项目的持续集成： Nexus + Maven2 + SVN + Hudson

项目的持续集成离不开功能强大而又成熟的工具，其中最重要的三个工具是：构建工具（Maven2），源码控制系统(SVN)，持续集成服务器(Hudson)。

三者基本的合作流程是：

1. 持续集成服务器从源码控制系统中签出最新代码；

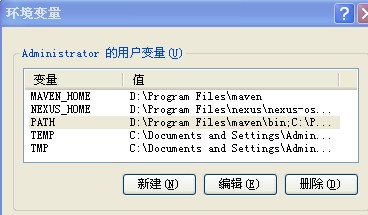
2. 持续集成服务器用构建工具进行构建；

3. 持续集成服务器发布构建结果。

项目在构建过程中，如果有多个项目，可能会依赖一些公用的插件和jar包，每次jar包或插件更新每个项目都得重新载入一遍，而且这些jar包和插件管理起来也会相当麻烦。所以我们引入数据仓库的概念，采用Nexus来管理这些jar包和插件，它可以很好的做到jar包版本控制和复用。

首先我们来看看maven2 的安装和使用，以及如何建立自己的数据仓库：

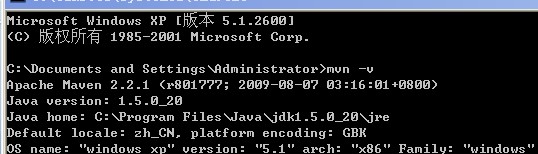
1. 安装：
2. 去<http://maven.apache.org/download.html>选择相应的ZIP包下载；
3. 把下载的包解压缩到任意指定目录，比如解压到D:\Program Files\maven;
4. 添加环境变量，MAVEN\_HOME=D:\Program Files\maven,如下图所示：



4．在用户变量path中加入%MAVEN\_HOME%\bin，这样以后就可以在命令行中使用

mvn命令了。

1. 打开一个新命令行 输入 mvn –v 验证安装是否成功。成功的话如下图所示：



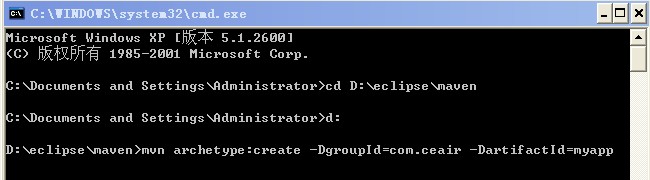
这样我们就可以在命令行下进行maven项目的开发了，下面我们就来创建第一个

Java工程。

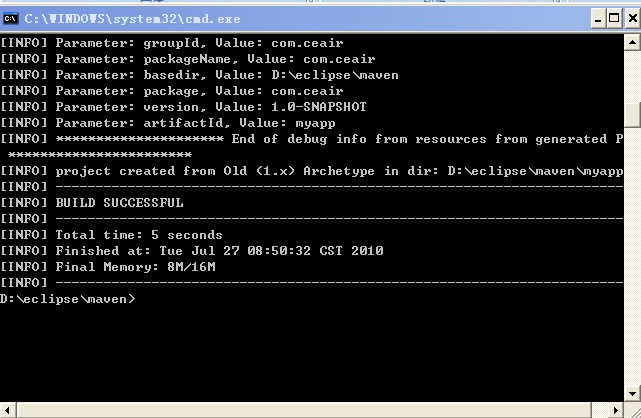
1. 用maven创建第一个java工程（大致有两种方式）

(一) 命令行创建，然后导入IDE ,步骤如下

1. 选择一个根目录，例如D:\eclipse\maven。
2. 打开命令行窗口采用cd命令进入D:\eclipse\maven目录下，输入以下命令（这里我们选择创建一个简单的java工程）：



