# Jenkins + Svn + Ant持续集成

tags : DevOps

## 一、Jenkins基础配置

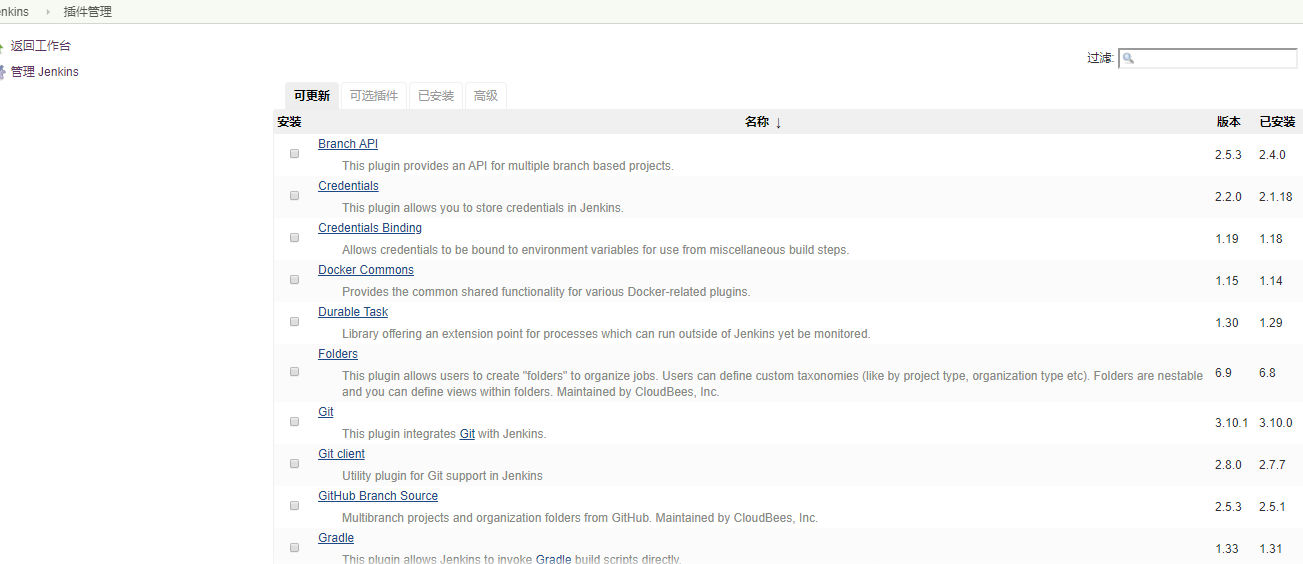
### 1.配置中文

主界面-->系统管理-->插件管理-->可选插件  
安装插件locale plugin  
系统管理-->系统设置-->Locale  
填入：zh\_CN   
保存应用

