kira719527102@hotmail.com

Steven

摘要

Maven仓库，配置文件，生命周期等大致讲解

Maven文档

目录

[一．Maven仓库结构： 3](#_Toc467835390)

[1. 结构说明 3](#_Toc467835391)

[2. 对于仓库的改动： 4](#_Toc467835392)

[代理仓库 4](#_Toc467835393)

[版本控制 4](#_Toc467835394)

[仓库权限配置 5](#_Toc467835395)

[二．Maven配置文件（聚合与继承）： 6](#_Toc467835396)

[1. 结构说明： 6](#_Toc467835397)

[2. 依赖管理: 7](#_Toc467835398)

[父项目： 7](#_Toc467835399)

[子项目： 8](#_Toc467835400)

[3. 打包环境配置： 8](#_Toc467835401)

[4. 上传私库配置： 9](#_Toc467835402)

[5. 更新策略： 9](#_Toc467835403)

[上传策略： 9](#_Toc467835404)

[下载策略： 10](#_Toc467835405)

[6. 对于pom文件的调整： 11](#_Toc467835406)

[1. 环境参数 11](#_Toc467835407)

[2. 仓库配置 11](#_Toc467835408)

[3. 依赖管理 11](#_Toc467835409)

[4. 更新策略 11](#_Toc467835410)

[1. 生命周期解读： 12](#_Toc467835411)

[2. 参数命令： 13](#_Toc467835412)

[四．约定优于配置： 14](#_Toc467835413)

[附录： 15](#_Toc467835414)

[Pom示例： 15](#_Toc467835415)

[父项目pom： 15](#_Toc467835416)

[子项目pom： 18](#_Toc467835417)

# 一．Maven仓库结构

## 结构说明

本地仓库从服务器仓库组下载项目依赖的jar包，服务器仓库又分为私库和远程公共库

墨迹绘图
Maven本地仓库
仓库组
墨迹绘图
私库A
私库B
墨迹绘图
墨迹绘图
墨迹绘图
代理仓库A
墨迹绘图
代理仓库B
墨迹绘图
墨迹绘图
墨迹绘图
远程仓库X
远程仓库Y
墨迹绘图
墨迹绘图
墨迹绘图
缓存到服务器
墨迹绘图
墨迹绘图
仓库寻址路径：配置文件配置顺序-->仓库组
墨迹绘图


私库又分为代理仓库，发布仓库及快照仓库

代理仓库用来缓存远程仓库的依赖，提升下载速度。

快照仓库用于上传下载项目开发时用到的依赖。

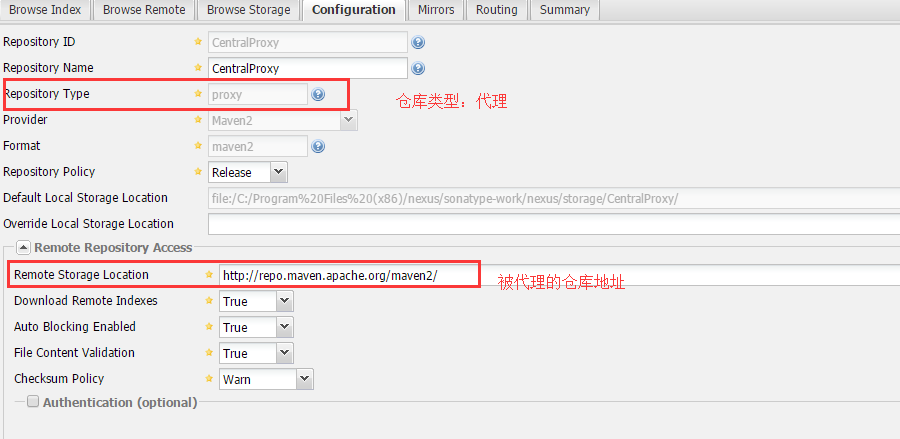
发布仓库用于上传下载项目发布到服务器上时用到的依赖。

私库
墨迹绘图
代理仓库
发布仓库
快照仓库
墨迹绘图
墨迹绘图
墨迹绘图
墨迹绘图
墨迹绘图
墨迹绘图


## 对于仓库的改动

### 代理仓库

设置代理仓库缓存远程仓库依赖



### 版本控制

1. 启用发布仓库，本地开发时使用快照仓库，当需要部署服务器时，使用发布版本，并禁止覆盖，避免依赖混乱

<!-- 发布私库配置 -->  
 <repository>  
 <id>releases</id>  
 <url>http://192.168.0.141:8081/nexus/content/repositories/releases</url>  
 </repository>

