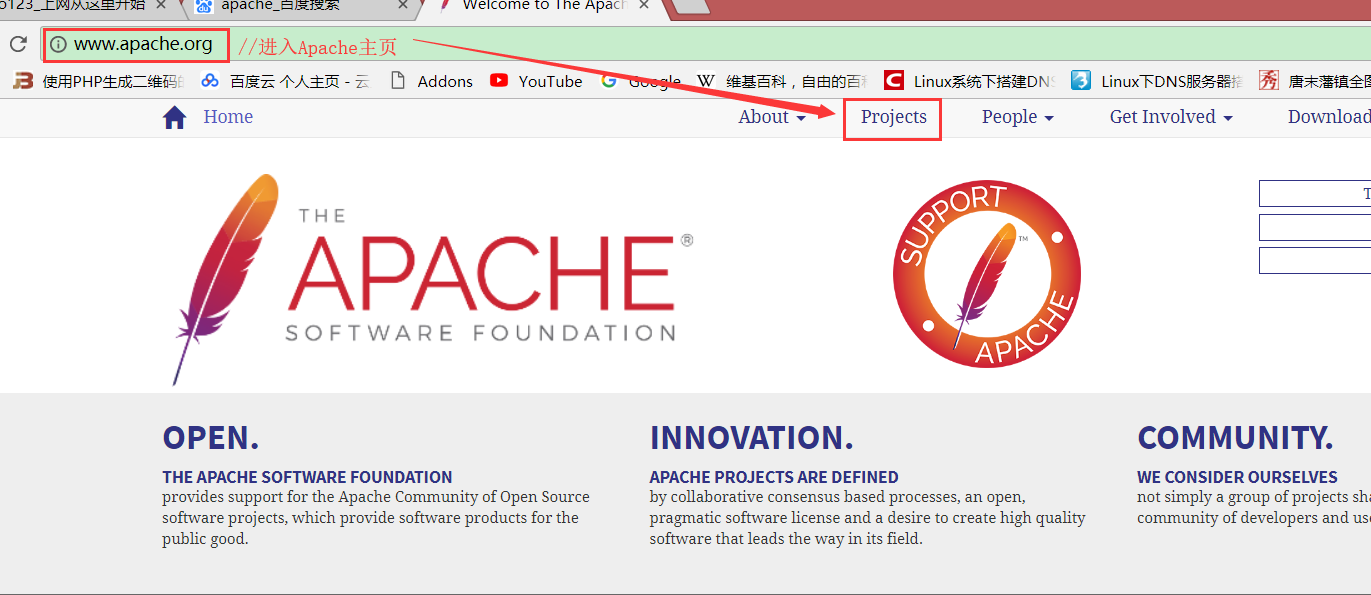
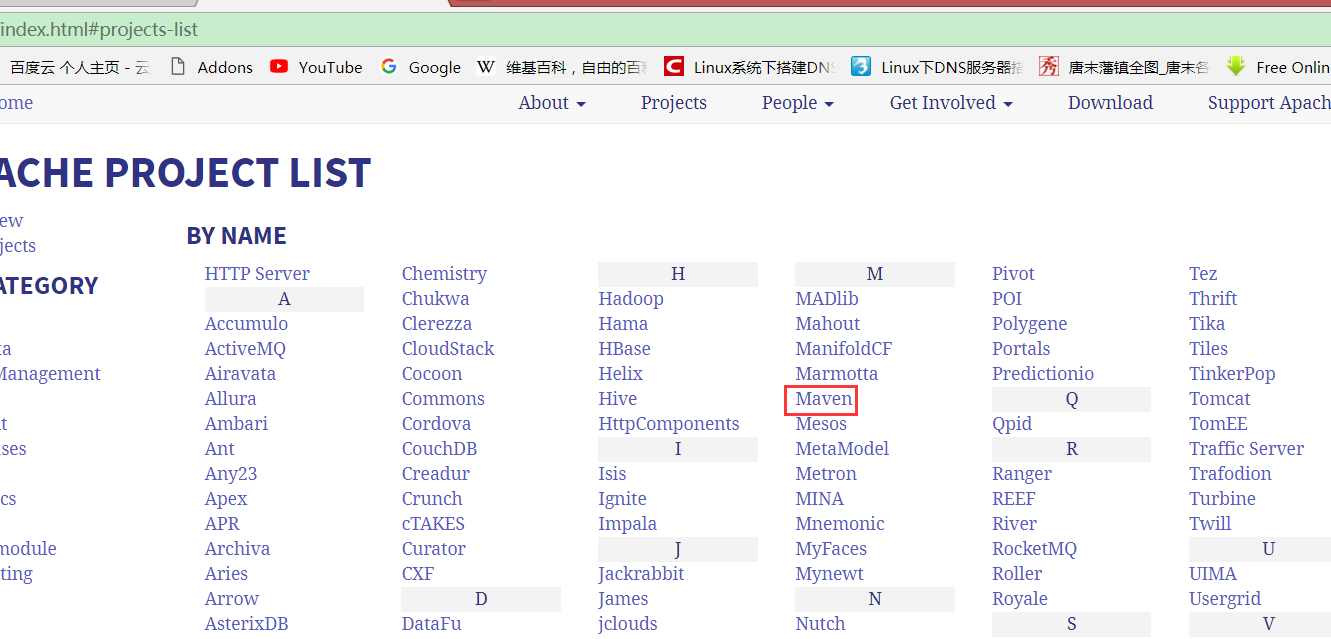
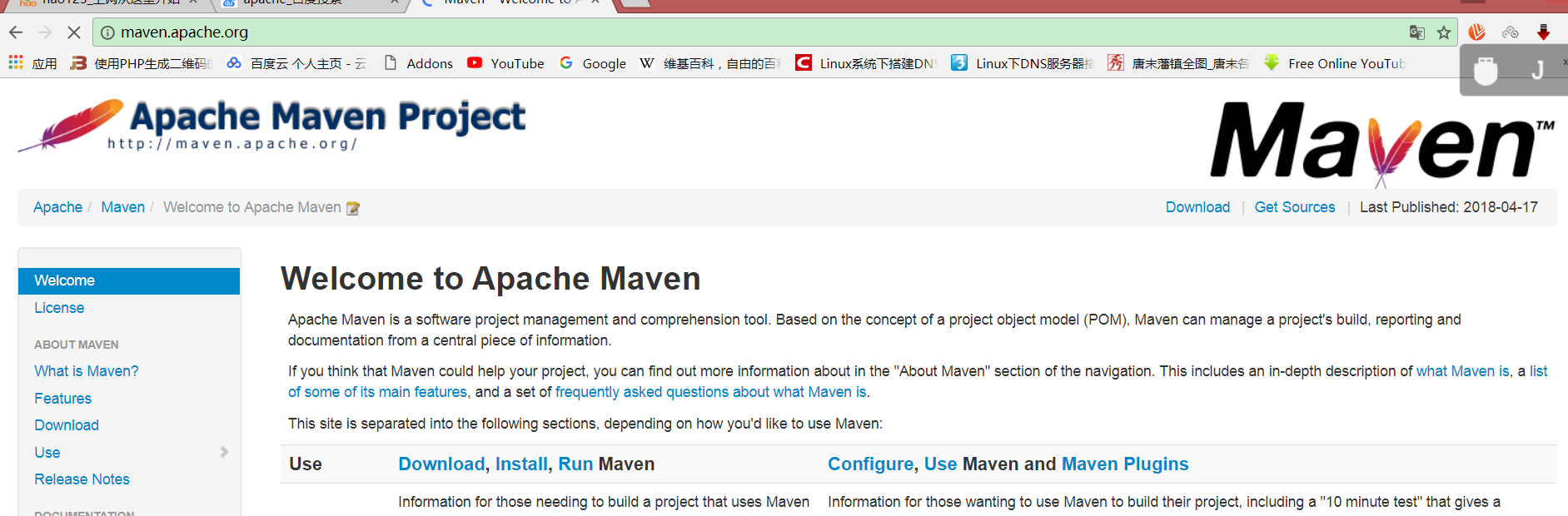
# Maven官网