### 2.插件管理

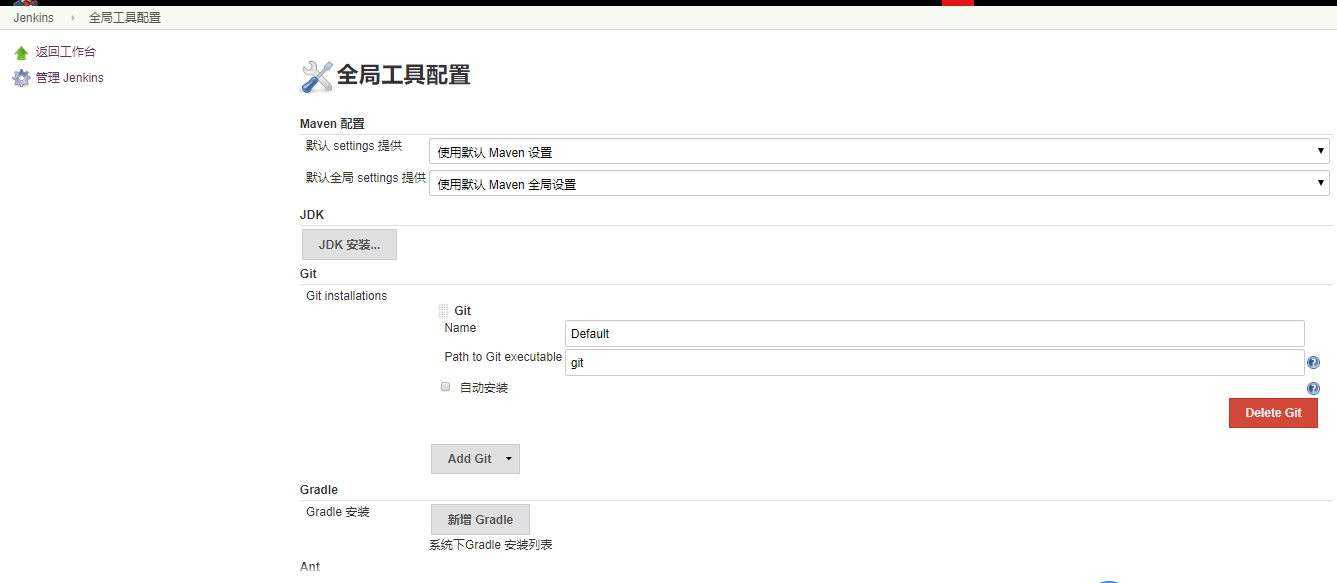
插件管理在 系统管理 -> 管理插件 里面。  
我们需要先完成的插件的安装才能配置和管理我们Job,有以下几种插件是我们需要的：

* Svn plugin(Svn 源码管理插件)
* antPlugin(ant 打包插件)
* Publish Over SSH(远程访问的SSH插件)



