

咕泡学院VIP课程

# 工程化专题之Maven

咕泡学院 James老师

QQ 2904270631

课程安排

1. 认识Maven
2. 优势
   1. 约定优于配置
   2. 简单
   3. 测试支持
   4. 构建简单
   5. CI
   6. 插件丰富
3. 下载
   1. <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
      1. Windows path
      2. Linux .bash\_profile
      3. MAVEN\_OPTS
      4. 配置setting.xml

<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>

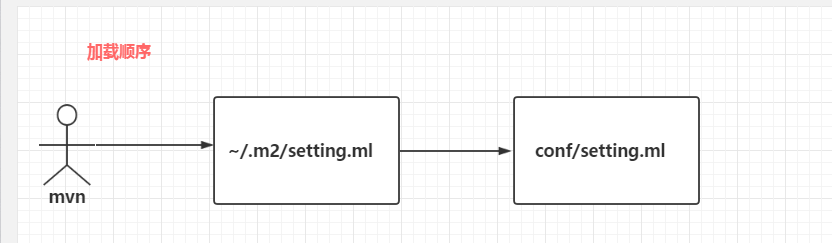
<mirror>

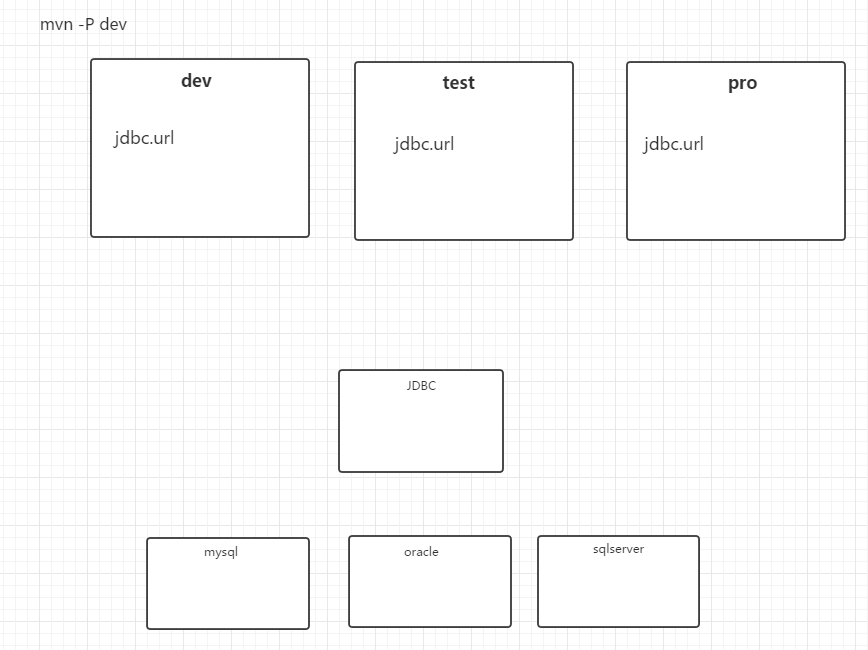
<id>osc\_thirdparty</id>

<mirrorOf>thirdparty</mirrorOf>

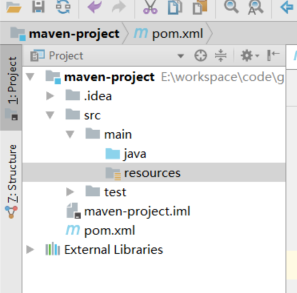
<url>http://maven.oschina.net/content/repositories/thirdparty/</url>

</mirror>





1. 新建一个Maven项目
   1. 项目结构



* 1. pom.xml
     1. groupId com.gupaoedu
     2. artfactId 功能命名
     3. version 版本号
     4. packaging 打包方式 默认是jar
     5. dependencyManagement
        1. 只能出现在父pom
        2. 统一版本号
        3. 声明 (子POM里用到再引)
     6. Dependency
        1. Type 默认jar
        2. Scope 在哪个阶段用 ，会不会被打到包里（优化pom，优化项目）
           1. compile 编译 例如spring-core
           2. test 测试，不会被打到包里
           3. provided编译 例如 servlet 我感觉也不会打到包里 因为tomcat里面有
           4. runtime运行时 例如JDBC驱动实现
           5. system 本地一些jar 例如短信jar （oss,阿里的）
           6. 依赖传递（mvn dependency:tree>d.txt）

