Maven

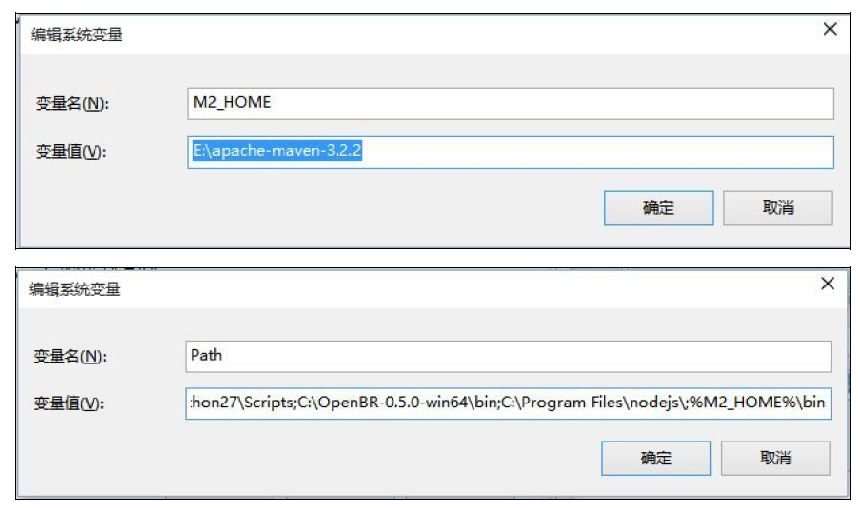
# Maven的安装

1,下载Maven

Maven下载地址：<https://maven.apache.org/download.cgi> 并解压到任意目录

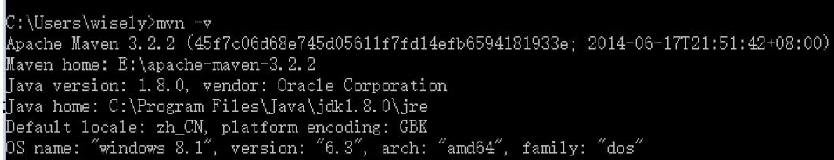
2,配置Maven

在系统属性→高级→环境变量中分别配置**M2\_HOME**和**Path**



3,测试安装

在控制台输入“mvn–v”，如下图表示安装成功



# Pom.xml

1.**dependencies**元素

包含多个项目依赖需要使用的<dependency></dependency>元素

2.**dependency**元素

通过groupId、artifactId以及version确定唯一的依赖，

**groupId**：组织的唯一标识。

**artifactId**：项目的唯一标识。

**version**：项目的版本。

<dependency>

　　 <groupId>org.springframework</groupId>

　　 <artifactId>spring-webmvc</artifactId>

　　 <version>4.1.5.RELEASE</version>

</dependency>

3.变量定义**properties**

<properties></properties>可定义变量在dependency中引用，如下

<properties>

<spring-framework.version>4.1.5.RELEASE</spring-framework.version>

</properties>

<dependency>

　　 <groupId>org.springframework</groupId>

　　 <artifactId>spring-webmvc</artifactId>

　　 <version>${spring-framework.version}</version>

</dependency>

4.编译插件**plugins**

Maven提供了编译插件，可在编译插件中涉及Java的编译级别，如

<build>

　　　<plugins>

<plugin>

　　　　　　　　 <groupId>org.apache.maven.plugins</groupId>

　　　　　　　　 <artifactId>maven-compiler-plugin</artifactId>

　　　　　　　　 <version>2.3.2</version>

　　　　　　　　 <configuration>

　　　　　　　　　　 <source>1.7</source>

　　　　　　　　　　 <target>1.7</target>

　　　　　　　　 </configuration>

　　　　　　 </plugin>

　　　　 </plugins>

</build>

5.**Maven运作方式**

Maven会自动根据dependency中的依赖配置，直接通过互联网在Maven中心库下载相关依赖包到.m2目录下，.m2目录下是你本地Maven库。

如果你不知道你所依赖jar包的dependency怎么写的话，推荐到http://mvnrepository.com网站检索。

若Maven中心库中没有你需要的jar包（如Oracle），你需要通过下面的Maven命令打到本地Maven库后应用即可，如安装Oracle驱动到本地库：

mvn install:install-file -DgroupId=com.oracle "-DartifactId=ojdbc14"

"-Dversion=10.2.0.2.0" "-Dpackaging=jar" "-Dfile=D:\ojdbc14.jar"

# Maven command

maven 命令的格式为 mvn [plugin-name]:[goal-name]，可以接受的参数如下，

-D 指定参数，如 -Dmaven.test.skip=true 跳过单元测试；

-P 指定 Profile 配置，可以用于区分环境；

-e 显示maven运行出错的信息；

-o 离线执行命令,即不去远程仓库更新包；

-X 显示maven允许的debug信息；

-U 强制去远程更新snapshot的插件或依赖，默认每天只更新一次。

常用maven命令

创建maven项目：mvn archetype:create

指定 group： -DgroupId=packageName

指定 artifact：-DartifactId=projectName

创建web项目：-DarchetypeArtifactId=maven-archetype-webapp

创建maven项目：mvn archetype:generate

验证项目是否正确：mvn validate

maven 打包：mvn package

只打jar包：mvn jar:jar

生成源码jar包：mvn source:jar

产生应用需要的任何额外的源代码：mvn generate-sources

编译源代码： mvn compile

编译测试代码：mvn test-compile

运行测试：mvn test

运行检查：mvn verify

清理maven项目：mvn clean

生成eclipse项目：mvn eclipse:eclipse

清理eclipse配置：mvn eclipse:clean

生成idea项目：mvn idea:idea

安装项目到本地仓库：mvn install

发布项目到远程仓库：mvn:deploy

在集成测试可以运行的环境中处理和发布包：mvn integration-test

显示maven依赖树：mvn dependency:tree

显示maven依赖列表：mvn dependency:list

下载依赖包的源码：mvn dependency:sources

安装本地jar到本地仓库：

mvn install:install-file -DgroupId=packageName -DartifactId=projectName -Dversion=version -Dpackaging=jar -Dfile=path

mvn -version/-v -- 显示版本信息

mvn archetype:generate -- 创建mvn项目

mvn archetype:create -DgroupId=com.oreilly -DartifactId=my-app -- 创建mvn项目

mvn package -- 生成target目录，编译、测试代码，生成测试报告，生成jar/war

文件

mvn jetty:run -- 运行项目于jetty上,

mvn compile -- 编译

mvn test -- 编译并测试

mvn clean -- 清空生成的文件

mvn site -- 生成项目相关信息的网站

mvn -Dwtpversion=1.0 eclipse:eclipse -- 生成Wtp插件的Web项目

mvn -Dwtpversion=1.0 eclipse:clean -- 清除Eclipse项目的配置信息(Web项目)

mvn eclipse:eclipse -- 将项目转化为Eclipse项目

mvn install -Dmaven.test.skip=true 给任何目标添加maven.test.skip 属性就能跳过测试

mvn install assembly:assembly 构建装配Maven Assembly 插件是一个用来创建你应用程序特有分发包的插件

mvn jetty:run 调用 Jetty 插件的 Run 目标在 Jetty Servlet 容器中启动 web 应用

mvn compile 编译你的项目