[**maven学习（上）- 基本入门用法**](http://www.cnblogs.com/yjmyzz/p/3495762.html)

**一、下载及安装**

1.1 下载maven 3.1.1

先到官网<http://maven.apache.org/download.cgi> 下载最新版本（目前是[3.1.1](http://mirrors.cnnic.cn/apache/maven/maven-3/3.1.1/binaries/apache-maven-3.1.1-bin.zip) ），下载完成后，解压到某个目录（本文中是C:\Java\maven-3.1.1）

2.1 配置环境变量

系统环境变量里，添加MAVEN\_HOME(或M2\_HOME)，其值为C:\Java\maven-3.1.1，然后PATH环境变量最后附加上";%MAVEN\_HOME%\bin"

检测方法：

a) 重新进入命令行(DOS窗口)模式，输入 echo %MAVEN\_HOME% 如果能显示 C:\Java\maven-3.1.1 说明环境变量起作用了

b) 输入 mvn -version，正常情况下会显示maven及jdk的版本号

（前提：jdk环境必须先安装好，否则后面无法正常编译项目）

3 理解“仓库”

首次运行完mvn -version后，会在用户目录下创建一个.m2的目录(比如：C:\Users\当前用户名\.m2\)，这个目录是maven的“本地仓库”，仓库是maven中一个很重要的概念。

试想一下，我们会在工作中同时创建很多项目，每个项目可能都会引用一些公用的jar包(.NET中是dll文件)，一种作法是每个项目里，都复制一份这些依赖的jar包(或dll文件)，这样显然不好，相同的文件在硬盘上保存了多份，太占用空间，而且这些依赖的jar包(或dll文件)的版本也不太好管理(比如某个公用的jar包，从1.0升级到2.0，如果所有引用这个jar包的项目都需要更新，必须一个个项目的修改)。

maven的仓库则很好的解决了这些问题，它在每台机器上创建一个本机仓库，把本机上所有maven项目依赖的jar包统一管理起来，而且这些jar包用“坐标”来唯一标识(注：坐标是另一个重要的概念，后面还会讲到，这里只要简单理解成“唯一识别某个jar包文件名、版本号”的标识即可)，这样所有maven项目就不需要再象以前那样把jar包复制到lib目录中，整个maven项目看起来十分清爽。

4 配置代理服务器（可选）

maven项目在编译、测试、打包里，会需要从maven的中央仓库(即：maven组织公布在互联网上的一个站点，里面已经收录了目前绝大多数主流的jar包)下载jar包等文件，如果使用代理服务器上网，需要配置代理服务器。

把%MAVEN\_HOME%\conf\settings.xml复制一份到本地仓库C:\Users\当前用户名\.m2\下，然后编辑该文件，找到下面这段

[复制代码](javascript:void(0);)

<proxies>

<!-- proxy

| Specification for one proxy, to be used in connecting to the network.

|

<proxy>

<id>optional</id>

<active>true</active>

<protocol>http</protocol>

<username>proxyuser</username>

<password>proxypass</password>

<host>proxy.host.net</host>

<port>80</port>

<nonProxyHosts>local.net|some.host.com</nonProxyHosts>

</proxy>

-->

</proxies>

[复制代码](javascript:void(0);)

把注释去掉，host这里填写代理服务器的地址（可以用IP）以及port端口，如果需要用户名/密码认证，则填写username/password节点，否则username/password这二个节点去掉，nonProxyHosts表示某些地址不需要经过代理服务器，多个地址之间用|分隔，支持通配符，比如172.156.\*

**二、创建项目“骨架”**

下面用命名行创建一个最基本的maven项目

2.1 mvn archetype:generate

先创建项目的根目录，比如c:\test，命令行窗口下输入

cd /d c:\test

mvn archetype:generate

首次运行时，mvn会从远程"中央仓库"下载一些必需的文件到"本地仓库" - （如果你有兴趣，可以在等待下载过程中，观察一下"C:\Users\当前用户名\.m2\repository"到底下载了些啥东东）