第一列表示直接依赖的scope，第一行表示间接依赖的scope

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | compile | test | provided | runtime |
| compile | compile | - | - | runtime |
| test | test | - | - | test |
| provided | provided | - | provided | provided |
| runtime | runtime | - | - | runtime |

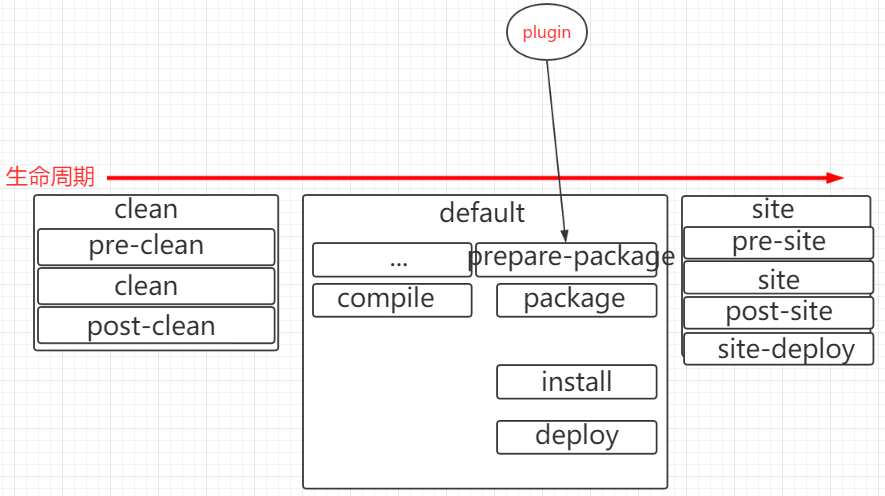
* + - * 1. 依赖仲裁

最短路径原则

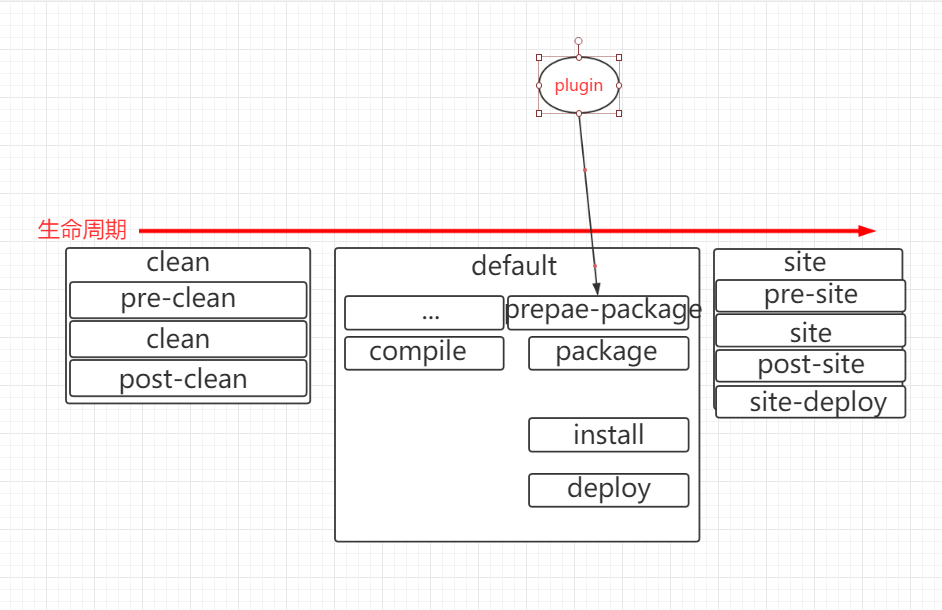
加载先后原则(书写顺序)

