Maven

# 1 认识maven

## maven的优势

1. **约定由于配置**
2. **简单**
3. **测试支持**
4. **构建简单**
5. **CI**
6. **插件丰富**

## 下载

* + - * 1. **下载地址：** [**https://maven.apache.org/download.cgi**](https://maven.apache.org/download.cgi)
        2. **安装**
        3. **maven-model-builder-3.3.9.jar/org/apache/maven/model**
        4. **配置 MVM\_HOME**

Windows path

Linux .bash\_profile

MAVEN\_OPTS

配置setting.xml

配置国内镜像：

**<mirror>**

**<id>alimaven</id>**

**<name>aliyun maven</name>**

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

**<mirrorOf>central</mirrorOf>**

**</mirror>**

**<mirror>**

**<id>ui</id>**

**<mirrorOf>central</mirrorOf>**

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

**<url>http://uk.maven.org/maven2/</url>**

**</mirror>**

**<mirror>**

**<id>osc</id>**

**<mirrorOf>central</mirrorOf>**

**<url>http://maven.oschina.net/content/groups/public/</url>**

**</mirror>**

**<mirror>**

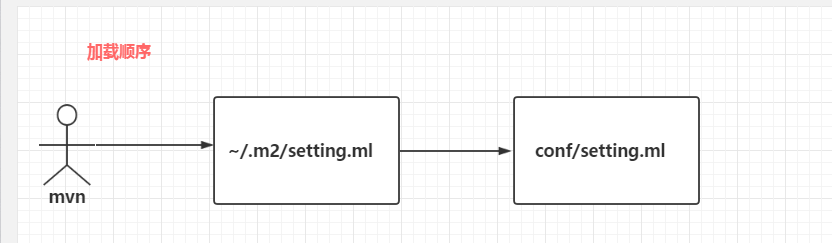
**<id>osc\_thirdparty</id>**

**<mirrorOf>thirdparty</mirrorOf>**

**<url>http://maven.oschina.net/content/repositories/thirdparty/</url>**

**</mirror>**

**配置文件加载顺序：**



## 3. pom

groupId ：com.maven => 公司名称

artifactId ：rxl-maven => 功能名称

version ：1.0-SNAPSHOT => 版本号

packaging ：打包方式 默认是jar

dependcyManagement :

* + - 1. 只能出现在父pom
      2. 统一版本号
      3. 声明(子pom在引用)

dependcy :

* + - 1. type 默认为jar
      2. scope

1. compile 编译 例如 spring-core
2. test 测试
3. provided 编译 例如 servlet
4. runtime 运行时 例如JDBC驱动实现
5. system 本地一些jar 例如短信jar

## 4. 依赖传递

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Compile | Test | provided | Runtime |
| Compile | Compile | - | - | Runtime |
| test | test | - | - | test |
| Provided | Provided | - | Provided | Provided |
| Runtime | Runtime | - | - | Runtime |

## 5. 依赖仲裁

1. 最短路径原则

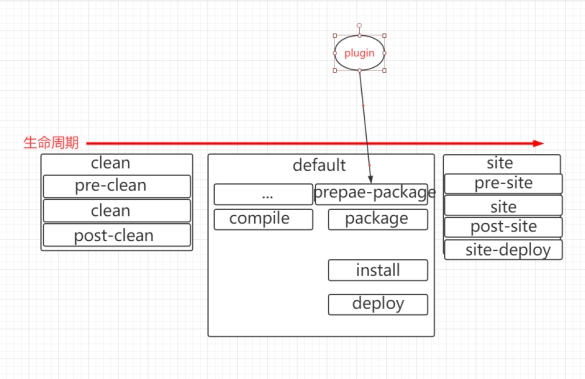
2. 加载先后原则

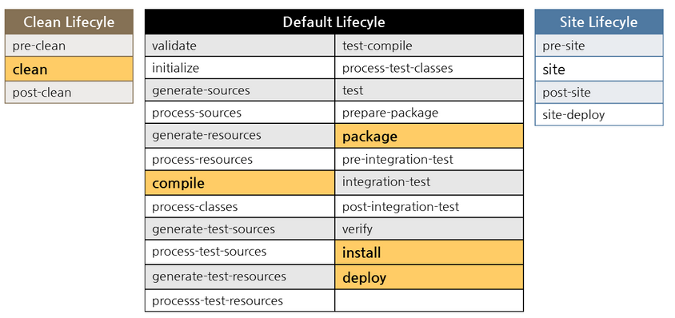
## 6. 排除包 (exclusions)

