**持续集成环境搭建**

目录

[1. Maven3安装 1](#_Toc328751189)

[1.1下载maven 1](#_Toc328751190)

[1.2安装maven 2](#_Toc328751191)

[2.Maven安装 Eclipse3.7插件 2](#_Toc328751192)

[2.1插件地址 2](#_Toc328751193)

[2.2安装时出现问题 2](#_Toc328751194)

[3.搭建局域网Mave服务器 3](#_Toc328751195)

[3.1下载Nexus 3](#_Toc328751196)

[3.2安装Nexus 3](#_Toc328751197)

[3.3运行Nexus 4](#_Toc328751198)

[3.4配置Nexus 6](#_Toc328751199)

[4. 搭建SVN服务器 8](#_Toc328751200)

[4.1安装SVN 8](#_Toc328751201)

[4.2使用SVN 9](#_Toc328751202)

[5.maven搭建eclipse工程 10](#_Toc328751203)

[5.1构建Project 10](#_Toc328751204)

[5.2部署maven项目 17](#_Toc328751205)

[6.jenkins环境搭建 19](#_Toc328751206)

[6.1安装 19](#_Toc328751207)

[6.2 Jenkins构建项目 20](#_Toc328751208)

# 1. Maven3安装

## 1.1下载maven

版本：3.0.3

下载地址：http://maven.apache.org/download.html

## 1.2安装maven

JDK：jdk1.6

[1] 解压缩apache-maven-3.0.3-bin.zip 到你想安装的位置，如D:/maven3。

[2]设置Maven系统环境变量，M2\_HOME= D:/maven3/maven-3.0.3。

[3]添加Maven bin目录至系统环境变量PATH中， %M2\_HOME%\bin。

[4]确认Maven的安装：

   cmd > mvn -version

Apache Maven 3.0.3 (r1075438; 2011-03-01 01:31:09+0800)

Maven home: D:\maven3

Java version: 1.6.0\_18, vendor: Sun Microsystems Inc.

Java home: C:\jdk1.6\jre

Default locale: zh\_CN, platform encoding: GBK

OS name: "windows xp", version: "5.1", arch: "x86", family: "windows"

如果报错Exception in thread "main" java.lang.NoClassDefFoundError,解决：

    删除系统环境变量中M2\_HOME和统环境变量PATH中， %M2\_HOME%\bin，然后重新设置一遍就OK。 这是maven3.0.3的bug，当然如果用M2\_HOME以外的变量名就不会出现这个bug

# 2.[Maven安装 Eclipse3.7插件](http://www.iteye.com/blogs/tag/Maven%20%E5%AE%89%E8%A3%85%20Eclipse3.7%E6%8F%92%E4%BB%B6)

Eclipse：Eclipse 3.7

## 2.1插件地址

  [1]先安装gef插件 地址：http://download.eclipse.org/tools/gef/updates/interim/

   选择3.7版本下载，maven插件安装的时候会依赖这个插件

   [2]安装svn插件    地址：http://subclipse.tigris.org/update\_1.6.x

   [3]maven插件

    m2eclipse-core Update Site: http://m2eclipse.sonatype.org/sites/m2e

    m2eclipse-extras Update Site: http://m2eclipse.sonatype.org/sites/m2e-extras

   注意 ：必须两个插件都安装，不安装extras 插件时，点击“run as server”来运行项目时会报错：“the selected did not contain any resources that can run on server maven project”

## 2.2安装时出现问题

    直接在线安装maven2 会出现依赖插件找不到的问题，无法安装。必须先安装gef 插件后才能安装m2eclipse-core 插件，然而安装m2eclipse-extras 插件又依赖subclipse 插件。所以，三个插件的正确的安装顺序是：

gef插件->subclipse插件->m2eclipse插件

  m2eclipse-core插件 ——依赖gef插件

  m2eclipse-extras插件 ——依赖subclipse插件

安装完成后，启动eclipse

注意:安装m2eclipse插件后，重启Eclipse老是提示

Eclipse is running in a JRE, but a JDK is required

-vm配置项必须在 –vmargs配置项前，是因为执行到-vmargs配置项已经选择了一个默认的JVM了。

所以在–vmargs前面加上这一句：

-vm D:/Java/jdk1.6.0\_29/bin/javaw.exe

|  |
| --- |
| -vm  D:/dev/j2sdk6/bin/javaw.exe  -vmargs  -Dosgi.requiredJavaVersion=1.5 |

启动后测试一切正常。

# 3.搭建局域网Mave服务器

Nexus是Maven资源仓库管理器。如果在公司里使用Maven，通常应该在本地架设一个Maven资源仓库服务器，在代理远程资源仓库的同时维护本地资源仓库，以节省带宽和时间，Nexus就可以满足这样的需要。此外，Nexus还提供了强大的资源仓库管理功能，构件搜索功能，它基于REST，友好的UI是一个extjs的 REST客户端，它占用较少的内存，基于简单文件系统而非数据库。这些优点使其日趋成为最流行的Maven资源仓库管理器。

## 3.1下载Nexus

从http://nexus.sonatype.org/downloads/下载最新版本的Nexus，Nexus有两种版本：开源版本和专业版本。这里我们下载开源版本。

