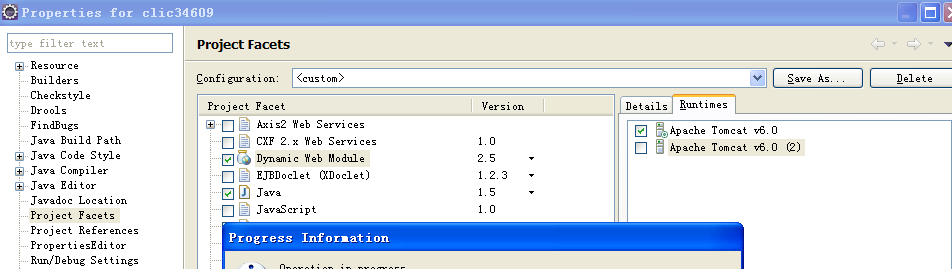
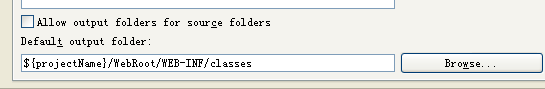
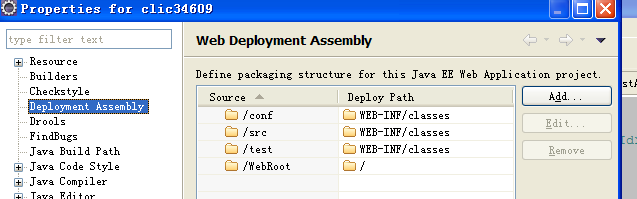
Maven,hudson持续集成

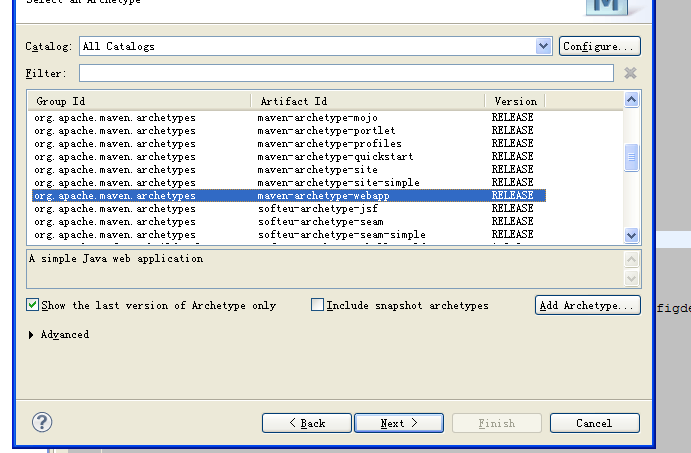
# 构建maven项目.

## 移植myeclispe中的项目至eclispe中.

1. 项目从svn做为普通java项目载下来后,更新source folder.
   1. 
   2. 把项目从 普通项目变成 web项目 ,如图:
   3. 
   4. 更新 class 生成输出地址如图:
   5. 
   6. 更新项目 webapp 的目录 ,把eclipse下的webContent 换成 myeclispe下的 webRoot, 如图. 之后删除项目中的webContent文件夹.
   7. 
2. 这样,我们的myEclipse中的web项目 就 可以在eclispe中成功部署了.

## 转换普通web项目为maven项目.

1. 新建 maven Project ,选择骨架为 webApp ,如图:



1. 在项目下新建src.main.java文件夹,同时把这个文件夹指定为source包. 再新建config文件夹用来放置 checkstyle 插件等需要使用的 配置文件.,再新建 src.test.java文件夹,同时指定为source,再新建src.test.resource文件夹,指定为source.
2. 拷贝原项目中的src 目录下的所有文件至 src.main.java文件夹中, 拷贝原test目录下的所有文件至src.test.java文件夹中.拷贝原webRoot目录下内容至src.main.webApp下(注意,请删除原webRoot下 WEB-INF 目录下的classes文件夹)
3. 把已经写好的pom文件,加入工程的最外级目录下.运行build命令. 如果成功则证明原项目已经成功转换成maven项目. 当然其实困难的过程只是在于pom文件的编写.详见

