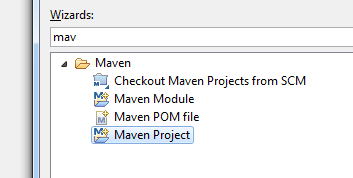
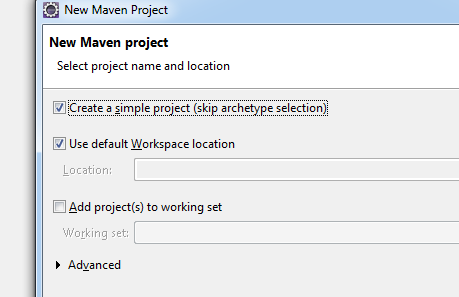
**使用eclipse插件创建一个web project**

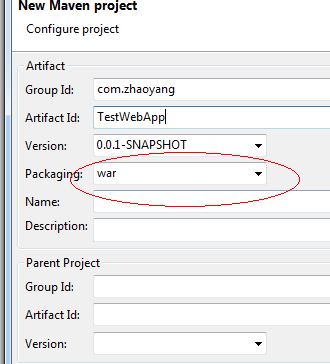
**首先创建一个Maven的Project如下图**



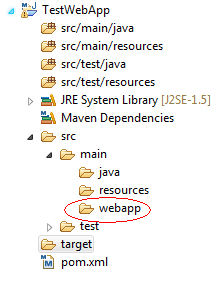
**我们勾选上Create a simple project （不使用骨架）**



**这里的Packing 选择 war的形式**

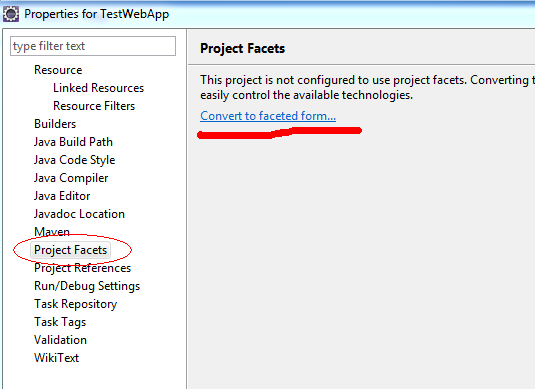


**由于packing是war包，那么下面也就多出了webapp的目录**

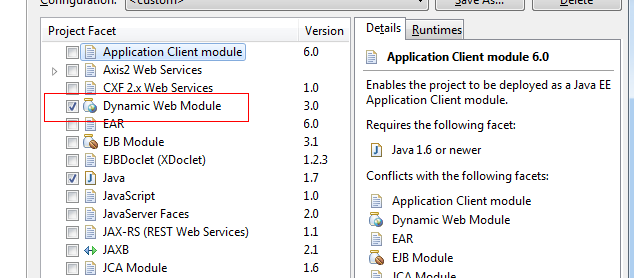


**由于我们的项目要使用eclipse发布到tomcat下面，这里我们需要先把项目转成dynamic web project**

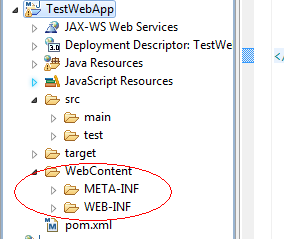
**在我们的项目上点击右键，选择properties 并找到 Project Facets ，并点击Convert to faceted form…   如下图：**



**然后勾选Dynamic Web Module 并点击ok  如下图：(3.0只有tomcat7才支持)**

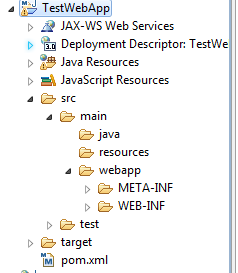


**接下来观察我们的项目结构，多了一个web content目录**

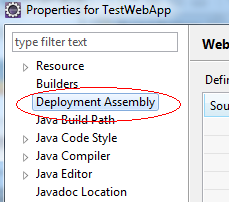


**虽然此时我们可以发布到tomcat中，但这不符合maven的结构，我们还要做如下修改**

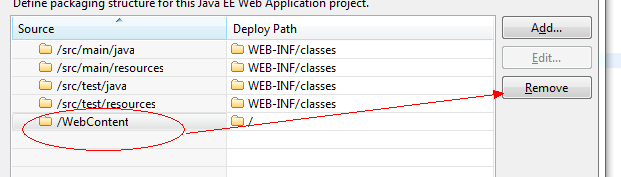
**把上图WebContent下面两个目录 META-INF ，WEB-INF 直接剪切到src/main/webapp目录下，并删掉WebContent目录，那么现在的项目结构如下图：**



