1 下载源代码：mvn dependency:sources -DdownloadSources=true -DdownloadJavadocs=true

[maven常用命令介绍](http://www.cnblogs.com/lexus/archive/2012/01/26/2329544.html)

这里主要是在eclipse中使用maven，因此只使用到了一部分命令，整理下来方便以后查阅。

生成清除Eclipse项目结构：  
mvn eclipse:eclipse  
mvn eclipse:clean

清理（删除target目录下编译内容）  
mvn clean

仅打包Web页面文件  
mvn war:exploded

编译项目  
mvn compile

打包发布  
mvn package

打包时跳过测试  
mvn package -Dmaven.test.skip=ture

还有很多命令目前还没有使用到，以后遇到再补充

本文地址：http://blog.csdn.net/kongxx/article/details/6993501

Maven用了很久了，命令一直记不住，其实想想就那个几个常用的，今天写下来，帮着记忆吧

* 创建一个简单的Java工程：mvn archetype:create -DgroupId=com.mycompany.example -DartifactId=Example
* 创 建一个java的web工程：mvn archetype:create -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-webapp -DgroupId=com.mycompany.app -DartifactId=my-webapp
* 打包：mvn package
* 编译：mvn compile
* 编译测试程序：mvn test-compile
* 清空：mvn clean
* 运行测试：mvn test
* 生成站点目录: mvn site
* 生成站点目录并发布：mvn site-deploy
* 安装当前工程的输出文件到本地仓库: mvn install
* 安 装指定文件到本地仓库：mvn install:install-file -DgroupId=<groupId> -DartifactId=<artifactId> -Dversion=1.0.0 -Dpackaging=jar -Dfile=<myfile.jar>
* 查看实际pom信息: mvn help:effective-pom
* 分析项目的依赖信息：mvn dependency:analyze 或 mvn dependency:tree
* 跳过测试运行maven任务：    mvn -Dmaven.test.skip=true XXX
* 生成eclipse项目文件: mvn eclipse:eclipse
* 查看帮助信息：mvn help:help 或 mvn help:help -Ddetail=true
* 查看插件的帮助信息：mvn <plug-in>:help，比如：mvn dependency:help 或 mvn ant:help 等等。

**常用命令**

**1. 创建Maven的普通java项目：   
   mvn archetype:create   
   -DgroupId=packageName   
   -DartifactId=projectName    
2. 创建Maven的Web项目：     
    mvn archetype:create   
    -DgroupId=packageName      
    -DartifactId=webappName   
    -DarchetypeArtifactId=maven-archetype-webapp      
3. 编译源代码： mvn compile   
4. 编译测试代码：mvn test-compile      
5. 运行测试：mvn test     
6. 产生site：mvn site     
7. 打包：mvn package     
8. 在本地Repository中安装jar：mvn install   
9. 清除产生的项目：mvn clean     
10. 生成eclipse项目：mvn eclipse:eclipse    
11. 生成idea项目：mvn idea:idea    
12. 组合使用goal命令，如只打包不测试：mvn -Dtest package     
13. 编译测试的内容：mvn test-compile    
14. 只打jar包: mvn jar:jar    
15. 只测试而不编译，也不测试编译：mvn test -skipping compile -skipping test-compile   
      ( -skipping 的灵活运用，当然也可以用于其他组合命令)    
16. 清除eclipse的一些系统设置:mvn eclipse:clean**

**http://cloudera.iteye.com/blog/424795**

[**maven 简单实用教程**](http://cloudera.iteye.com/blog/424795)

**博客分类：**

* [java ee 技术](http://cloudera.iteye.com/category/61475)

[maven](http://www.iteye.com/blogs/tag/maven)[Tomcat](http://www.iteye.com/blogs/tag/Tomcat)[Eclipse](http://www.iteye.com/blogs/tag/Eclipse)[Hibernate](http://www.iteye.com/blogs/tag/Hibernate)[项目管理](http://www.iteye.com/blogs/tag/%E9%A1%B9%E7%9B%AE%E7%AE%A1%E7%90%86)