# Maven的使用

<http://mvnrepository.com> ：jar包的下载地址

Maven主要用于项目中jar包的管理和项目构建管理。

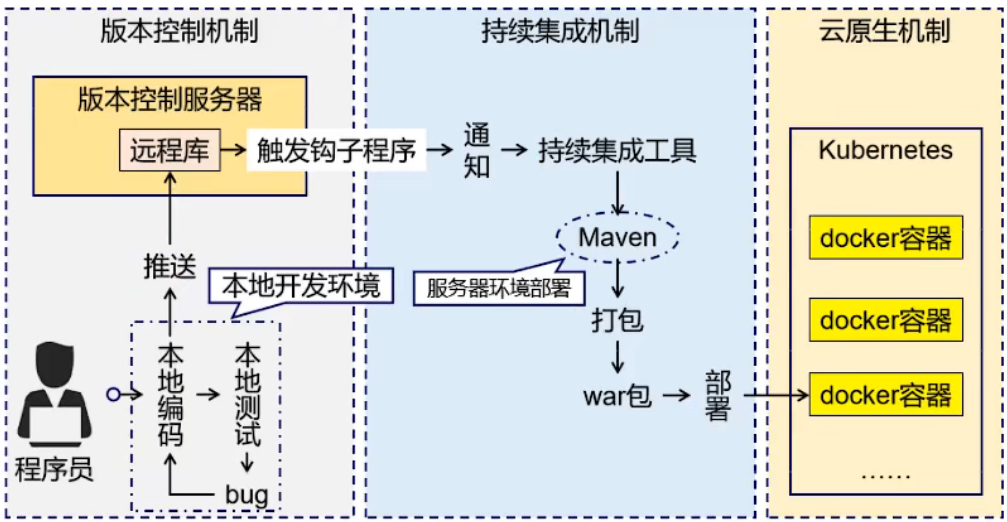
## Maven的用途

### jar包依赖管理

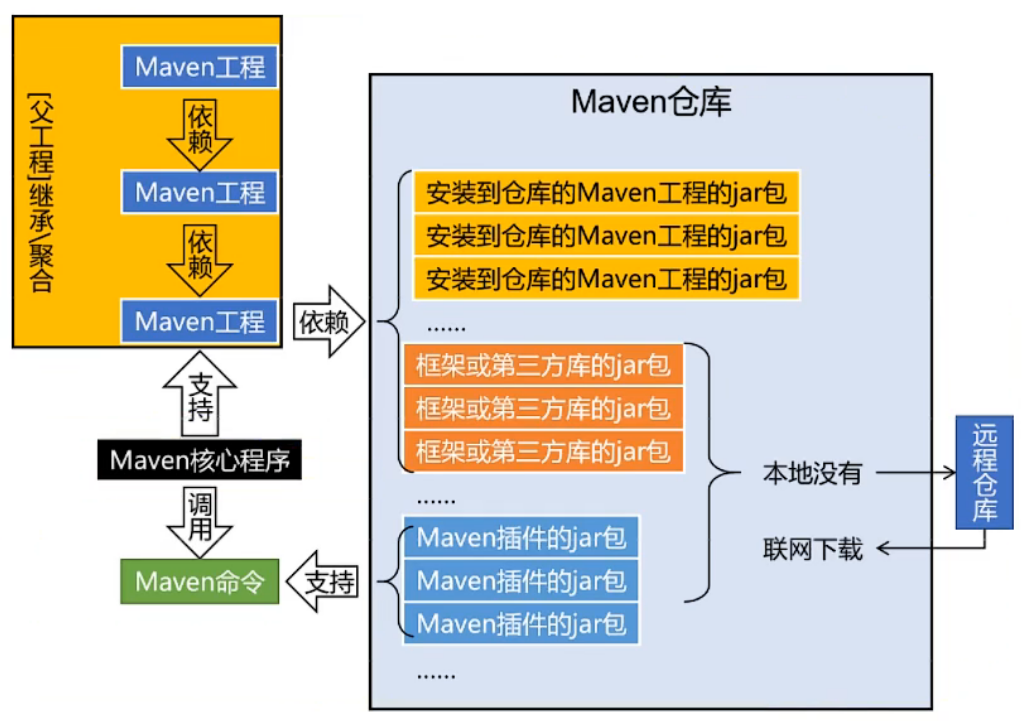
### 项目构建管理

#### 与IDE结合的项目构建

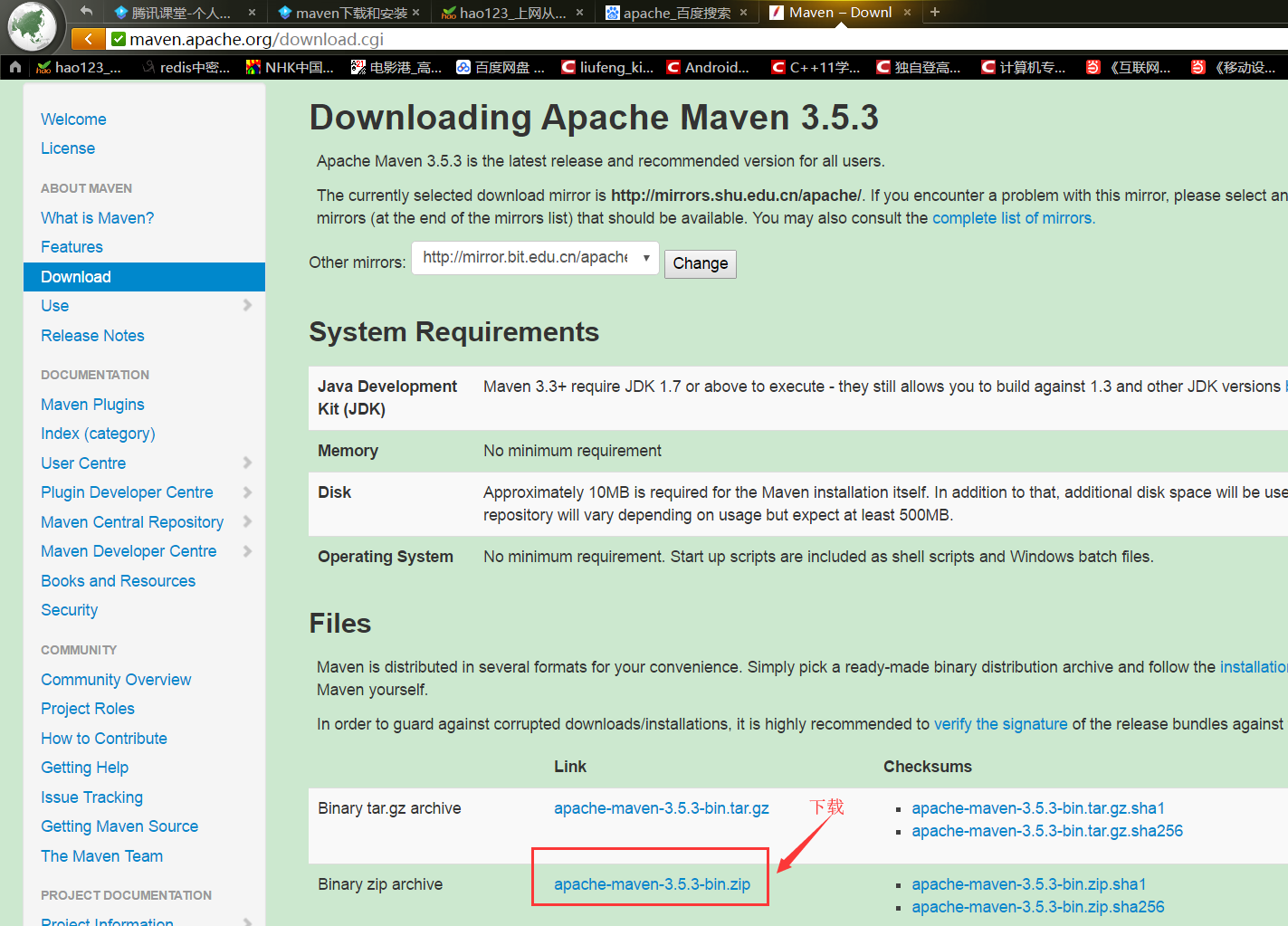
#### 脱离IDE环境的项目构建



## Maven的工作机制



## 下载与安装

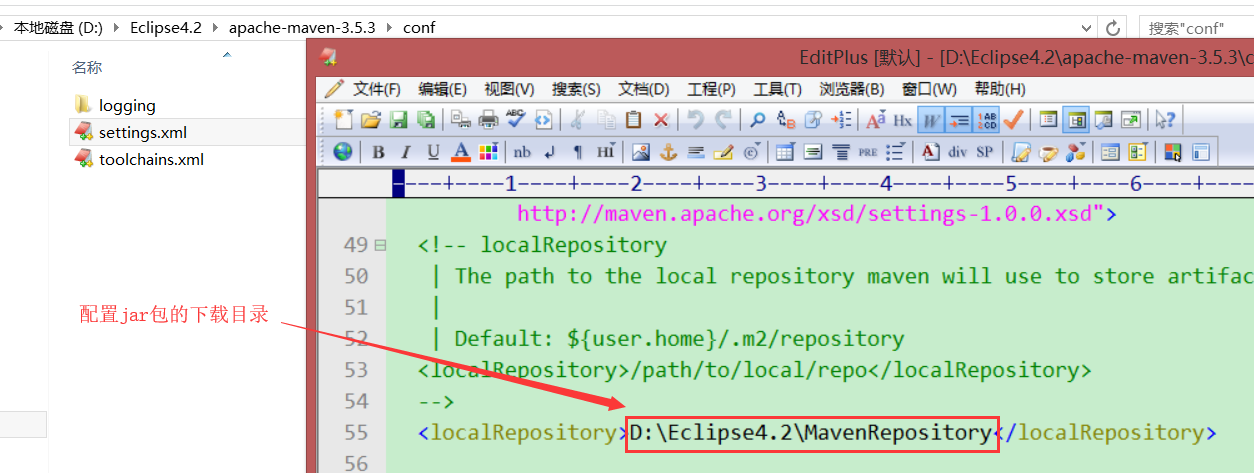




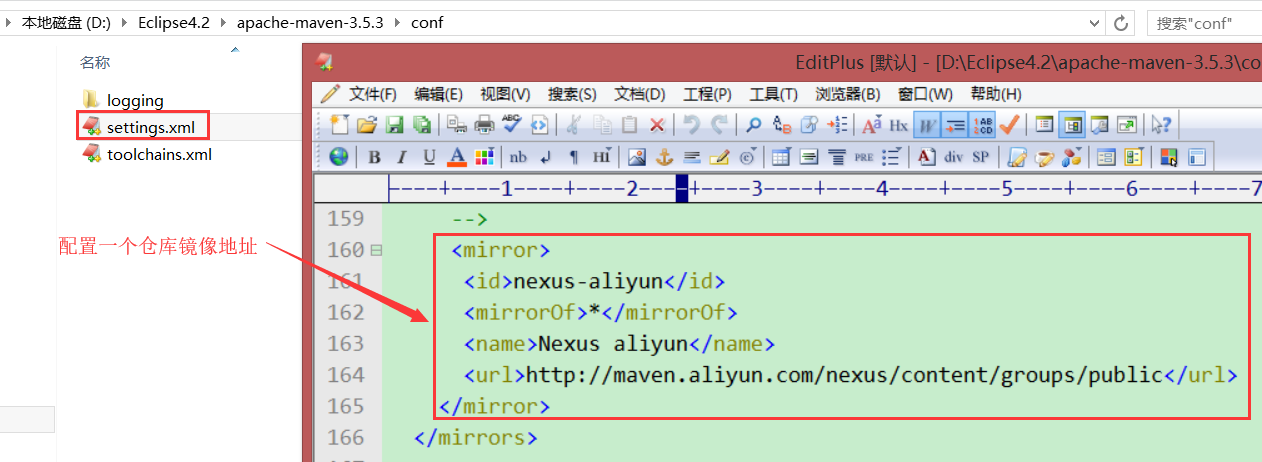
## 配置全局Maven管理环境

Maven仓库：<http://mvnrepository.com>

