Maven 的使用与技巧

**Maven 的安装**

**下载地址：http://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.5.0/binaries/apache-maven-3.5.0-bin.tar.gz**

**配置环境变量：**

**M2\_HOME**

**Maven 对应项目结构**

Project name

--src

-----main

--------java

--------resources

-----test

--------java

--------resources

**POM 文件介绍与基本组成**

说明：全称是Project Object Model,通俗点的话说就是要对构建的项目进行建模。

组成的基本元素：

|  |  |  |
| --- | --- | --- |
| 元素 | 可选值 | 描述 |
| groupId |  | 分组ID |
| artifactId |  | 模块id |
| version |  | 版本 |
| packaging |  | 打包类型：pom、jar、war |
| modelVersion |  | 对应的超级pom 版本 |
| dependencies |  | 项目依懒包 |

### **maven repository 与镜像地址**

[http://mvnrepository.com](http://mvnrepository.com/) maven 仓库用于查找所需要pom项目

<http://repo1.maven.org/maven2/> 全球总仓库1

<http://repo2.maven.org/maven2/> 全球总仓库2

http://maven.aliyun.com/nexus/content/groups/public/ 阿里云镜像仓库

http://maven.oschina.net/content/groups/public/ oschina 镜像仓库

镜像配置

修改settings.xml 文件

<mirrors>

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

<mirrorOf>central</mirrorOf>

</mirror>

<!-- 中央仓库1 -->

<mirror>

<id>repo1</id>

<mirrorOf>central</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://repo1.maven.org/maven2/</url>

</mirror>

<!-- 中央仓库2 -->

<mirror>

<id>repo2</id>

<mirrorOf>central</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://repo2.maven.org/maven2/</url>

</mirror>

</mirrors>

mvn 基本命令

mvn clean // 清理

mvn compile // 编译

mvn test // 测试

mvn package // 打包

mvn install // 打包并上传到本地仓库

mvn depeloy // 上传到远程仓库

mvn -Dmaven.test.skip=true // 跳过测试

## **maven 私服搭建**

**nexus 下载地址：**

https://sonatype-download.global.ssl.fastly.net/nexus/oss/nexus-2.14.5-02-bundle.tar.gz

**解压并设置环境变量**

#解压

shell>tar -zxvf [nexus-2.14.5-02-bundle.tar.gz](https://sonatype-download.global.ssl.fastly.net/nexus/oss/nexus-2.14.5-02-bundle.tar.gz)

#在环境变量当中设置启动用户

shell> vim /etc/profile

#添加profile文件。安全起见不建议使用root用户，如果使用其它用户需要加相应权限

RUN\_AS\_USER=root

**配置启动参数：**

shell> vi ${nexusBase}/conf/nexus.properties

#端口号

application-port=8081

#仓库地址

nexus-work=${bundleBasedir}/../sonatype-work/nexus

启动与停止nexus

#启动

shell> ${nexusBase}/bin/nexus start

#停止

shell> ${nexusBase}/bin/nexus stop

登录nexus 界面

**地址**：http://{ip}:8081/nexus/

**用户名**:admin

**密码**：admin123

**手动更新索引**

*前往maven中央仓库下载*[indexer-cli-5.1.1.jar](http://search.maven.org/" \l "search|gav|1|g%3A"org.apache.maven.indexer" AND a%3A"indexer-cli")

http://search.maven.org/#search%7Cgav%7C1%7Cg%3A%22org.apache.maven.indexer%22%20AND%20a%3A%22indexer-cli%22

**下载索引文件**

<http://repo.maven.apache.org/maven2/.index/>

nexus-maven-repository-index.gz

nexus-maven-repository-index.properties

**解压索引引文件**

java -jar indexer-cli-5.1.1.jar -u nexus-maven-repository-index.gz -d indexer

**拷贝索引文件至索引目录**

{nexus-home}/sonatype-work/nexus/indexer/central-ctx

## **基本插件的应用**

常用的插件-----------

**设置jre 版本**

两种方式：第一种是**局部项目**修改 pom.xml 设置compiler 插件。第二种是**全局**修改setting.xml 文件 配 置profile

<!-- 修改compiler 插件已设置jre 源码版本和编译版本-->

**<plugin>**

**<groupId>**org.apache.maven.plugins**</groupId>**

**<artifactId>**maven-compiler-plugin**</artifactId>**

**<configuration>**

**<source>**1.7**</source>**

**<target>**1.7**</target>**

**</configuration>**

**</plugin>**

<!-- 设置setting.xml 文件中 为profile 添加属性-->

**<properties>**

**<maven.compiler.source>**1.7**</maven.compiler.source>**

**<maven.compiler.target>**1.7**</maven.compiler.target>**

**<maven.compiler.compilerVersion>**1.7**</maven.compiler.compilerVersion>**

**</properties>**

**生成一个源码包：**

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-source-plugin</artifactId>

<version>2.4</version>

<executions>

<execution>

<id>attach-source</id>

<phase>verify</phase>

<goals>

<goal>jar-no-fork</goal>

</goals>

</execution>

</executions>

</plugin>

**测试指定范围**

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.20</version>

<configuration>

<includes>

<include>\*\*/User\*.java</include>

</includes>

</configuration>

</plugin>

## **生命周期 lifecycle**

**clean lifecycle** : 构建前的清理工作

* pre-clean 执行一些需要在clean之前完成的工作
* clean 移除所有上一次构建生成的文件
* post-clean 执行一些需要在clean之后立刻完成的工作

**Default lifecycle**： 构建的核心部分，编译、打包、部署、上传

* validate 项目及所必须的环境验证
* initialize 初始化构建状态，例如设置属性或创建目录。
* generate-sources 生成次源包
* process-sources
* generate-resources
* process-resources 复制并处理资源文件，至目标目录，准备打包。
* compile 编译项目的源代码。
* process-classes
* generate-test-sources
* process-test-sources
* generate-test-resources
* process-test-resources 复制并处理资源文件，至目标测试目录。
* test-compile 编译测试源代码。
* process-test-classes
* test 使用合适的单元测试框架运行测试。这些测试代码不会被打包或部署。
* prepare-package
* package 接受编译好的代码，打包成可发布的格式，如 JAR 。
* pre-integration-test
* integration-test
* post-integration-test
* verify
* install 将包安装至本地仓库，以让其它项目依赖。
* deploy 将最终的包复制到远程的仓库，以让其它开发人员与项目共享。

**site lifecycle**：项目报告生成，站点文档生成

* pre-site 执行一些需要在生成站点文档之前完成的工作
* site 生成项目的站点文档
* post-site 执行一些需要在生成站点文档之后完成的工作，并且为部署做准备
* site-deploy 将生成的站点文档部署到特定的服务器上

*原官方档：*

*http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html*