目录

[Maven学习笔记 2](#_Toc769)

[环境搭建 2](#_Toc18410)

[Maven 命令 3](#_Toc7749)

[Maven自动建立目录骨架 3](#_Toc28447)

[本地仓库，远程仓库，镜像仓库 3](#_Toc30528)

[Maven的生命周期和插件 4](#_Toc19327)

[Maven的生命周期 4](#_Toc13005)

[Pom文件详解 5](#_Toc4851)

[依赖范围 6](#_Toc10689)

[依赖传递 6](#_Toc4933)

[依赖冲突 7](#_Toc4578)

[Maven聚合和继承 7](#_Toc21950)

# Maven学习笔记

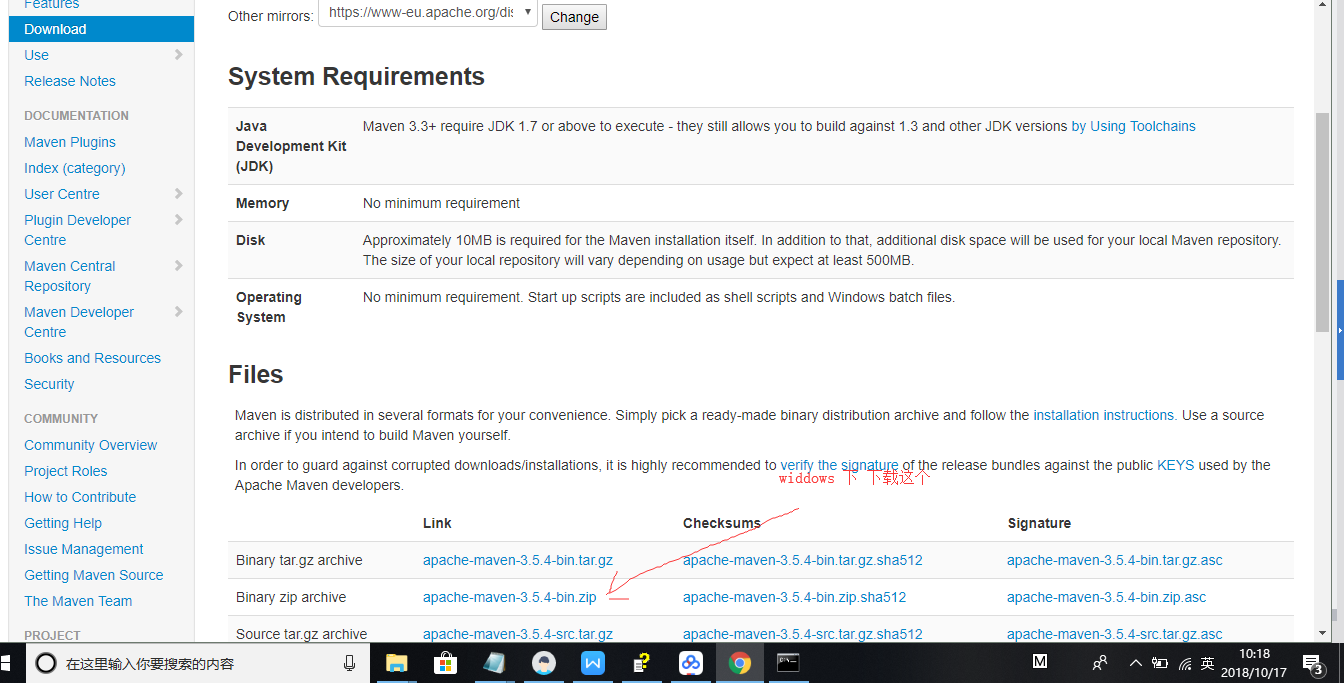
所谓的依赖（jar文件） 实际上也是 通过项目，打包成jar文件得来的。无非就是引用这个项目中的字节码（类）的方法。

