# Maven基本命令

-v:查询Maven版本   
本命令用于检查maven是否安装成功。   
Maven安装完成之后，在命令行输入mvn -v，若出现maven信息，则说明安装成功。

compile：编译   
将[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "http://blog.csdn.net/hcmony/article/details/_blank)源文件编译成class文件

test:[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "http://blog.csdn.net/hcmony/article/details/_blank)项目   
执行test目录下的[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "http://blog.csdn.net/hcmony/article/details/_blank)用例

package:打包   
将项目打成jar包

clean:删除target文件夹

install:安装   
将当前项目放到Maven的本地仓库中。供其他项目使用

maven的生命周期  
  
maven把项目的构建划分为不同的生命周期(lifecycle)，在我看来，划分的已经是非常仔细了，大家可以参考这里。粗略一点的话，它这个过程(phase)包括：编译、测试、打包、集成测试、验证、部署。maven中所有的执行动作(goal)都需要指明自己在这个过程中的执行位置，然后maven执行的时候，就依照过程的发展依次调用这些goal进行各种处理。  
  
这个也是maven的一个基本调度机制。一般来说，位置稍后的过程都会依赖于之前的过程。当然，maven同样提供了配置文件，可以依照用户要求，跳过某些阶段。

# 什么是Maven仓库？

Maven仓库用来存放Maven管理的所有Jar包。分为：本地仓库 和 中央仓库。

本地仓库   
Maven本地的Jar包仓库。

中央仓库   
Maven官方提供的远程仓库。

# 什么是“坐标”？

在Maven中，坐标是Jar包的唯一标识，Maven通过坐标在仓库中找到项目所需的Jar包。   
如下代码中，groupId和artifactId构成了一个Jar包的坐标。

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.1.1</version></dependency>

groupId:所需Jar包的项目名

artifactId:所需Jar包的模块名

version:所需Jar包的版本号

# 传递依赖 与 排除依赖

传递依赖：如果我们的项目引用了一个Jar包，而该Jar包又引用了其他Jar包，那么在默认情况下项目编译时，Maven会把直接引用和间接引用的Jar包都下载到本地。

排除依赖：如果我们只想下载直接引用的Jar包，那么需要在pom.xml中做如下配置：(将需要排除的Jar包的坐标写在中)

<exclusions>

<exclusion>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

</exclusion></exclusions>

一、假设目前有三个maven项目，分别是project.A、project.B、project.C

要求B依赖A、C依赖B但不依赖A

2、C添加对B的依赖，同时排除对A的依赖

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <dependency>              <groupId>com.project</groupId>              <artifactId>project.B</artifactId>              <version>0.0.1-SNAPSHOT</version>              <exclusions>                <exclusion>                      <groupId>com.project</groupId>                      <artifactId>project.A</artifactId>                </exclusion>              </exclusions>          </dependency> |

# 依赖冲突

若项目中多个Jar同时引用了相同的Jar时，会产生依赖冲突，但Maven采用了两种避免冲突的策略，因此在Maven中是不存在依赖冲突的。

短路优先

本项目——>A.jar——>B.jar——>X.jar

本项目——>C.jar——>X.jar

若本项目引用了A.jar，A.jar又引用了B.jar，B.jar又引用了X.jar，并且C.jar也引用了X.jar。   
在此时，Maven只会引用引用路径最短的Jar。

声明优先   
若引用路径长度相同时，在pom.xml中谁先被声明，就使用谁。

# 聚合

什么是聚合？   
将多个项目同时运行就称为聚合。

如何实现聚合？   
只需在pom中作如下配置即可实现聚合：

<modules>

<module>../模块1</module>

<module>../模块2</module>

<module>../模块3</module>

</modules>

# 继承

什么是继承？   
在聚合多个项目时，如果这些被聚合的项目中需要引入相同的Jar，那么可以将这些Jar写入父pom中，各个子项目继承该pom即可。

如何实现继承？

父pom配置：将需要继承的Jar包的坐标放入标签即可。

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.apache.shiro</groupId>

<artifactId>shiro-spring</artifactId>

<version>1.2.2</version>

</dependency>

</dependencies>

</dependencyManagement>

子pom配置：

<parent>

<groupId>父pom所在项目的groupId</groupId>

<artifactId>父pom所在项目的artifactId</artifactId>

<version>父pom所在项目的版本号</version></parent>