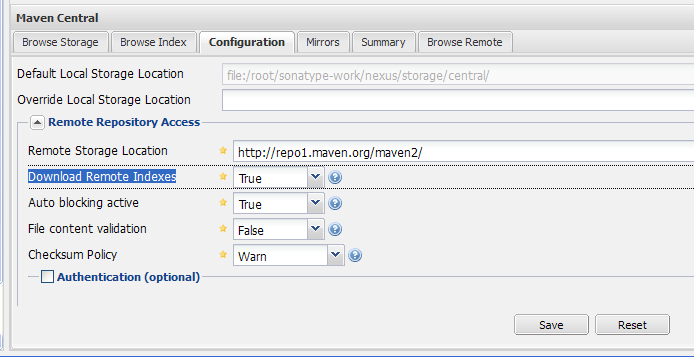
# 搭建maven私服

## 安装私服

* 1. 从http://nexus.sonatype.org 下载nexus 最新包,直接部署至web容器中.

## 私服配置

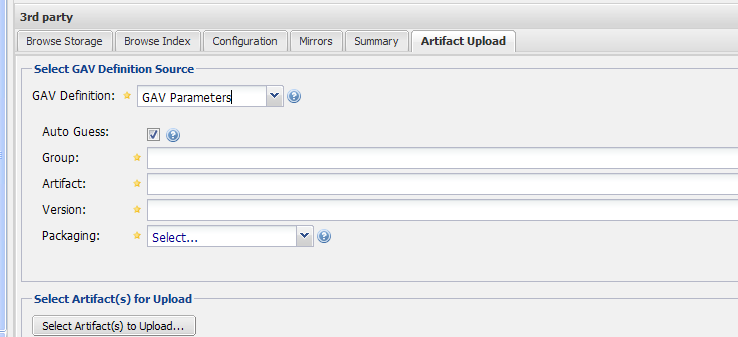
* 1. 打开主页面,使用用户admin 进行登陆. 密码 admin123
  2. 打开左侧[Repositories](http://192.168.101.40:8080/nexus/index.html) 菜单,在仓库页面中选择Maven Central,如图,配置Download Remote Indexes 属性值为 true



点击左侧菜单[Scheduled Tasks](http://192.168.101.40:8080/nexus/index.html) ,查看是否有任务存在,有的话,等待任务运行完毕后,就已经打开私服与maven中心仓库的下载通道.

## 上传中央仓库中没有的包.

1)点击[Repositories](http://192.168.101.40:8080/nexus/index.html)菜单,在 ‘3rd party’仓库中,上传我们自己的lib包.如图:



## 设置镜像

* 1. 这样我们就拥有了我们的私服了,但是我们需要让所有的开发者都使用我们的私服进行lib包的维护,那我们就需要设置私服镜像了.文件如下. 所有开发者在自己机器的 /.m2 目录 把settings.xml这个文件放置进去,这样我们开发所有的lib库就都取自我们自己的私服了.