**1. Maven介绍**

**1.1. 简介**

java编写的用于构建系统的自动化工具。

目前版本是2.0.9，注意maven2和maven1有很大区别，阅读第三方文档时需要区分版本。

**1.2. Maven资源**

* 见[官方网站](http://maven.apache.org/)；
* [The 5 minute test](http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html)，官方简易入门文档；
* [Getting Started Tutorial](http://maven.apache.org/guides/getting-started/index.html)，官方入门文档；
* [Build Cookbook](http://maven.apache.org/users/cookbook/index.html)，官方的cookbook；
* [POM Reference](http://maven.apache.org/pom.html)，POM文件的设置参考
* [Settings Reference](http://maven.apache.org/settings.html)，settings文件的设置参考
* [Better Builds with Maven](http://www.devzuz.com/web/guest/products/resources)，免费的电子书，下载需要注册。

**1.3. Maven和Ant的区别**

Maven正在逐渐取代Ant，很多java开源软件（Spring、Struts2 ……）已经使用maven。

* 不需要写复杂的处理脚本；
* 声明式的类库依赖管理。

**1.4. Maven的基本功能**

* 构建：比如生成class、jar、war或者ear文件
* 生成文档：比如生成javadoc、网站文档
* 生成报告：比如junit测试报告
* 生成依赖类库：生成文档，说明项目多其他软件的依赖
* 有关SCM：SCM（Software Configuration Management），软件配置管理，比如版本控制，比如bug管理等等
* 发布：生成供发布的分发包，比如生成Struts2的分发包，供提交给用户使用
* 部署：比如，web应用程序，自动部署到指定的服务器上

通过[我写的商品管理的小例子](http://code.google.com/p/simplelife/wiki/SimpleLife_Product)，演示结合maven和svn的功能。

**2. Maven使用**

**2.1. Maven的安装和配置**

* 从官方网站下载最新的Maven分发包<http://maven.apache.org/download.html>，当前为2.0.9；
* 解压缩到本地；
* 配置maven, 将maven/bin目录设置到windows环境变量Path中
* 检查maven是否安装成功, 在命令行中执行
* mvn -version

**2.2. Maven的基本使用**

*Maven的基本使用*介绍通过*命令行*编写简单的java和web项目。

**2.2.1. 实现Java项目**

通过maven在命令行下创建普通java项目，也就是main方法执行的项目或者jar文件的类库。

**2.2.1.1. 创建Maven项目**

执行：

mvn archetype:generate

在交互界面中：

* Choose a number: 回车即可，也就是选择15
* Define value for groupId: 输入组织id，比如easymorse.com
* Define value for artifactId：输入项目名称，比如helloworld
* Define value for version: 输入版本号，可以直接回车，默认是1.0-SNAPSHOT
* Define value for package: java的包名，比如com.easymorse
* 然后回车表示确认上述输入即可。

观察helloworld目录（Define value for artifactId输入的项目名称）下生成的文件和目录：

* 项目构建文件: pom.xml
* 代码框架: src\main\java\com\easymorse\App.java
* 测试代码: src\test\java\com\easymorse\[AppTest](http://localhost:9922/AppTest).java

**2.2.1.2. 运行Maven项目**

命令行进入helloworld目录Define value for artifactId输入的项目名称）。

**项目打包**

mvn package

检查命令生成了什么？

* target目录
* 编译了代码
* 编译了测试代码
* 使用junit测试并生成的报告
* 生成代码的jar文件

运行打包的jar文件:

java -cp target\helloworld-1.0-SNAPSHOT.jar com.easymorse.App

**编译源程序**

mvn compile

**编译并测试**

mvn test

**清空生成的文件**

mvn clean

**将maven项目转化为eclipse项目**

命令行运行：

mvn eclipse:eclipse

打开eclipse，菜单选择：file>import>general>existing projects into workspace，在对话框中选中目录，导入即可。

如果要清除有关eclipse项目的配置信息：

mvn -Dwtpversion=1.0 eclipse:clean

**联合使用**

mvn eclipse:clean clean

**2.2.2. 实现Web项目**

通过maven在命令行下创建java web项目。

**2.2.2.1. 创建Maven项目**

在命令行输入，这一步和创建java项目类似：

mvn archetype:generate

交互步骤说明：

* Choose a number: 回车即可，也就是选择18，这里和java普通项目不一样
* Define value for groupId: 输入组织id，比如easymorse.com
* Define value for artifactId：输入项目名称，比如helloworld
* Define value for version: 输入版本号，可以直接回车，默认是1.0-SNAPSHOT
* Define value for package: java的包名，比如com.easymorse
* 然后回车表示确认上述输入即可。

需要在pom.xml文件中增加servlet容器的插件：

<build>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>tomcat-maven-plugin</artifactId>