下载界面如下图所示：



## 3.2安装Nexus

安装Nexus只有一个前提，就是必须先安装JRE，Java 5或Java 6都可以（推荐使用Java 6）。

Nexus提供有两种安装方式，一种是内嵌Jetty的bundle绑定包，只要有JRE就能直接运行。第二种方式是WAR，只须简单的将其发布到web容器中即可使用。这里我们重点讲解第一种安装方式。这种方式可以将Nexus（开源版本）作为一个单独的服务器来安装。Nexus绑定有一个Jetty，Jetty监听主机（0.0.0.0）上所有配置的IP地址并默认运行在端口8081。

解压nexus-oss-webapp-1.9.1.1-bundle.zip 至任意目录，如D:\nexus，然后配置环境变量${NEXUS\_HOME}，指向目录D:\nexus\nexus-oss-webapp-1.9.1.1，此即为nexus的安装目录。

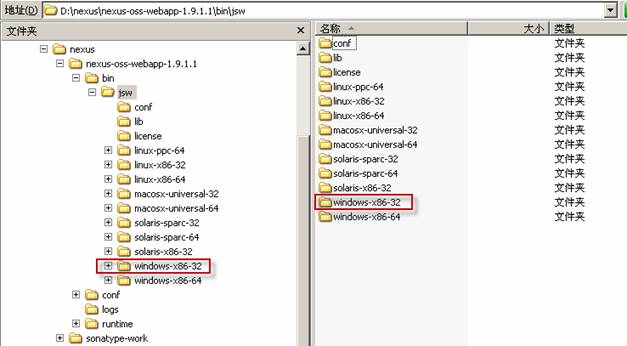
Nexus安装目录总有一个兄弟目录，名为“sonatype-work”。这个目录包含有所有资源库和Nexus的配置信息，这样配置在Nexus安装目录之外，使得Nexus的版本更新更加容易。默认情况下，该目录总是作为nexus安装目录的兄弟目录，用来保存资源内容和配置。如果想要改变该目录的位置，可以在文件${NEXUS\_HOME}/conf/plexus.properties中修改nexus-work属性。

## 3.3运行Nexus

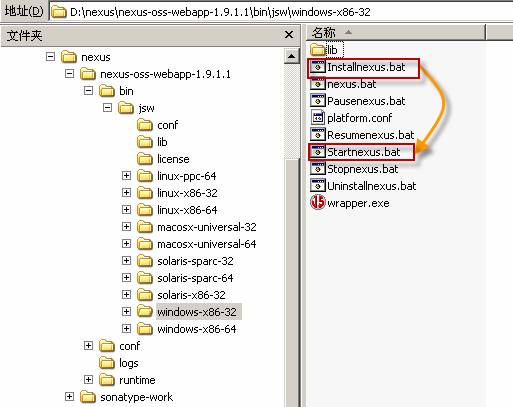
进入到以下目录位置：

D:\nexus\nexus-oss-webapp-1.9.1.1\bin\jsw

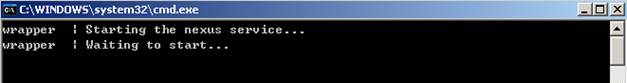
在这个目录下，有许多Nexus运行的平台可供选择，如下图所示：



我用的是windows系统，所以选择图中红框选中的平台：windwos-x86-32。进到目录以后，可以看到有许多批处理文件。如下图：

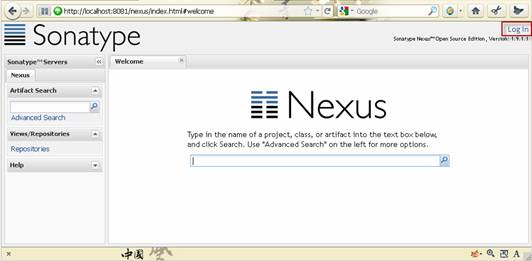


如果是第一次运行，首先运行Installnexus.bat批处理程序，将其安装为Windows服务，然后运行Startnexus.bat批处理程序，启动Nexus服务器。如下图：



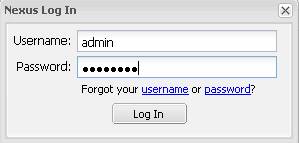
当我们启动Nexus时，我们是在启动一个web服务器，默认端口号为0.0.0.0:8081。Nexus运行在名为Jetty的Servlet容器中，它由一个名为“Tanuki Java Service Wrapper”的本地服务包装器启动。可以通过配置这个service wrapper，将Nexus作为一个Windows服务或一个Unix daemon运行。

启动以后，Nexus将处于运行状态并在8081端口上监听为当前主机配置的所有IP地址（0.0.0.0）。要使用Nexus，首先打开一个浏览器，键入URL：http://localhost:8081/nexus。（也可用127.0.0.1或其他配置的IP地址）。这时会打开Nexus的初始欢迎界面，如下图所示：



单击页面右上角的【log in】链接，会弹出登录对话框，默认的用户名是“admin”，密码是“admin123”。

如下图所示：



登录后界面如下：



## 3.4配置Nexus

以管理员身份登陆成功后，会看到左边的导航栏增加了很多内容。在这里，我们可以管理仓库，配置Nexus系统，管理任务，管理用户，角色，权限，查看系统的RSS源，管理及查看系统日志，等等。可以看到Nexus的功能十分丰富和强大。在本教程中，我们只介绍一些最基本的管理和操作。