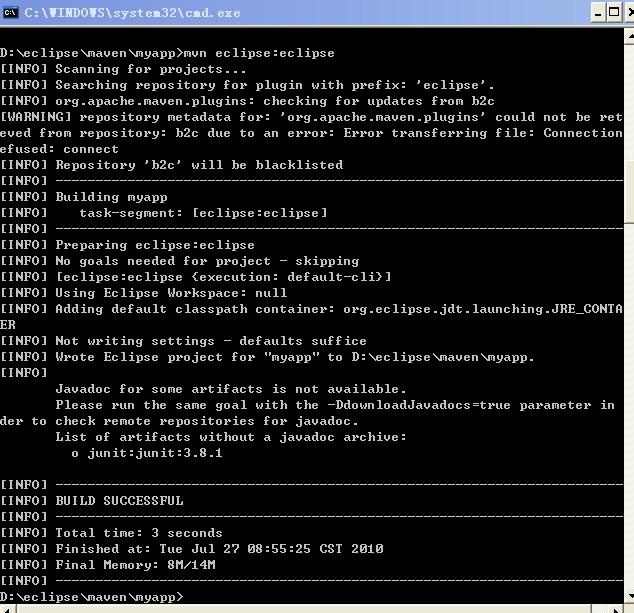
其中myapp是工程名，com.ceair是包名，出现以下信息则说明创建成功：



3．下面我们需要生成eclipse工程，步骤如下：

a> 命令行进入myapp目录

b> 输入指令mvn eclipse:eclipse,如下所示：



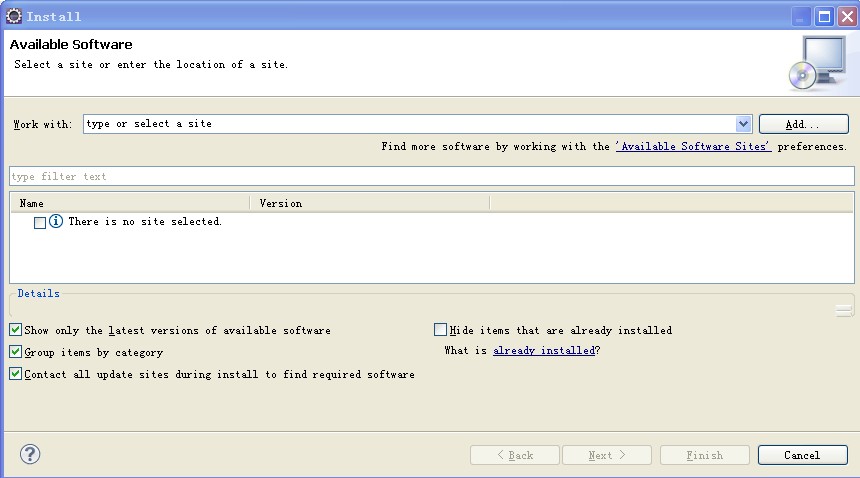
指令执行成功后，发现多了两个文件：.classpath和.project,接下来只需将该工 工程导入eclipse即可。



