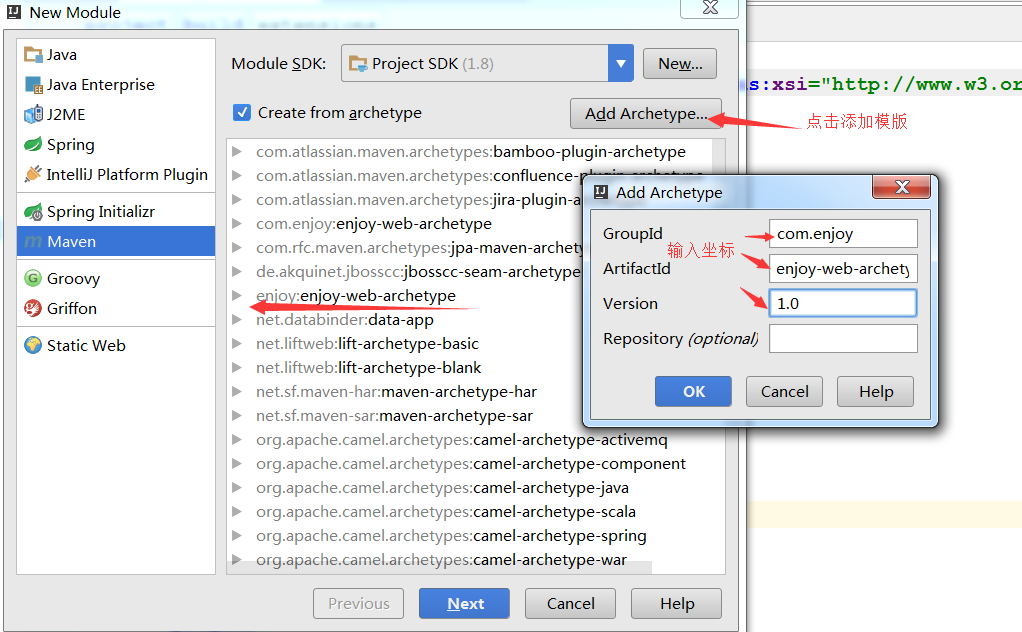
进入项目根目录，执行命令：mvn clean install



运行完毕，查看本地仓库，已经有了此构件包



## IDE工具加载archetype，新建工程



以后建立工程时，选用此插件，即可得到与enjoy-web一样结构的项目了，如：

