



(/blog/index?

username=zhouenshan)

zhouenshan (/blog/index?

username=zhouenshan)

持续集成之路——搭建Maven私服

发表于2015/5/5 18:33:40 160人阅读

分类：Java (<http://blog.csdn.net/mydeman/article/category/312353>) 敏捷实践 (<http://blog.csdn.net/mydeman/article/category/693446>)

2010-09-25 16:15 15451人阅读 评论 (<http://blog.csdn.net/mydeman/article/details/5905424#comments>)(10) 收藏 举报 (<http://blog.csdn.net/mydeman/article/details/5905424#report>)

[maven](http://www.csdn.net/tag/maven) (<http://www.csdn.net/tag/maven>)

[服务器](http://www.csdn.net/tag/%e6%9c%8d%e5%8a%a1%e5%99%a8) (<http://www.csdn.net/tag/%e6%9c%8d%e5%8a%a1%e5%99%a8>)

[wrapper](http://www.csdn.net/tag/wrapper) (<http://www.csdn.net/tag/wrapper>) [junit](http://www.csdn.net/tag/junit) (<http://www.csdn.net/tag/junit>)

[浏览器](http://www.csdn.net/tag/%e6%b5%8f%e8%a7%88%e5%99%a8) (<http://www.csdn.net/tag/%e6%b5%8f%e8%a7%88%e5%99%a8>)

在开发过程中，有时候会使用到公司内部的一些开发包，显然把这些包放在外部是不合适的。另外，由于项目一直在开发中，这些内部的依赖可能也在不断的更新。可以通过搭建公司内部的Maven服务器，将第三方和内部的依赖统一管理。

这里使用Nexus来搭建本地的Maven服务器，过程比较简单。

一、安装服务器

1、下载

我们可以在nexus的官网 (<http://nexus.sonatype.org/>)上找到它的相关介绍，下载地址是：<http://nexus.sonatype.org/downloads/> (<http://nexus.sonatype.org/downloads/>)，在这里可以找到最新的版本，如果需要以前的版本，在官网上应该也可以找到下载地址。我下载的是：[nexus-oss-webapp-1.8.0-bundle.tar.gz](http://nexus.sonatype.org/downloads/nexus-oss-webapp-1.8.0-bundle.tar.gz) (<http://nexus.sonatype.org/downloads/nexus-oss-webapp-1.8.0-bundle.tar.gz>)。关于Nexus的详细使用方法可以参照：[Repository Management with Nexus](http://www.sonatype.com/books/nexus-book/reference/) (<http://www.sonatype.com/books/nexus-book/reference/>)。

2、安装

解压下载的文件：

```
# tar xzvf nexus-oss-webapp-1.8.0-bundle.tar.gz
```

解压后会在同级目录中，出现两个文件夹：nexus-oss-webapp-1.8.0和sonatype-work，前者包含了nexus的运行环境和应用程序，后者包含了你自己的配置和数据。

3、启动nexus

在上面的提到，nexus的运行环境在nexus-oss-webapp-1.8.0目录，下面就进入这个目录启动：

```
# cd nexus-oss-webapp-1.8.0/bin/jsw/linux-x86-64/
```

在这个目录下包含了一个文件夹和三个文件：lib、nexus、platform和wrapper，其中nexus就是启动命令。

```
# ./nexus
```

执行上面的命令，可以得到nexus命令的用法提示：start 命令启动，stop命令停止。下面启动nexus：

```
# ./nexus start
```

```
Starting Nexus OSS...
```