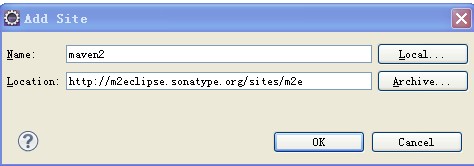
（二）利用maven插件，直接在IDE中创建（推荐）

1．安装maven插件

Eclipse->help->Install New Software,弹出安装对话框



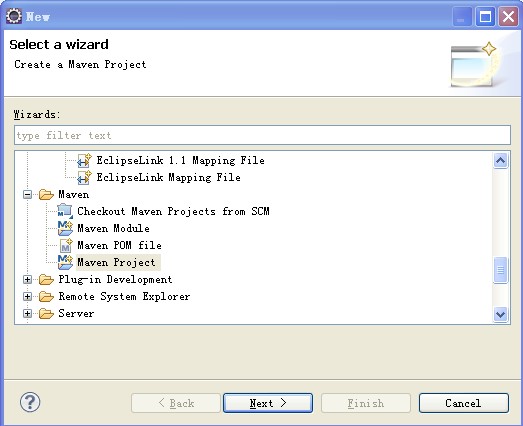
点击Add按钮，按图示输入，确定



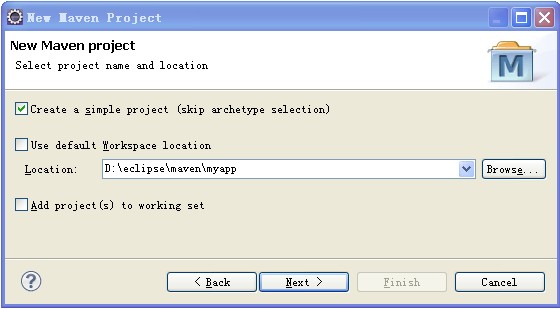
接下来按提示一直next下去，知道安装完毕。

2. 创建maven工程

File->new->other,弹出选择框，我们选Maven Project



点击Next,我们选择Create a simple Project,同时选择工作目录



点击Next,按提示填入信息，其中：

Artifact模块是新建工程的内容：

Group Id: 包名

Artifact Id: 工程名

Version: 版本信息

Packing: 默认打包类型

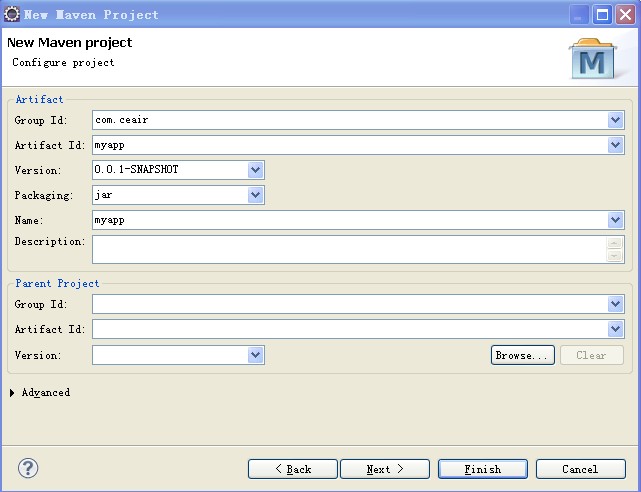
Name: 打包工程名

Description: 打包描述

Parent Project模块是父工程内容（Maven2支持Model依赖），我们这里不考虑

模块依赖问题，所以可以不用填写，大家如果有兴趣，可以参见这篇文章

<http://juvenshun.javaeye.com/blog/305865>



接下来按提示next下去，直到完成。生成的工程结构如下：



其中

src/main/java： 默认存放主流程编写的java代码

src/main/resources: 默认存放主流程资源文件

src/test/java: 默认存放单元测试用例

src/test/resources: 默认存放单元测试所需资源文件

target: 默认打包路径及classes文件生成目录

pom.xml: maven工程配置文件，类似于java web工程中的web.xml文件

做完这些我们就可以在maven项目中添加自己的java类了，当然这其中还会

涉及到一些配置信息，主要是settings.xml和pom.xml两个文件。

（三）settings.xml和pom.xml的简单配置

1．Settings.xml的配置

settings.xml对于maven来说是全局性配置，用于所有的项目。在maven2安

装目录中存在两个settings.xml,一个位于maven2的安装目录conf下，作为默认的

全局性配置。另一个是用户自己的本地配置，默认目录为：

${user.dir}/.m2/settings.xml，其中*${user.dir}指windows中的户目录，如C:\Documents*

*and Settings\Administrator*。

如果是第一次使用maven，${user.dir}/.m2/目录下的settings.xml文件可能不存在，需要我们手动将安装目录conf下的settings.xml拷贝到该目录下，然后根据自身情况具体配置。

settings.xml基本结构如下：  
    <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">  
  <localRepository/>  
  <interactiveMode/>  
   <usePluginRegistry/>  
  <offline/>  
  <pluginGroups/>  
  <servers/>  
  <mirrors/>  
  <proxies/>  
  <profiles/>  
  <activeProfiles/>  
 </settings>

简单介绍一下几个主要的配置因素：

**localRepository：**表示本地库的保存位置，也就是maven2主要的jar保存位置，默认在

${user.dir}/.m2/repository，如果需要另外设置，就换成其他的路径。

**offline：**如果不想每次编译，都去查找远程中心库，那就设置为true。当然前提是你已经下载

了必须的依赖包。

**Servers:**   在POM中的 distributionManagement元素定义了开发库。然而，特定的sername和pwd不能使用于pom.xml，所以通过此配置来保存server信息。

**Mirrors** : 表示镜像库，指定库的镜像，用于增加其他库。

**Proxies** : 此设置，主要用于无法直接访问中心的库用户配置。

**Active Profiles：**表示激活的profile,通过profile id来指定。

**。**。。。。。

**这里不一一列举了，具体配置信息见文档《mvn使用命令》**

**2**．Pom.xml的配置

pom作为项目对象模型。通过xml表示maven项目，使用pom.xml来实现。主

要描述了项目：包括配置文件；开发者需要遵循的规则，缺陷管理系统，组织和

licenses，项目的url，项目的依赖性，以及其他所有的项目相关因素。

pom.xml基本结构如下：

<project>

<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>

**具体如何配置参见文档《mvn使用命令》**

**三、 建立自己的数据仓库 Nexus**

如果不做任何配置，Maven2会默认去maven的数据中心下载所需jar包及插件，

如果没有会报错。这样不利于我们的项目控制，所以我们需要建立自己的数据仓库，这

里推荐一款开源软件Nexus,基本配置如下：

1. 下载[nexus-oss-webapp-1.7.0-bundle.zip](http://nexus.sonatype.org/downloads/nexus-oss-webapp-1.7.0-bundle.zip)，然后解压缩到任意目录下（如E:\nexus）

下载地址：<http://nexus.sonatype.org/downloads/>

1. 建立用户环境变量NEXUS\_HOME  值为E:\nexus\nexus-oss-webapp-1.7.0
2. 如果操作系统是windows-x86-32 则 进入目录

E:\nexus\nexus-oss-webapp-1.7.0\bin\jsw\windows-x86-32运行nexus.bat在浏览器中输入<http://localhost:8081/nexus/index.html>

即可进入登录页面 默认用户名密码是admin/admin123

值得注意的一点是：开启远程索引下载。这样我们可以在检索栏直接输入资源名，如果仓库有相应信息，系统会给我们全部列出来，就像本地检索一样方便。如果没有索引，即使仓库里面有相应信息，检索栏也会检索不到。

