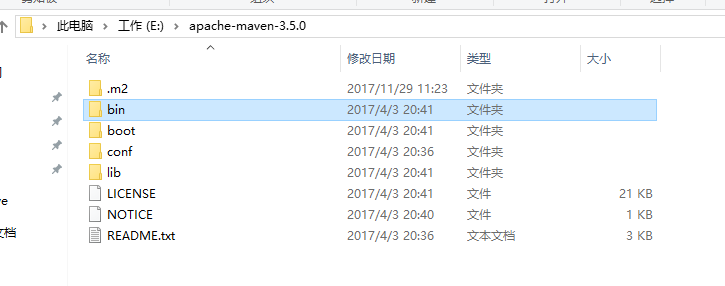
# Maven

## Maven目录结构

图1



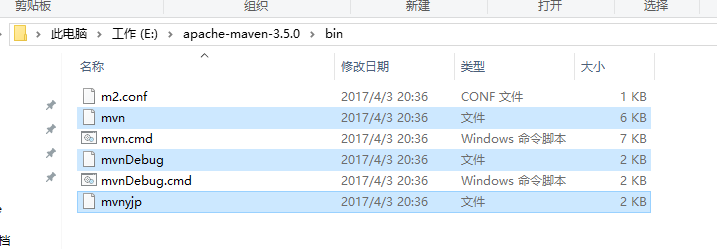
### 1.1.1 bin

没有后缀的就是.sh文件（基于UNIX平台的shell脚本）；

cmd或者bat后缀的就是Windows上执行的脚本；

m2.conf是classworlds的配置文件。

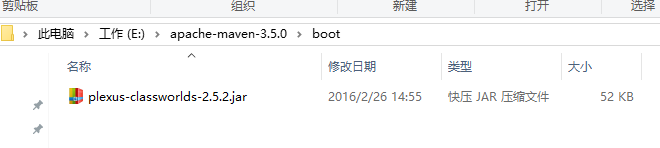
图1



### 1.1.2 boot

boot目录下只有一个jar包，它是一个类加载框架，Maven使用这个框架来加载自己的类库。

图1

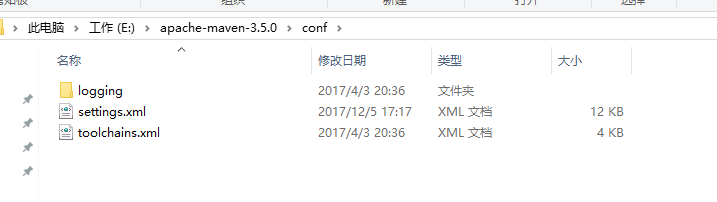


### 1.1.3 conf

conf目录下的settings.xml可以全局定制Maven的行为。

（注意：~/.m2/settings.xml（~是用户目录）可以用户范围定制Maven的行为，一般将conf下面的拷到~/.m2下面修改，这样升级maven的时候不需要重新修改conf下面的settings.xml）