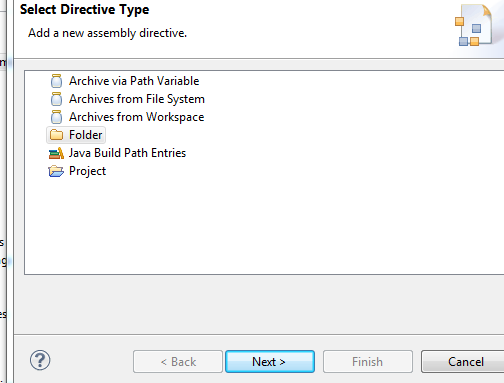
**然后我们要修改发布规则，右键点击项目， 选择 Deployment Assembly**



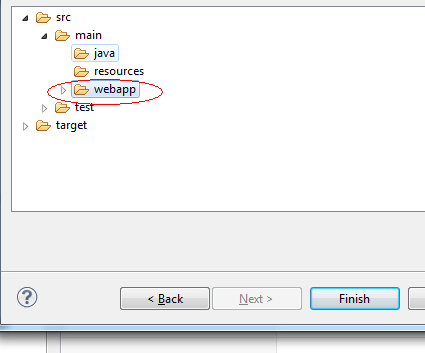
**选择WebContent，把它remove掉，测试类我们也不需要发布，test的两个目录页可以remove**



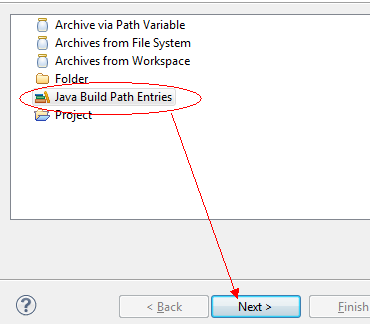
**接着重新指定一个web的路径，点击add，选择Folder -- 〉  next**



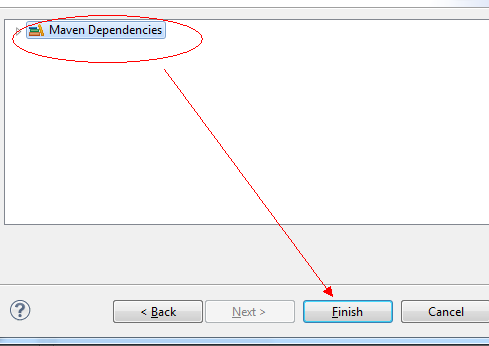
**在src下找到webapp目录，然后finish**



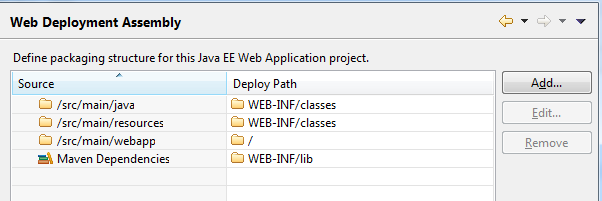
**最后一步，我们要把当前的build path 指向 Maven Dependency， 直接点击add，选择Java Build Path Entries 然后next**



**然后再点击finish完成**



**完成后如下图：**



**至此一个基于maven的webapp就建立好了，并可以直接从eclipse中发布到tomcat中**

补充：我们需要在src/main/webapp/WEB-INF下面创建一个web.xml

**导入我们的Spring mvc依赖jar包**

<dependencies>

    <dependency>

        <groupId>org.springframework</groupId>

        <artifactId>spring-core</artifactId>

        <version>3.0.7.RELEASE</version>

    </dependency>

    <dependency>

        <groupId>org.springframework</groupId>

        <artifactId>spring-web</artifactId>

        <version>3.0.7.RELEASE</version>

    </dependency>

    <dependency>

        <groupId>org.springframework</groupId>

        <artifactId>spring-webmvc</artifactId>

        <version>3.0.7.RELEASE</version>

    </dependency>

    <dependency>

        <groupId>org.codehaus.jackson</groupId>

        <artifactId>jackson-mapper-asl</artifactId>

        <version>1.7.1</version>

    </dependency>

  </dependencies>

**直接保存，maven就会自动为我们下载所需jar文件**

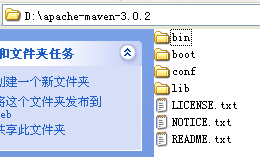
最后一步,将项目发布到tomcat上:   
点击eclipse中servers->new->server,全部finish以后双击该server,切换到modules窗口->点击add external module->   
在document base中browse到项目webapp路径   
在path中输入"/项目名称"   
点击OK后配置全部完成,这样配置的好处不只是热部署,因为eclipse自动编译java文件经常出现问题,在这样的情况下随时可以在项目根路径下用mvn compile命令编译项目