Nexus默认是关闭远程索引下载的，主要是担心会造成对服务器的巨大负担。常用的需要开启的三个仓库分别是Maven Center,Apache Snapshots,Codehaus Snapshots;具体的开启方式（以Maven Center为例）：

左侧窗口点击Repositories,右侧会将所有仓库信息显示出来，选中代理仓库Maven Center， 更改Configuration->Download Remote Indexes为true,保存配置后，右击代理仓库Maven Center然后选择ReIndex,如此即可触发Nexus下载远程的索引文件。

4. 配置MAVEN使用NEXUS作为镜像仓库，首先找到${home.dir}/.m2/settings.xml

文件 如果没有.m2文件夹的话，说明你没有运行过MVN，请在命令行中输入MVN命

令，首次运行MAVEN后会自动创建.M2文件夹，然后进入MAVEN的安装路径下的

CONF文件夹，把默认settings.xml 拷贝到${home.dir}/.m2下。接着在此文件中填入以下信息：

<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0

http://maven.apache.org/xsd/settings-1.0.0.xsd">

<mirrors>

<mirror>

<!--This sends everything else to /public -->

<id>nexus</id>

<mirrorOf>\*</mirrorOf>

<url>http:// 主机ip:端口/nexus/content/groups/public</url>

</mirror>

</mirrors>

<profiles>

<profile>

<id>nexus</id>

<!--Enable snapshots for the built in central repo to direct -->

<!--all requests to nexus via the mirror -->

<repositories>

<repository>

<id>b2c</id>

<url>http://主机ip:端口/nexus/content/groups/public</url>