1. 发布私库版本配置：

开发服务器使用后缀dev的发布版本号，测试服务器使用后缀test的发布版本号，正式服务器使用后缀prod的发布版本号

<package.version>1.0.0.dev</package.version>

<package.version>1.0.0.test</package.version>

1. 快照版本：

<package.version>1.0.0-SNAPSHOT</package.version>

1. 版本号写入profiles，根据maven环境命令选择不同的依赖版本编译或打包

<!--打包环境配置-->  
 <profiles>  
 <profile>  
 <!--本地开发用配置-->  
 <id>dev</id>  
 <!--默认激活-->  
 <activation>  
 <activeByDefault>true</activeByDefault>  
 </activation>  
 <properties>  
 <package.environment>${project.basedir}/config/dev</package.environment>  
 <package.version>1.0.0-SNAPSHOT</package.version>  
 </properties>  
</profile>  
 </profiles>

### 仓库权限配置

设置仓库角色，权限，禁止非管理员修改仓库，上传发布版本依赖

# 二．Maven配置文件（聚合与继承）：

## 结构说明

Maven配置文件有面向用户的settings文件及面向项目的pom文件

Setting：可配置用户名，密码，仓库，环境参数等

其他setting（一般在用户目录下）
软件安装目录下全局setting
墨迹绘图
继承（覆写）
墨迹绘图
墨迹绘图




Pom：可配置依赖，构建，仓库，环境参数等

软件安装目录超级pom


主项目目录下针对项目的pom文件


子项目目录下针对子项目的pom文件
墨迹绘图
墨迹绘图
继承     覆写
墨迹绘图
墨迹绘图
墨迹绘图
墨迹绘图


依赖可以配置在主pom中，子项目会继承该依赖

## 依赖管理

依赖可以配置在主pom中，子项目会继承该依赖，也可以在主pom中配置dependencyManagement依赖统一管理，子项目中仍需声明该依赖，但可以省略版本号及有效范围，统一管理整个项目的依赖版本。例如：

### 父项目

<!--父项目依赖管理（子项目中仍需声明依赖，但可以省略版本号，使得整个项目的依赖版本统一）-->  
<dependencyManagement>  
 <dependencies>  
 <!-- deecent -->  
 <dependency>  
 <groupId>com.deecent</groupId>  
 <artifactId>jiabaiwen-dao</artifactId>  
 <version>${package.version}</version>  
 </dependency>

<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
 <scope>test</scope>  
 </dependency>

</dependencies>

</dependencyManagement>

### 子项目

<dependencies>

<dependency>  
 <groupId>com.deecent</groupId>  
 <artifactId>jiabaiwen-dao</artifactId>  
</dependency>

<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 </dependency>

</dependencies>

## 打包环境配置

<!--打包环境配置-->  
 <profiles>  
 <profile>  
 <!--本地开发用配置-->  
 <id>dev</id>  
 <!--默认激活-->  
 <activation>  
 <activeByDefault>true</activeByDefault>  
 </activation>  
 <properties>  
 <package.environment>${project.basedir}/config/dev</package.environment>  
 <package.version>1.0.0-SNAPSHOT</package.version>  
 </properties>  
</profile>  
 <profile>  
 <!--开发服务器用配置-->  
 <id>serverDev</id>  
 <properties>  
 <package.environment>${project.basedir}/config/dev</package.environment>  
 <package.version>1.0.0.dev</package.version>  
 </properties>  
</profile>  
<profile>  
 <!--测试服务器用配置-->  
 <id>serverTest</id>  
 <properties>  
 <package.environment>${project.basedir}/config/test</package.environment>  
 <package.version>1.0.0.test</package.version>  
 </properties>  
</profile>  
</profiles>

## 上传私库配置

分为发布用私库repository及快照私库snapshotRepository

<!-- 上传私库配置 -->  
 <distributionManagement>  
 <!-- 发布私库配置 -->  
 <repository>  
 <id>releases</id>  
 <url>http://192.168.0.141:8081/nexus/content/repositories/releases</url>  
 </repository>  
 <!-- 开发快照私库配置 -->  
 <snapshotRepository>  
 <id>snapshots</id>  
 <url>http://192.168.0.141:8081/nexus/content/repositories/snapshots</url>  
 </snapshotRepository>  
 </distributionManagement>

## 更新策略

更新策略配置在pom文件仓库设置中：

### 上传策略

快照仓库使用版本号+时间戳命名文件后上传至私库，不断更新至最新版

发布仓库使用版本号命名文件后上传至私库，同一版本始终只保留一个文件

### 下载策略

１更新策略always： 每次打包都到服务器上检查最新依赖并下载到本地，打包

２更新策略never： 每次打包都直接使用本地依赖打包主项目

３更新策略daily（默认）&interval：每天（间隔时间）内，只检查一次更新

<!-- 下载仓库组配置 -->  
 <repositories>  
 <repository>  
 <id>nexus</id>  
 <!--仓库地址-->  
 <url>http://192.168.0.141:8081/nexus/content/groups/public/</url>   
 <releases>  
 <!--是否下载发布版本-->  
 <enabled>true</enabled>  
 **<!--更新策略（十分钟以内最多检查一次更新）-->**  **<updatePolicy>interval:10</updatePolicy>** </releases>  
 <snapshots>  
 <!--是否下载快照版本-->  
 <enabled>true</enabled>  
 **<!--更新策略（十分钟以内最多检查一次更新）-->  
 <updatePolicy>interval:10</updatePolicy>** </snapshots>   
 </repository>  
 </repositories>