**[1]修改密码**

在管理界面左侧栏，选择【Security】|【Change Password】，会弹出密码修改对话框。建议修改默认密码，以保证安全。



**[2]设置远程资源库索引**

在Nexus中，用户管理的资源库有三种类型（virtual除外）：

hosted，本地资源库，通常我们会部署自己的构件到这一类型的资源仓库。

proxy，代理资源仓库，它们被用来代理远程的公共资源仓库，如maven中央资源仓库。

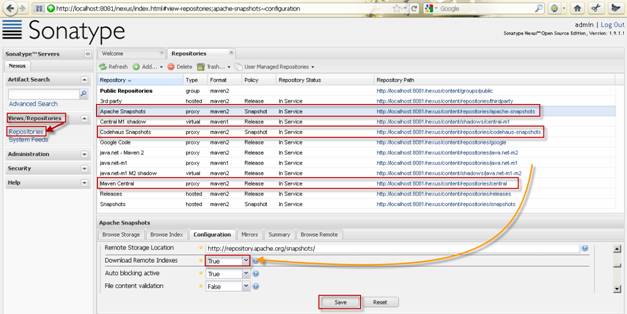
group，资源仓库组，用来合并多个hosted/proxy资源仓库，通常我们配置maven依赖资源仓库组。

Nexus代理三个重要的远程资源库：

Maven Central资源库、Apache Snapshot资源库和Codehaus Snapshot资源库。

每个远程资源库都包含有成千上万的artifact，全部下来是不切实际的。所以Nexus中只维护它们的索引，使用这些索引来搜索artifact，需要的时候再下载内容。不过默认情况下是禁止下载这些索引的，这就需要通过配置允许下载索引。步骤如下：

在左侧栏中，选择【Views/Repositories】|【Repositories】，右侧会打开用户管理的资源列表。找到上述的三个远程资源库，在下方的属性窗口，将“Download Remote Indexes”设为“true”，并保存修改。如下图所示：



在Nexus下载远程资源库索引文件之后，我们就可以在本地搜索远程资源库的所有构件（artifact）了。

**[3]管理资源库**

在左侧栏中，选择【Views/Repositories】|【Repositories】，右侧会打开用户管理的资源列表。

在最上面一栏，我们可以更新、添加和删除资源库。如下图：

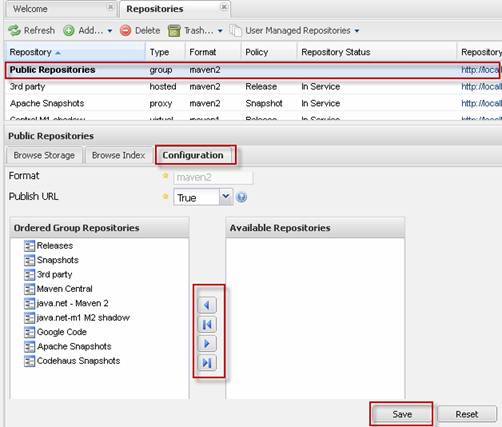


**[4]管理Maven仓库组**

Nexus中仓库组的概念是Maven没有的，在Maven看来，不管你是hosted也好，proxy也好，或者group也好，对我都是一样的，我只管根据groupId，artifactId，version等信息向你要构件。为了方便Maven的配置，Nexus能够将多个仓库，hosted或者proxy合并成一个group，这样，Maven只需要依赖于一个group，便能使用所有该group包含的仓库的内容。

Nexus预定义了“Public Repositories”仓库组，默认合并所有预定义的Release仓库。

点击列表中的“Public Repositories”，然后选择下方的"Configuration" Tab，在配置面板中，将右边“Avaiable Repositories”中的资源库移动到左边的“Ordered Group Repository”中即可，如图：



**[5]配置Maven使用本地Nexus**

要使用Nexus，我们需要配置我们的Maven检查Nexus，而不再是公共资源库。要达到这个目的，我们需要在~/.m2/settings.xml（Maven 3.0.3 为%MAVEN\_HOME%\conf\settings.xml）文件中编辑镜像设置（mirror setting）

setting.xml配置如下

|  |
| --- |
| <mirrors>  <mirror>  <id>Nexus</id>  <name>Nexus Public Mirror</name>  <url>http://localhost:8081/nexus/content/groups/public</url>  <mirrorOf>central</mirrorOf>  </mirror>  </mirrors> |

# 4. 搭建SVN服务器

## 4.1安装SVN

**[1]软件下载**

下载Subversion服务器程序。

到[官方网站](http://subversion.tigris.org/)的下载二进制安装文件，来到[二进制包下载部分](http://subversion.tigris.org/getting.html)，找到 Windows NT, 2000, XP and 2003部分，然后选择[Apache 2.0](http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=91) 或者 [Apache 2.2](http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=8100)，这样我们可以看到许多下载的内容，目前可以下载[Setup-Subversion-1.5.3.msi](http://subversion.tigris.org/files/documents/15/44049/Setup-Subversion-1.5.3.msi) 。

下载Subversion的Windows客户端TortoiseSVN。

TortoiseSVN是扩展Windows Shell的一套工具，可以看作Windows资源管理器的插件，安装之后Windows就可以识别Subversion的工作目录。   
官方网站是[TortoiseSVN](http://tortoisesvn.net/) ，下载方式和前面的svn服务器类似，在[Download](http://tortoisesvn.net/downloads)页面的我们可以选择下载的版本，目前的最高稳定版本的安装文件为[TortoiseSVN-1.5.5.14361-win32-svn-1.5.4.msi](http://downloads.sourceforge.net/tortoisesvn/TortoiseSVN-1.5.5.14361-win32-svn-1.5.4.msi?download)。