<releases><enabled>true</enabled></releases>

<snapshots><enabled>true</enabled></snapshots>

</repository>

</repositories>

<pluginRepositories>

<pluginRepository>

<id>b2c</id>

<url>http:// 主机ip:端口/nexus/content/groups/public</url>

<releases><enabled>true</enabled></releases>

<snapshots><enabled>true</enabled></snapshots>

</pluginRepository>

</pluginRepositories>

</profile>

</profiles>

<activeProfiles>

<!--make the profile active all the time -->

<activeProfile>nexus</activeProfile>

</activeProfiles>

</settings>

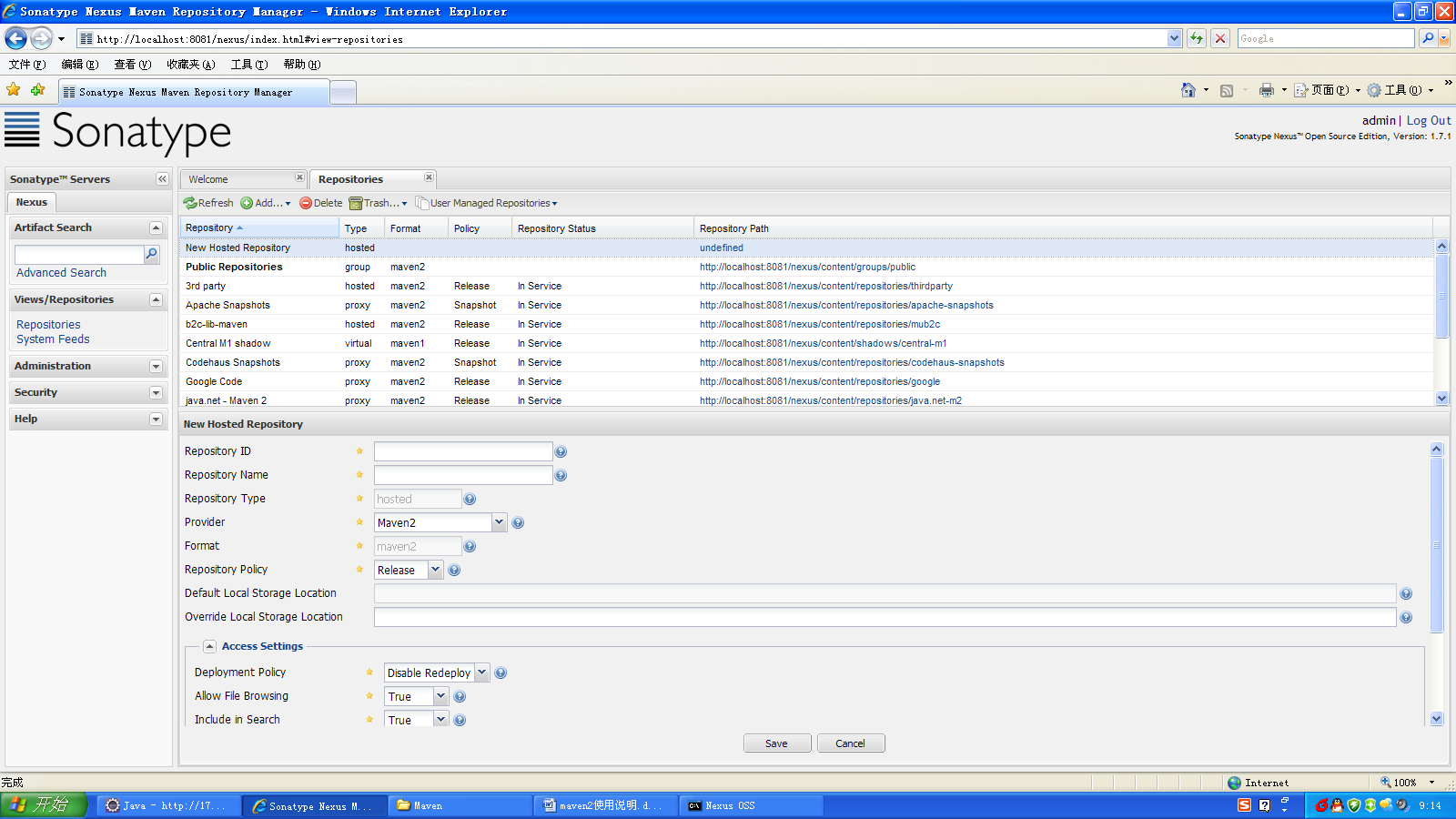
其他更详细信息配置可参见文档《**mvn使用命令**》

**四、 Nexus中配置自己的数据仓库，及仓库管理**

1．添加一个新的仓库（以添加host仓库为例）

首先以管理员身份登录Nexus，左边导航栏Views/repositories点击Repositories,这样会在右边新窗口列出所有的已知仓库，。

然后点击新窗口左上角的Add按钮，下拉选项中我们选在Host Repositories,这样就打开了一个新增Host数据仓库页面，如下图‘



接下来要做的就是根据自己的需要按照提示填写配置信息，填写完成记得点击save按钮保存。

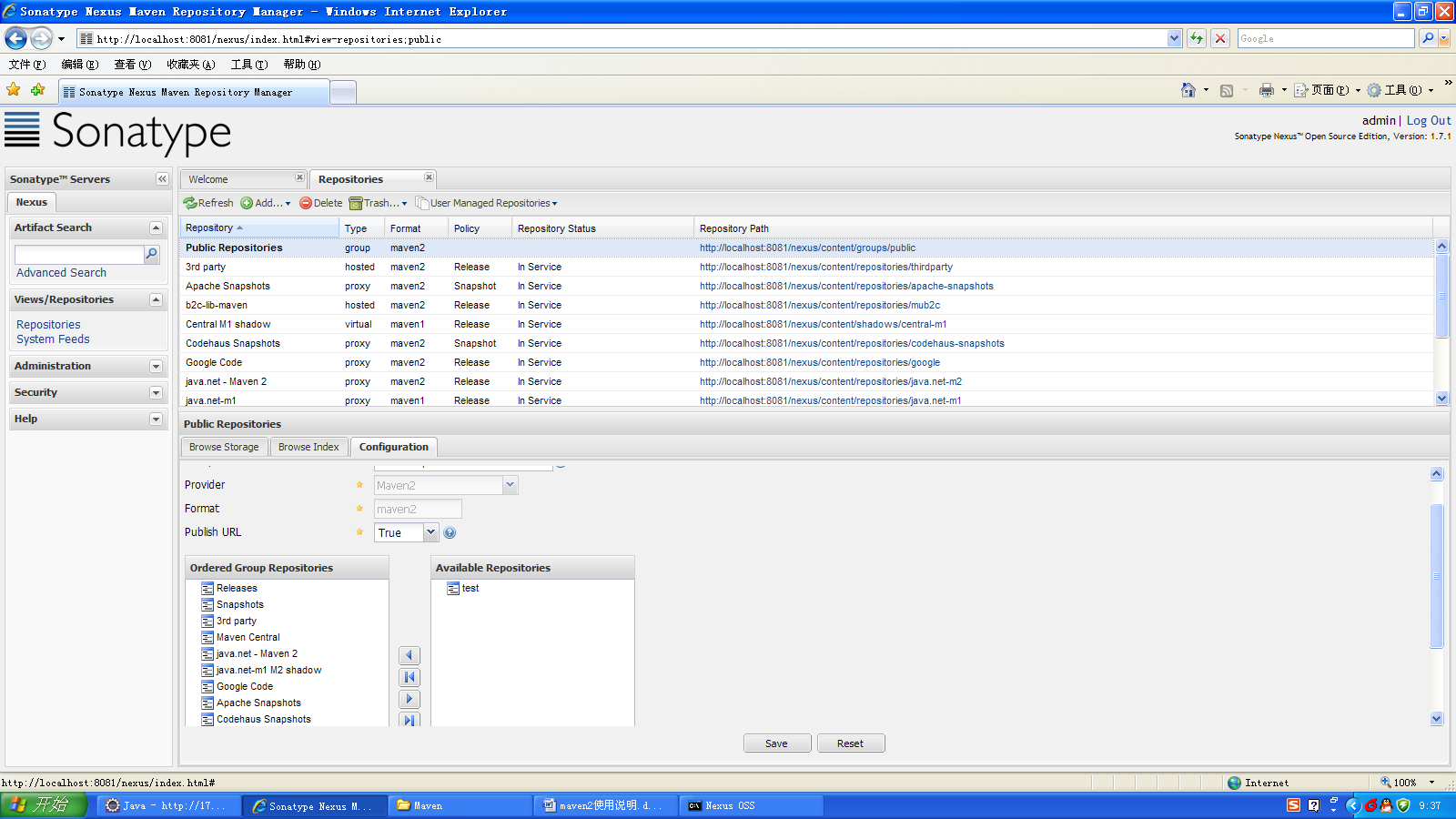
默认的Local Storage Location会保存在Nexus安装主目录下的\sonatype-work\nexus\storage路径下，如果需要改动，在Overridding Local Storage Location项输入相应的路径即可。