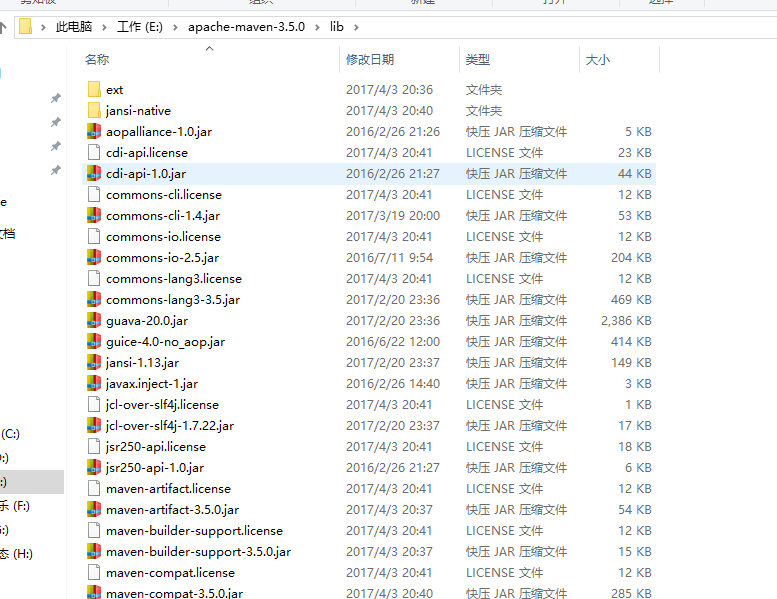
图1



### 1.1.4 lib

lib目录下是Maven运行时需要的Java类库。

图1



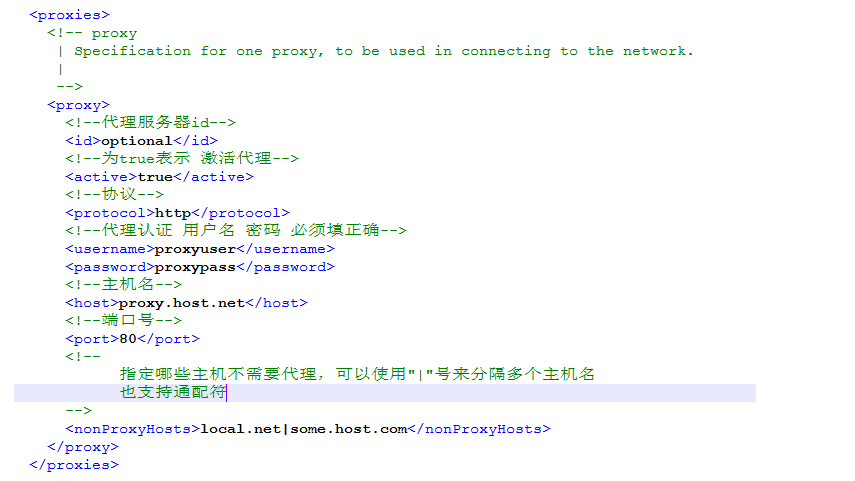
## 设置代理服务器

一般出于安全原本，本机->代理服务器->因特网，本机是无法访问因特网的。

设置代理服务器，修改settings.xml文件，如图1：

（注意：在配置了多个代理时，默认取第一个激活的代理）

图1



## Maven坐标

Maven坐标由groupId，artifactId，version组成。

groupId：一般是“公司类型.公司名称.实际项目名称”（实际项目，一个公司下有很多个实际项目），比如：com.googlecode.myapp。

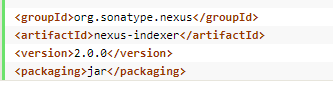
artifactId：一般是“实际项目-实际项目下的Maven项目”（Maven项目，隶属于某个实际项目），比如：myapp-util或myapp-domain等。

version：一般是版本号-快照/稳定版，比如：1.1-SNAPSHOT。

jar：打包方式，默认是jar。

（注意：一般jar包会以artifactId-version（快照/稳定标志可要可不要）.jar命名，比如：nexus-indexer-2.0.0.jar，其中groupId是org.sonatype.nexus，artifactId是nexus-indexer）

图1



## 依赖范围（scope）

Maven在编译，测试，运行会使用不同的classpath，共3套。

1.test：当scope为test时，只对测试有效，编译和执行阶段使用会报错，常见的是JUnit。

2.compile：默认是compile，对编译，测试，运行都有效。

（注意：主代码就是src/main/java，测试代码就是src/test/java）

3.provider：只对编译和测试有效，运行时使用会报错，常见的是servlet-api，运行时由容器提供。

4.runtime：只对测试和运行有效，编译时使用会报错，常见的是JDBC驱动的实现，编译时只需要JDBC驱动的接口，不需要实现。

5.system：和provider一样，只对编译和测试有效，但需要通过systemPath显示指定依赖文件的路径，往往与本机系统绑定导致不可移植，慎用，一般是Maven仓库没有的。