**[2]服务器和客户端安装**

服务器安装，直接运行[Setup-Subversion-1.5.3.msi](http://subversion.tigris.org/files/documents/15/44049/Setup-Subversion-1.5.3.msi) ，根据提示安装即可，这样我们就有了一套服务器可以运行的环境。

安装TortoiseSVN，同样直接运行[TortoiseSVN-1.5.5.14361-win32-svn-1.5.4.msi](http://downloads.sourceforge.net/tortoisesvn/TortoiseSVN-1.5.5.14361-win32-svn-1.5.4.msi?download)按照提示安装即可，不过最后完成后会提示是否重启，其实重启只是使svn工作拷贝在windows中的特殊样式生效，与所有的实际功能无关，这里为了立刻看到好的效果，还是重新启动机器。

**[3]建立版本库（Repository）**

运行Subversion服务器需要首先要建立一个版本库（Repository），可以看作服务器上存放数据的数据库，在安装了Subversion服务器之后，可以直接运行，如：

svnadmin create F:\Subversion\SVNdemo\repository

就会在目录E:\svndemo\repository下创建一个版本库。

我们也可以使用TortoiseSVN图形化的完成这一步：   
在目录E:\svndemo\repository下"右键->TortoiseSVN->Create Repository here...“， 然后可以选择版本库模式， 这里使用默认即可， 然后就创建了一系列目录和文件。

**[4]配置用户和权限**

来到E:\svndemo\repository\conf目录，修改svnserve.conf：

# [general]

# password-db = passwd

改为：

[general]

password-db = passwd

然后修改同目录的passwd文件，去掉下面三行的注释：

# [users]

# harry = harryssecret

# sally = sallyssecret

最后变成：

[users]

harry = harryssecret

ally = sallyssecret

 在passwd文件中，“=”前的字符就是用户名，后面的就是密码。还要注意“[users]”前面的注释“#”一定要删除掉。

## 4.2使用SVN

**[1]运行独立服务器**

在任意目录下运行：

svnserve -d -r F:\Subversion\SVNdemo\repository 我们的服务器程序就已经启动了。注意不要关闭命令行窗口，关闭窗口也会把svnserve停止。

-d是demon守护进程，-r根目录

**[2]初始化导入**

来到我们想要导入的项目根目录，在这个例子里是E:\svndemo\initproject，目录下有一个readme.txt文件：

右键->TortoiseSVN->Import...   
在URL of repository输入“svn://localhost/trunk”   
在Import Message中输入你的日志信息

完成之后目录没有任何变化，如果没有报错，数据就已经全部导入到了我们刚才定义的版本库中。

需要注意的是，这一步操作可以完全在另一台安装了TortoiseSVN的主机上进行。例如运行svnserve的主机的IP是133.96.121.22，则URL部分输入的内容就是“svn://133.96.121.22/trunk”。

**[3]基本客户端操作**

取出版本库到一个工作拷贝：

来到任意空目录下，在本例中是E:\svndemo\wc1，运行右键->Checkout，在URL of repository中输入svn://localhost/trunk，这样我们就得到了一份工作拷贝。

在工作拷贝中作出修改并提交：

打开readme.txt，作出修改，然后右键->Commit...，这样我们就把修改提交到了版本库，我们可以运行。

察看所作的修改：

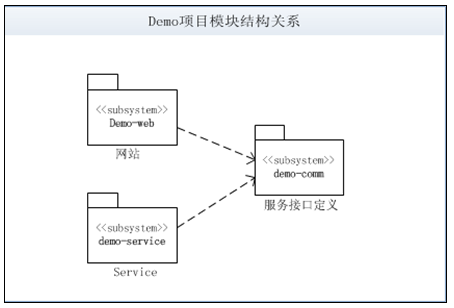
readme.txt上右键->TortoiseSVN->Show Log，这样我们就可以看到我们对这个文件所有的提交。在版本1上右键->Compare with working copy，我们可以比较工作拷贝的文件和版本1的区别。