### 3.全局工具配置

全局工具配置在 系统管理 -> Global Tool Configuration



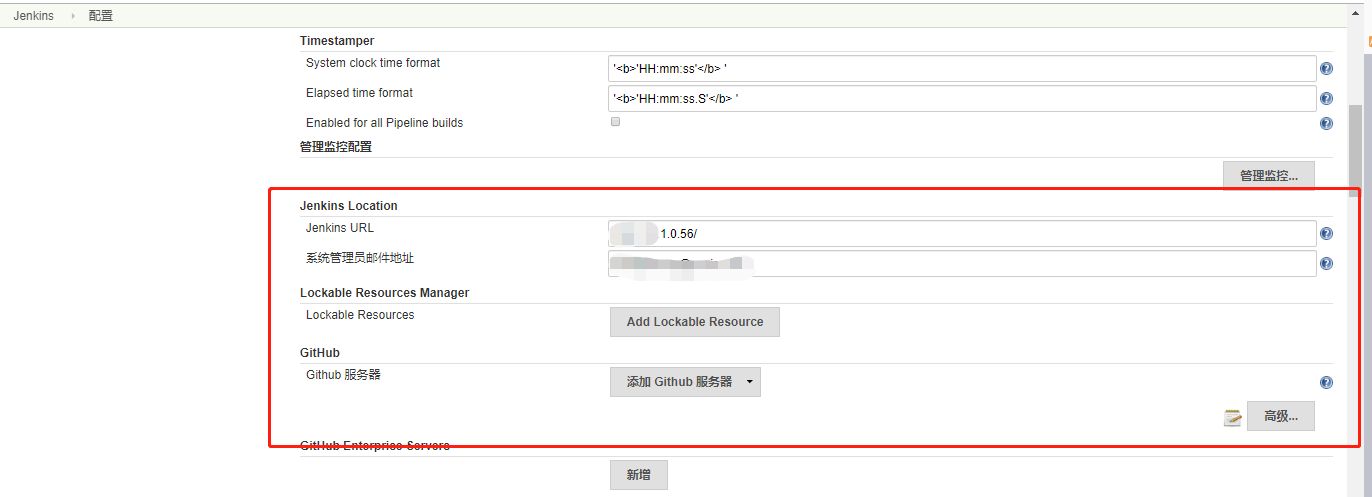
* JDK配置
* Git配置
* Maven配置
* Ant配置

Name ：ant\_1.9.6  
 ANT\_HOME： /home/data/apache-ant-1.9.6

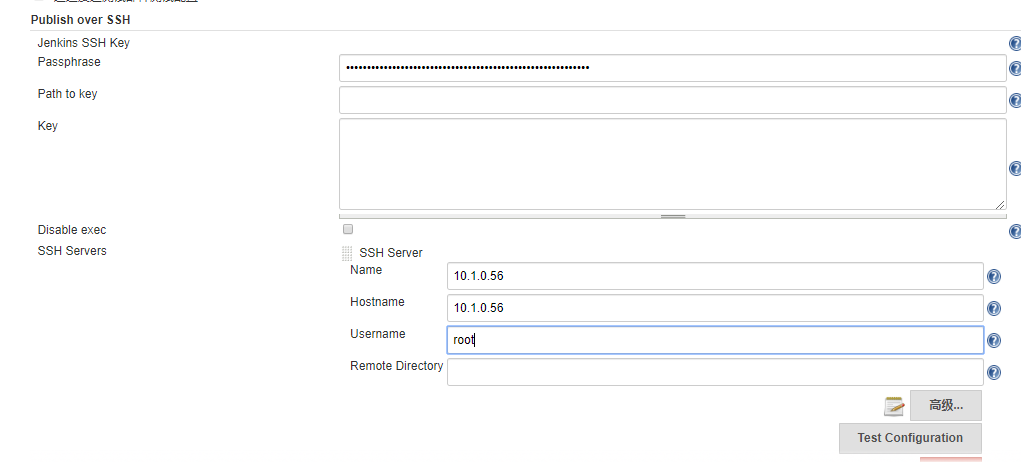
* Gradle配置

### 4.系统配置

**Jenkins Location**  
邮件配置图



**SSH配置**  
  
配置完成后点击TestConfiguration进行测试，是否连通。  
如果是服务器集群的话，就再增加一个 SSH Server。



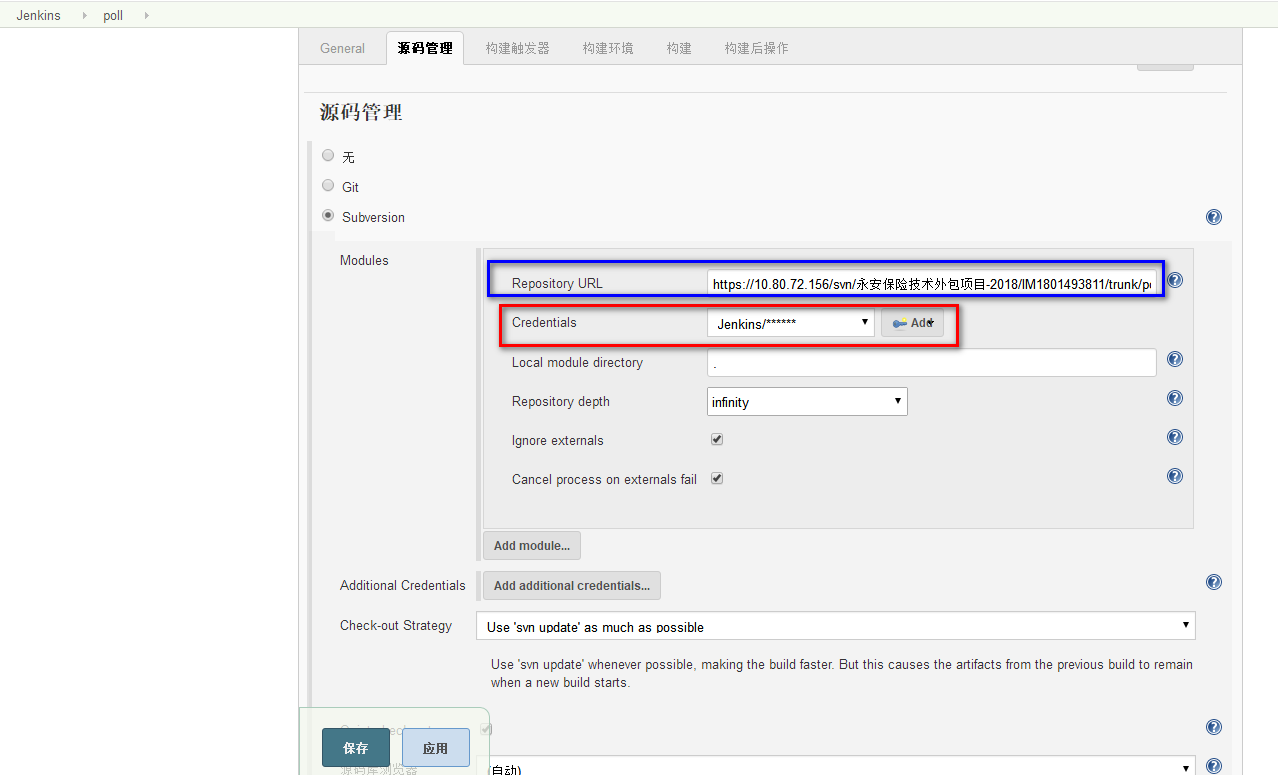
## 新建任务



### 添加任务描述



### 源码管理



选择Subversion绑定svn

Check-out Strategy各个选项详细

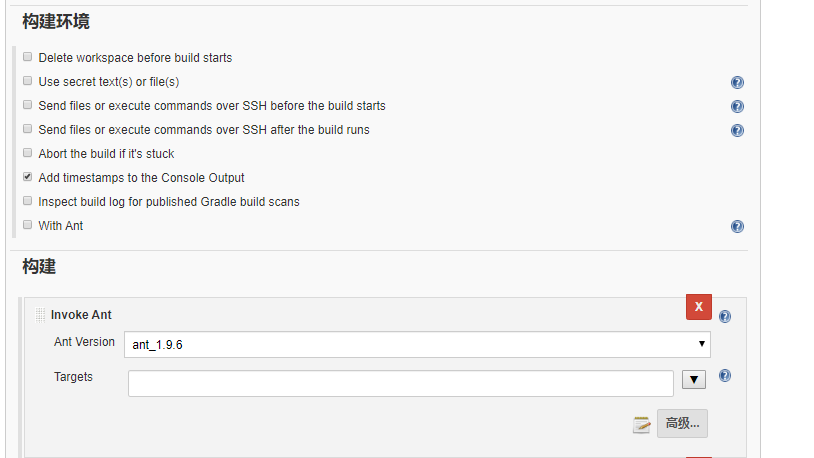
|  |  |  |
| --- | --- | --- |
| Check-out Strategy | 第一次build | 第n次build(除第一次) |
| Use 'svn update' as much as possible | 将workspace下的所有文件清空，然后从svn上check out一份完整的项目到workspace下 | update前不会revert |
| Always check out a fresh copy | 删除workspace下的所有文件，然后重新check out一份完整的项目到workspace下。 |
| Emulate clean checkout by first deleting unversioned/ignored files, then 'svn update' | update前先删除unversioned/ignored文件 |
| Use 'svn update' as much as possible, with 'svn revert' before updat | update前先revert |

参考博客：[russ44](https://blog.csdn.net/russ44/article/details/52261781)