1. 添加一个仓库至一个组

因为我们maven中配置的公共仓库组是在public Repostitories组中,所以我们

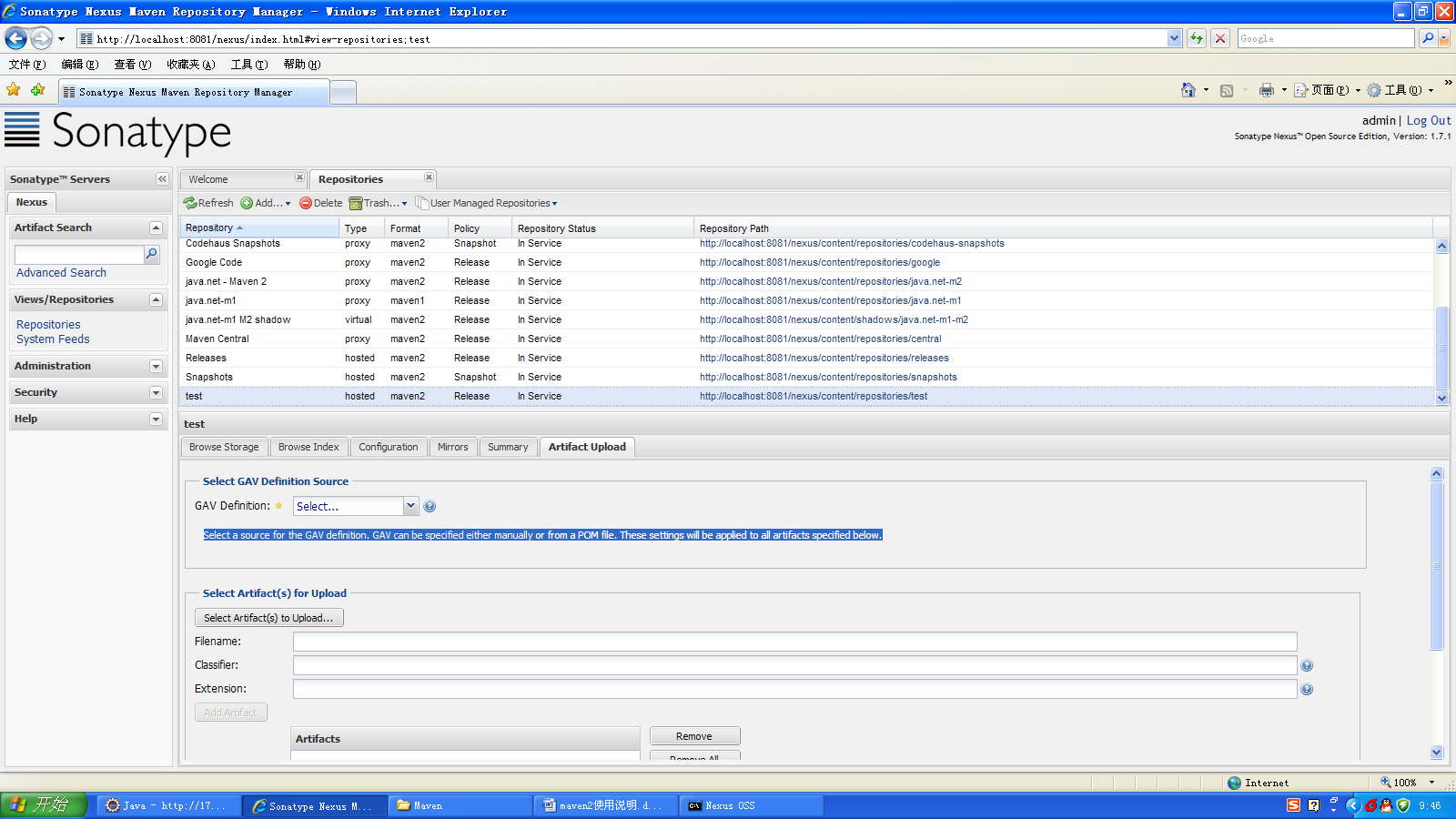
需要把新建的仓库添加至public Repostitories.具体操做如下：

在仓库列表中选中public Repostitories，Configuration –>Available Repositories下将新建仓库（我们这里的新建仓库名为test）拖拽至Ordered Group Repositories,点击save按钮保存配置。