# 5.maven搭建eclipse工程

## 5.1构建Project

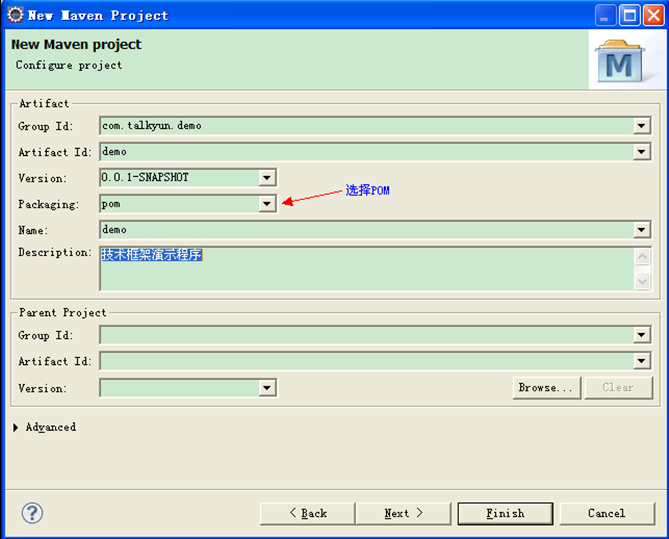
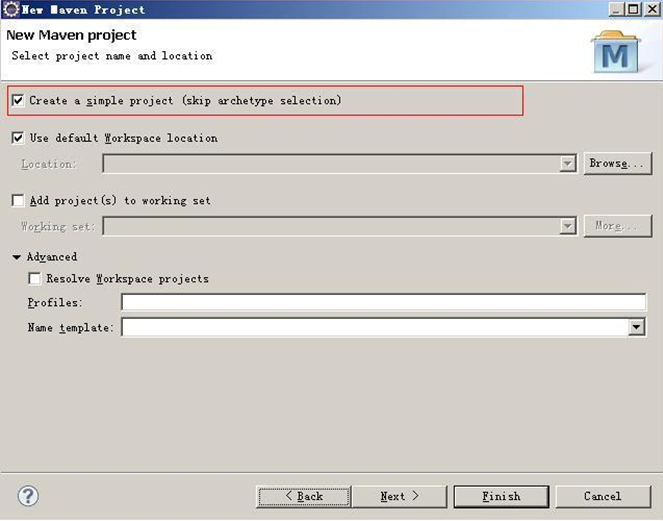
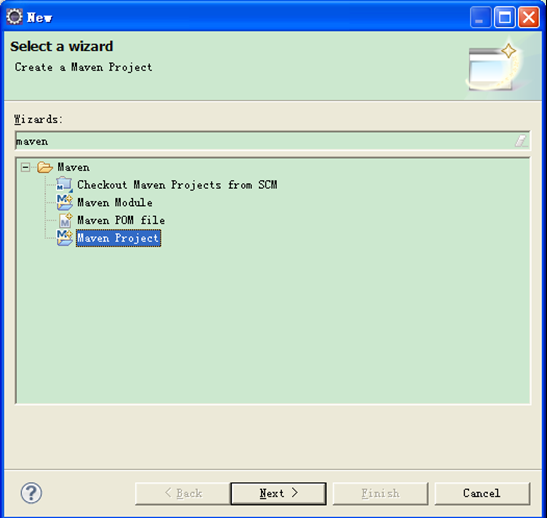
|  |  |
| --- | --- |
|  | 必备条件 在开始构建项目前，请确定您的eclipse + maven环境已经安装正确。 |

demo模块结构示意图



不同于我们之前可能习惯的建立一个大项目，包含mvc+service+dao的模式，为了让项目能够服务分离，水平扩展，我们需要将一个大项目拆分成几个模块来开发。下面我们开始构建项目.

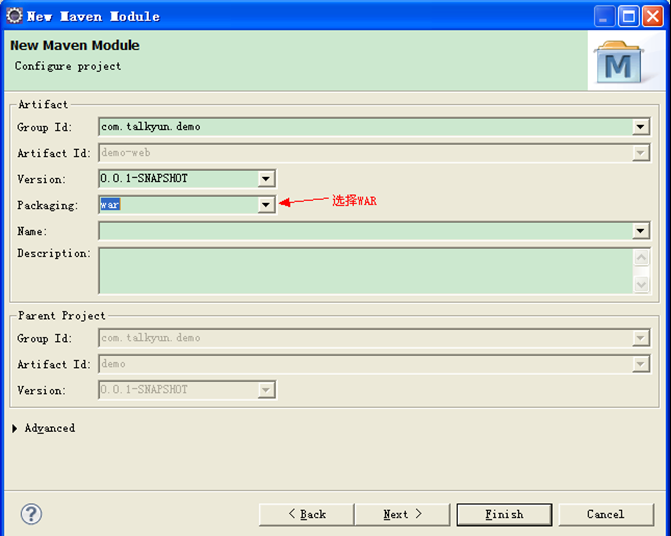
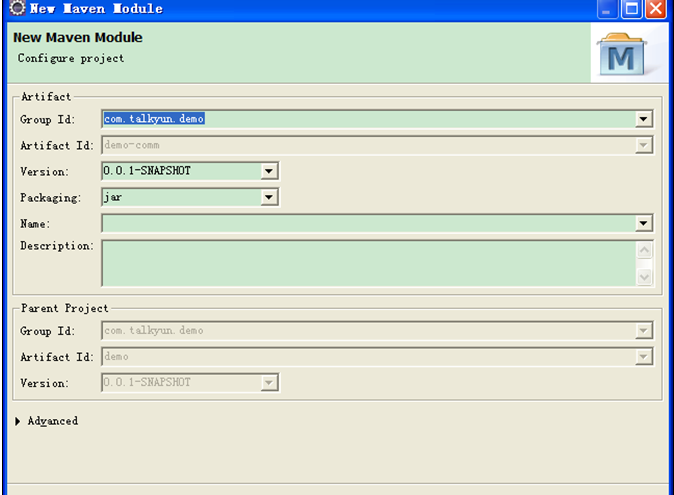
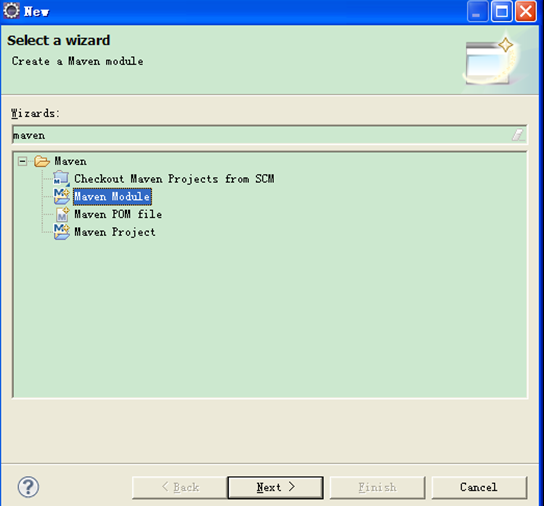
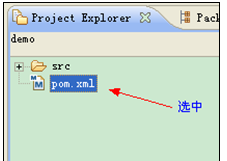
eclipse 新建项目流程截图