所以 自己的maven项目，也可以 让另一个maven项目直接用依赖引入。这样就可以使用另外一个项目的类了

## 环境搭建

mvnrepository.com ： maven依赖搜索

官网 <http://maven.apache.org/> maven需要java支持



.bat 文件 一般在 bin 目录下 ，用 windows 命令行 可以运行这些脚本。

比如 tomcat中的 startup.bat ： 进入这个目录，输入startup就可以运行。或者双击运行。

环境变量设置， 在path 下 添加 maven安装目录/bin

或者 定义一个 系统变量 M2\_HOME 值为 maven安装目录，在

Path中 这样引用， %M2\_HOME%/bin

mvn -v : 查看maven 版本号

Maven

Pom文件 中

Groupid 是 项目的包名 com.imooc.maven01 : com.imooc 是组织（公司名）,maven01项目名

artifactId 是模块名 ：maven01-model （规范 项目名+ 模块名）

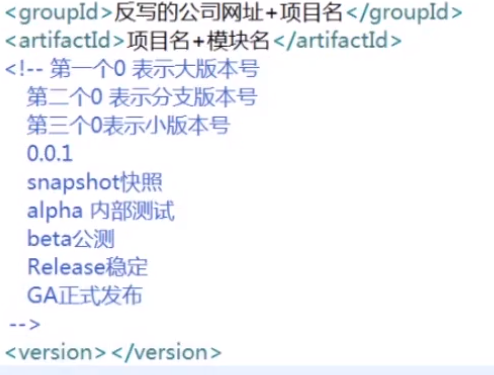
Version 是项目的版本

<groupId>cn.myusen</groupId>

<artifactId>chess</artifactId>

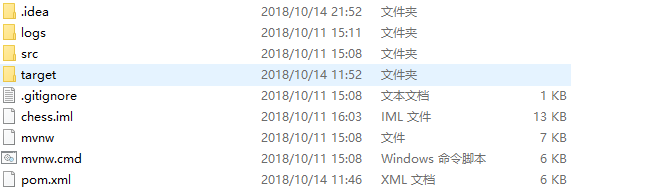
<version>1.0</version>

<packaging>jar</packaging>



## Maven 命令

命令 都在 maven 项目的根目录（下图就是一个根目录）下执行。用 windows cmd ，执行



mvn compile : 编译项目，根据依赖坐标到本地仓库中看是否有依赖，有的话，将依赖加入到项目的classpath中，没有，从远程中央仓库下载依赖到本地。（和插件），并在项目根目录生成 target文件夹， 生成 java字节码等 文件。