下载完成后（下一篇会讲解如何，在局域网环境中搭建“私服”，直接从局域网的代理仓库中下载这些依赖项），会自动进入交互模式，会让你输入一些基本信息，类似下面这样：

*...*

*[INFO] Generating project in Interactive mode (这里会卡一会儿，因为要联网获取项目模板)*  
*[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.archetypes:maven-archetype-quickstart:1.0)*  
*Choose archetype:*  
*1: remote -> br.com.ingenieux:elasticbeanstalk-service-webapp-archetype (A Maven Archetype Encompassing RestAssured, Jetty, Jackson, Guice and Jersey for Publishing JAX-RS-based Services on AWS' Elastic Beanstalk Service)*  
*... (这里会自动列出很多项目模板，每种模板前面会有一个数字序号)*

*336: remote -> org.apache.maven.archetypes:maven-archetype-quickstart (An archetype which contains a sample Maven project.)*

*...*

*Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 336: (这里根据每个人机器上安装的eclipse插件不同，可能默认的数字不是这个，先不管，直接回车)*

*Choose org.apache.maven.archetypes:maven-archetype-quickstart version:*  
*1: 1.0-alpha-1*  
*2: 1.0-alpha-2*  
*3: 1.0-alpha-3*  
*4: 1.0-alpha-4*  
*5: 1.0*  
*6: 1.1*  
*Choose a number: 6: (直接回车)*  
*Define value for property 'groupId': : cnblogs （可暂时先理解成类似package或namespace的名称，通常我们填写组织机构名称缩写）*  
*Define value for property 'artifactId': : maven-hello-world （组件名称，可暂时理解成项目名称）*  
*Define value for property 'version':  1.0-SNAPSHOT: : （版本号，直接回车，默认1.0-SNAPSHOT）*  
*Define value for property 'package':  cnblogs: : （打包后的jar文件名，相当于.net中项目最后生成的程序集dll名称）*  
*Confirm properties configuration:*  
*groupId: cnblogs*  
*artifactId: maven-hello-world*  
*version: 1.0-SNAPSHOT*  
*package: cnblogs*  
*Y: :  (直接回车确认)*  
*[INFO] ----------------------------------------------------------------------------*  
*[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.1*  
*[INFO] ----------------------------------------------------------------------------*  
*[INFO] Parameter: groupId, Value: cnblogs*  
*[INFO] Parameter: packageName, Value: cnblogs*  
*[INFO] Parameter: package, Value: cnblogs*  
*[INFO] Parameter: artifactId, Value: maven-hello-world*  
*...*  
*[INFO] ------------------------------------------------------------------------*  
*[INFO] BUILD SUCCESS (看到这个，表示项目创建成功！)*  
*[INFO] ------------------------------------------------------------------------*  
*...*

2.2 maven项目的目录结构

C:\test\maven-hello-world>tree  
Folder PATH listing for volume win7  
Volume serial number is AA2C-6E70  
C:.  
├───src  
│   ├───main  
│   │   └───java  
│   │       └───cnblogs  
│   └───test  
│       └───java  
│           └───cnblogs  
└───target  
    └───classes  
        └───cnblogs

注意上面带红色的目录名，maven项目采用“约定优于配置”的原则，src/main/java约定用于存放源代码，src/main/test用于存放单元测试代码，src/target用于存放编译、打包后的输出文件。这是全世界maven项目的通用约定，请记住这些固定的目录结构。

**三、编译项目**

先进入刚才创建项目的根目录

cd /d c:\test\maven-hello-world

然后执行 mvn clean compile

这样就能对项目进行编译了，编译后会自动在target目录中生成class文件，如果编译成功，会输出类似下面的信息

*yangjunmingmatoMacBook-Pro-7:maven-hello-world jimmy$ mvn clean compile*  
*[INFO] Scanning for projects...*  
*[INFO]*  
*[INFO] ------------------------------------------------------------------------*  
*[INFO] Building maven-hello-world 1.0-SNAPSHOT*  
*[INFO] ------------------------------------------------------------------------*  
*[INFO]*  
*[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ maven-hello-world ---*  
*[INFO]*  
*[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ maven-hello-world ---*  
*[INFO] Using 'UTF-8' encoding to copy filtered resources.*  
*[INFO] skip non existing resourceDirectory /Users/jimmy/Desktop/study/maven-hello-world/src/main/resources*  
*[INFO]*  
*[INFO] --- maven-compiler-plugin:2.5.1:compile (default-compile) @ maven-hello-world ---*  
*[INFO] Compiling 1 source file to /Users/jimmy/Desktop/study/maven-hello-world/target/classes*  
*[INFO] ------------------------------------------------------------------------*  
*[INFO] BUILD SUCCESS*  
*[INFO] ------------------------------------------------------------------------*  
*[INFO] Total time: 1.515s*  
*[INFO] Finished at: Wed Jan 01 19:08:32 CST 2014*  
*[INFO] Final Memory: 10M/156M*  
*[INFO] ------------------------------------------------------------------------*

