Java的包管理与Maven初步

什么是包

- JVM的工作被设计地相当简单：
  - 执行一个类的字节码
  - 假如这个过程中碰到了新的类，加载它
- 那么，去哪里加载这些类呢?

类路径（Classpath）

- 在哪里可以找到类
  - –classpath/–cp
- 类的全限定类名（目录层级）唯一确定了一个类
- 包就是把许多类放在一起打的压缩包

别急，还没完

- 传递性依赖
  - 你依赖的类还依赖了别的类
- Classpath hell
  - 全限定类名是类的唯一标识
  - 当多个同名类同时出现在Classpath中，就是噩梦的开始

什么是包管理

- 你要使用一些第三方类，总要告诉JVM从哪里找吧?
- 包管理的本质就是告诉JVM如何找到所需的第三方类库
- 以及成功地解决其中的冲突问题

没有Maven的蛮荒年代

黑暗岁月

- 手动写命令进行编译运行

启蒙时代

- Apache Ant
  - 手动下载jar包，放在一个目录中
  - 写XML配置，指定编译的源代码目录、依赖的jar包、输出目录等
- 缺点
  - 每个人都要自己造一套轮子
  - 依赖的第三方类库都需要手动下载，费时费力
    - 假如你的应用依赖了一万个第三方的类库呢?
  - 没有解决Classpath地狱的问题

Maven时代

Maven——划时代的包管理

- Convention over configuration
- 约定优于配置
- 必须强调，Maven远远不止是包管理工具
- Maven的中央仓库
  - 按照一定的约定存储包
- Maven的本地仓库
  - 默认位于~/.m2
  - 下载的第三方包放在这里进行缓存

Maven——划时代的包管理

- Maven的包
  - 按照约定为所有的包编号，方便检索
  - groupId/artifactId/version
    - 扩展：语义化版本
  - SNAPSHOT快照版本

Maven——划时代的包管理

- 传递性依赖的自动管理
  - 原则：绝对不允许最终的classpath出现同名不同版本的jar包
- 依赖冲突的解决：原则：最近的胜出
- 依赖的scope
- 当你看到如下的异常的时候
- AbstractMethodError
- NoClassDefFoundError
- ClassNotFoundException
- LinkageError

Maven——划时代的包管理

- 实战：解决Maven的包冲突

Maven——自动化构建工具

- Maven实战哪些章节是值得看的?
- Maven项目的基本结构（传世经典）
- 基本概念：坐标和依赖/生命周期/仓库/聚合和继承
- 使用Maven进行测试
- 如果需要开发插件的话：
  - Maven的插件

真实世界中的Maven

- 分析若干真实世界的Maven仓库