6.import：不对三套classpath造成实际影响。

## compile插件默认只编译Java 1.3

由于历史原因，Maven的compile插件默认只支持编译Java 1.3，如果要支持1.3以上版本，需要配置compile插件。

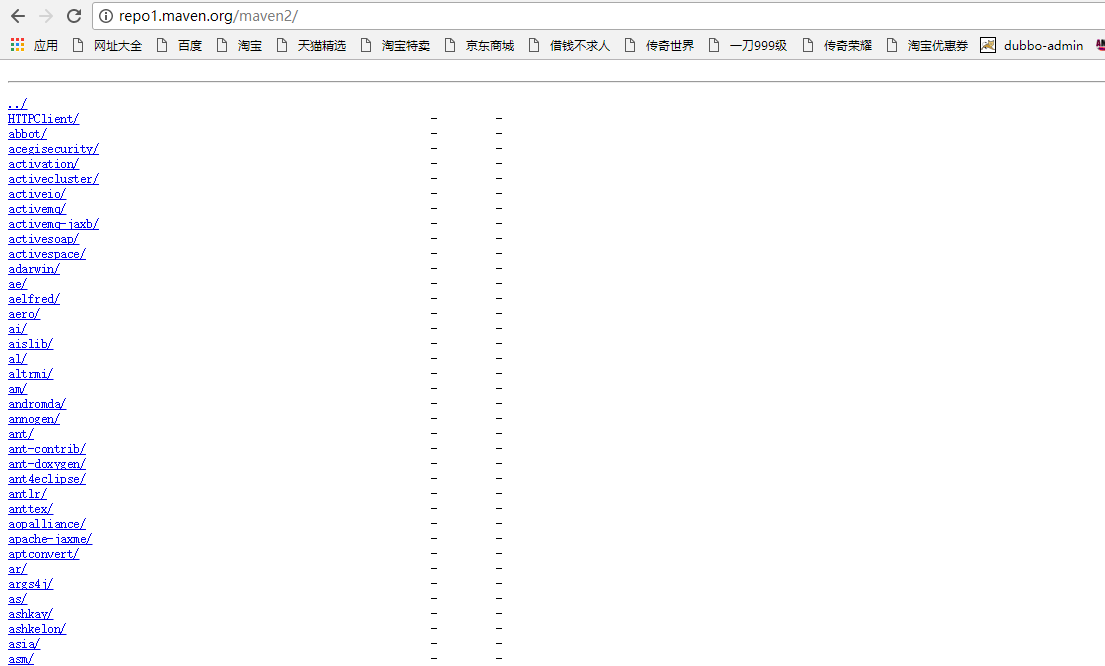
图1



## Maven中央仓库

Maven的中央仓库地址：<http://repo1.maven.org/maven2/>

图1



## 1.7 依赖配置

<dependencies>标签包含一个或多个<dependency>依赖。

图1



## 1.8 传递性依赖

## 1.9 继承

继承后，父pom中的大多数元素都能被子pom继承。

图1（能被子pom继承的元素）

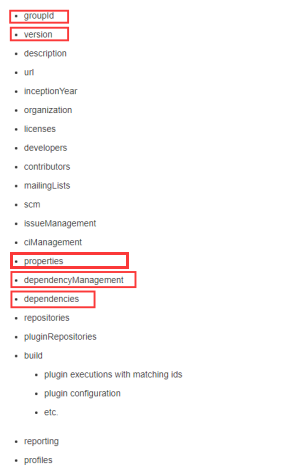


图2（不能被子pom继承的元素）

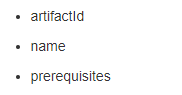


图3（使用parent标签继承父pom）



## 1.10 聚合

将多个子模块（module）聚合成一个组合模块（modules），对组合模块构建时能够一次完成多个子模块的构建。

图1（使用modules和module将多个子模块聚合成一个组合模块）