**四、单元测试**

mvn clean test

这样就能做单元测试了，so easy !

如果单元测试不通过，会提示出错信息，注意看输出。

注：从输出上可以发现，test前，会先执行compile，即先编译，再执行单元测试.

有兴趣的朋友，可以修改下/src/test/java/cnblogs/AppTest.java里的内容，把testApp()方法中的assertTrue( true );改成assertTrue( false );再跑下单元测试，看下有什么不同

**五、项目打包**

通常我们会把java项目打包成jar包或war包，maven中打包的命令为

mvn clean package

运行完后，会在target目录下生成jar包

注：从输出 可以发现，package前，会先执行compile，再执行test，最后才是package打包

**六、项目运行**

该项目中的App.java中有main方法，可以直接运行，常规方式下，我们如果想直接运行class文件，得敲一段很长的命令，maven中不必这么复杂，先用记事本打开项目根目录下的pom.xml文件，增加下面这节内容：

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif View Code

上述这段内容插入在</project>之前即可。

然后在命令行下，输入

mvn exec:exec

即可直接运行，下面是输出：

*C:\test\maven-hello-world>mvn exec:exec*  
*[INFO] Scanning for projects...*  
*[INFO]*  
*[INFO] ------------------------------------------------------------------------*  
*[INFO] Building maven-hello-world 1.0-SNAPSHOT*  
*[INFO] ------------------------------------------------------------------------*  
*[INFO]*  
*[INFO] --- exec-maven-plugin:1.2.1:exec (default-cli) @ maven-hello-world ---*  
*Hello World!*  
*[INFO] ------------------------------------------------------------------------*  
*[INFO] BUILD SUCCESS*  
*[INFO] ------------------------------------------------------------------------*  
*[INFO] Total time: 1.175s*  
*[INFO] Finished at: Mon Jan 13 22:35:02 CST 2014*  
*[INFO] Final Memory: 6M/111M*  
*[INFO] ------------------------------------------------------------------------*

**七、项目部署**

如果是web项目，使用命令

mvn clean jboss-as:deploy

就能自动将web项目部署到jboss中（前提是jboss web server已经成功启动），因为刚才我们创建的是一个最基本的maven项目，并非web项目，所以执行这条命令，应该会失败，后面会讲如何在eclipse中用插件部署web项目，这里可以先跳过。

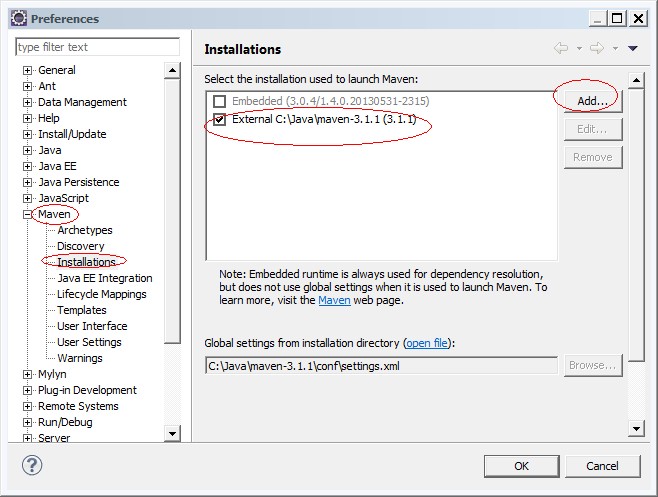
另外：

有时候，我们的项目是一个类库，只是封装一些方法供其它项目引用，对于这种项目，我们可以用 mvn clean install 把生成的jar包，安装到“本地仓库”中，这样本机其它项目需要使用该jar包时，只要在pom里配置依赖项即可，不用把jar包复制到当前项目中。