## 7. 构建生命周期

1. 生命周期 lifecycle/phase/goal

* 1. A Build Lifecycle is Made Up of Phases

3. A Build Phase is Made Up of Plugin Goals



# 2 常用命令及常用插件

<https://maven.apache.org/plugins/> <http://www.mojohaus.org/plugins.html>

## 1. 统一设置版本插件

<build>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>versions-maven-plugin</artifactId>

<version>2.3</version>

</plugin>

</plugins>

</build>

命令 ： mvn versions:set -DnewVersion=1.1-SNAPSHOT

## 查看依赖树

命令 ： mvn dependency:tree > d.txt

## 静态代码检查- findbugs

https://gleclaire.github.io/findbugs-maven-plugin/usage.html

## 打包源代码- source

<http://maven.apache.org/plugins/maven-source-plugin/usage.html>

## 打包源代码- assembly

<http://maven.apache.org/components/plugins/maven-assembly-plugin/>

## 服务器插件-tomcat

# 3 自定义插件

<https://maven.apache.org/guides/plugin/guide-java-plugin-development.html>

## 步骤

1 新建一个maven项目

2 <packaging>maven-plugin</packaging>

3 extends AbstractMojo

4 mvn install

5 参数传递

# 4 Profiles

使用场景 dev/test/pro

配置：

<**profiles**>  
 <**profile**>  
 <**id**>dev</**id**>  
 <**properties**>  
 <**profiles.active**>dev</**profiles.active**>  
 </**properties**>  
 <**activation**>  
 <**activeByDefault**>true</**activeByDefault**>  
 </**activation**>  
 </**profile**>  
 <**profile**>  
 <**id**>pro</**id**>  
 <**properties**>  
 <**profiles.active**>pro</**profiles.active**>  
 </**properties**>  
 </**profile**>  
 <**profile**>  
 <**id**>test</**id**>  
 <**properties**>  
 <**profiles.active**>test</**profiles.active**>  
 </**properties**>  
 </**profile**>  
</**profiles**>  
  
<**build**>  
 <**resources**>  
 <**resource**>  
 <**directory**>${project.basedir}/src/main/resources</**directory**>  
 <**excludes**>  
 <**exclude**>conf/\*\*</**exclude**>  
 </**excludes**>  
 </**resource**>  
 <**resource**>  
 <**directory**>${project.basedir}/src/main/resources/conf/${profiles.active}</**directory**>  
 </**resource**>  
 </**resources**>

打包命令：mvn clean install -P dev/pro/test

# 5 Maven私服

## Linux下安装

<https://www.sonatype.com/download-oss-sonatype>

### 下载

官网下载即可

### 解压

[root@MiWiFi-R3-srv ~]# tar -zxvf nexus-3.6.0-02-unix.tar.gz -C /usr/local/

### 启动

[root@MiWiFi-R3-srv ~]# cd /usr/local/nexus-3.6.0-02/bin/

[root@MiWiFi-R3-srv ~]# ./nexus run &

### 开放防火墙

[root@MiWiFi-R3-srv bin]# firewall-cmd --zone=public --add-port=8081/tcp --permanent

[root@MiWiFi-R3-srv bin]# firewall-cmd –reload

### 设置开机自启动

**创建一个服务**

[root@MiWiFi-R3-srv ~]# vim /usr/lib/systemd/system/nexus.service

**填入相关内容**

[Unit]

Description=nexus service

[Service]

Type=forking

LimitNOFILE=65536 #警告处理

ExecStart=/usr/local/nexus/nexus-3.7.1-02/bin/nexus start

ExecReload=/usr/local/nexus/nexus-3.7.1-02/bin/nexus restart

ExecStop=/usr/local/nexus/nexus-3.7.1-02/bin/nexus stop

Restart=on-failure

[Install]