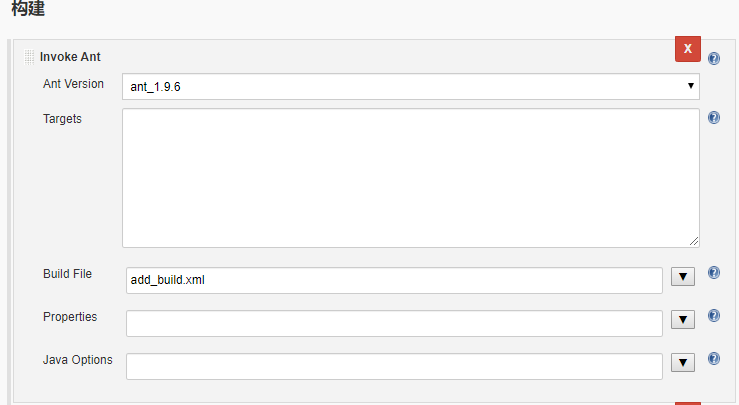
### 3.构建触发器

可选择定时构建，轮询构建等多种方式。

### 构建环境



#### 4.1配置Ant



不指定时默认使用build.xml，我这里新写了一个add\_build.xml配置后执行。

add\_build.xml：

<?xml version="1.0" encoding="UTF-8"?>  
<project name="policy\_vhl\_110" basedir="." default="package">  
 <property environment="env" />  
 <property name="ant.home" value="${env.ANT\_HOME}" />  
 <property name="java.home" value="${env.JAVA\_HOME}" />  
 <property name="src.dir" value="${basedir}/src/path" />  
 <property name="src.lib" value="${basedir}/src/templib" />  
 <property name="src.dir\_1" value="${basedir}/src/main/java" />  
 <property name="src.dir\_2" value="${basedir}/src/main/resources" />  
 <property name="src.dir\_3" value="${basedir}/src/interface" />  
 <property name="web.dir" value="${basedir}/WebRoot" />  
 <property name="build.classes" value="${web.dir}/WEB-INF/classes" />  
 <property name="dist.dir" value="${basedir}/policy\_vhl\_jar" />  
 <property name="conf.dir" value="${web.dir}/WEB-INF/classes/conf" />  
 <property name="temp\_build.dir" value="${basedir}/build" />  
 <property name="conf.dir\_1" value="${web.dir}/WEB-INF/"/>  
  
 <!-- 依赖路径，用于编译 -->  
 <path id="classpath">  
 <!--<fileset dir="${basedir}/build">  
 <include name="\*\*/\*.jar" />  
 </fileset> -->  
 <fileset dir="${web.dir}/WEB-INF/lib">  
 <include name="\*\*/\*.jar" />  
 </fileset>  
 </path>  
 <!-- 排除java源文件的模式集 -->  
 <patternset id="no.java">  
 <exclude name="\*\*/\*.java" />  
 </patternset>  
 <!-- 排除conf下配置文件 -->  
 <patternset id="properties">  
 <exclude name="\*\*/applicationDeployment.properties" />  
 <exclude name="\*\*/pcisv6DataSource.properties" />  
 <exclude name="\*\*/\*.java" />  
 </patternset>  
  
 <echo>\*\*\*\*\*\*删除pcis目录下包</echo>  
 <delete dir="${src.dir}" />  
 <delete dir="${src.lib}" />  
 <delete dir="${build.classes}" />  
 <delete dir="${basedir}/conf" />  
 <echo level="info">清理完毕</echo>  
  
 <!-- 初始化,建立目录,将多个src目录复制到同一src目录，编译-->  
 <target name="init" description="初始化,建立目录,复制文件">  
 <echo>\*\*\*\* init dir copy \*\*\*\*</echo>  
 <mkdir dir="${build.classes}" />  
 <mkdir dir="${dist.dir}" />  
 <mkdir dir="${src.dir}" />  
 <copy todir="${src.dir}">  
 <fileset dir="${src.dir\_1}"/>  
 <fileset dir="${src.dir\_2}"/>  
 <fileset dir="${src.dir\_3}"/>  
 </copy>  
 <copy todir="${build.classes}">  
 <fileset dir="${src.dir\_1}">  
 <patternset refid="no.java" />  
 <patternset refid="properties" />  
 </fileset>  
 <fileset dir="${src.dir\_2}">  
 <patternset refid="properties" />  
 </fileset>  
 <fileset dir="${src.dir\_3}">  
 <patternset refid="properties" />  
 </fileset>  
 </copy>  
 </target>  
 <!-- 编译 -->  
 <target name="compile" depends="init">  
 <echo>\*\*\*\* compile to classes dir \*\*\*\*</echo>  
 <javac srcdir="${src.dir}" destdir="${build.classes}" target="1.6" source="1.6" debug="true" debuglevel="lines,vars,source" fork="true" executable="/home/data/jdk1.6.0\_45/bin/javac" memoryMaximumSize="1024m" nowarn="on" includeantruntime="true">  
 <compilerarg line="-encoding UTF-8" />  
 <classpath refid="classpath" />  
 </javac>  
 </target>  
 <path id="lib\_classpath">  
 <fileset dir="${basedir}/">  
 <include name="svnPacket.jar"/>  
 </fileset>  
 </path>  
 <target name="package" depends="compile">   
 <echo>开始.......</echo>  
 <!-- <java classname="com.glory.svn.SVNKitUtil" classpathref="lib\_classpath">  
 </java>-->  
 <echo>结束.......</echo>  
 </target>   
</project>