点击【finish】后，主项目就已经建立完毕。

|  |  |
| --- | --- |
|  | 这里将主项目建立为pom项目，而web,service,api建立为他的模块，这样做的目的是为了统一管理，统一发布。 |

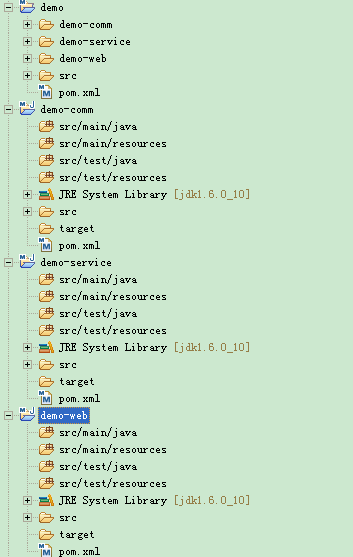
下面开始建立各个模块



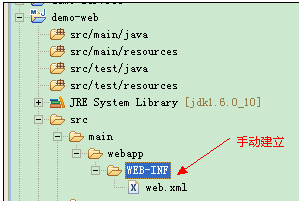
|  |  |
| --- | --- |
|  | 当前建立的是web部分，工程类型需要选择为 war。 |

重复上面的同样步骤，分别再建立demo-comm, demo-service，不过记得，选择类型为 【jar】

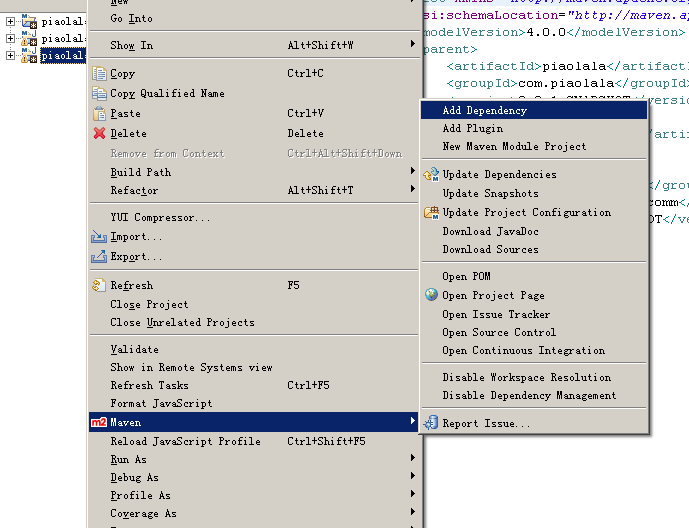
建立完毕后，整个工程包截图如下



|  |  |
| --- | --- |
|  | 这里需要注意，maven创建的war工程不会自动添加web.xml，需要手工建立WEB-INF，手动维护web.xml |



注意：如果demo-web工程依赖demo-service工程，请手动添加依赖关系



5.2部署maven项目

**[1]maven打包部署**

pom.xml添加

|  |
| --- |
| <!-- maven打包部署 -->  <distributionManagement>  <repository>  <id>nexus-releases</id>  <name>Nexus Release Repository</name>  <url>http://localhost:8081/nexus/content/repositories/releases/</url>  </repository>  <snapshotRepository>  <id>nexus-snapshots</id>  <name>Nexus Snapshot Repository</name>  <url>http://localhost:8081/nexus/content/repositories/snapshots/</url>  </snapshotRepository>  </distributionManagement> |

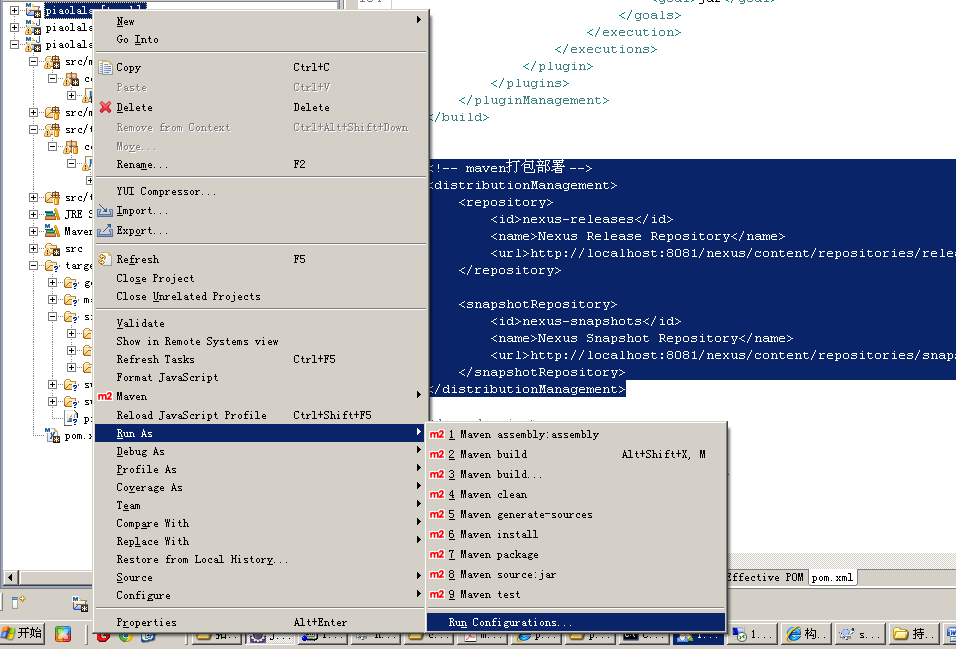
[2]maven的配置文件 (maven3/config/setting.xml)