</plugin>

<plugin>

<groupId>org.mortbay.jetty</groupId>

<artifactId>maven-jetty-plugin</artifactId>

<version>6.1.6</version>

</plugin>

<plugin>

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

<configuration>

<source>1.6</source>

<target>1.6</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

</plugins>

</build>

* tomcat插件
* jetty插件
* 编译插件的配置

**repository目录的作用**

repository的位置，在用户目录的.m2目录下。

repository目录的作用，对依赖类库的缓存。

**2.2.2.2. 运行Maven项目**

**项目打包**

mvn package

**启动tomcat**

mvn tomcat:run

**启动jetty**

mvn jetty:run

**转化为eclipse项目**

mvn -Dwtpversion=1.5 eclipse:eclipse

这样生成wtp插件的web项目。

打开eclipse，菜单选择：file>import>general>existing projects into workspace，在对话框中选中目录，导入即可。

另外，需要在eclipse里创建一个classpath变量，名称为：M2\_REPO，值为系统用户下.m2/repository目录。

**3. POM文件的基本配置**

**3.1. POM介绍**

**3.1.1. 什么是POM**

Project Object Model，项目对象模型。

通过xml格式保存的pom.xml文件。

作用类似ant的build.xml文件，功能更强大。

该文件用于管理：源代码、配置文件、开发者的信息和角色、问题追踪系统、组织信息、项目授权、项目的url、项目的依赖关系等等。

**3.1.2. 快速浏览**

一个完整的pom.xml文件，放置在项目的根目录下。

<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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<!-- The Basics -->

<groupId>...</groupId>

<artifactId>...</artifactId>

<version>...</version>

<packaging>...</packaging>

<dependencies>...</dependencies>

<parent>...</parent>

<dependencyManagement>...</dependencyManagement>

<modules>...</modules>

<properties>...</properties>

<!-- Build Settings -->

<build>...</build>

<reporting>...</reporting>

<!-- More Project Information -->

<name>...</name>

<description>...</description>

<url>...</url>

<inceptionYear>...</inceptionYear>

<licenses>...</licenses>

<organization>...</organization>

<developers>...</developers>

<contributors>...</contributors>

<!-- Environment Settings -->

<issueManagement>...</issueManagement>

<ciManagement>...</ciManagement>

<mailingLists>...</mailingLists>

<scm>...</scm>

<prerequisites>...</prerequisites>

<repositories>...</repositories>

<pluginRepositories>...</pluginRepositories>

<distributionManagement>...</distributionManagement>

<profiles>...</profiles>

</project>

**3.2. 基本设置**

**3.2.1. 协作关系**

<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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.codehaus.mojo</groupId>

<artifactId>my-project</artifactId>

<version>1.0</version>

<packaging>war</packaging>

</project>

* groupId : 组织标识，例如：org.codehaus.mojo，在M2\_REPO目录下，将是: org/codehaus/mojo目录。
* artifactId : 项目名称，例如：my-project，在M2\_REPO目录下，将是：org/codehaus/mojo/my-project目录。
* version : 版本号，例如：1.0，在M2\_REPO目录下，将是：org/codehaus/mojo/my-project/1.0目录。
* packaging : 打包的格式，可以为：pom , jar , maven-plugin , ejb , war , ear , rar , par

**3.2.2. POM间关系**

**3.2.2.1. 依赖关系**

依赖关系列表（dependency list）是POM的重要部分。

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

...

</dependencies>

* groupId , artifactId , version :
* scope : compile(default),provided,runtime,test,system
* exclusions

**如何查到依赖的类库？**

一般可以通过这个网站：[http://www.mvnrepository.com](http://www.mvnrepository.com/)

比如查询hibernate，可以找到结果列表中的hibernate类库条目。

点击：[http://www.mvnrepository.com/artifact/org.hibernate/hibernate，](http://www.mvnrepository.com/artifact/org.hibernate/hibernate%EF%BC%8C)

选择版本，比如3.2.6ga，即：<http://www.mvnrepository.com/art> ... /hibernate/3.2.6.ga

复制文章中的：

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate</artifactId>

<version>3.2.6.ga</version>

</dependency>

到pom.xml文件中即可。

**是否还需要找到hibernate依赖的pom？**

不需要，hibernate也会有pom，maven会通过它的pom自动找到它依赖的类库。

**3.2.2.2. 继承关系**

继承其他pom.xml配置的内容。

maven提供了一个类似java.lang.Object的顶级父pom.xml文件。

可以通过下面命令查看当前pom.xml受到超pom.xml文件的影响：

mvn help:effective-pom

创建一个各种项目可复用的pom.xml文件：<http://easymorse.googlecode.com/svn/trunk/pom/pom.xml>

部署要复用的pom.xml文件：

mvn install

在自己的pom文件中继承上述pom：

<parent>

<groupId>com.easymorse</groupId>

<artifactId>pom</artifactId>

<version>0.1</version>

</parent>

**3.2.2.3. 聚合关系**

用于将多个maven项目聚合为一个大的项目。

比如目录结构如下：

.