#### 4.2执行shell



因为项目目前的打包方式是增量包，所以在这里使用java写了一个脚本，作用是将想发布的内容以版本号为标的，将两个版本号范围内提交的内容提取到一个文件内（svn\_version.txt），再使用打包工具将文件内的列表进行打包。  
其中主要使用的是svnkit的jar包获取svn对应版本号内提交的内容，将路径进行转换成class路径输出到目标文件内。  
具体代码如下：

private static String line = System.getProperty("line.separator", "/n");  
 private static SVNRepository repository ;  
 private static Properties properties = new Properties();  
 static {  
 try {  
 InputStream is = new BufferedInputStream(new FileInputStream("svnconfig.properties"));  
 BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(is, "UTF-8"));  
 properties.load(bufferedReader);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void setupLibrary() {  
 DAVRepositoryFactory.setup();  
 SVNRepositoryFactoryImpl.setup();  
 FSRepositoryFactory.setup();  
 String uName;  
 try {  
 uName = (String)properties.get("SVN\_URL");  
 repository = SVNRepositoryFactory.create(SVNURL.parseURIEncoded(uName));  
 } catch (SVNException e) {  
 e.printStackTrace();  
 }  
  
 uName = (String)properties.get("SVN\_USERNAME");  
 String passwd = (String)properties.get("SVN\_PASSWORD");  
 ISVNAuthenticationManager authManager = SVNWCUtil.createDefaultAuthenticationManager(uName, passwd.toCharArray());  
 repository.setAuthenticationManager(authManager);  
// try {  
// System.out.println("Repository Root:" + repository.getRepositoryRoot(true));  
// System.out.println("Repository UUID:" + repository.getRepositoryUUID(true));  
// } catch (SVNException e) {  
// e.printStackTrace();  
// }  
 }  
  
 public static void filterCommitHistoryTest() throws Exception {  
 long startRevision = 100L;  
 String versionStr = "100";  
 InputStream inputStream = null;  
 try {  
 File vFile = new File("svn\_version.txt");  
 inputStream = new FileInputStream(vFile);  
 byte[] byets = new byte[(int)vFile.length()];  
 inputStream.read(byets);  
 versionStr = new String(byets,"utf-8");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }finally {  
 if(null != inputStream) inputStream.close();  
 }  
  
 if(null != versionStr && !"".equals(versionStr)){  
 String[] versions = versionStr.split(",");  
  
 long endRevision = -1L;  
 final List<String> history = new ArrayList();  
 final List<Long> versionList = new ArrayList();  
 final Set<String> delPathSet = new HashSet();  
 for(String version : versions){  
 startRevision = Long.parseLong(version);  
 repository.log(new String[]{""}, startRevision, endRevision, true, true, new ISVNLogEntryHandler() {  
 public void handleLogEntry(SVNLogEntry svnlogentry) throws SVNException {  
 versionList.add(svnlogentry.getRevision());  
 Map<String, SVNLogEntryPath> mapLog = svnlogentry.getChangedPaths();  
 Set<String> keySet = mapLog.keySet();  
 Iterator var5 = keySet.iterator();  
 while(var5.hasNext()) {  
 String logkey = (String)var5.next();  
 char cType = ((SVNLogEntryPath)mapLog.get(logkey)).getType();  
 if ('D' == cType) {  
 delPathSet.add(logkey);  
 }  
 }  
 history.addAll((Collection)svnlogentry.getChangedPaths().keySet());  
 }  
 });  
 }  
 Set<String> fileNameSet = new HashSet();  
 StringBuilder sBuilder = new StringBuilder();  
 Iterator hisIter = history.iterator();  
  