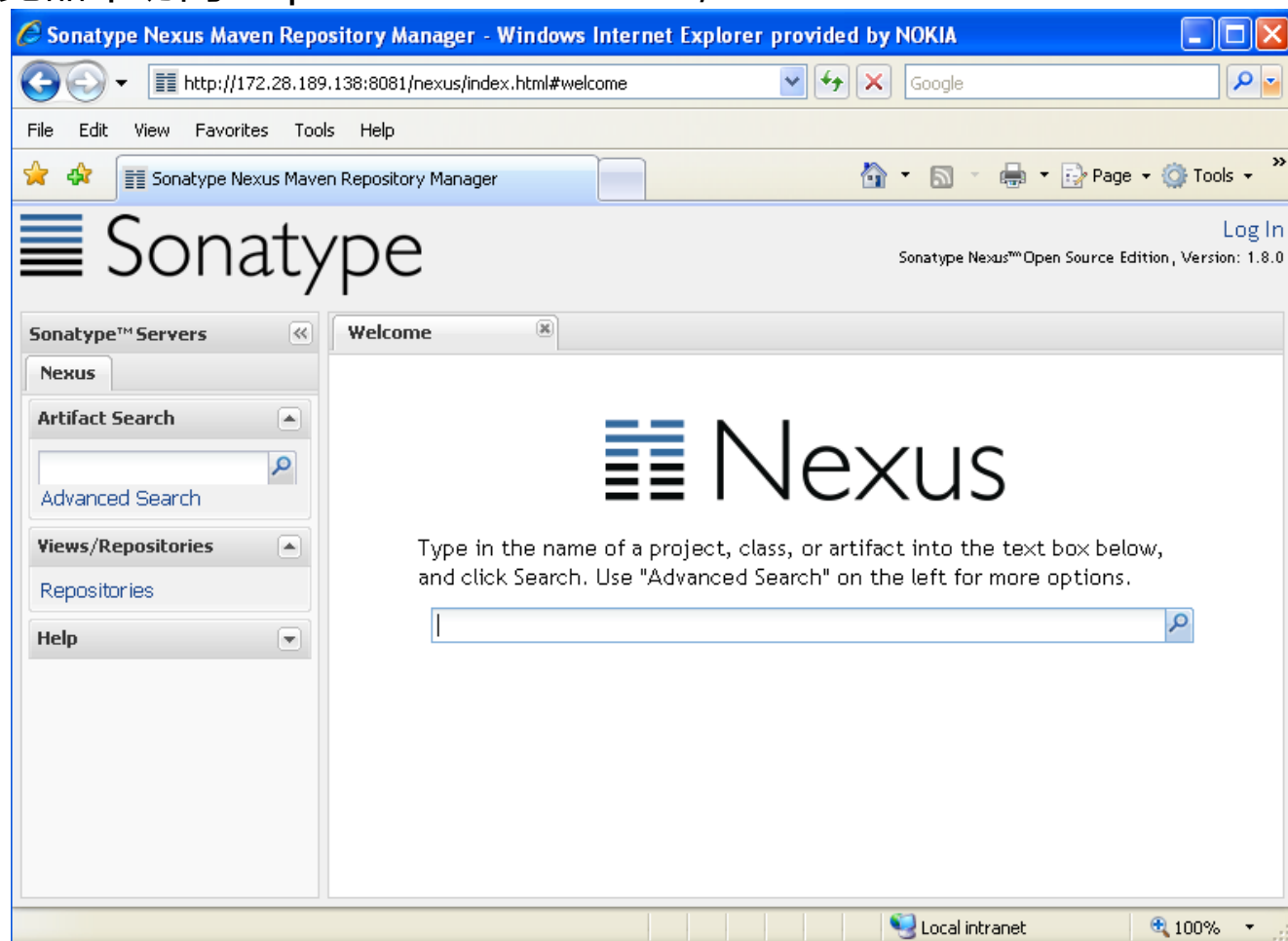
```
Started Nexus OSS
```

从控制台输出可以看到Nexus已经启动成功，我们可以通过log文件查看更详细的信息：

```
# cd ~/nexus-oss-webapp-1.8.0/log
```

```
# tail -f wrapper.log
```