**八、eclipse中集成maven**

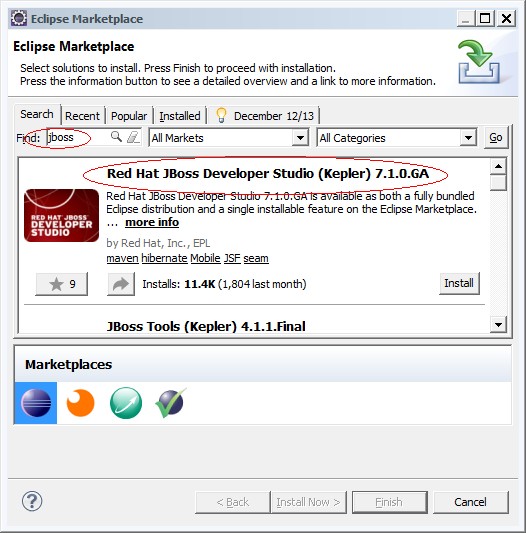
8.1 修改eclipse中使用的maven版本

eclipse自带了maven3.0.4，如果想使用最新的3.1.1，可以在eclipse -> window -> perferences -> Maven -> Installations 里参考下图设置一下



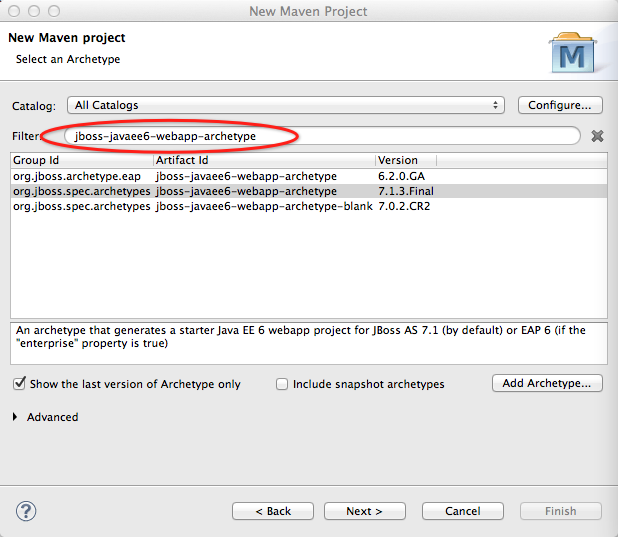
8.2 安装Red Hat JBoss Developer Studio(可选，纯个人喜欢)

eclipse -> help -> Eclipse Marketplace 搜索jboss，找到Red Hat JBoss Developer Studio，安装即可

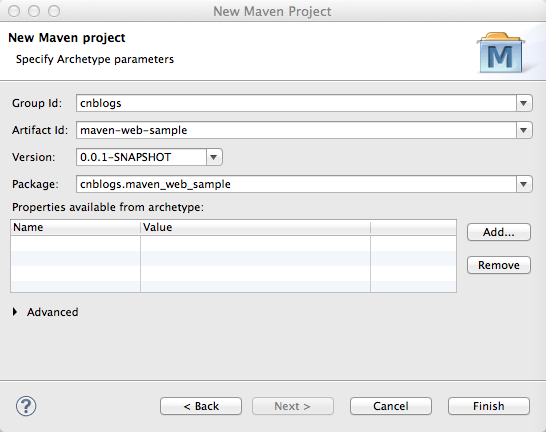


8.3 创建一个web的maven示例项目

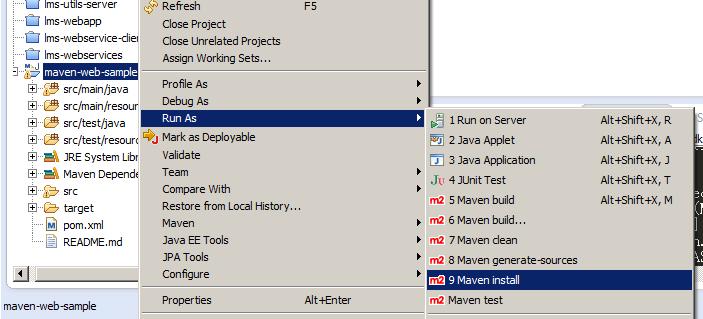
File -> New -> Maven Project -> Next ,在接下来的窗口中会看到一大堆的项目模板，Filter栏里输入“jboss-javaee6-webapp-archetype”,参考下图



