Maven 依赖管理 Maven Web 应用

Maven 自动化部署

项目开发过程中，部署的过程包含需如下步骤：

- 将所的项目代码提交到 SVN 或者代码库中并打上标签。
- 从 SVN 上下载完整的源代码。
- 构建应用。
- 存储构建输出的 WAR 或者 EAR 文件到一个常用的网络位置下。
- 从网络上获取文件并且部署文件到生产站点上。
- 更新文档并且更新应用的版本号。

问题描述

通常情况下上面的提到开发过程中会涉及到多个团队。一个团队可能负责提交代码，另一个团队负责构建等等。很有可能由于涉及的人为操作和多团队环境的原因，任何一个步骤都可能出错。比如，较旧的版本没有在网络机器上更新，然后部署团队又重新部署了较早的构建版本。

解决方案

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 SubVersion， 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。

修改项目的 pom.xml

我们将会使用 Maven 发布的插件来创建一个自动化发布过程。

例如，bus-core-api 项目的 pom.xml 文件代码如下：

```
<project 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/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>bus-core-api</groupId>
<artifactId>bus-core-api</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
<scm>
<url>http://www.svn.com</url>
<connection>scm:svn:http://localhost:8080/svn/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>
</project>
```

在 pom.xml 文件中，我们常用到的一些重要元素节点如下表所示：

元素节点	描述
SCM	配置 SVN 的路径，Maven 将从该路径下将代码取下来。
repository	构建的 WAR 或 EAR 或JAR 文件的位置，或者其他源码构建成功后生成的构件的存储位置。
Plugin	配置 maven-release-plugin 插件来实现自动部署过程。

Maven Release 插件

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。

```
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
```

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，你就可以可以验证在你仓库下上传的 JAR 文件是否生效。

← Maven 依赖管理 Maven Web 应用 →