在log中可以看到nexus默认监听的端口是8081。那么我们就可以在浏览器中访问：<http://host:8081/nexus>，



二、配置Nexus

由于在新搭建的nexus环境中只是一个空的仓库，所以第一步就是要和远程的Maven中心仓库进行同步。

The screenshot shows the Sonatype Nexus Open Source Version 1.8.0-01 interface. The top right corner has a login bar with 'admin' and 'Log Out'. The left sidebar contains 'Nexus', 'Artifact Search', 'Advanced Search', 'Views/Repositories', 'Repositories', 'Administration', 'Security', and 'Help'. The 'Repositories' tab is selected, showing a table of repositories. The 'Public Repositories' section is expanded, and the 'Configuration' tab is selected. The 'Configuration' page shows the 'Provider' set to 'Maven2', 'Format' set to 'maven2', and 'Publish URL' set to 'True'. The 'Ordered Group Repositories' list includes '3rd party', 'Maven Central', 'java.net - Maven 2', 'Apache Snapshots', 'Releases', 'Snapshots', 'java.net-m1 M2 shadow', 'Google Code', and 'Codehaus Snapshots'. The 'Available Repositories' list includes '3rd party', 'Maven Central', and 'java.net - Maven 2'. The 'Save' and 'Reset' buttons are at the bottom.

1. Sonatype Nexus™ Open Source Edition - Version 1.8.0-01

2. Repositories

3. Public Repositories

4. Configuration

5. Ordered Group Repositories

从这里可以删除不需要的Repository，我只保留了3rd party, Maven Central和java.net - Maven 2

以admin登录，默认的用户名密码admin/admin123

The screenshot shows the Sonatype Nexus Open Source Version 1.8.0-01 interface. The 'Repositories' tab is selected, and the 'Public Repositories' section is expanded. The 'Browse Index' tab is selected. A context menu is open over the 'Public Repositories' list, showing options: 'Expire Cache', 'Rebuild Metadata', 'ReIndex', 'Incremental ReIndex', and 'ReIndex'. The 'ReIndex' option is highlighted.

6. ReIndex

在Public Repository上点击右键，在弹出菜单中选择ReIndex

稍等一会儿，就可以在Browse Storage下面看到多了index文件夹

The screenshot shows the Sonatype Nexus Open Source Version 1.8.0-01 interface. The 'Browse Storage' tab is selected. The 'Public Repositories' folder is expanded, showing a list of files: 'nexus-maven-repository-index.gz', 'nexus-maven-repository-index.gz.md5', 'nexus-maven-repository-index.gz.sha1', 'nexus-maven-repository-index.properties', 'nexus-maven-repository-index.properties.md5', 'nexus-maven-repository-index.properties.sha1', and '.meta'.

如果在Reindex之后，并没有同步到远程的仓库，可以检查每个仓库的设置。下面是Maven Central的设置：

Maven Central	proxy	maven2	Release	In Service	http://10.68.34.11/nexus/content/repositories/central
Releases	hosted	maven2	Release	In Service	http://10.68.34.11/nexus/content/repositories/releases

Maven Central

Browse Storage Browse Index **Configuration** Mirrors Summary Browse Remote

Repository ID

Repository Name

Repository Type

Provider

Format

Repository Policy

Default Local Storage Location

Override Local Storage Location

Remote Repository Access

Remote Storage Location

Download Remote Indexes

Auto blocking active

Checksum Policy

☐ Authentication (optional)