 while(hisIter.hasNext()) {  
 String path = (String)hisIter.next();  
 if (!delPathSet.contains(path)) {  
 String className;  
 if (path.contains("WebRoot")) {  
 className = path.substring(path.indexOf("WebRoot"));  
 fileNameSet.add(className);  
 }  
  
 if (path.contains("src")) {  
 className = path.substring(path.indexOf("src"));  
 className = className.replace("src/main/java", "WebRoot/WEB-INF/classes");  
 className = className.replace("src/interface/com", "WebRoot/WEB-INF/classes/com");//只针对车承保  
 className = className.replace("src/com", "WebRoot/WEB-INF/classes/com");  
 className = className.replace("src/main/resources", "WebRoot/WEB-INF/classes");  
 className = className.replace("src", "WebRoot/WEB-INF/classes");  
 className = className.replace(".java", ".class");  
 fileNameSet.add(className);  
 }  
 }  
 }  
 int i = 0;  
 Iterator fileIter = fileNameSet.iterator();  
 while(fileIter.hasNext()) {  
 String namePath = (String)fileIter.next();  
 ++i;  
 sBuilder.append(namePath);  
 if (i != fileNameSet.size()) {  
 sBuilder.append(line);  
 }  
 }  
 OutputStream outputStream = null;  
 try {  
 File file = new File("file\_list.txt");  
 if (!file.exists()) {  
 try {  
 file.createNewFile();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 outputStream = new FileOutputStream(file);  
 outputStream.write(sBuilder.toString().getBytes());  
 outputStream.flush();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }finally {  
 if(null != outputStream) outputStream.close();  
 }  
 }  
 }  
 public static void main(String[] args) throws Exception {  
 System.out.println("打包工程start");  
 setupLibrary();  
 filterCommitHistoryTest();  
 System.out.println("打包工程end");  
 }

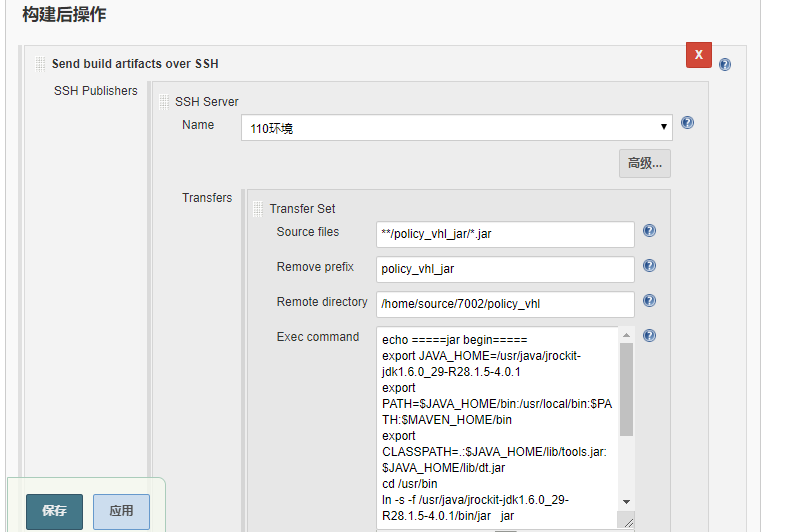
命令：

echo =====gen file list begin=====   
cd /var/lib/jenkins/workspace/policy\_vhl\_110/   
java -jar svnPacket.jar  
echo =====gen file list end=====   
echo =====make jar======  
jar cvf policy\_vhl\_jar/policy\_vhl.jar @file\_list.txt  
echo ====make jar over=====

简述下流程

1. 首先代码提交开发版本库
2. 由开发版本库将内容合并至DEV版本库
3. 提交svn*version.txt文件，里面记录着想发布到DEV环境的SVN version编号。例如：svn列表由上到下为1、2、3、4代码无交叉，svn*version记录着2，3 ，则虽然1、2、3、4代码已经发布了DEV环境的代码基线，并且其代码已经构建，但是只会将2，3中提交的代码进行打包。
4. 根据file\_list.txt记录的具体class路径进行打包，获得指定版本编号范围内的代码。
5. 将jar发布至远程服务，解压重启。