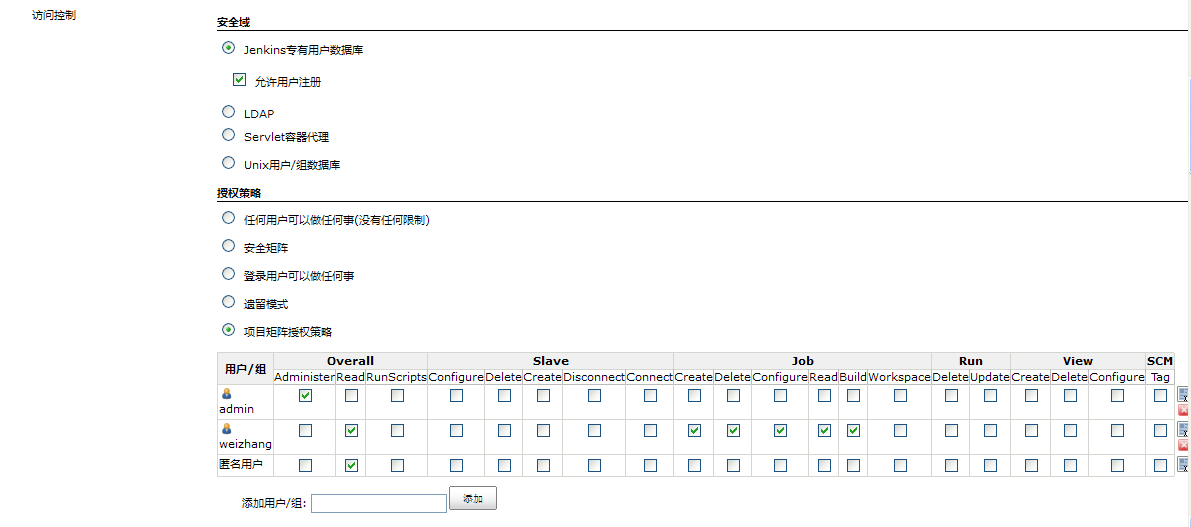
**在经过以上4 步之后, 我们的私服就创建完毕了.**

# Hudson的安装,部署

## 下载安装hudson.,地址为jenkins-ci.org,请对应选择服务器版本.

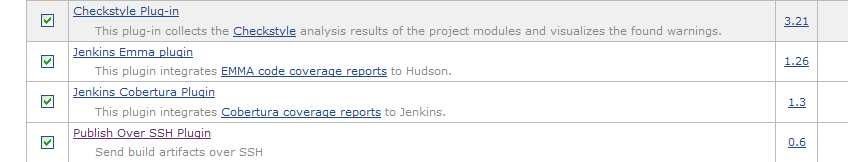
## hudson配置

* 1. 打开hudson 页面.初始是不需要登陆的,在系统管理中配置各项. Jdk,maven,svn,以及之后的各类插件的一些属性配置,例如[Publish Over SSH Plugin](http://wiki.jenkins-ci.org/display/JENKINS/Publish+Over+SSH+Plugin) 等.
  2. 安全 也在系统配置中 进行配置,一般是按照下图进行hudson用户配置:

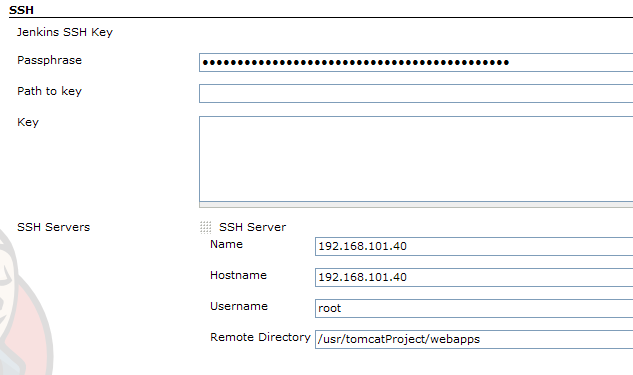


## 插件安装,需要安装的插件大致如下图:





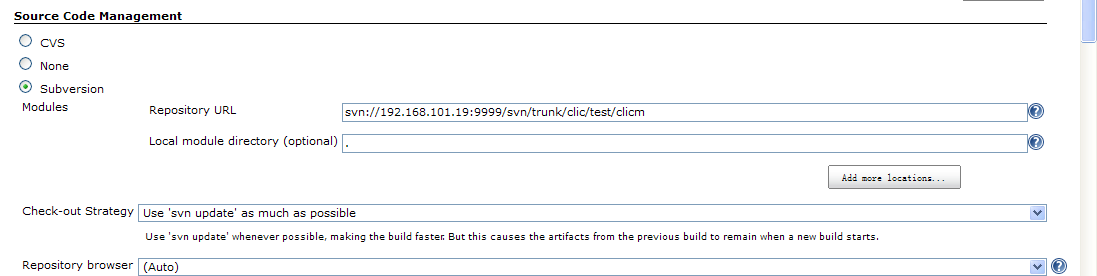
各个插件的作用,如图中的英文所述. 其中emma为ant版本的 测试覆盖率统计 插件,这里我们可以忽略.我们使用 cobertura 为我们做 测试覆盖率统计. 在 ssh 插件安装完成后,我们需要在 系统配置 中 配置 需要ssh的远程服务器的地址,登陆账号,需要上传文件的路径,等信息,如图:



