# 起步

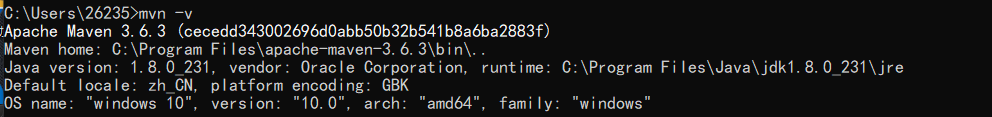
## 介绍

Maven下载并自动关联所有依赖的jar

Maven功能：清理、编译、测试、报告、打包、安装、部署

## Maven下载安装

1. 到官网https://maven.apache.org/download.cgi下载maven压缩包
2. 解压文件，然后配置MAVEN\_HOME环境变量【同JAVA\_HOME的配置方式一样】，最后加入path。
3. 验证是否配置正确：进入CMD输入**mvn –v**，看结果；若如下，则正确



## 配置Maven本地仓库

Maven默认的本地仓库路径是C:/User/…/.m2/repository中，为节约C盘空间，设置Maven仓库路径

1. 新建一个名为mavenRepository的文件夹；拷贝路径
2. 打开apache-maven-3.6.3\conf文件夹中setting.xml文件，搜索localReposlocalitory，找到如下一段

把上述灰色底纹的代码拷贝出来后将标签中的内容替换成第1步拷贝的路径。并将‘/’改为‘\’，保存xml文件即可。

<!-- localReposlocalitory

| The path to the local repository maven will use to store artifacts.

|

| Default: ${user.home}/.m2/repository

<localRepository>/path/to/local/repo</localRepository>

-->

## 配置Maven中央仓库

从国外maven仓库下载jar包速度太慢，推荐使用淘宝镜像，setting.xml中此段内容

修改为：

<mirrors>

<!-- mirror

| Specifies a repository mirror site to use instead of a given repository. The repository that

| this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs are used

| for inheritance and direct lookup purposes, and must be unique across the set of mirrors.

|

<mirror>

<id>mirrorId</id>

<mirrorOf>repositoryId</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://my.repository.com/repo/path</url>

</mirror>

-->

</mirrors>

<mirrors>

<!-- mirror

| Specifies a repository mirror site to use instead of a given repository. The repository that

| this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs are used

| for inheritance and direct lookup purposes, and must be unique across the set of mirrors.

|-->

<mirror>

<id>**aliyunmaven**</id>

<mirrorOf>**central**</mirrorOf>

<name>**aliyun maven**</name>

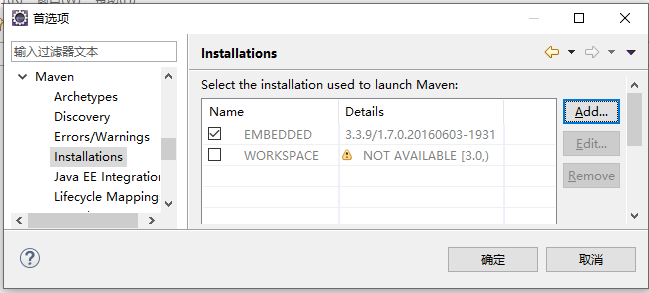
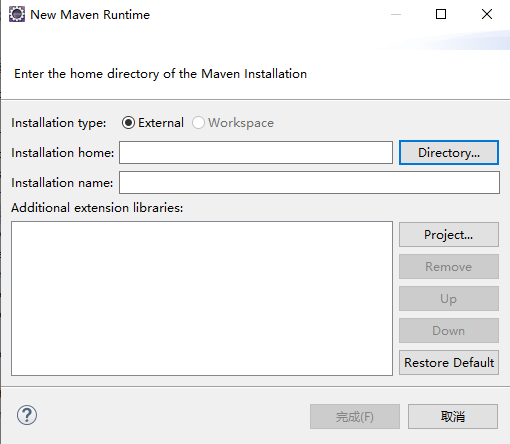
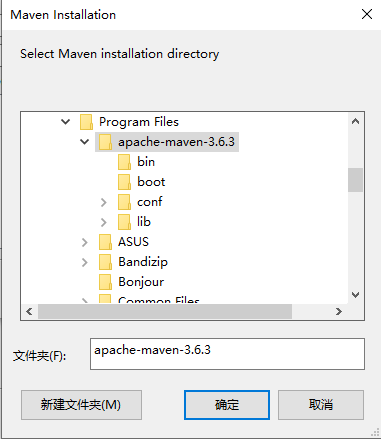
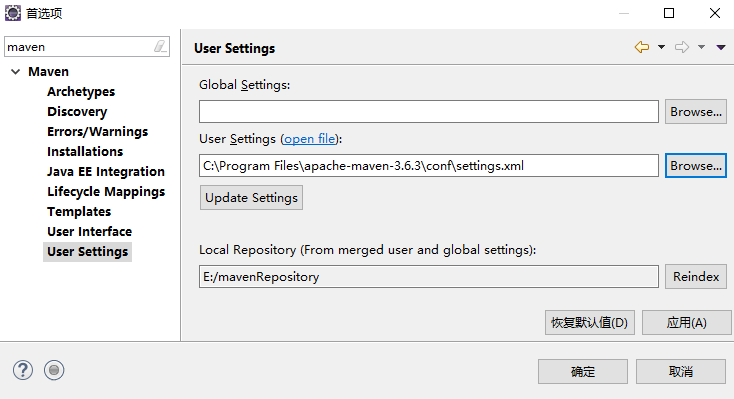
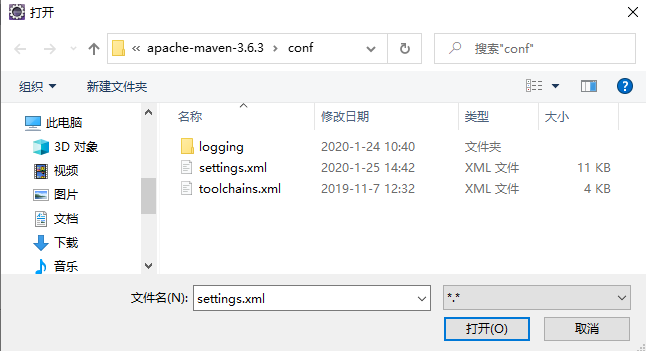
<url>**https://maven.aliyun.com/repository/public**</url>