## 6. 对于pom文件的调整

### 1. 环境参数

pom文件中转移生产环境参数目录至指定路径，可放置在jenkins服务器硬盘中，项目中移除生产环境参数配置目录

### 2. 仓库配置

上传仓库地址从setting中转移到pom文件中

### 3. 依赖管理

使用依赖管理将依赖配置在主项目中，子项目需要的依赖在子pom文件中再次配置，可省去版本号，保持整个项目的依赖版本统一

### 4. 更新策略

更新策略调整为interval:10

默认更新状态为daily，即每天更新一次，这样的更新策略太慢，跟不上改动速度，如果每次都检查更新，又会容易造成刚编译完的依赖马上被其他人的最新版本替代，所以使用与上次更新超过十分钟才会再次检查更新的策略。当需要更新时可以更改deploy -U命令强制更新。

# 三．maven生命周期及基本命令：

## 生命周期解读

maven生命周期分为三套，clean，default，site，他们是相互独立的，执行一个阶段不会同时执行到另外两个阶段，执行某个命令一般会从该命令开始一直执行到这套生命周期结束。，以下是列举常用的几个命令：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | Pre-clean |  |
|  | clean |  | Clean  post-clean |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  | …  Compile  …  Test  …  package  …  Install  …  Deploy  … |
|  | default |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Site |  |  |
|  |  | Pre-site  Site  Post-site  Site-deploy |
|  |  |  |

## 参数命令

1. 当需要强制更新依赖时，可以使用-U命令
2. 当需要同时编译依赖的项目时，可以使用-am命令
3. 当需要同时编译被依赖的项目时，可以使用-amd命令
4. 当需要为编译设定环境profile时，可以使用-P命令，例如-Pdev
5. Maven在编译时会自动测试那些需放在src/main/test目录下的类及方法，类名及方法名需以Test开头或结尾。
6. 当需要跳过测试时，可以使-Dmaven.test.skip=true
7. Maven是集成一套自动测试部署的编译插件，建议今后的项目为每个方法写好测试方法，每次有改动编译时都会自动测试整个项目有没有其他地方被影响，会生成测试报告及测试覆盖率报告。
8. 注意命令参数是区分大小写的

# 四．约定优于配置

Maven提倡约定优于配置，很多默认目录结构都配置在了超级pom中，例如：

源码目录为：src/main/java，

测试目录src/test/java,

web资源目录：src/main/resources,

编译输出目录为：target/classes/，

打包方式为jar，

包输出目录为target

遵循约定虽然损失了一定的灵活性，用户不能随意安排目录结构，但是却能减少配置。更重要的是，遵循约定能够帮助用户遵守构建标准。如果一定要灵活配置，可以在pom文件中覆写配置。

# 五．附录

## Pom示例：

### 父项目pom：

<?xml version="1.0" encoding="UTF-8"?>  
<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">  
 <!-- pom文件版本号 -->  
 <modelVersion>4.0.0</modelVersion>  
 <!-- 以下三个属性去顶maven依赖唯一坐标-->  
 <groupId>com.deecent</groupId>  
 <artifactId>jiabaiwen</artifactId>  
 <version>${package.version}</version>  
 <!--打包方式-->  
 <packaging>pom</packaging>  
 <!--项目名称-->  
 <name>HRJIABAIWEN</name>  
 <url>http://maven.apache.org</url>  
 <!--定义全局变量-->  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <org.springframework.version>4.2.4.RELEASE</org.springframework.version>  
 <druid.version>1.0.18</druid.version>  
 </properties>  
 <!--子模块声明-->  
 <modules>  
 <module>jiabaiwen-model</module>  
 <module>jiabaiwen-core</module>  
 <module>jiabaiwen-plugin</module>  
 <module>jiabaiwen-dao</module>  
 <module>hr-jiabaiwen</module>  
 </modules>  
 <!--父项目依赖管理（子项目中仍需声明依赖，但可以省略版本号，使得整个项目的依赖版本统一）-->  
 <dependencyManagement>  
 <dependencies>  
 <!-- deecent -->  
 <dependency>  
 <groupId>com.deecent</groupId>  
 <artifactId>jiabaiwen-dao</artifactId>  
 <version>${package.version}</version>  
 </dependency>  
 <!-- spring -->  
 <dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-redis</artifactId>  
 <version>1.6.4.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>redis.clients</groupId>  
 <artifactId>jedis</artifactId>  
 <version>2.8.0</version>  
 </dependency>   
 </dependencies>  
  
