<https://www.bilibili.com/video/av36557763/?p=18>

maven的作用

1.依赖管理

maven工程对jar包的管理，jar包冲突

节省了项目空间

2.编译

cmd 进入maven项目目录

mvn tomcat:run 运行项目

3.单元测试

4.一键构建

maven 工具目录介绍

-bin 命令

-boot 类加载器

-conf 主要配置

-lib 自身依赖jar包

加载jar包流程

本地仓库，远程仓库，中央仓库

maven 项目目录结构

**原始项目结构**

项目名

src

**/\***

**maven项目结构**

项目名

src/main/java 核心diamante部分

src/main/recources 配置文件部分

src/test/java 测试代码部分

src/test/recources 测试配置文件

src/main/webapp 页面资源，js,css,图片等

maven 命令

mvn clean 清除target（编译后的文件）

mvn compile 编译项目

mvn test-compile 编译主程序

mvn test 编译执行测试代码（会生成正式代码）

mvn package 编译打包 （生成war）

mvn install 把项目安装到本地仓库

mvn deploy 发布项目（需要进行配置）

mvn site 生成站点

maven 构建

清理：将以前编译得到的旧的class字节码文件删除，为下一次编译做准备

编译：将Java源程序编译成class字节码文件

测试：自动测试，自动调用junit程序

报告：测试程序执行的结果

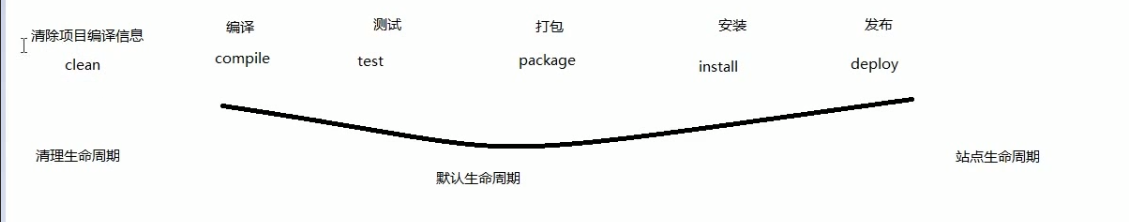
打包：动态Web工程打war包，Java工程打jar包

安装：Maven特定的概念——将打包得到的文件复制到“仓库”中的指定位置

部署：将动态Web工程恒诚的war包复制到Servlet容器的置顶目录下，使其可以运行

maven 生命周期

各个构建环节顺序执行



1.Clean 生命周期

pre-clean 置顶一些需要在clean之前完成的工作

clean 移除所有上一次构建生成的文件

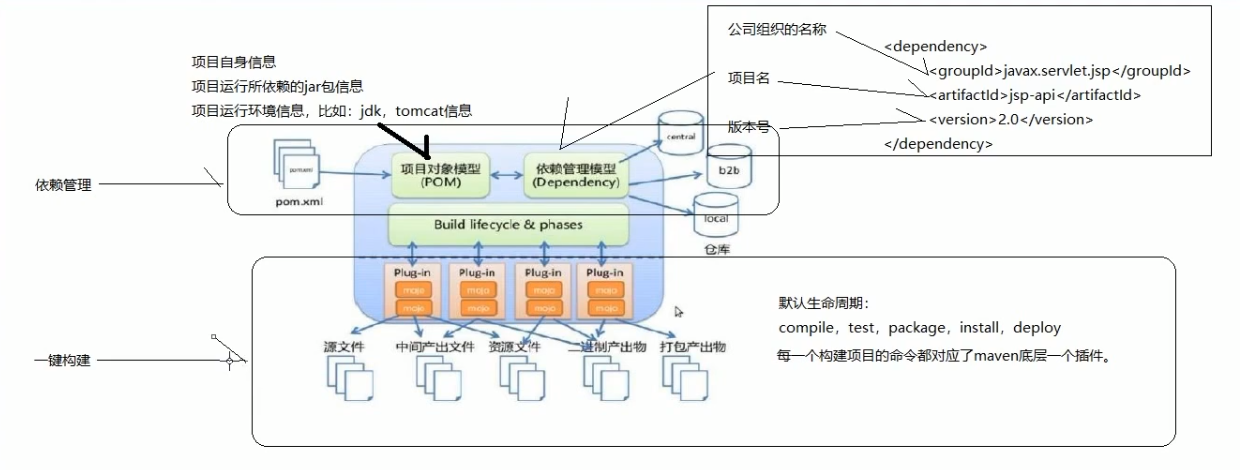
post-clean 执行一些需要在clean之后立刻完成的工作

2.Site 生命周期

mvn site 生成站点

3.Default生命周期

Maven 概念模型



# Maven 配置

apache-maven-3.1.1\conf\settings.xml

配置本地仓库

<settings …>

<localRepository>

E:\Development\Tool\apache-maven-3.1.1\pack\repository

</localRepository>

</settings>

配置远程中央仓库

<mirrors>

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

<mirrorOf>central</mirrorOf>

</mirror>

</mirrors>

**IDEA创建Maven 项目**

Create project

Maven

（推荐）

不勾选create from archetype 使用骨架创建

默认创建

（没啥用）

勾选create from archetype 使用骨架创建

org.apache.maven.archetypes:maven-archetype-quickstart

（web项目）

勾选create from archetype 使用骨架创建

org.apache.maven.archetypes:maven-archetype-webapp

# Maven 依赖

1.一个项目A依赖另一个自定义项目B，

使用B项目，需要将B项目mvn install生成jar包添加至本地仓库编译使用

Maven解析依赖信息时会到本地仓库中查找被依赖的jar包

对于我们自己开发的Maven工程，使用mvn install命令安装后就可以进入仓库

2.依赖范围

<dependency>

<groupId></> <artifactId></> <version></>

<scope>compile</ >

<>

compile：

对主程序有效，对测试程序有效，参与打包

test：

对主程序无效，对测试程序有效，不参与打包

provided：

开发阶段提供，部署时运行时忽略，

例如servlet-api.jar等，开发阶段需要用于运行，而部署运行后服务器可以提供。

3.依赖传递性

每个引用的jar包依赖的jar包都会被统一被引用

不必在每个工程中重复声明再引用依赖

4.依赖排除

被传递性传递过来的jar包给排除

<exclusions>

<exclusion>

<groupId></groupId>

<artifactId></ artifactId >

</exclusion>

</exclusions>

5.依赖原则