WantedBy=multi-user.target

**将服务加入开机启动**

[root@MiWiFi-R3-srv ~]# systemctl enable nexus.service

**重新加载配置文件**

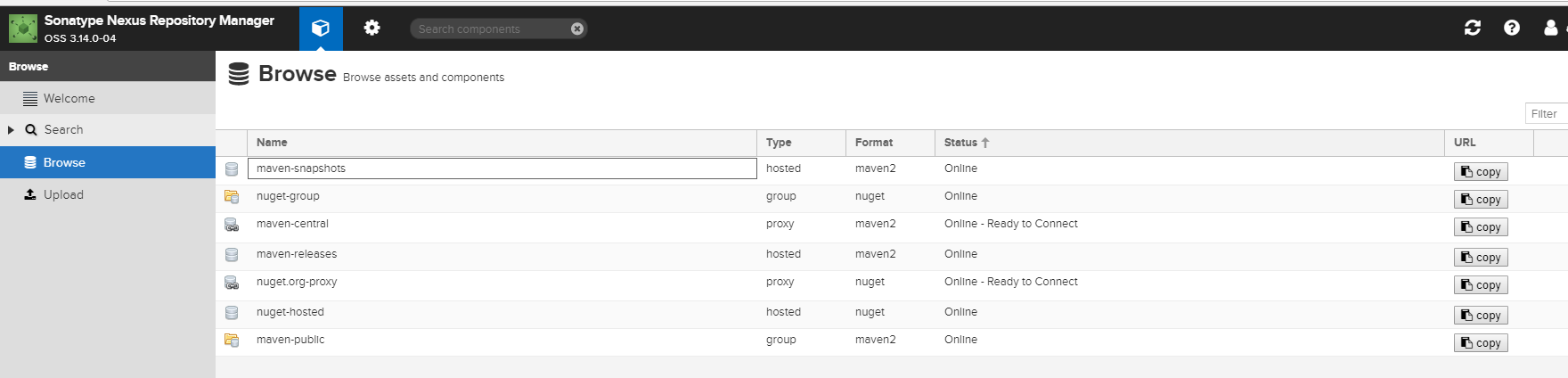
[root@MiWiFi-R3-srv ~]# systemctl daemon-reload

### 6. 修改nexus3的运行用户为root

[root@MiWiFi-R3-srv bin]# vim nexus.rc

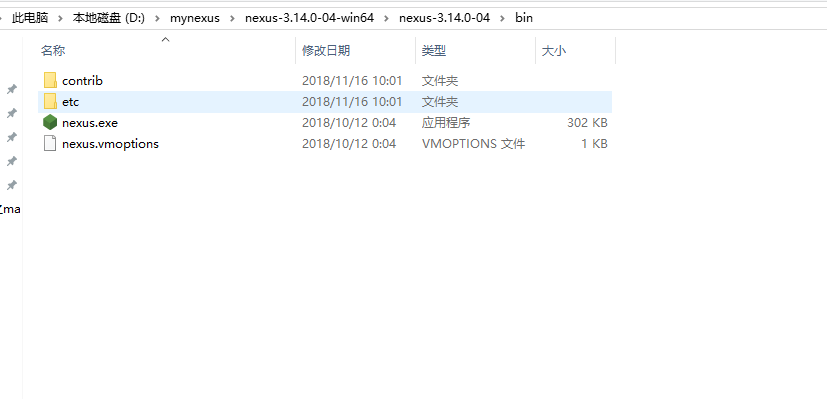
run\_as\_user="root"

### 7. 测试



默认：admin-admin123

## Windows下安装



直接下载压缩包解压，点击nexus.exe运行即可。

## 发布到私服

### 配置POM

<**distributionManagement**>  
 <**repository**>  
 <**id**>nexus-releases</**id**>  
 <**name**>Nexus Release Repository</**name**>  
 <**url**>http://10.30.171.161:8081/repository/maven-releases/</**url**>  
 </**repository**>  
 <**snapshotRepository**>  
 <**id**>nexus-snapshots</**id**>  
 <**name**>Nexus Snapshot Repository</**name**>  
 <**url**>http://10.30.171.161:8081/repository/maven-snapshots/</**url**>  
 </**snapshotRepository**>  
</**distributionManagement**>