</dependencyManagement>  
<!-- 上传私库配置 -->  
 <distributionManagement>  
 <!-- 发布私库配置 -->  
 <repository>  
 <id>releases</id>  
 <url>http://192.168.0.141:8081/nexus/content/repositories/releases</url>  
 </repository>  
 <!-- 开发快照私库配置 -->  
 <snapshotRepository>  
 <id>snapshots</id>  
 <url>http://192.168.0.141:8081/nexus/content/repositories/snapshots</url>  
 </snapshotRepository>  
 </distributionManagement>  
<!-- 下载仓库组配置 -->  
 <repositories>  
 <repository>  
 <id>nexus</id>  
 <!--仓库地址-->  
 <url>http://192.168.0.141:8081/nexus/content/groups/public/</url>   
 <releases>  
 <!--是否下载发布版本-->  
 <enabled>true</enabled>  
 <!--更新策略（十分钟以内最多检查一次更新）-->  
 <updatePolicy>interval:10</updatePolicy>  
 </releases>  
 <snapshots>  
 <!--是否下载快照版本-->  
 <enabled>true</enabled>  
 <!--更新策略（十分钟以内最多检查一次更新）-->  
 <updatePolicy>interval:10</updatePolicy>  
 </snapshots>   
 </repository>  
 </repositories>  
   
 <!--打包环境配置-->  
 <profiles>  
 <profile>  
 <!--本地开发用配置-->  
 <id>dev</id>  
 <!--默认激活-->  
 <activation>  
 <activeByDefault>true</activeByDefault>  
 </activation>  
 <properties>  
 <package.environment>${project.basedir}/config/dev</package.environment>  
 <package.version>1.0.0-SNAPSHOT</package.version>  
 </properties>  
 </profile>  
 <profile>  
 <!--开发服务器用配置-->  
 <id>serverDev</id>  
 <properties>  
 <package.environment>${project.basedir}/config/dev</package.environment>  
 <package.version>1.0.0.dev</package.version>  
 </properties>  
 </profile>  
 <profile>  
 <!--测试服务器用配置-->  
 <id>serverTest</id>  
 <properties>  
 <package.environment>${project.basedir}/config/test</package.environment>  
 <package.version>1.0.0.test</package.version>  
 </properties>  
 </profile>  
 </profiles>  
  
</project>

### 子项目pom：

<?xml version="1.0"?>  
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
 <modelVersion>4.0.0</modelVersion>  
 <!--声明父工程,继承属性-->  
 <parent>  
 <groupId>com.deecent</groupId>  
 <artifactId>jiabaiwen</artifactId>  
 <version>${package.version}</version>  
 </parent>  
 <groupId>com.deecent</groupId>  
 <artifactId>hr-jiabaiwen</artifactId>  
 <packaging>war</packaging>  
 <name>hr-jiabaiwen</name>  
 <url>http://maven.apache.org</url>  
 <!--依赖配置，由于已经在主项目里配置过，所以此处可省略版本号-->  
 <dependencies>  
 <!--common-->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.commons</groupId>  
 <artifactId>commons-lang3</artifactId>  
 </dependency>  
   
 <!--hr- jiabaiwen-->  
 <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.deecent</groupId>  
 <artifactId>jiabaiwen-dao</artifactId>  
 </dependency>  
   
 </dependencies>  
 <!--编译配置-->  
 <build>  
 <!-- 编译输出名字 -->  
 <finalName>hr-jiabaiwen</finalName>  
 <!-- 配置maven编译输出目录 -->  
<!-- <directory>C:\\.jenkins\\workspace\\jiabaiwen\\target</directory> -->  
 <plugins>  
 <!--maven测试用插件-->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.4.2</version>  
 <!--是否跳过测试-->  
 <configuration>  
 <skipTests>true</skipTests>  
 </configuration>  
 </plugin>  
<!-- war打包插件 -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-war-plugin</artifactId>  
 <version>2.1.1</version>  
 <configuration>  
 <archive>  
 <addMavenDescriptor>false</addMavenDescriptor>  
 </archive>  
<!-- 配置web资源目录 -->  
 <webResources>  
 <resource>  
 <directory>${package.environment}/</directory>  
 <targetPath>WEB-INF/classes</targetPath>  
 <filtering>true</filtering>  
 </resource>  
 </webResources>  
 </configuration>  
 </plugin>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.1</version>  
 <!-- maven编译插件不支持高版本jdk，此处配置使其支持高版本jdk -->  
 <configuration>  
 <source>1.7</source>  
 <target>1.7</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>