### 构建后操作



指定环境变量，解压jar包后调用jenkins.sh脚本进行重启。（容器为weblogic）

echo =====jar begin=====  
export JAVA\_HOME=/usr/java/jrockit-jdk1.6.0\_29-R28.1.5-4.0.1  
export PATH=$JAVA\_HOME/bin:/usr/local/bin:$PATH:$MAVEN\_HOME/bin  
export CLASSPATH=.:$JAVA\_HOME/lib/tools.jar:$JAVA\_HOME/lib/dt.jar  
cd /usr/bin   
ln -s -f /usr/java/jrockit-jdk1.6.0\_29-R28.1.5-4.0.1/bin/jar jar  
  
cd /home/source/7002/policy\_vhl  
jar -xvf policy\_vhl.jar  
echo =====jar end=====  
cd /home/oracle/Oracle/Middleware/user\_projects/domains/hhuat7002/  
./jenkins.sh &

jenkins.sh脚本主要是进行重启，最后将jar包按照时间戳进行重命名。  
脚本：

#!/bin/bash  
#jenkins 重启脚本  
  
#删除lok文件  
find /home/oracle/Oracle/Middleware/user\_projects/domains/hhuat7002/ -name "\*.lok" -print -exec rm -rf {} \;  
#清除服务器缓存  
cd /home/oracle/Oracle/Middleware/user\_projects/domains/hhuat7002/servers/AdminServer/tmp/\_WL\_user  
rm -rf pcisv7  
#查找并强制杀死进程  
ps -ef|egrep 7002|gawk '{print $2}'|while read pid  
do  
kill -9 $pid  
done  
#以当前时间命名后台日志，并显示  
str=$"/n"  
MYDATE=`date +%Y%m%d%H%M%S`   
echo "$MYDATE"  
#清除历史日志，防止占内存过多  
cd /home/oracle/Oracle/Middleware/user\_projects/domains/hhuat7002/logs  
rm -rf \*.log  
cd /home/oracle/Oracle/Middleware/user\_projects/domains/hhuat7002/bin  
nohup ./startWebLogic.sh>/home/oracle/Oracle/Middleware/user\_projects/domains/hhuat7002/logs/$MYDATE.log 2>&1 &  
# 重命名jar包  
mv /home/source/7002/policy\_vhl/policy\_vhl.jar policy\_vhl.$MYDATE.jar

### 立即构建



## 三、问题处理

### ant编译时乱码问题：

编译时遇到 “非法字符： \65279”的报错

[javac] /var/lib/jenkins/workspace/policy\_vhl\_110/src/path/interface/com/isoftstone/pcis/policy/foreignInterface/service/impl/ElectricityPinServiceImpl.java:1: 非法字符： \65279

解决:https://blog.csdn.net/netwalk/article/details/52005546

### 软件包 javax.servlet 不存在

软件包 javax.servlet 不存在  
[javac] import javax.servlet.ServletException;

解决办法：  
从tomcat lib目录下拷贝一个servlet-api.jar的包到“JDK\jre\lib\ext”目录下  
凡是出现找不到包的情况，都可以将找到的包放到JDK\jre\lib\ext下，然后再编译就能够通过。

## 四、扩展资料

### 1.Ant 构建相关连接

https://blog.csdn.net/Al\_assad/article/details/76285841  
https://blog.csdn.net/mevicky/article/details/72828554  
https://blog.csdn.net/xxdddail/article/details/21166161  
https://blog.csdn.net/yang3wei/article/details/7393399  
https://www.cnblogs.com/fnlingnzb-learner/p/6279189.html  
https://www.yiibai.com/ant/apache-ant-javac-task.html

### 2.Svnkit 组件使用：

https://www.cnblogs.com/douJiangYouTiao888/p/6138660.html

### 3. Jenkins+Ant

https://blog.csdn.net/xiaxiaozhang/article/details/74155300  
jekins svn任务配置：https://blog.csdn.net/russ44/article/details/52261781