项目构建工具Maven

# 系统需求

使用工具来管理工程的依赖，项目搭建及项目打包

# 项目负责

# 项目前思考

# 项目前攻破

# 项目中使用的技术总结

# 项目后期思考

# 项目中觉得亮点

## Maven优化

对于多项目的项目依赖，如果每个子项目都显示去依赖不同的jar，而可能导致版本管理起来，不能达到一致效果

1.优化依赖

使用dependencyManagement在最顶层的pom.xml来声明jar的版本，然后在子模块可以不使用version来达到继承父模块的pom.xml中的版本信息

2.优化插件

使用pluginManagement方式，类似于dependencyMangement

## 查看项目的jar依赖

mvn dependency:resolve查看项目依赖jar

mvn dependency:tree 查看项目的依赖jar，层级关系

mvn dependency:analyse 分析jar的直接依赖关系

## Maven的周期



1. 处理资源: 1).General(生成所有需要包含在打包过程中的资源文件) -> Process (复制并处理资源文件至目标目录，准备打包)
2. 编译->打包classes，这是主包的目标，每一步都依赖于上一步
3. 打包test资源->编译->执行test，这里执行测试包的构建，执行src/test/class执行对应的单元测试，使用-Dmaven.test.skip=true，可以跳过这一步
4. 打包jar，这里将处理的claess及resourcees文件打包成jar，并输出

## Dependency的scope

1. Provider：容器提供了，表示打包的时候，不用再将该dependency引入
2. test：只在测试阶段使用

## Maven 传递性依赖

有时候使用到了项目依赖于A，而A又依赖于B，我们引入A，也会依赖引入B，要排除对B的依赖可以这么处理

<dependency>

<exclutions>

<exclution>

B

</exclution>

</exclutions>

</dependency>

## Maven Profile的使用

1. Activation的使用

<profile>

<activation>

<activeByDefault>false…

<jdk>1.5(只有1.5的jdk版本，这个Profile才可激活)

</activation>

</profile>

注:如果在属性值前面加上!，表示不存在或者不是该属性

## 使用Idea在新建maven项目很慢

在新建项目中新增archetypeCatalog = internal

## maven仓库的顺序

maven本地仓库->mirror->profile->pom.xml的repository(按定义的顺序来)>中央仓库

如果配置mirror的话，那么只会从mirror下载，如果mirror下载失败，就不会从其它服务器下载。但是不配置mirror的话，会根据以上的顺序，一级一级查找。直接最后找不到。

这里碰到个问题：

1. deploy到snapshot的分支上，由于snapshot只能上传snapshotRepository，而且命名以SNAPSHOT结尾
2. deploy到release的上面时，需要使用命名为RELEASE结尾或者非SNAPSHOT结尾的版本
3. 必须在maven的配置文件中配置server节点的鉴权信息，id与pom.xml中的id一一对应。否则会鉴权失败，而导致上传失败
4. deploy也相当于本地install了，然后再upload到服务器上

mirror: 就是代理镜像，这样可以为代理的仓库建立镜像。可以将所有的库下载引导自身的代理服务器



以上是maven下载jar的流程信息

1. 从本地仓库查看是否存在
2. 从Profile中去查看是否存在Repository，和active..被激活，然后查看是否该repository是否被镜像了
3. 从pom.xml中的Repo...中查看配置的远程库，然后查看是否被镜像
4. 从中央仓库去查找，并且查看是否被镜像了