</mirror>

</mirrors>

eclipse配置maven

eclipse自带maven，即便不配置也可以使用，不过为了项目统一建议配置自身的maven。以下是配置步骤：

→→→→

# 使用

### 基本使用

Maven项目依赖pom.xml文件，项目所需jar包都得在pom.xml中定义；格式如下

<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.train</groupId>

<artifactId>mavenTrain</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!--jar包-->

</dependencies>

</project>

所需jar包到：https://mvnrepository.com/中查找，并拷贝依赖复制到pom.xml中即可；格式为

<dependency>

<groupId>org.apache.commons</groupId><!--组织id-->

<artifactId>commons-lang3</artifactId><!--jar包名称-->

<version>3.8.1</version><!--版本号-->

<scope>compile</scope><!--作用范围（生命周期）-->

</dependency>

作用范围（生命周期）

用来描述jar包在何种环境下起作用。

### Maven命令

mvn compile:编译项目，项目下生成target目录

mvn clean:清除target目录

mvn test:编译test下代码，项目下生成target目录，目录下生成test-classes文件夹，其中存放编译后代码。

mvn package:打包项目成jar包。项目下target目录中有一个jar或war包。

mvn install:将jar包或war包发布到本地仓库

|  |  |  |  |
| --- | --- | --- | --- |
| <scope> | compile | test | provided |
| 编译 | √ |  | √ |
| 测试 | √ | √ | √ |
| 运行 | √ |  |  |

### 排除依赖包

引入一个jar包后Maven将自动引入jar包的其他依赖jar包，若需要限制其他依赖的jar包，则手动排除依赖包；格式如下：

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.2.12.RELEASE</version>

**<exclusions><!--被排除的依赖包写在这个标签中-->**

**<exclusion><!--被排除的依赖包-->**

**<groupId>org.springframework</groupId>**

**<artifactId>spring-jcl</artifactId>**

**</exclusion>**

**</exclusions>**

</dependency>

Maven常用命令

编译mvn compile：编译结束后，项目下诞生target目录，里面存放maven发布所需以及打包的jar包

测试mvn test，

打包mvn package，

将开发的模块放入本地仓库mvn install，

清理mvn clean

# Maven小记

统一声明依赖包的版本号并引用：

申明：properties中的标签名可以随意定义

<properties>

<springversion>5.1.12.RELEASE</springversion>

</properties>

使用

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>**${springversion}**</version>

</dependency>

当然，properties不仅可以使用在版本统一方面，其他方面都可行。

maven的传递性：项目B依赖项目A，则项目B也会在自己的依赖中加入项目A的所有jar包。（注：仅compile范围jar可以被传递）