|-- pom.xml

|-- module-a

`-- pom.xml

|-- module-b

`-- pom.xml

|-- module-c

`-- pom.xml

|-- foo-all

`-- pom.xml

那么总的pom.xml文件类似：

...

<modules>

<module>module-a</module>

<module>module-b</module>

<module>module-c</module>

<module>foo-all</module>

</modules>

</project>

参考文档：<http://maven.apache.org/plugins/maven-eclipse-plugin/reactor.html>

原文示例有误，见：[modules.rar](http://localhost:9922/work/training/java/maven/get_start?action=AttachFile&do=get&target=modules.rar)

**3.2.3. 其他配置**

maven的属性，是值的占位符，类似EL，类似ant的属性，比如${X}，可用于pom文件任何赋值的位置。

有以下分类：

* env.X：操作系统环境变量，比如${env.PATH}
* project.x：pom文件中的属性，比如：<project><version>1.0</version></project>，引用方式：${project.version}
* settings.x：settings.xml文件中的属性，比如：<settings><offline>false</offline></settings>，引用方式：${settings.offline}
* Java System Properties：java.lang.System.getProperties()中的属性，比如java.home，引用方式：${java.home}
* 自定义：在pom文件中可 以：<properties><installDir>c:/apps/cargo-installs< /installDir></properties>，引用方式：${installDir}

**3.3. 其他**

**3.3.1. 设置离线**

在.m2目录下创建settings.xml文件（如果没有的话）

<settings 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/settings-1.0.0.xsd">

<offline>true</offline>

</settings>

**3.3.2. 安装第三方包**

经常有第三方包，因为一些原因，在网上repository上没有，需要自己动手安装。

比如sun某些版本的jar文件，比如oracle的驱动。

已oracle驱动程序为例，比如驱动路径为c:/driver/ojdbc14.jar，是10.2.0.3.0版本

在该网址能够查到：<http://www.mvnrepository.com/artifact/com.oracle/ojdbc14> artifactId和groupId。

mvn install:install-file -DgroupId=com.oracle -DartifactId=ojdbc14 -Dversion=10.2.0.3.0 -Dpackaging=jar -Dfile=c:/driver/ojdbc14.jar

这样就可以在pom中依赖引用了：

<dependency>

<groupId>com.oracle</groupId>

<artifactId>ojdbc14</artifactId>

<version>10.2.0.3.0</version>

</dependency>

**3.3.3. 部署到tomcat**

tomcat配置有管理权限的用户：conf\tomcat-users.xml

<?xml version='1.0' encoding='utf-8'?>

<tomcat-users>

<role rolename="manager"/>

<user username="marshal" password="password" roles="manager"/>

</tomcat-users>

在pom文件的tomcat插件中添加：

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>tomcat-maven-plugin</artifactId>

<configuration>

<url>http://localhost:8080/manager</url>

<server>myserver</server>

<path>/mycontext</path>

</configuration>

</plugin>

在.m2/settings.xml文件中增加：

<settings 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/settings-1.0.0.xsd">

<servers>

<server>

<id>myserver</id>

<username>marshal</username>

<password>password</password>

</server>

</servers>

</settings>

运行打包部署，在maven项目目录下：

mvn tomcat:deploy

然后访问：<http://localhost:8080/mycontext/> 即可。

撤销部署：

mvn tomcat:undeploy

启动web应用：

mvn tomcat:start

停止web应用：

mvn tomcat:stop

重新部署：

mvn tomcat:redeploy

部署展开的war文件：

mvn war:exploded tomcat:exploded

**3.3.4. 未讲到的内容**

* settings.xml的配置
* pom.xml的详细配置
* 自定义插件的方法：<http://maven.apache.org/plugins/maven-archetype-plugin/examples/archetype.html>
* 自定义goal的执行：<preGoal><postGoal>
* 插件的查询和使用
* 搭建镜像repository
* 在maven中调用ant