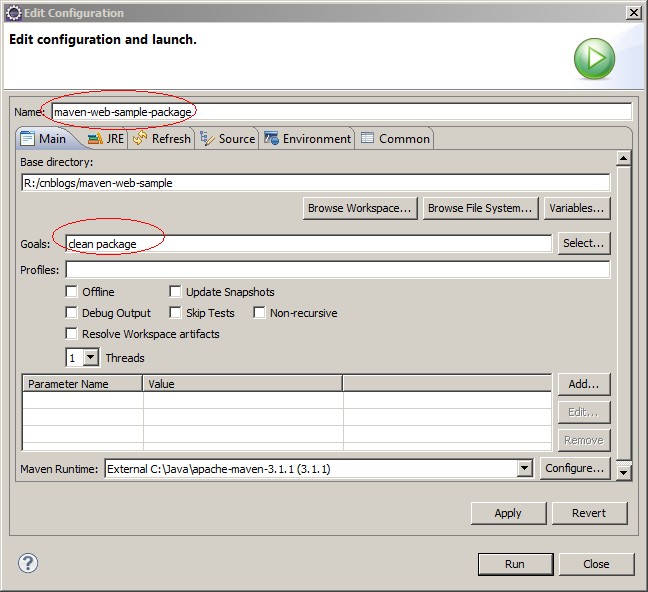
然后Next，在接下来的界面中填写groupId之类



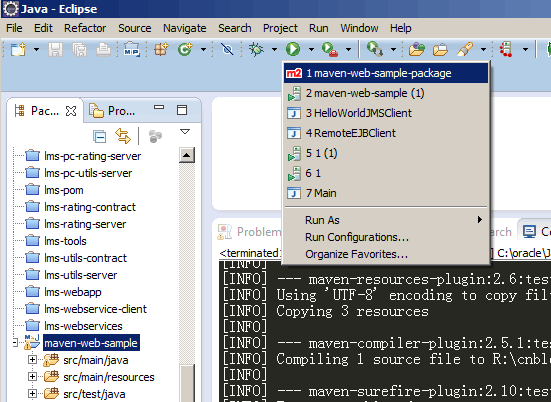
然后Finish完成项目创建。整个过程其实跟前面提到的纯手式命令行完全类似，只不过借助于插件，以图形化的方式更友好而已，直接Finish



项目上右击->Run As 就能看到很多Maven的选项，顾名思义，如何选择Maven install，等效于命令行执行mvn clean install，其它几项的含义一看便知，一般来讲Maven test 和 Maven install这二个基本上已经够用，如果还想定义自己的命令，比如 maven clean package，可以选择“6 Maven build...”，会弹出一个界面，如下图，在name输入框填写名称:maven-web-sample-package(名字可以随便写，只要方便自己记忆就行)，Goals栏输入clean package



最后点击Run关闭这个窗口，以后可以直接从工具栏调出这个选择，如下图：



实际使用中，发现eclipse与maven之间并非整合得天衣无缝，有时候eclipse中项目图标经常会莫名其妙的出现一些红叉，以下是一些个人经验：

a) 尝试先到Dos命令行窗口，用mvn clean compile(或install)之类的命令试下能否正常编译

b) 如果a) 步骤能正常编译，在回到eclipse中，项目右击->Maven-Update Project ，通常这样就能消除红叉

c) 有时候b)步骤操作完以后，仍然会有红色感叹号或红叉，但是eclipse编译、调试都OK，这就要具体问题具体分析了，可以尝试把Problems面板调出来，逐个分析

d) 以上3个步骤都尝试无效后，最后还可以尝试 Windows -> Preferences -> Validation -> Disable All 把所有验证都关掉

[下一篇](http://www.cnblogs.com/yjmyzz/p/3519373.html)，将介绍如何搭建"仓库私服"等其它内容