### 配置本地jar包存放目录



### 配置国内阿里云镜像地址



### 配置Maven的jdk编译版本

在conf/settins.xml的profiles节点下配置



### 配置JAVA\_HOME和MAVEN\_HOME

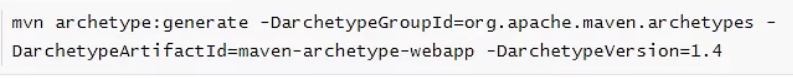
配置到环境变量即可

## 命令行创建Maven项目

### 创建普通Java工程

mvn archetype:generate

### 创建Web工程

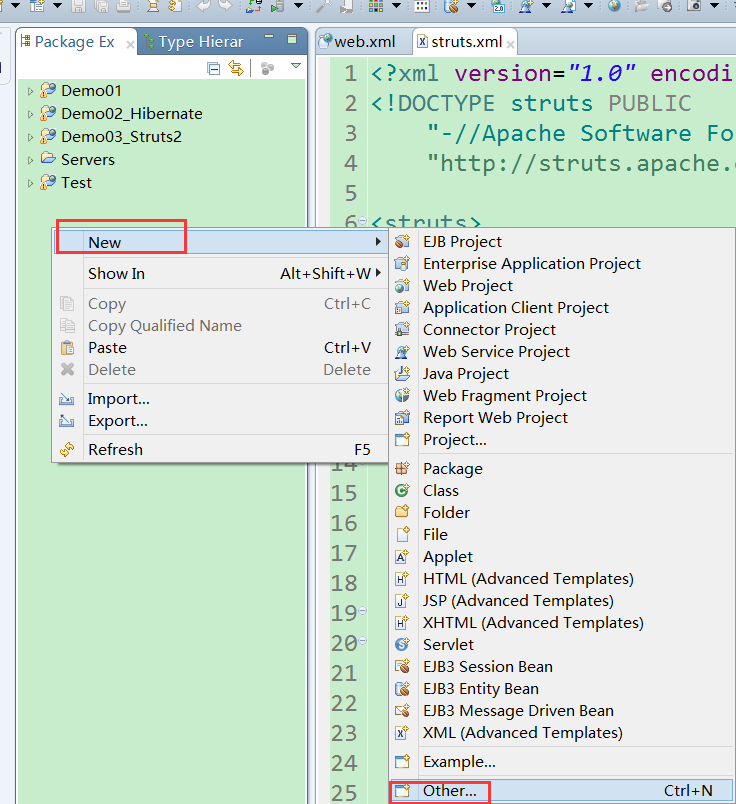


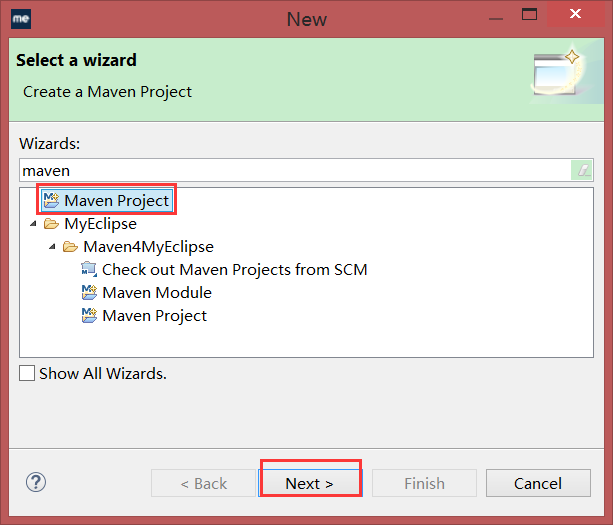


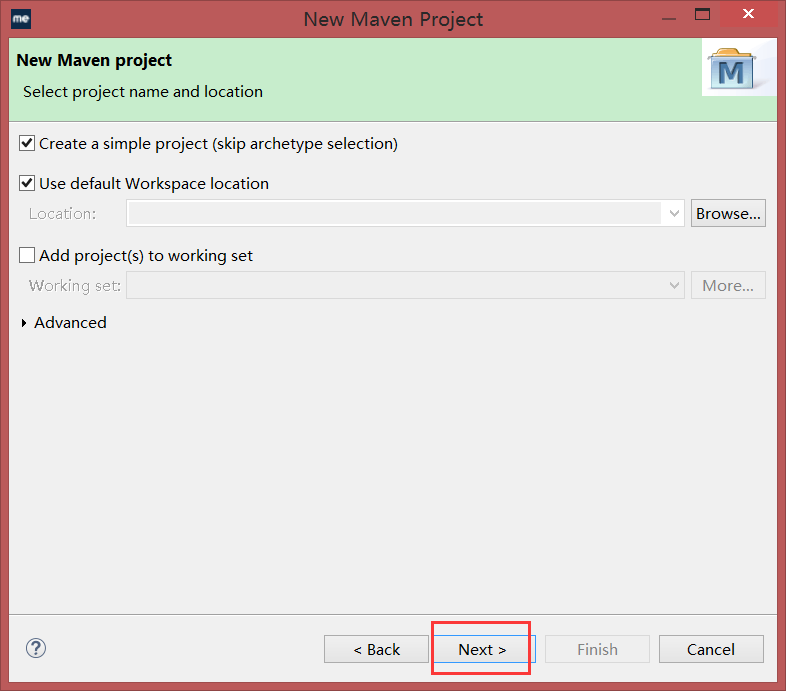
web.xml位于WEB-INF目录下

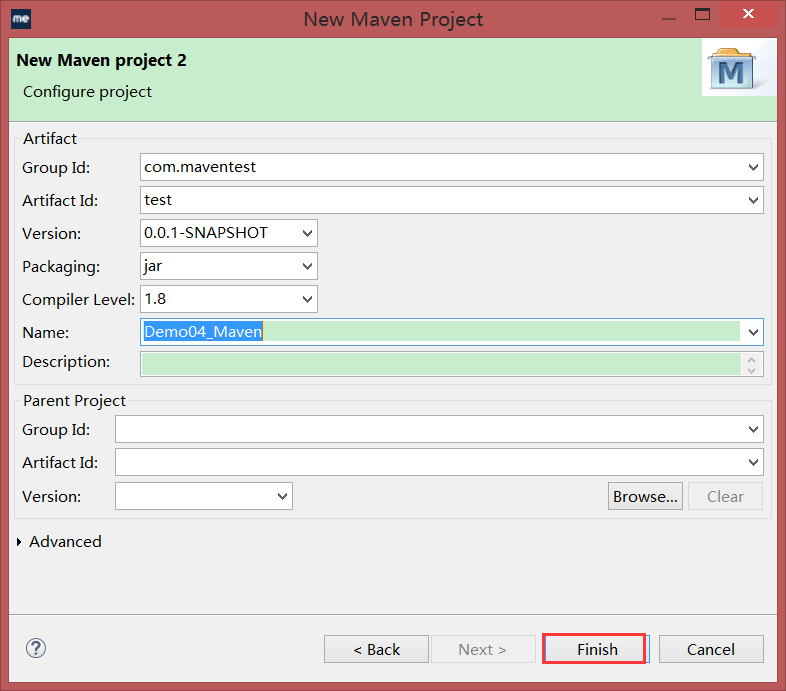
## MyEclipse创建Maven项目

MyEclipse中自动集成了一个Maven环境，但是也可以自定义配置自己下载好的。

