Maven项目自动化部署

## 19.1 思路

通过结合以下方案来实现自动化部署：

* 使用 Maven 构建和发布项目
* 使用 git， 源码仓库来管理源代码
* 使用远程仓库管理软件（Jfrog或者Nexus） 来管理项目二进制文件。

## 19.2 自动化部署工具类项目

### 19.2.1 配置pom.xml文件

<scm>  
 <url>http://www.git.com</url>  
 <connection>scm:git:http://localhost:8080/git/jrepo/trunk/  
 Framework</connection>  
 <developerConnection>scm:svn:${username}/${password}@localhost:8080:  
 common\_core\_api:1101:code</developerConnection>  
 </scm>  
 <distributionManagement>  
 <repository>  
 <id>Core-API-Java-Release</id>  
 <name>Release repository</name>  
 <url>http://localhost:8081/nexus/content/repositories/  
 Core-Api-Release</url>  
 </repository>  
 </distributionManagement>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-release-plugin</artifactId>  
 <version>2.0-beta-9</version>  
 <configuration>  
 <useReleaseProfile>false</useReleaseProfile>  
 <goals>deploy</goals>  
 <scmCommentPrefix>[bus-core-api-release-checkin]-<  
 /scmCommentPrefix>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
  
  
###

### 19.2.2 Maven Release 插件

#### 19.2.2.1 清理

Maven 使用 maven-release-plugin 插件来完成以下任务。

mvn release:clean

清理工作空间，保证最新的发布进程成功进行。

#### 19.2.2.2 回滚

mvn release:rollback

在上次发布过程不成功的情况下，回滚修改的工作空间代码和配置保证发布过程成功进行。

mvn release:prepare

执行多种操作：

* 检查本地是否存在还未提交的修改
* 确保没有快照的依赖
* 改变应用程序的版本信息用以发布
* 更新 POM 文件到 SVN
* 运行测试用例
* 提交修改后的 POM 文件
* 为代码在 SVN 上做标记
* 增加版本号和附加快照以备将来发布
* 提交修改后的 POM 文件到 SVN

mvn release:perform

将代码切换到之前做标记的地方，运行 Maven 部署目标来部署 WAR 文件或者构建相应的结构到仓库里。

打开命令终端，进入到 C:\ > MVN >bus-core-api 目录下，然后执行如下的 mvn 命令。

C:\MVN\bus-core-api>mvn release:prepare

#### 19.2.2.2 部署

Maven 开始构建整个工程。构建成功后即可运行如下 mvn 命令。

C:\MVN\bus-core-api>mvn release:perform

构建成功后，验证在你仓库下上传的 JAR 文件是否生效。

## 19.3 部署war包

maven的自动部署功能可以很方便的将maven工程自动部署到远程tomcat服务器

### 19.3.1 配置 Tomcat 访问权限

首先，我们需要先打开 Tomcat 的 manager 功能，找到 conf 文件夹下的 tomcat-users.xml文件中的 标签，然后添加如下内容（可以直接在其文档注释部分找到对应的模版，然后进行修改）：

<role rolename="manager-gui"/>   
<role rolename="manager-script"/>  
<role rolename="manager-jmx"/>  
<role rolename="manager-status"/>  
<user password="1234" username="admin"  
roles="manager-gui,manager-script,manager-jmx,manager-status" />

配置好之后，ctrl+s 保存文件。紧接着，双击 tomcat 解压包中 bin 目录下的 startup.bat 命令进行启动Tomcat服务器。在浏览器地址来中进行访问http://localhost:8080/manager，

按下 Enter 回车键，即可看到弹窗，需要我们输入上面配置好的用户名和密码，才能进行登录，如果顺利则请进入下一步。

### 19.3.2 配置maven的settings.xml

在 conf/settings.xml 文件中的标签 添加子标签。通过标签名字，我们知道这主要是为了让 maven 去关联我们的 Tomcat 服务器。

注意，这里配置的 username 和 password 一定要和 tomcat 中的 tomcat\_user.xml 中一致，否则关联不起来。

<server>   
 <id>tomcat9</id>  
 <username>admin</username>  
 <password>1234</password>  
</server>

### 19.3.3 修改pom.xml文件

最后，回到我们的 Eclipse 中，然后在 pom.xml 文件中，在原来 tomcat7 插件的基础上，往 下添加 子标签进行配置即可。

```  
<build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.3</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
  
  
 <plugin>  
 <groupId>org.apache.tomcat.maven</groupId>  
 <artifactId>tomcat7-maven-plugin</artifactId>  
 <version>2.2</version>  
 <configuration>  
 <!-- 直接访问 Tomcat 服务器的 manager -->  
 <url>http://localhost:8080/manager/text</url>  
 <server>tomcat9</server>  
 <username>admin</username>  
 <password>1234</password>  
 <update>true</update>  
 <path>/webapp</path>  
 </configuration>  
 </plugin>  
  
 </plugins>  
 </build>

### 19.3.4 执行命令

(执行过程中，tomcat9要先启动)

1）Run as → clean install  
2）Run as → tomcat7:deploy 注：第1次部署执行  
3）Run as → tomcat7:redeploy 注：第2次或以后需要重新发布执行  
4）Run as → tomcat7:run 注：部署到 tomcat 中启动（执行之前先关闭tomcat9，如果部署到本地，防止端口占用启动不了）