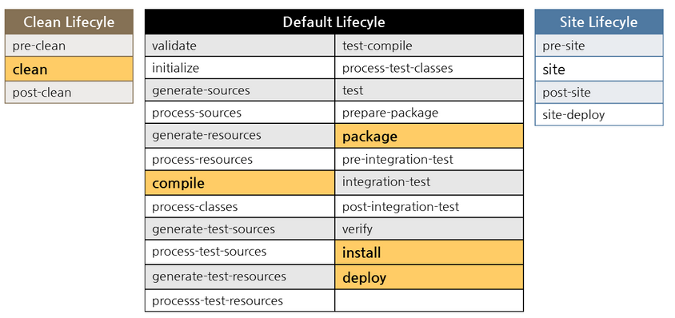
* + - * 1. exclusions

排除包



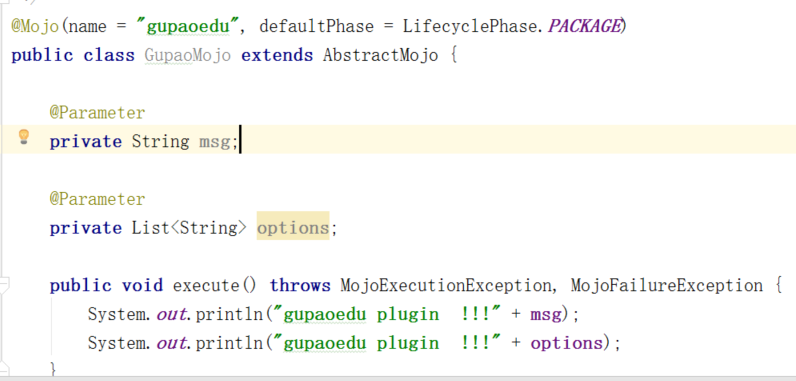
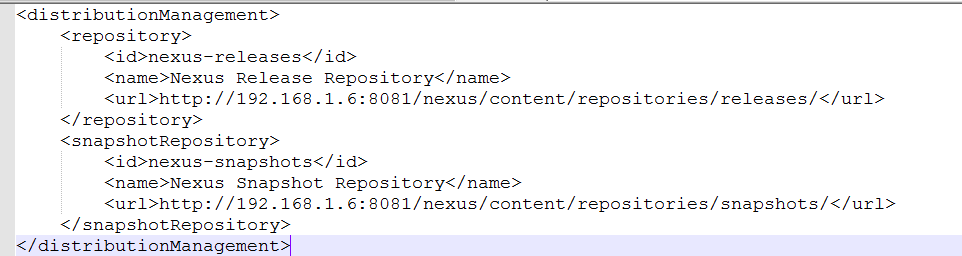
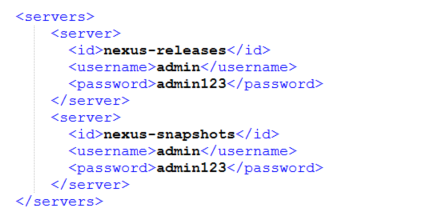
* + 1. 生命周期 lifecycle/phase/goal
       1. A Build Lifecycle is Made Up of Phases
       2. A Build Phase is Made Up of Plugin Goals





1. 版本管理
   1. 1.0-SNAPSHOT
      1. repository 删除
      2. mvn clean package -U (强制拉一次)
   2. 主版本号.次版本号.增量版本号-<里程碑版本>

1.0.0-RELAESE

1. 常用命令
   1. compile
   2. clean 删除target/
   3. test test case junit/testNG
   4. package 打包
   5. install 把项目install到local repo
   6. deploy 发本地jar发布到remote
2. 插件
   1. 常用插件
      1. <https://maven.apache.org/plugins/>
      2. <http://www.mojohaus.org/plugins.html>
      3. findbugs 静态代码检查mvn findbugs:findbugs,检查语法和语意方面
      4. versions 统一升级版本号 得配置插件
         1. mvn versions:set -DnewVersion=1.1
      5. source 打包源代码
      6. assembly 打包zip、war 生成可执行的包 jar –with dependency
      7. tomcat7
3. 自定义插件 <https://maven.apache.org/guides/plugin/guide-java-plugin-development.html>
   1. <packaging>maven-plugin</packaging>
   2. extends AbstractMojo
   3. 
   4. mvn install
   5. 参数传递 <configuration> 插件没有生命周期 插件要挂载（plugin要挂载到phase）
4. Profile
   1. 使用场景 dev/test/pro
   2. setting.xml 家和公司两套 profile切换私服 如果没有vpn的话
5. 仓库
   1. 下载
   2. 安装 解压
   3. 使用<http://books.sonatype.com/nexus-book/reference3/index.html>
      1. <http://192.168.1.6:8081/nexus>
      2. admin/admin123
   4. 发布
      1. pom.xml 配置
         1. 
         2. 
   5. 下载jar配置
      1. 配置mirror
      2. Profile
   6. Privilege 不同部门 权限
6. 索引下载 定时任务 brower index
7. archetype 模版化
   1. 生成一个archetype
      1. mvn archetype:create-from-project
      2. cd /target/generated-sources/archetype
      3. mvn install
   2. 从archetype创建项目 mvn archetype:generate -DarchetypeCatalog=local 如果本地骨架过多 会有选号 然后输入gropuID ….

创建一个新的项目 然后走12的步骤 生成骨架 然后 在先创建项目 走12（b）