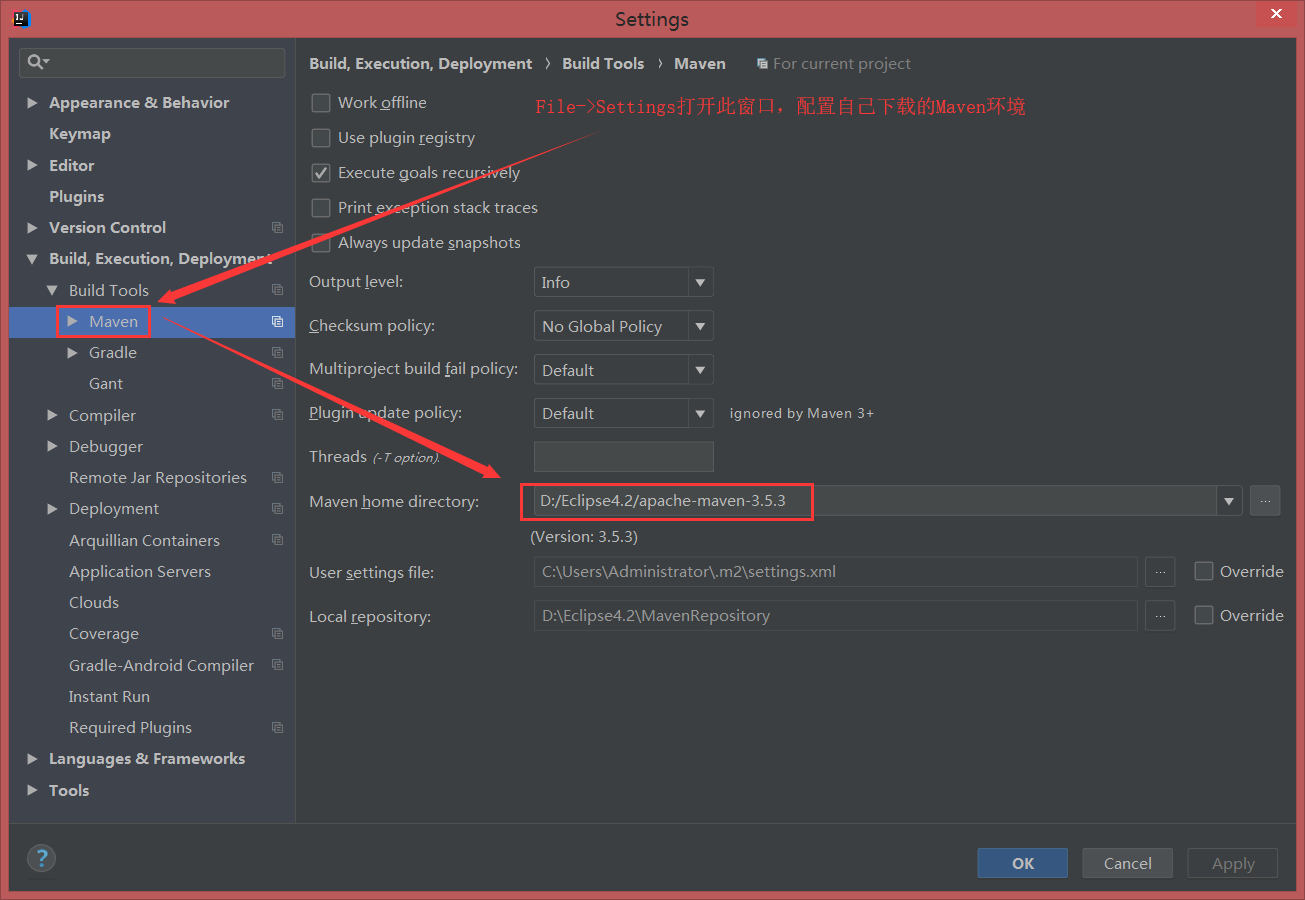


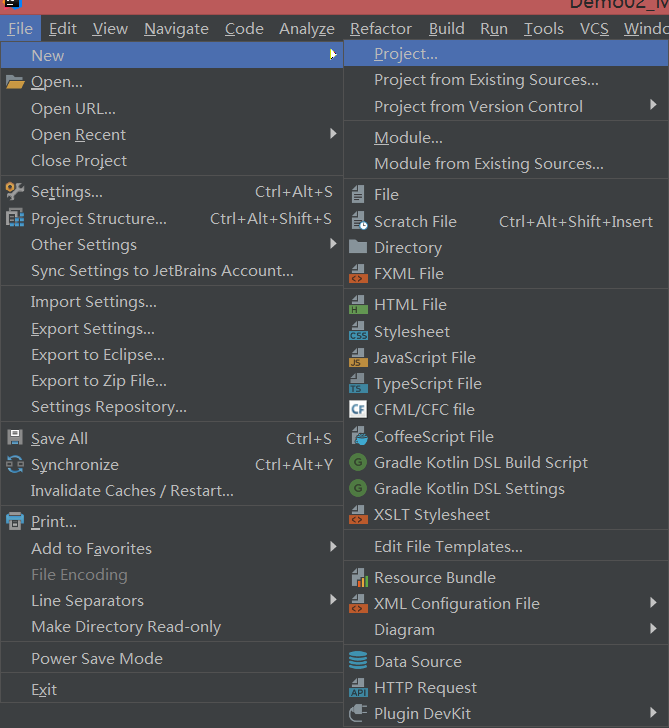


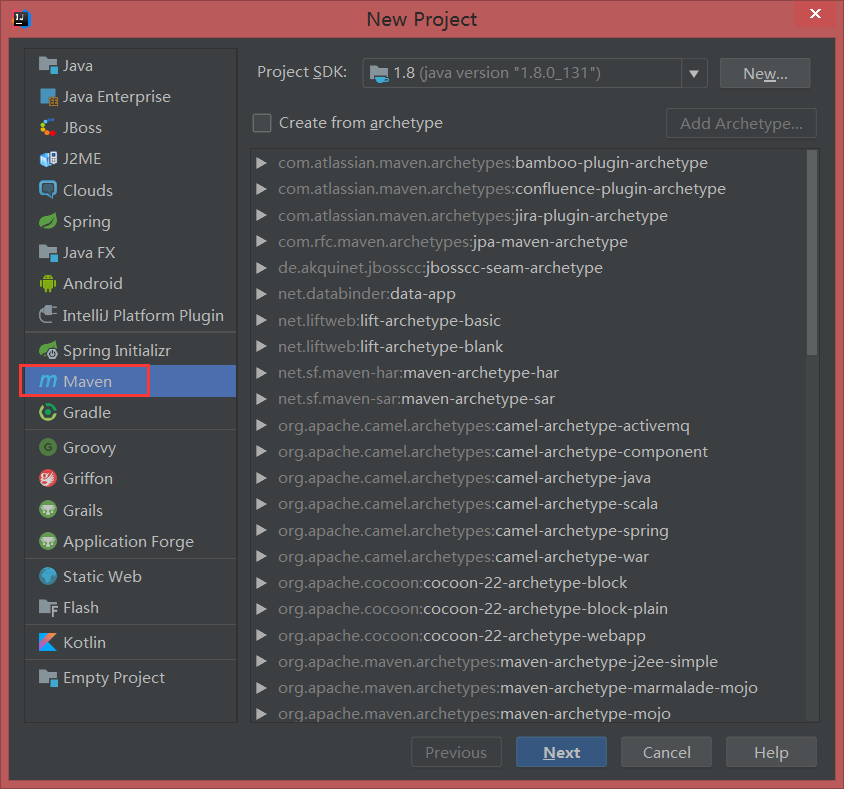




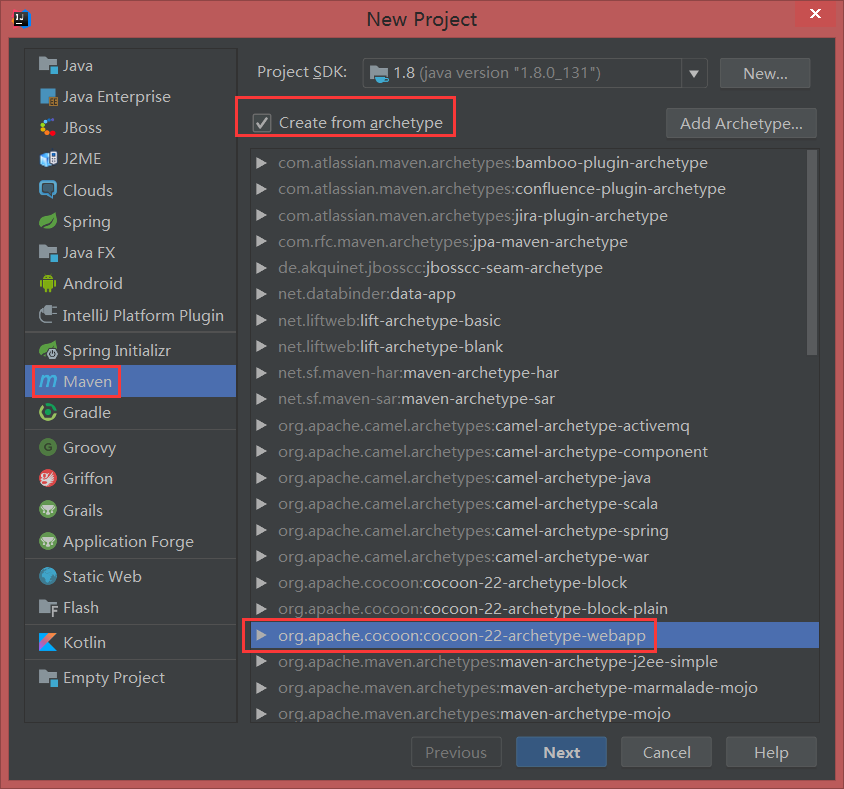
## Idea创建Maven项目

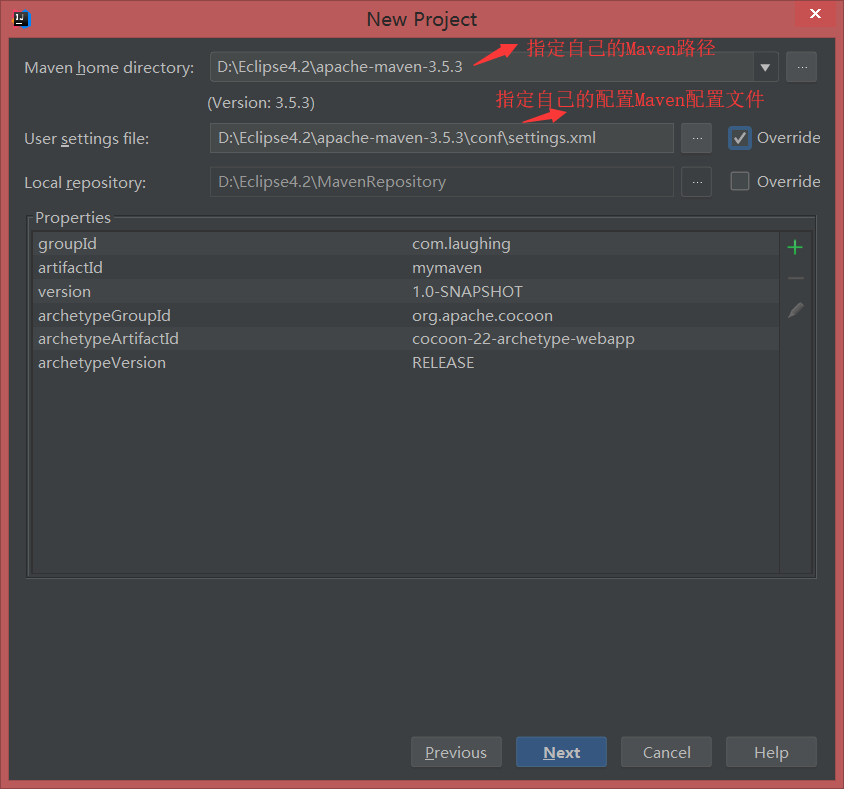






## idea创建Maven Web项目



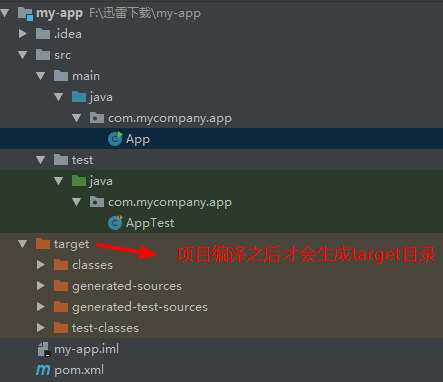


## 2.7 maven项目的目录结构

如果需要maven工具来构建管理项目就必须按照maven的规范约束创建目录结构。

* 创建普通Java项目：

mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false



* 创建web项目：

mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-WebApp -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false



## 2.8 maven常用命令

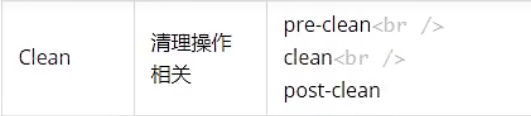
* mvn clean：删除清理编译后产生的target目录；
* mvn compile：编译src/main下面的源码并生成target目录，在target目录下的classes目录下生成对应的.class文件；
* mvn test-compile：编译src/test下面的源代码;
* mvn test：编译src/test下的源码，在target目录下生成test-classes目录并生成对应的.class文件；
* mvn package：按照pom文件中配置的打包方式打包当前项目；
* mvn install：把当前项目打包后发布到本地配置的maven仓库目录中；
* mvn site：在target目录下生成项目说明文档；
* mvn dependency:tree 查看当前项目的依赖信息；

## maven生命周期

compile<test<package<install（执行范围从小到大，后面的命令执行时会预先依次执行前面的命令）;

* 有3个主要的独立生命周期，各个之间相互独立;
* 生命周期内部无论直接执行哪个命令都会自动先执行其之前的所有命令；

### Clean

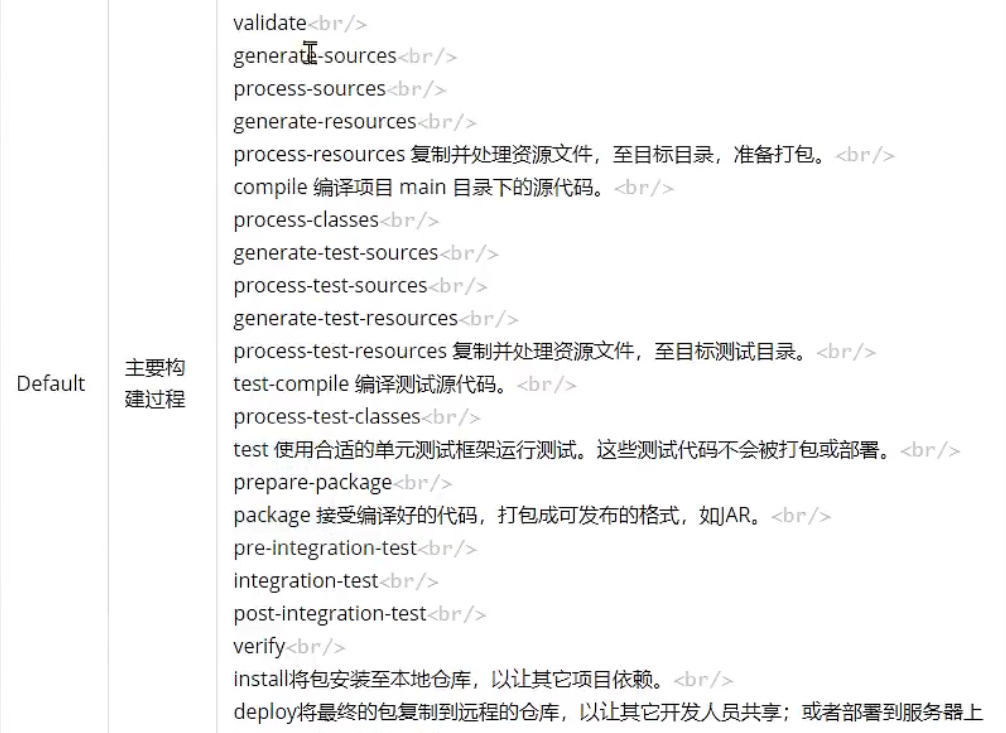


### Site



以HTML站点的形式生成项目的清单信息

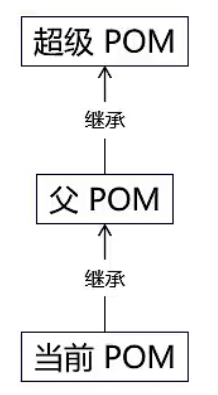
### Default



## pom文件

### 超级pom

* 超级pom是所有pom的祖父级pom，类似于Java中的Object。当一个pom没有明确指定父pom时，超级pom就是其默认的父pom；
* 超级pom的作用：约定各种通用的规则定义，定义了maven项目的默认设置；