# Maven命令使用

Maven是一个采用纯Java编写的开 源项目管理工具。Maven采用了一种被称之为project object model (POM)概念来管理项目，所有的项目配置信息都被定义在一个叫做POM.xml的文件中，通过该文件，Maven可以管理项目的整个声明周期，包括编 译，构建，测试，发布，报告等等。目前Apache下绝大多数项目都已经采用Maven进行管理。而Maven本身还支持多种插件，可以方便更灵活的控制 项目。

## 1：http://maven.apache.org/download.html 下载最新版本 Maven 3.0.2 (Binary zip) http://dl.iteye.com/upload/attachment/410248/903c257f-ffe2-3224-83d4-8315c3ec4400.png

## **2：解压到 D:**

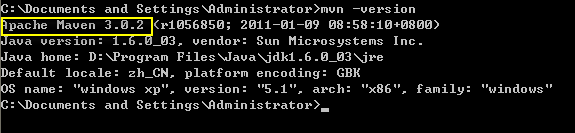


## **3：配置环境变量**

   MAVEN\_HOME : D:\apache-maven-3.0.2   
   MAVEN : %MAVEN\_HOME%\bin   
  (可选） MAVEN\_OPTS : -Xms256m -Xmx512m

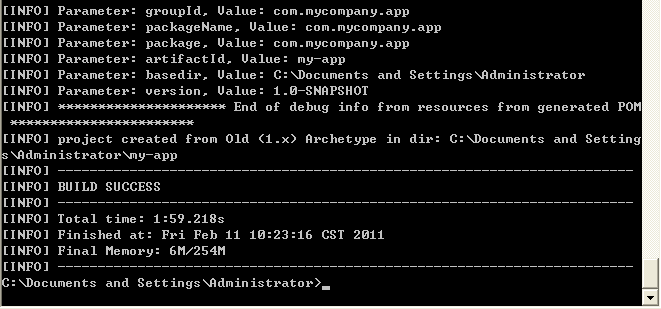
path 前面 加上 %MAVEN%;   


## **4：验证安装是否成功**

在命令行上输入 ： mvn -version; 回车，如看到下面信息表示安装成功：   


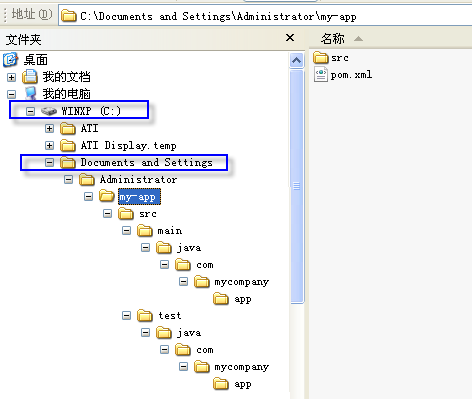
## **5：创建项目**

在命令行上输入 ：mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=my-app 回车，   
  
http://dl.iteye.com/upload/attachment/410281/63c26a8f-7fe0-3b43-bc66-7047af1b6834.png  
  
如果你是首次运行该命令(goal)，maven将要花一些时间去把最新的工具包(Maven 把它叫作artifacts)下载到你的本地仓库。



如果出现创建失败，先把repository文件夹下的（除cache）删除，然后运行上述命令。

命令执行完后你将看到maven生成了一个名为my-app的目录，这个名字就是你在命令中指定的artifactId,进入该目录，你将发现以下标准的项目结构：



其中:src/main/java 目录包含了项目的源代码，src/test/java 目录包含了项目的测试代码，pom.xml是项目的项目对象模型（Project Object Model or POM）。

## 6：POM

pom.xml 文件是maven对一个项目的核心配置，这个文件将包含你希望如何构建项目的大多数配置信息。POM大而复杂，但你不用全部去了解，只要使用一些常用的配置就可以了。下面列出这个POM的内容：