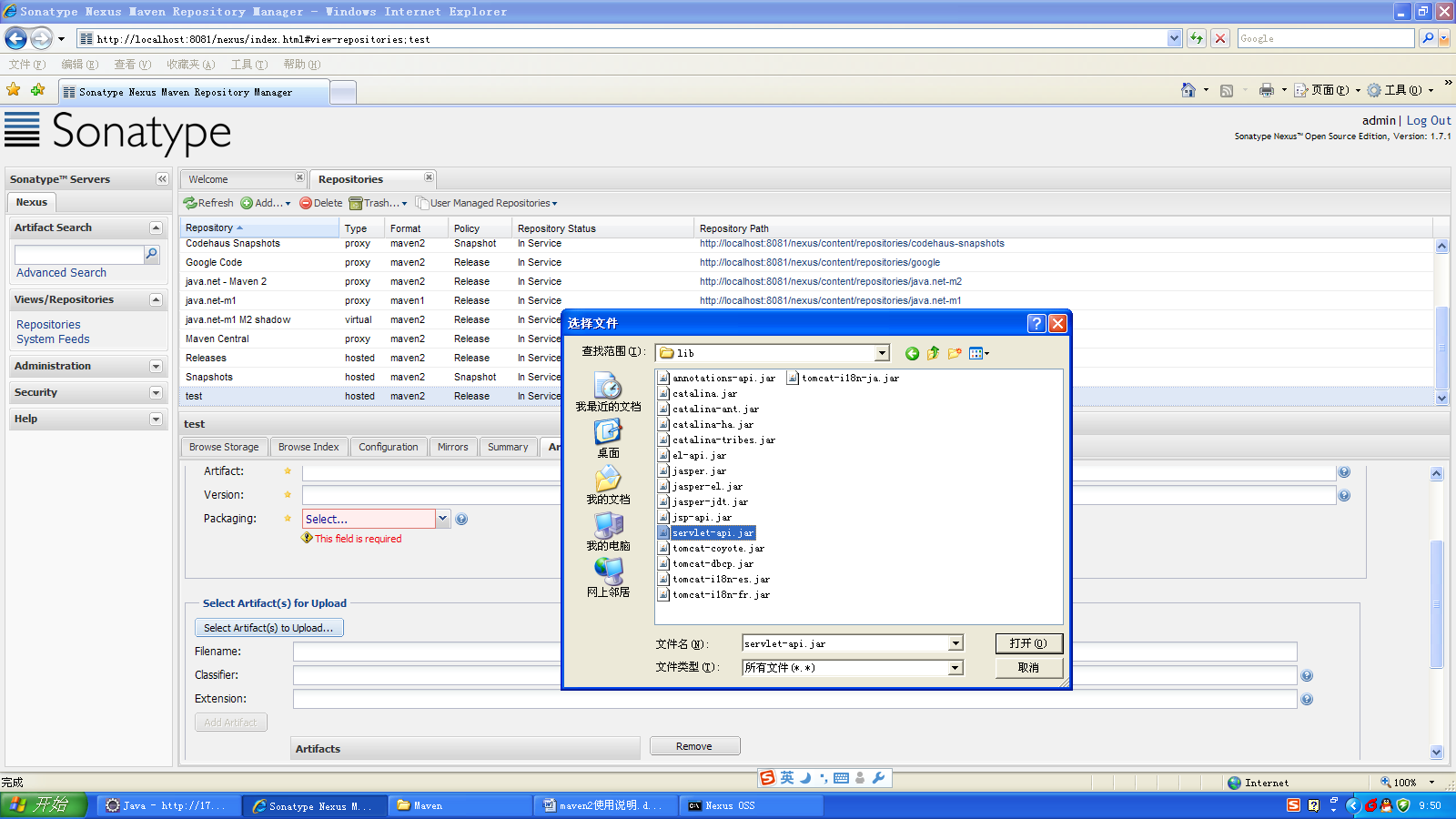


3．为数据仓库添加构件

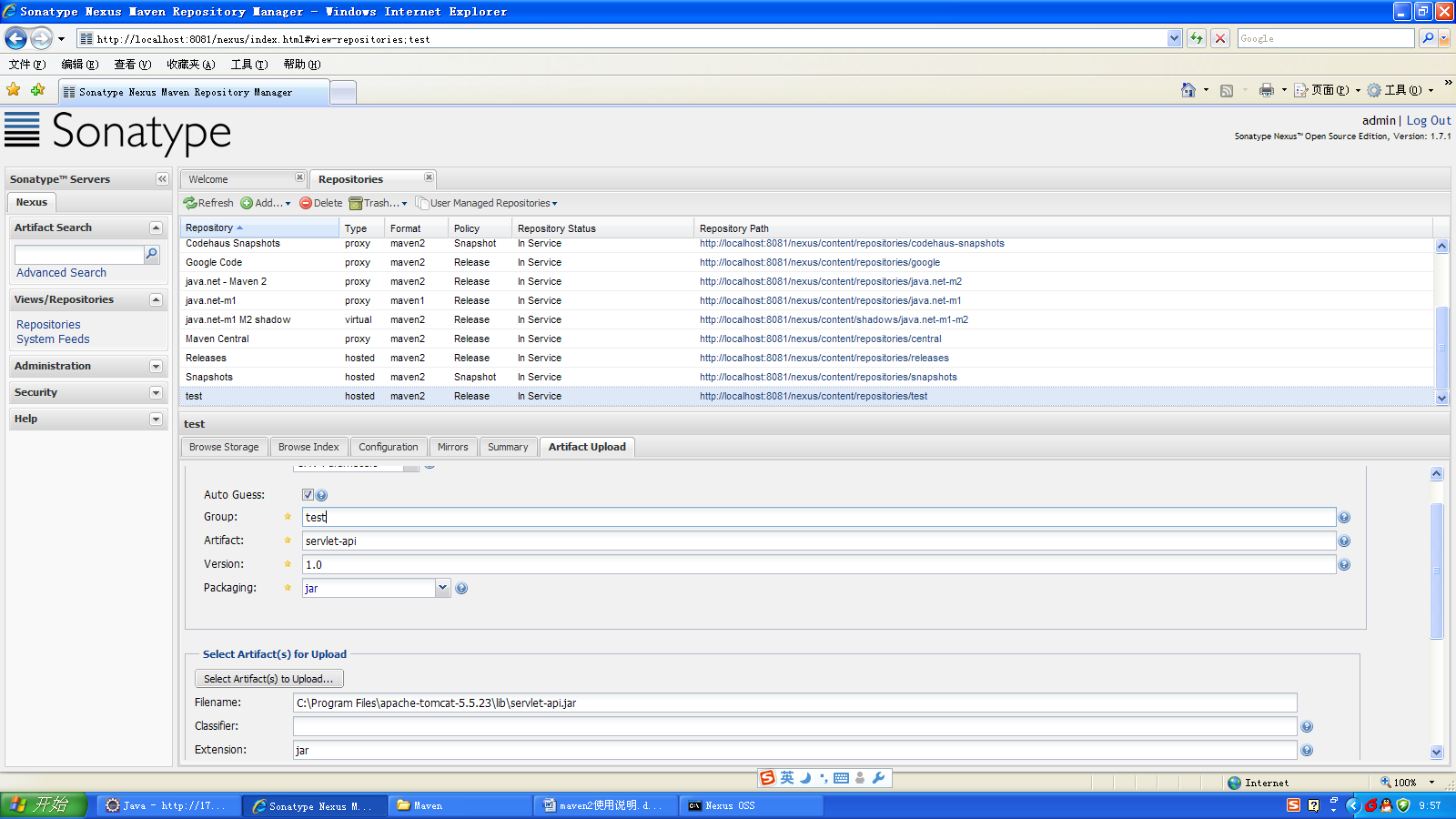
我们以添加jar包为例。首先选中要添加的host仓库test,选择Artifact Upload



Artifact Upload下一共有两个模块：select GAV Defination source 和 select Artifact(s) for Upload,我们首先在select Artifact(s) for Upload 模块下点击select Artifact(s) for upload按钮上传构件



然后在select GAV Defination source模块填写相关构件配置参数，如下图：



Classifier是用来区分同一版本号的不同版本，如poi-3.0-FINAL.jar，FINAL就是它的Classifier.另外Nexus对构件命名校验很严格，一般规则是“构建名+版本号[+Classifier].构建类型” ，上传时的相关参数填写和Maven的pom.xml配置依赖稍微不匹配就可能导致取不到构件。这点一定要注意。

填写完所有构件信息保存后就完成了构件的添加，这时候再重建下仓库索引即可。