### 配置Settings.xml

配置Nexus Repository访问权限

**<servers>**

**<server>**

**<id>nexus-releases</id>**

**<username>admin</username>**

**<password>admin123</password>**

**</server>**

**<server>**

**<id>nexus-snapshots</id>**

**<username>admin</username>**

**<password>admin123</password>**

**</server>**

**</servers>**

### 发布

mvn clean deploy

## 从私服下载jar

### 配置mirror

**<****mirror>**

**<id>nexusMirror</id>**

**<name>raoxianlong maven</name>**

**<url>** **http://localhost:8081/repository/maven-public/</url>**

**<****mirrorOf>central</mirrorOf>**

**</mirror>**

配置远程仓库镜像，当访问中央仓库时，默认去私服找，如果没有找到有，也不会去中央仓库找。

### 配置Profile(Settings.xml文件中)

<**profiles**>  
 <**profile**>  
 <**id**>LocalRepository</**id**>  
 <**repositories**>  
 <**repository**>  
 <**id**>local nexus</**id**>  
 <**name**>rao xian long de sf</**name**>  
 <**url**>http://localhost:8081/repository/maven-public/</**url**>  
 <**releases**>  
 <**enabled**>true</**enabled**>  
 <**updatePolicy**>daily</**updatePolicy**>  
 </**releases**>  
 <**snapshots**>  
 <**enabled**>false</**enabled**>  
 </**snapshots**>  
 </**repository**>  
 </**repositories**>  
 </**profile**>  
 <**profile**>  
 <**id**>CentralRepository</**id**>  
 <**repositories**>  
 <**repository**>  
 <**id**>central</**id**>  
 <**name**>Central Repository</**name**>  
 <**url**>https://repo.maven.apache.org/maven2</**url**>  
 <**layout**>default</**layout**>  
 <**snapshots**>  
 <**enabled**>false</**enabled**>  
 </**snapshots**>  
 </**repository**>  
 </**repositories**>  
 </**profile**>  
 </**profiles**>

<**activeProfiles**>  
 <**activeProfile**>LocalRepository</**activeProfile**>  
</**activeProfiles**>

**repository：**在repositories元素下，可以使用repository子元素声明一个或者多个远程仓库。

**id：**仓库声明的唯一id，尤其需要注意的是，Maven自带的中央仓库使用的id为central，如果其他仓库声明也使用该id，就会覆盖中央仓库的配置。

**name：**仓库的名称，让我们直观方便的知道仓库是哪个，暂时没发现其他太大的含义。

**url：**指向了仓库的地址，一般来说，该地址都基于http协议，Maven用户都可以在浏览器中打开仓库地址浏览构件。

**releases和snapshots：**用来控制Maven对于发布版构件和快照版构件的下载权限。需要注意的是**enabled**子元素，该例中releases的enabled值为true，表示开启JBoss仓库的发布版本下载支持，而snapshots的enabled值为false，表示关闭JBoss仓库的快照版本的下载支持。根据该配置，Maven只会从JBoss仓库下载发布版的构件，而不会下载快照版的构件。

**layout：**元素值default表示仓库的布局是Maven2及Maven3的默认布局，而不是Maven1的布局。基本不会用到Maven1的布局。

**其他：**对于releases和snapshots来说，除了enabled，它们还包含另外两个子元素updatePolicy和checksumPolicy。

元素**updatePolicy**用来配置Maven从远处仓库检查更新的频率，默认值是daily，表示Maven每天检查一次。其他可用的值包括：never-从不检查更新；always-每次构建都检查更新；interval：X-每隔X分钟检查一次更新（X为任意整数）。

元素**checksumPolicy**用来配置Maven检查校验和文件的策略。当构建被部署到Maven仓库中时，会同时部署对应的检验和文件。在下载构件的时候，Maven会验证校验和文件，如果校验和验证失败，当checksumPolicy的值为默认的warn时，Maven会在执行构建时输出警告信息，其他可用的值包括：fail-Maven遇到校验和错误就让构建失败；ignore-使Maven完全忽略校验和错误。

### Profile、Repository、Mirror关系

Profile ：组合仓库，一般放在Settings文件中配置，属于全局配置。

Repository：配置仓库，一般放在POM.xml文件中配置

Mirror：镜像，配置仓库的关系。配置多个mirror是根据id的字母进行排序。

# 6 创建脚手架-archetype

**步骤**：

1.创建一个项目之后

mvn archetype:create-from-project

1. 到下面的目录中去

cd /target/generated-sources/archetype

1. install到本地仓库

mvn install

1. 通过archetype创建项目

mvn archetype:generate-DarchetypeCatalog=local