* 查看超级pom内容：在命令行中输入 mvn help:effective-pom

<?xml version="1.0" encoding="GBK"?>

<!-- ====================================================================== -->  
<!-- -->  
<!-- Generated by Maven Help Plugin on 2022-07-26T13:00:10+08:00 -->  
<!-- See: http://maven.apache.org/plugins/maven-help-plugin/ -->  
<!-- -->  
<!-- ====================================================================== -->  
<!-- ====================================================================== -->  
<!-- -->  
<!-- Effective POM for project -->  
<!-- 'com.servlet.maven:servlet-maven:war:1.0-SNAPSHOT' -->  
<!-- -->  
<!-- ====================================================================== -->  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.servlet.maven</groupId>  
 <artifactId>servlet-maven</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>war</packaging>  
 <properties>  
 <maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
 <repositories>  
 <repository>  
 <snapshots>  
 <enabled>false</enabled>  
 </snapshots>  
 <id>central</id>  
 <name>Central Repository</name>  
 <url>https://repo.maven.apache.org/maven2</url>  
 </repository>  
 </repositories>  
 <pluginRepositories>  
 <pluginRepository>  
 <releases>  
 <updatePolicy>never</updatePolicy>  
 </releases>  
 <snapshots>  
 <enabled>false</enabled>  
 </snapshots>  
 <id>central</id>  
 <name>Central Repository</name>  
 <url>https://repo.maven.apache.org/maven2</url>  
 </pluginRepository>  
 </pluginRepositories>  
 <build>  
 <sourceDirectory>E:\JavaTest\servletmaven\src\main\java</sourceDirectory>  
 <scriptSourceDirectory>E:\JavaTest\servletmaven\src\main\scripts</scriptSourceDirectory>  
 <testSourceDirectory>E:\JavaTest\servletmaven\src\test\java</testSourceDirectory>  
 <outputDirectory>E:\JavaTest\servletmaven\target\classes</outputDirectory>  
 <testOutputDirectory>E:\JavaTest\servletmaven\target\test-classes</testOutputDirectory>  
 <resources>  
 <resource>  
 <directory>E:\JavaTest\servletmaven\src\main\resources</directory>  
 </resource>  
 </resources>  
 <testResources>  
 <testResource>  
 <directory>E:\JavaTest\servletmaven\src\test\resources</directory>  
 </testResource>  
 </testResources>  
 <directory>E:\JavaTest\servletmaven\target</directory>  
 <finalName>servlet-maven-1.0-SNAPSHOT</finalName>  
 <pluginManagement>  
 <plugins>  
 <plugin>  
 <artifactId>maven-antrun-plugin</artifactId>  
 <version>1.3</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-assembly-plugin</artifactId>  
 <version>2.2-beta-5</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-dependency-plugin</artifactId>  
 <version>2.8</version>  
 </plugin>  
 <plugin>  
 <artifactId>maven-release-plugin</artifactId>  
 <version>2.5.3</version>  
 </plugin>  
 </plugins>  
 </pluginManagement>  
 <plugins>  
 <plugin>  
 <artifactId>maven-clean-plugin</artifactId>  
 <version>2.5</version>  
 <executions>  
 <execution>  
 <id>default-clean</id>  
 <phase>clean</phase>  
 <goals>  
 <goal>clean</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 <plugin>  
 <artifactId>maven-resources-plugin</artifactId>  
 <version>2.6</version>  
 <executions>  
 <execution>  
 <id>default-testResources</id>  
 <phase>process-test-resources</phase>  
 <goals>  
 <goal>testResources</goal>  
 </goals>  
 </execution>  
 <execution>  
 <id>default-resources</id>  
 <phase>process-resources</phase>  
 <goals>  
 <goal>resources</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 <plugin>  
 <artifactId>maven-war-plugin</artifactId>  
 <version>2.2</version>  
 <executions>  
 <execution>  
 <id>default-war</id>  
 <phase>package</phase>  
 <goals>  
 <goal>war</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 <plugin>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.1</version>  
 <executions>  
 <execution>  
 <id>default-compile</id>  
 <phase>compile</phase>  
 <goals>  
 <goal>compile</goal>  
 </goals>  
 </execution>  
 <execution>  
 <id>default-testCompile</id>  
 <phase>test-compile</phase>  
 <goals>  
 <goal>testCompile</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 <plugin>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.12.4</version>  
 <executions>  
 <execution>  
 <id>default-test</id>  
 <phase>test</phase>  
 <goals>  
 <goal>test</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 <plugin>  
 <artifactId>maven-install-plugin</artifactId>  
 <version>2.4</version>  
 <executions>  
 <execution>  
 <id>default-install</id>  
 <phase>install</phase>  
 <goals>  
 <goal>install</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 <plugin>  
 <artifactId>maven-deploy-plugin</artifactId>  
 <version>2.7</version>  
 <executions>  
 <execution>  
 <id>default-deploy</id>  
 <phase>deploy</phase>  
 <goals>  
 <goal>deploy</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 <plugin>  
 <artifactId>maven-site-plugin</artifactId>  
 <version>3.3</version>  
 <executions>  
 <execution>  
 <id>default-site</id>  
 <phase>site</phase>  
 <goals>  
 <goal>site</goal>  
 </goals>  
 <configuration>  
 <outputDirectory>E:\JavaTest\servletmaven\target\site</outputDirectory>  
 <reportPlugins>  
 <reportPlugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-project-info-reports-plugin</artifactId>  
 </reportPlugin>  
 </reportPlugins>  
 </configuration>  
 </execution>  
 <execution>  
 <id>default-deploy</id>  
 <phase>site-deploy</phase>  
 <goals>  
 <goal>deploy</goal>  
 </goals>  
 <configuration>  
 <outputDirectory>E:\JavaTest\servletmaven\target\site</outputDirectory>  
 <reportPlugins>  
 <reportPlugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-project-info-reports-plugin</artifactId>  
 </reportPlugin>  
 </reportPlugins>  
 </configuration>  
 </execution>  
 </executions>  
 <configuration>  
 <outputDirectory>E:\JavaTest\servletmaven\target\site</outputDirectory>  
 <reportPlugins>  
 <reportPlugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-project-info-reports-plugin</artifactId>  
 </reportPlugin>  
 </reportPlugins>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
 <reporting>  
 <outputDirectory>E:\JavaTest\servletmaven\target\site</outputDirectory>  
 </reporting>  
</project>