mvn test : 在 target目录中生成测试报告

mvn package： 在 target目录 下生成 项目的jar 或者 war 文件

mvn clean: 删除 target 目录

mvn install: 将jar包安装到本地仓库中

## Maven自动建立目录骨架

archetype 插件，用于创建符合maven规定的目录骨架

在 一个目录下（不是项目的根目录），执行

mvn archetype:generate 命令之后，一步一步 跟着提示创建骨架

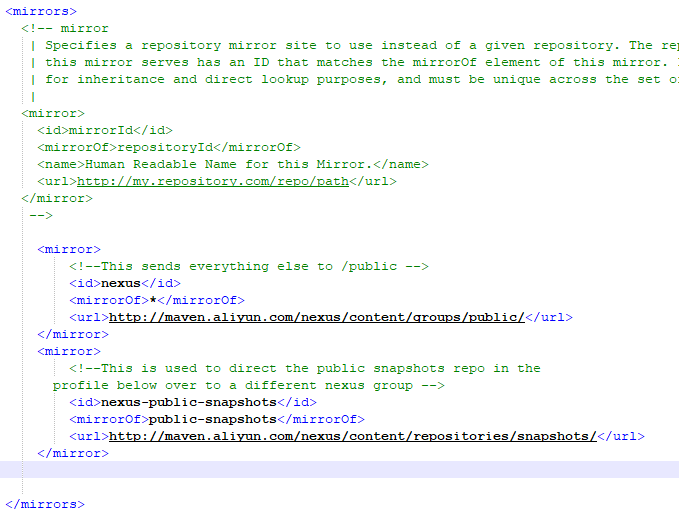
## 本地仓库，远程仓库，镜像仓库

远程仓库 ：在国外有个maven的全球仓库 （连接不到 国外网络的情况下，就无法下载依赖）

Maven的 lib目录下，有下图jar，解压之后，org\apache\maven\model目录下有个pom文件，url 是 国外 的 远程仓库

TIM截图20181017111040

镜像仓库：国内的 远程仓库。 这时候再下载依赖，就从国内远程仓库下载



本地仓库：默认 本地仓库 是在 用目录下的 .m2目录中

修改 本地仓库的 位置， 在maven的 conf 目录下的setting.xm文件

TIM截图20181017112254

修改之后 ，再从远程仓库下载，依赖就存放在 自己配置的 仓库路径中

## Maven的生命周期和插件

### Maven的生命周期

clean ：清理项目

default ：构建项目 (核心) compile test package 这几个阶段

site : 生成项目站点

Mvn 生命周期

clean compile test package install

执行 后面的命令，前面的命令都会执行

**插件**

Maven的插件很多，比如有 生成源码的插件 等等。。 具体用法查资料

## Pom文件详解

Maven

Pom文件 中

Groupid 是 项目的包名 com.imooc.maven01 : com.imooc 是组织（公司名）,maven01项目名

artifactId 是模块名 ：maven01-model （规范 项目名+ 模块名）

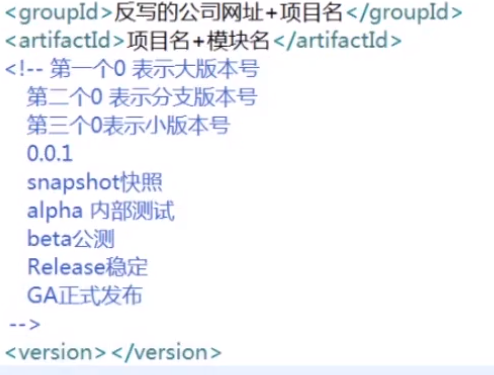
Version 是项目的版本

<groupId>cn.myusen</groupId>

<artifactId>chess</artifactId>

<version>1.0</version>

<packaging>jar</packaging>



<name></name> 项目描述名

<url></url> 项目的地址

<description></description> 项目的描述 .......

依赖管理，本身 不引入 管理的这些依赖，继承这个pom ，就可以将这些依赖引入项目的classpath

<dependencyManagement>

<dependcies>

................

</dependcies>

</dependencyManagement>

<parent></parent> 继承指定的父模块

聚合多个子模块，一起编译

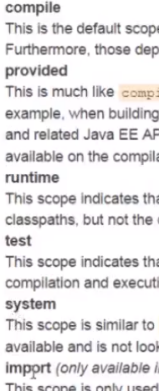
<modules>

<module></module>

.......

</modules>

### 依赖范围



Maven 的 3 中 classpath ：

编译 测试 运行

<scope>test</scope> 此依赖 只在 测试的 classpath

Socpe 的值

compile ：默认范围，编译，测试，运行都有效

Provided :在编译和测试时有效 ，运行时 不会将次依赖 加入 classpath

比如 发布 web项目是，servlet-api 就不必 加入到 classpath，因为tomcat有servlet-api

运行时，加入的话，可能存在冲突

runtime : 在测试和运行时有效

test ： 测试范围有效

### 依赖传递

ABC 实际上 是 项目的坐标

B 依赖 引用 A 依赖 ，C依赖 引用 B 依赖 ，

间接的 C 依赖 引用 A依赖

排除 依赖

比如 在 C依赖 中 将 A依赖 排除,就不再将 A的 jar包加入到classpath

<dependency>

<groupid>C</groupid>

.......

<exclusions>

<exclusion>

<groupid>A</groupid>

....

</exclusion>

</exclusions>

</dependency>

## 依赖冲突

1 依赖 有短路 优先的 特点，加入 A要 引入D依赖

1. >B->C->D (jar)

A->X->D(jar)

则 A 引入的依赖是 下面 的那个D

2 依赖 谁先 声明， 谁有效 。

## Maven聚合和继承

聚合：

将 多个 项目，一同 编译 ，打包

举例： 创建一个新的maven项目，pom文件中

打包方式 pom , 下面的标签实现聚合

<modules>

<module>

子模块的坐标

</module>

.......

</modules>

继承 ：

父 pom文件中 定义，这个标签（dependencyManagement）下面的依赖，不会引入到本项目中

<packaging>pom</packaging>

<dependencyManagement>

<dependcies>

................

</dependcies>

</dependencyManagement>

谁去继承它，就在 pom文件中用

<parent>坐标</parent> 引入就可以了