# 概念解释

一、lifecycle：

生命周期，maven最高级别控制单元，由phase组成。

二、phase：

可以理解为任务单元，理解为：lifecycle是总任务，phase就是总任务分出来的 一个个 子任务。

三、goal:

这是执行任务的最小单元，它可以绑定到任意个phase中，一个phase有一个或多个goal，goal也是按顺序执行的，一个phase被执行时，绑定到phase里的goal会按绑定的顺序执行。

四、mojo:

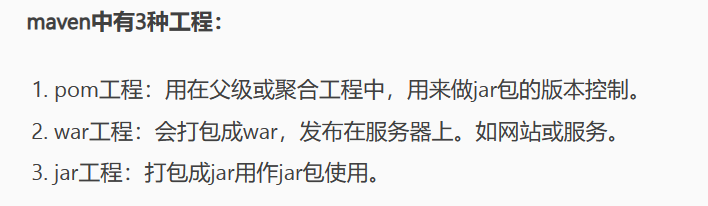
lifecycle与phase与goal都是概念上的东西，mojo才是做具体事情的**，可以简单理解mojo为goal的实现**类，它继承于AbstractMojo，有一个execute方法，goal等的定义都是通过在mojo里定义一些注释的anotation来实现的，maven会在打包时，自动根据这些anotation生成一些xml文件，放在plugin的jar包里。

**注意：**

**1、一个lifecycle可以包含任意个phase，phase的执行是按顺序的，一个phase可以绑定很多个goal，至少为一个。**

**2、没有goal的phase是没有意义的。**

# pom文件结构



<packaging>pom</packaging>

<packaging>war</packaging>

<packaging>jar</packaging>

## 添加依赖

在dependencies标签下添加

<dependencies>

**<!--添加依赖-->**

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.1.3</version>

</dependency>

.......

</dependencies>

## 二、 排除依赖

   <dependency>

   <groupId>org.apache.struts</groupId>

   <artifactId>struts2-spring-plugin</artifactId>

   <version>2.3.24</version>

**<!--排除依赖-->**

**<exclusions>**

**<exclusion>**

**<groupId>org.springframework</groupId>**

**<artifactId>spring-beans</artifactId>**

**</exclusion>**

**</exclusions>**

   </dependency>

## 三、锁定版本号

**（1）添加自定义版本号**

<properties>

<!-- 自定义版本号，spring.version为自定义 -->

<spring.version>4.2.4.RELEASE</spring.version>

</properties>

**（2）添加 版本锁定依赖 <!-- 锁定版本，spring4.2.4 -->**

**<dependencyManagement>**

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

**<version>${spring.version}</version>**

</dependency>

</dependencies>

......

</dependencyManagement>

**（3）添加依赖-无需指定版本号**

**<dependencies>**

**<dependency>**

**<groupId>org.springframework</groupId>**

**<artifactId>spring-context</artifactId>**

**<!-- 不添加版本号的原因是，此依赖版本号已锁定 -->**

**</dependency>**

**<dependencies>**

## 四、jar 版本同一管理

<!-- 对于jar包冲突，在此进行同一管理 -->

<dependencyManagement>

<dependencies>

<dependency>

<groupId>io.netty</groupId>

<artifactId>netty-all</artifactId>

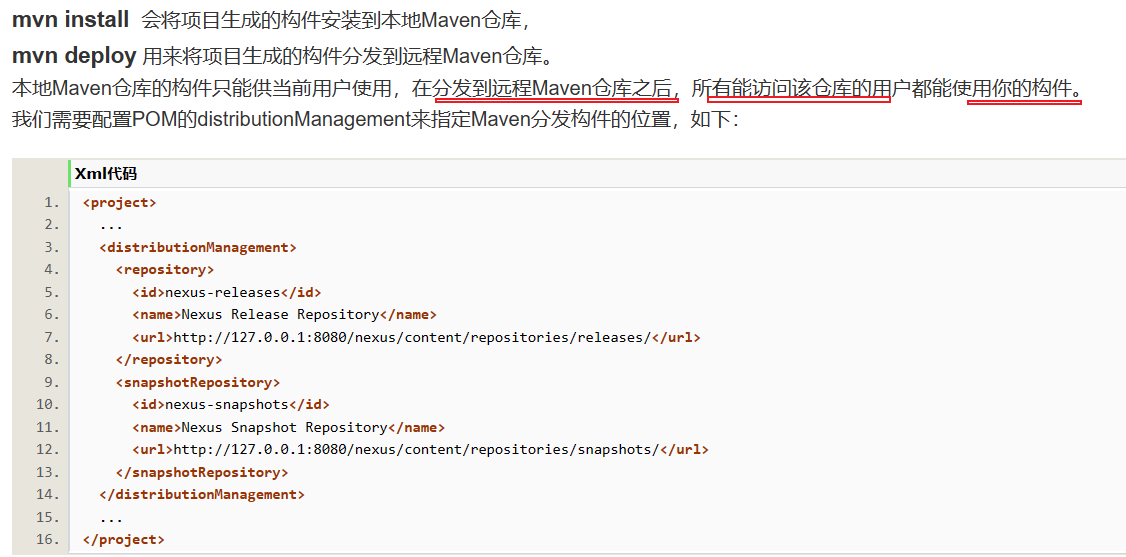
<version>4.0.23.Final</version>

</dependency>

</dependencies>

</dependencyManagement>

## 五、同步构建至远程服务器



[maven中的distributionManagement的作用\_distributionmanagement作用\_](https://blog.csdn.net/qq_31924435/article/details/53745811)

# pom继承

## 1、父工程

**（1）父工程的标志：pom.xml文件中，packaging 为pom。**

例如：<packaging>pom</packaging>

**（2）父工程包含子工程的标签<modules>。**

例如：<modules>

<module>logsplit</module>

<module>logformat</module>

<module>common</module>

</modules>

## 2、子工程

**子工程标志：pom.xml文件中，会出现 <parent>标签，并看到 父工程 坐标信息。**

例如： <parent>

<artifactId>parent</artifactId>

<groupId>com.djt.bigdata</groupId>

<version>1.0-SNAPSHOT</version>

</parent>

## 3、作用

**类似于Java中继承，子类 可以继承 父类的的方法，属性（必须是公共的），maven也是一样，**

**子工程 直接继承 父工程中 的pom.xml 的所有依赖。**

**注意：**

**1、去除不必要的重复依赖，方便管理。**

**2、父工程是一个抽象工程，只能定义 pom依赖。**

**父工程注意事项：**