引用

|  |
| --- |
| <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.mycompany.app</groupId>    <artifactId>my-app</artifactId>    <version>1.0-SNAPSHOT</version>    <packaging>jar</packaging>     <name>my-app</name>    <url>http://maven.apache.org</url>     <properties>      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>    </properties>     <dependencies>      <dependency>        <groupId>junit</groupId>        <artifactId>junit</artifactId>        <version>3.8.1</version>        <scope>test</scope>      </dependency>    </dependencies>  </project> |

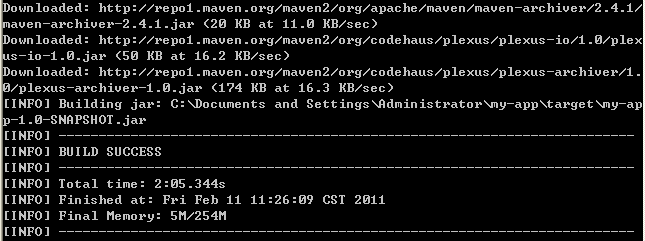
## 7：第5步我们做了什么

引用

mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=my-app

执行Maven 命令（goal）archetype:create，并且设置了一些参数( -DgroupId=com.mycompany.app -DartifactId=my-app)

## 8：Build 项目

在命令行上 输入 ： cd my-app 回车，进入到 项目路径下   
  
http://dl.iteye.com/upload/attachment/410299/0b4337f1-8cc0-380a-9fa5-1fbef0c13c74.png  
  
再输入 mvn package 回车这时命令行将会打印出各种动作，并且以下面一段信息结束：   
  


与首次执行的命令(**那是一个goal**)

引用

archetype:create

不同,这次执行的只是一个简单的命令---package。不同于goal,**这是一个phase(阶段)**，一个phase是指构建生命周期的一个阶段，构建生命周期是指有序的一系列phase。当给出一个phase，Maven将执行所有的在此阶段前的phase及其自身，例如，如果我们执行compile阶段，实际上执行的阶段有：

**引用**

|  |
| --- |
| validate  generate-sources  process-sources  generate-resources  process-resources  compile |

你可以使用以下的命令来测试新编译和打包出来的jar包，

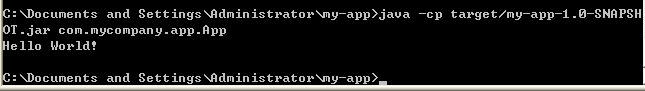
**引用**

|  |
| --- |
| java -cp target/my-app-1.0-SNAPSHOT.jar com.mycompany.app.App |

这时将打印出最经典的：

**引用**

Hello World!



## 9：运行Maven工具

虽然很难列出一张非常全面的表，但在此可先列出最普通的默认的生命周期阶段:

**引用**

|  |
| --- |
| **validate：**验证工程是否正确，所有需要的资源是否可用。  **compile：**编译项目的源代码。    **test：**使用合适的单元测试框架来测试已编译的源代码。这些测试不需要已打包和布署。  **Package：**把已编译的代码打包成可发布的格式，比如jar。  **integration-test：**如有需要，将包处理和发布到一个能够进行集成测试的环境。  **verify：**运行所有检查，验证包是否有效且达到质量标准。  **install：**把包安装在本地的repository中，可以被其他工程作为依赖来使用。  **Deploy：**在集成或者发布环境下执行，将最终版本的包拷贝到远程的repository，使得其他的开发者或者工程可以共享。  **clean：**清除先前构建的artifacts（在maven中，把由项目生成的包都叫作artifact）。  **site：**为项目生成文档站点。 |

上面列出的各个Phases（阶段）实际上对应着潜在的goals，特殊的每个阶段执行的特殊的goals由项目的类型所决定，例如：如果项目的 类型是jar,package阶段将默认执行jar:jar的goals,如果项目类型是war,那么package阶段执行的goals将是 war:war。   
有件有趣的事情需要注意，那就是phases和goals需要按一定的顺序来执行。   
mvn clean dependency:copy-dependencies package   
这个命令将先清除项目，然后拷贝依赖，最后把项目打包，当然，在打包之前，会先执行此阶段之前的阶段.如compile,test等.   
生成站点   
mvn site   
这个阶段生成基于pom.xml配置的项目信息。你可以在target/site目录下看到生成的文档。