setting.xml添加

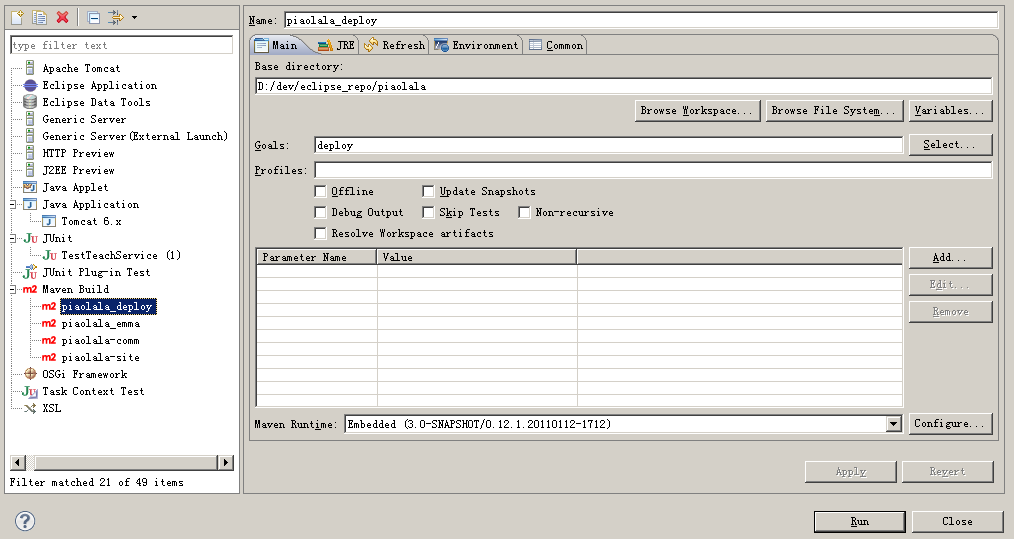
|  |
| --- |
| <servers>  <server>  <id>nexus-releases</id>  <username>admin</username>  <password>123</password>  </server>  <server>  <id>nexus-snapshots</id>  <username>admin</username>  <password>123</password>  </server>  </servers> |