Save Reset

指定远程仓库的地址，下载远程的索引！

三、在项目中使用私服

在完成了上面的配置后，就可以将项目中默认的Repository切换为本地的私服了，只需要在pom.xml中增加repositories就可以了：

```
[xhtml] view plain (http://blog.csdn.net/mydeman/article/details/5905424#) copy (http://blog.csdn.net/mydeman/article/details/5905424#) print (http://blog.csdn.net/mydeman/article/details/5905424#) ? (http://blog.csdn.net/mydeman/article/details/5905424#)
01. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
02. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
03.   <modelVersion>4.0.0</modelVersion>
04.
05.   <groupId>org.maven.demo</groupId>
06.   <artifactId>MavenDemo</artifactId>
07.   <version>0.0.1-SNAPSHOT</version>
08.   <packaging>jar</packaging>
09.
10.   <name>MavenDemo</name>
11.   <url>http://maven.apache.org</url>
12.
13.   <repositories>
14.     <repository>
15.       <snapshots>
16.         <enabled>true</enabled>
17.       </snapshots>
18.       <id>public</id>
19.       <name>Public Repositories</name>
20.       <url>http://172.28.189.138:8081/nexus/content/groups/public</url>
21.     </repository>
22.   </repositories>
23.   <pluginRepositories>
```

```
24.     <pluginRepository>
25.         <id>public</id>
26.         <name>Public Repositories</name>
27.         <url>http://172.28.189.138:8081/nexus/content/groups/public</url>
28.     </pluginRepository>
29. </pluginRepositories>
30. <dependencies>
31.     <dependency>
32.         <groupId>junit</groupId>
33.         <artifactId>junit</artifactId>
34.         <version>4.8.1</version>
35.         <type>jar</type>
36.         <scope>compile</scope>
37.     </dependency>
38. </dependencies>
39. <properties>
40.     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
41. </properties>
42. </project>
```

将pom.xml保存后，再回过头来了看去nexus管理界面看，就会发现junit已经被下载到本地的nexus服务器中了。

转载地址：<http://blog.csdn.net/mydeman/article/details/5905424>

maven--私服的搭建（Nexus的使用）和注意的问题

私服是什么

私服，私有服务器，是公司内部Maven项目经常需要的东东，不总结一下，不足以体现出重视。Nexus是常用的私用Maven服务器，一般是公司内部使用。下载地址是<http://www.sonatype.org/nexus/go> (<http://www.sonatype.org/nexus/go>)。默认端口8081,这里我选择最新版nexus-2.5.0-04。

常用功能

Nexus常用功能就是：指定私服的中央地址、将自己的Maven项目指定到私服地址、从私服下载中央库的项目索引、从私服仓库下载依赖组件、将第三方项目jar上传到私服供其他项目组使用。

开启Nexus服务后访问url地址<http://localhost:8081/nexus/> (<http://localhost:8081/nexus/>)(推荐使用自己的ip地址)，之后登录系统，用户名密码分别是：admin/admin123.

最频繁的就是点击左侧菜单栏的Repositories按钮

一般用到的仓库种类是hosted、proxy。Hosted代表宿主仓库，用来发布一些第三方不允许的组件，比如oracle驱动、比如商业软件jar包。Proxy代表代理远程的仓库，最典型的就是Maven官方中央仓库、JBoss仓库等等。如果构建的Maven项目本地仓库没有依赖包，那么就会去这个代理站点去下载，那么如果代理站点也没有此依赖包，就回去远程中央仓库下载依赖，这些中央仓库就是proxy。代理站点下载成功后再下载至本机。笔者认为，其实Maven这个自带的默认仓库一般情况下已经够大多数项目使用了。特殊情况时在配置新的仓库，指定url即可，一般熟悉ExtJS的人操作这个Nexus都没什么问题，单词不是很难，不明白的查查单词基本差不多。就是如果Sonatype公司对其做了国际化的处理就更好了。

hosted 类型的仓库，内部项目的发布仓库

releases 内部的模块中release模块的发布仓库

snapshots 发布内部的SNAPSHOT模块的仓库

3rd party 第三方依赖的仓库，这个数据通常是由内部人员自行下载之后发布上去

proxy 类型的仓库，从远程中央仓库中寻找数据的仓库

group 类型的仓库，组仓库用来方便我们开发人员进行设置的仓库

maven项目索引

下载Maven项目索引，项目索引是为了使用者能够在私服站点查找依赖使用的功能

保存后后台会运行一个任务，点击菜单栏的Scheduled Tasks选项即可看到有个任务在RUNNING。下载完成后，Maven索引就可以使用了，在搜索栏输入要搜索的项，就可以查到相关的信息。例如spring-core