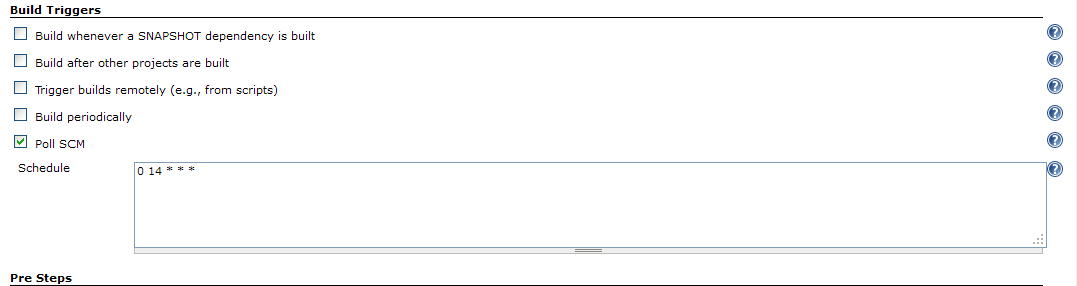
## hudson job的配置,项目的持续集成,在这里就体现出来了.

1. 创建maven job.
2. Job 图示:

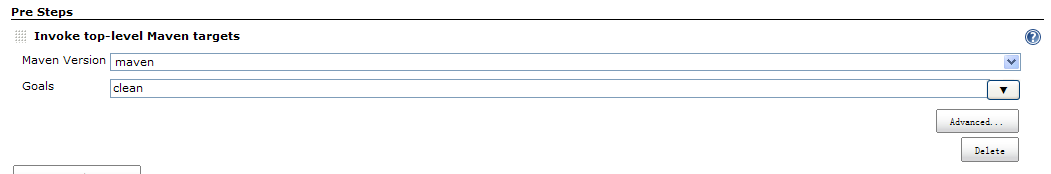
第一步:svn配置



第二步:设置自动部署的轮询,schedule按照 分,时,日,月,年的 pattern 方式 排列,下图的意思是说每天的14点 自动部署.



第三步: 执行build 前的操作,这里是maven clean .当然这步不是必须的.



第四步: 设定 具体 build 内容,这里是给项目打包,同时进行 checkstyle的检查,当然还可以同时进行 cobertura的检查,格式在后面追加空格和cobertura: cobertura,当然除了**goal命令**之外,想要使用插件检查一些东西的话,还需要对 job 设置中对应的插件进行配置.





第五步:**在build成功后,所做的事**,这里我们是先ssh到需要部署 项目的服务器上,之后先关闭容器,,再删除上次部署的成果物,之后我们再启动容器,为最后的 deploy 插件部署war 包做准备.



第六步: 在所有build完成后,要做的最后一件事,就是部署成果物至远程容器中了. 这里需要选择下容器类型,如果是tomcat则需要配置具有 manager 项目权限的 用户及密码以及容器的访问地址. 至此我们就完成hudson 与 maven,svn 的持续集成了. 当然,如果这种 ssh +deploy 插件 部署方式您认为不行的话,完全可以只采用 ssh 的方式 进行 1停服务 2 删除成果物 3上传成果物 4启动服务 这种单一的操作方式.



# 过程中需要注意的地方

* 1. 在linux上安装完 jdk,maven后,需要修改 /etc/profile文件,在文件最后加上 如下内容:

M2\_HOME=/usr/apache-maven-3.0.3

JAVA\_HOME=/usr/java/jdk1.6.0\_29

PATH=$JAVA\_HOME/bin:$M2\_HOME/bin:$PATH

CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

export M2\_HOME

export JAVA\_HOME

export PATH

export CALSSPATH

* 1. 在hudson 使用 ssh 插件后,如果需要 ssh 时候正常执行 关闭 以及 启动 容器命令,这里拿tomcat为例子,则需要改变 /tomcat/bin/setclasspath.sh 这个文件,在最上方加上如下内容:

JAVA\_HOME=/usr/java/jdk1.6.0\_29

JRE\_HOME=/usr/java/jdk1.6.0\_29/jre

export JAVA\_HOME

export JRE\_HOME