[3]在eclipse中如图操作

选择maven运行配置



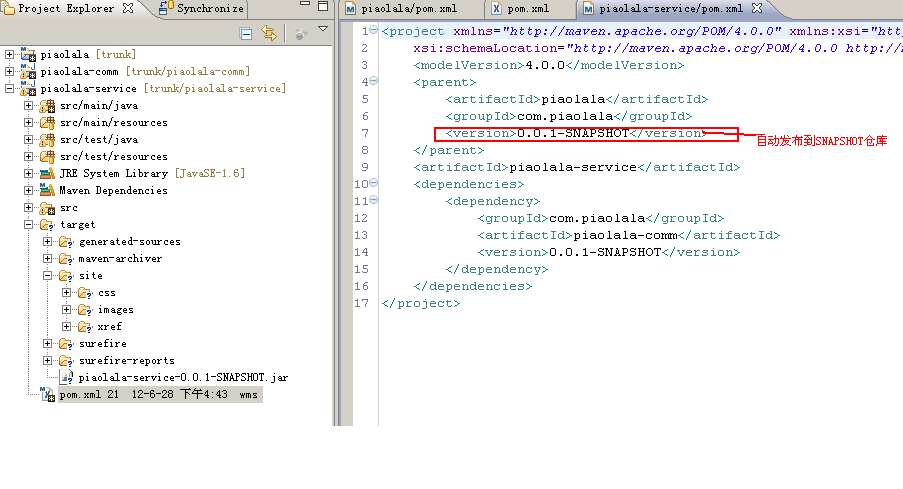
使用maven的deploy命令



点击运行，完成部署

注意：maven自动发布的jar，如果结尾时SANPSHOT自动发布到快照仓库，其他的默认到

release仓库



# 6.jenkins环境搭建

Jenkins，之前叫做Hudson，是基于Java开发的一种持续集成工具，用于监控秩序重复的工作，包括：

　　1、持续的软件版本发布/测试项目。

　　2、监控外部调用执行的工作。

## 6.1安装jenkins

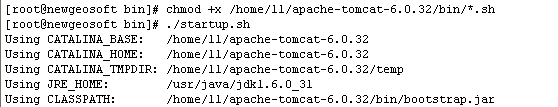
**[1]下载地址**

官网 http://jenkins-ci.org/

**[2]安装步骤**

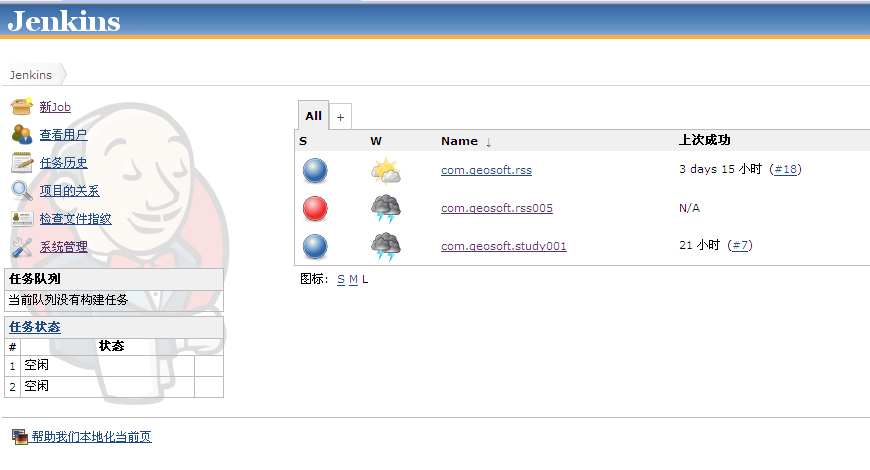
把jenkins.jar下载到Linux服务器, 放入tomcat 的webapp下

启动tomcat



待部署好后 进入jenkins管理页面:

访问http://localhost:8080 , jenkins的主界面如下



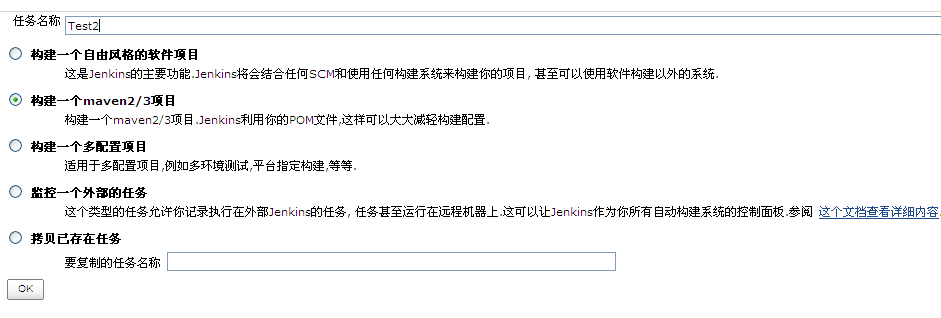
## 6.2 Jenkins构建项目

注意：我们知道Jenkins通过master/slave来支持分布式的job运行，这里的项目运行在master，即Jenkins所在的机器。

**[1]创建Job任务**

在ie中打开http://localhost:8080， 单击 新 job链接，新建job，且编译job的配置如下：

注意jenkins默认已经安装了svn的plugin了。



点击ok



源码管理选择svn 输入svn 的url 如果设置了用户名密码 会有提示输入。

如果没有安装maven,这里也会提示安装一下maven.点进去安装一下就可以 。



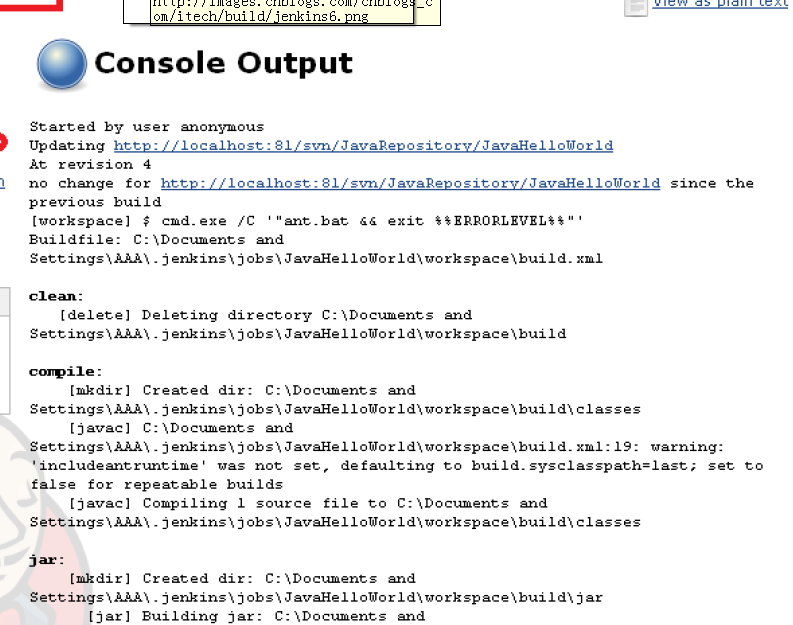
配置好后进入如下页面 点击立即构建

蓝色为成功构建,红色为失败



进入项目的主页面，点击build now链接进行build，build后可以在此主页面上看到所有的build历史：

然后还可以点击某个build的链接，查看某个build的详细日志，如下：



发布成功的话 之前项目启用的cargo插件会自动部署到web容器中,即可直接访问