就可以检索出它的相关信息，包括怎么配置依赖信息。我们要想使用这个私服仓库，先在项目pom中配置相关私服信息
指定仓库

[html] view plaincopy

```
<repositories>
  <repository>
    <id>nexus</id>
    <name>nexus</name>
    <url>http://192.168.1.103:8081/nexus/content/groups/public/ (http://192.168.1.103:8081/nexus/content/groups/public/)</url>
    <releases>
      <enabled>>true</enabled>
    </releases>
    <snapshots>
      <enabled>>true</enabled>
```

```

    </snapshots>
  </repository>
</repositories>

```

指定插件仓库

[html] view plaincopy

```

<pluginRepositories>
  <pluginRepository>
    <id>nexus</id>
    <name>nexus</name>
    <url>http://192.168.1.103:8081/nexus/content/groups/public/ (http://192.168.1.103:8081/nexus/content/groups/public/)</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
  </pluginRepository>
</pluginRepositories>

```

这样只有本项目才在私服下载组件

这样这个Maven项目构建的时候会从私服下载相关依赖。当然这个配置仅仅是在此项目中生效，对于其他项目还是不起作用。如果相对Maven的其他项目也生效的话。需要修改全局的settings.xml文件。

修改settings.xml为

追加激活profile

[html] view plaincopy

```

<activeProfiles>
  <activeProfile>central</activeProfile>

</activeProfiles>

```

之后所有本机的Maven项目就在私服下载组件。（ 这样比较好 ）

项目的发布

[html] view plaincopy

```

<distributionManagement>
  <repository>
    <id>user-release</id>
    <name>User Project Release</name>
    <url>http://192.168.1.103:8081/nexus/content/repositories/releases/ (http://192.168.1.103:8081/nexus/content/repositories/releases/)</url>
  </repository>

  <snapshotRepository>
    <id>user-snapshots</id>
    <name>User Project SNAPSHOTs</name>
    <url>http://192.168.1.103:8081/nexus/content/repositories/snapshots/ (http://192.168.1.103:8081/nexus/content/repositories/snapshots/)</url>
  </snapshotRepository>

```

</distributionManagement>

注意配置了还是发布项目到私服失败，原因为没有权限，会出现401错误码，原因就是权限不够。

配置权限在settings.xml

注意Repository中的id一定要和server下的id一致，切记！！否则出现权限问题。

然后运行发布

clean deploy

在控制台发布成功

然后进入到私服上的仓库中，看一下确实存在刚刚发布的项目

上一篇 (/article/details?

下一篇 (/article/details?

id=45504817)

id=45507139)

暂无评论，我去发表 (/comment/post?id=45506121)~



我的热门文章

spring注解@Service("service")括号中的service有什么用 (/article/details?id=48470557)

在windows上部署Redis系统服务 (/article/details?id=51607654)

如何给web项目添加redis服务 JAVA几种缓存技术 ehcache和redis哪个更好 (/article/...

Redis服务器段和客户端 最详细的说明 (/article/details?id=49127707)

Jenkins安装与配置 (/article/details?id=47011485)

相关博文

Jenkins学习总结3Jenkins+Maven+Git搭建持续集成和自动化部署的 (http://blog.csdn...

Jenkins+Maven+SVN快速搭建持续集成环境转 (http://blog.csdn.net/zhousenshan/ar...

搭建基于Jenkins+SVN+Maven持续集成环境CI (http://blog.csdn.net/wlp_name/article...)

利用jenkinssvnmaven搭建持续集成环境 (<http://blog.csdn.net/colorandsong/article/d...>)

Hudson+Maven+SVN 快速搭建持续集成环境 (<http://blog.csdn.net/xiebinghu/article/...>)

从头搭建maven+jenkins持续集成环境 (<http://blog.csdn.net/MayFlower19870918/arti...>)

CruiseControl+SVN+Maven+Tomcat6持续集成搭建总结 (http://blog.csdn.net/Allen_...)

Jenkins+Maven+SVN快速搭建持续集成环境转 (<http://blog.csdn.net/lwjshuai/article/...>)

Hudson Maven SVN快速搭建持续集成环境 (<http://blog.csdn.net/w565911788/article...>)