### 依赖版本冲突解决



### 依赖包的作用范围

即<dependency>/<scope>节点的取值

* 运行时无效表示在打包时相关依赖不会参与打包；

#### compile(默认值)

在main和test下都有效，编译时和运行时都有效

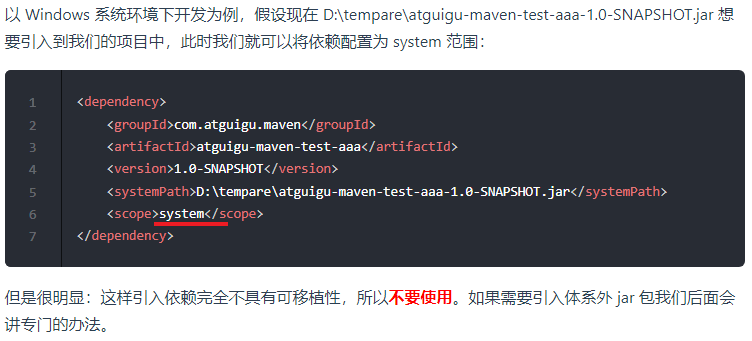
#### test

只在运行单元测试时有效，只对test目录有效，对main目录无效；编译时有效，运行时无效；

#### provided

对main、test目录有效，运行时无效。用于生产环境已经提供了相关依赖包的前提下。

#### system



#### runtime



#### import

import 依赖范围使用要求：

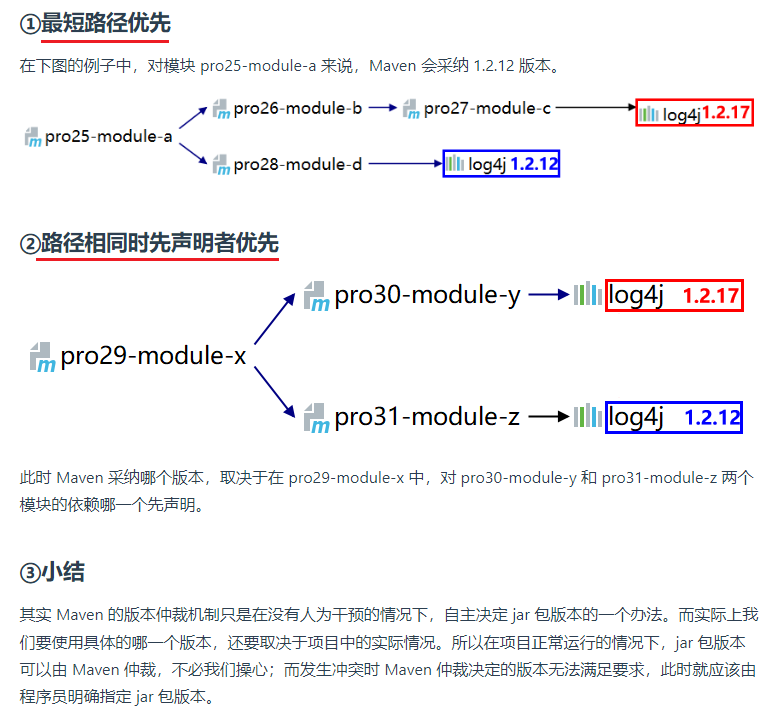
* 打包类型必须是 pom
* 必须放在 dependencyManagement 中



### maven打包方式

* jar
* war
* pom：作为管理其他工程的工程打包方式

### 版本仲裁



### 依赖排除



### 插件

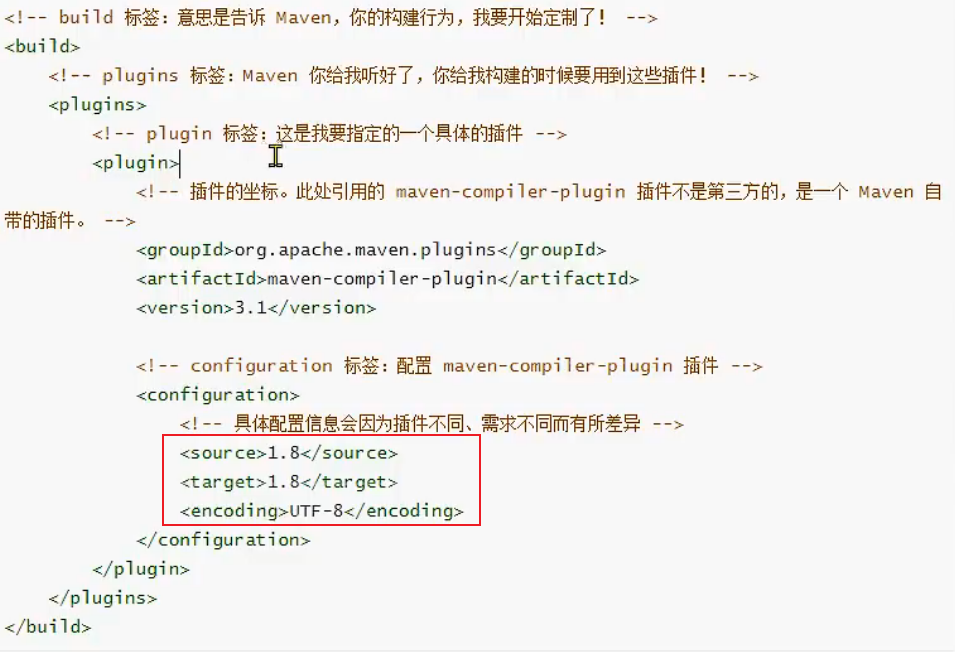
#### help插件



### build标签

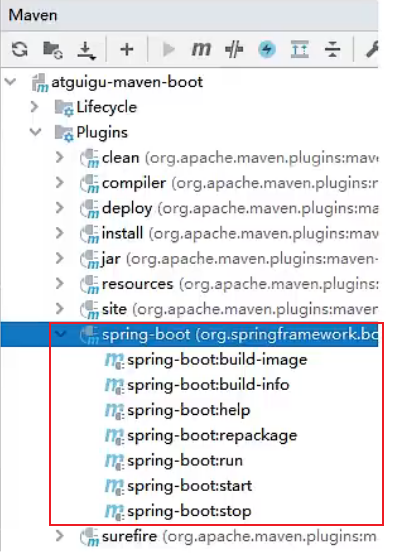
自定义的构建配置项会覆盖超级pom中相同项的默认配置。

#### 配置jdk编译版本



#### SpringBoot定制化打包





#### Mybatis逆向工程

<!-- 控制 Maven 在构建过程中相关配置 -->

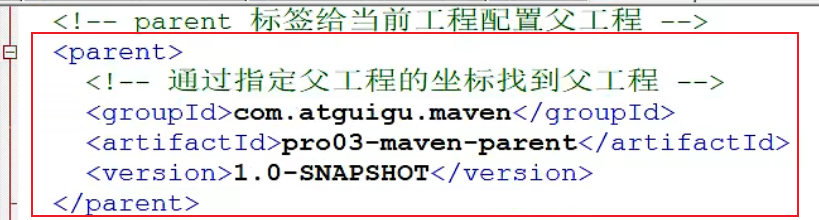
<build>  
 <!-- 构建过程中用到的插件 -->  
 <plugins>  
  
 <!-- 具体插件，逆向工程的操作是以构建过程中插件形式出现的 -->  
 <plugin>  
 <groupId>org.mybatis.generator</groupId>  
 <artifactId>mybatis-generator-maven-plugin</artifactId>  
 <version>1.3.0</version>  
  
 <!-- 插件的依赖 -->  
 <dependencies>  
  
 <!-- 逆向工程的核心依赖 -->  
 <dependency>  
 <groupId>org.mybatis.generator</groupId>  
 <artifactId>mybatis-generator-core</artifactId>  
 <version>1.3.2</version>  
 </dependency>  
  
 <!-- 数据库连接池 -->  
 <dependency>  
 <groupId>com.mchange</groupId>  
 <artifactId>c3p0</artifactId>  
 <version>0.9.2</version>  
 </dependency>  
  
 <!-- MySQL驱动 -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.8</version>  
 </dependency>  
 </dependencies>  
 </plugin>  
 </plugins>  
</build>

## 父子工程

* 只有打包方式为pom的工程才能管理其他工程；
* 父工程一般不写业务代码；
* 父工程pom包含子模块：



* 子模块pom指定父工程：



* 在父工程pom中统一管理依赖信息



* 子模块要使用哪些依赖包，还得在自己的pom中明确加上(版本不用再指定)；