MAVEN

1. maven给我的感觉：
   1. 包管理器（类似composer）
      1. 通过dependency来引入外部和内部的包，当然直接导入的架包是可以的。
      2. 分布式管理
      3. 版本控制方便
      4. 团队协作
2. Maven的配置文件
   1. Setting.xml
      1. <localRepository>d:/m2/repository</localRepository>是很重要的，决定了你本地仓库的位置，即下载的包的位置。
      2. Setting.xml可以自己定义，$M2\_HOME/conf/setting.xml为全局setting。
   2. Pom.xml
      1. <properties>

<ctg.mq.version>1.0.18</ctg.mq.version>

<junit.version>4.12</junit.version>

<spring.version>3.0.5.RELEASE</spring.version>

<aspect.vesion>1.8.5</aspect.vesion>

<cglib.version>3.1</cglib.version>

<java.version>1.6</java.version>

<kryo.version>2.22</kryo.version>

</properties>

**设置全局变量通过${sprint.version}来引用**

* + 1. 第三方打撒

1. Localrepository
   1. 文件结构
      1. GroupId/ArtifactId/Version
         1. GroupId与包名没有必然联系。可以设置为包名也可以其他（项目组名）。如果使用.方式命名则保存为目录格式
2. 构建周期->构建阶段s(复数)
   1. 构建周期->一系列构建阶段 总的一个命令
      1. Default：不能直接用。要细化到构建阶段目标。
      2. Clean：清除构建的
      3. Site：生成文档的
   2. 执行某构建阶段会将该阶段之前的所有阶段
3. 本地仓库 远程仓库（例如：项目组服务器上的） 中央仓库（社区仓库）
4. Mvn shell
   1. Mvn [构建周期][构建阶段][:构建目标] 允许多个阶段
5. Maven通过GroupId,ArtifactId,Version来管理包定位包
6. Dependency的scope？？？？
   1. Compile 默认值 适用于所有阶段，会随项目一起发布
   2. Provided 希望使用者提供包，不会被打包到jar
   3. Runtime 运行时使用，编译时不会使用
   4. Test 仅测试编译与运行，不会随项目发布
   5. System

测试编译

打包

源码编译

测试运行

源码运行

1. 问题：Missing artifact jdk.tools:jdk.tools:jar:1.6
2. dependences与dependencymanagement
   1. 后者在parent中定义规定子pom使用哪个版本还有scope的依赖。这些依赖需要子pom显示声明
   2. 前者会在子pom中自动写入依赖
3. 兄弟pom不会相互依赖，要显示声明依赖的兄弟pom