# Maven

## Maven简介

Maven是一种基于项目对象模型(POM)的管理工具,可以通过一小段的描述信息来管理项目的构建,以及添加项目所依赖的jar包等,并且Maven还可以给我们项目中所需要的报告文档等.

Maven主要功能:

1. 项目构建;
2. 依赖管理.

## 构建的概念

### 1.构建的概念:

从SVN/GIT服务器上下载项目,进行项目的编写,测试,部署---Tomcat服务器上运行.整个这一系列的流程,就被称为一个项目的构建.

项目的构建过程:

清理-🡪编译-🡪测试-🡪报告-🡪打包-🡪部署.

### 2.构建的方式

①.手动管理项目:

一个类一个类的进行编写,编译,运行;

②.IDE工具(Eclipse,MyEclise)--🡪Ant.

Ant是Eclipse中默认的打包工具.

③.Maven:

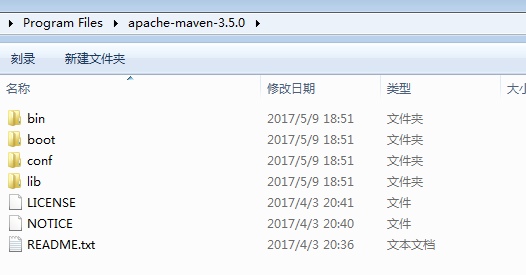
Maven可以对项目的目录结构进行约束,可以知道项目的源码应该放在哪个地方,也知道class字节码应该放在哪个地方.

④. IntelliJ IDEA(Android Studio):

都是基于Gradle进行项目的构建和依赖管理.

## Maven环境的搭建

### 1.Maven工具目录介绍



bin:maven的命令目录,存放maven的可执行文件;

boot:里面存放一个类加载器,一般用不着.

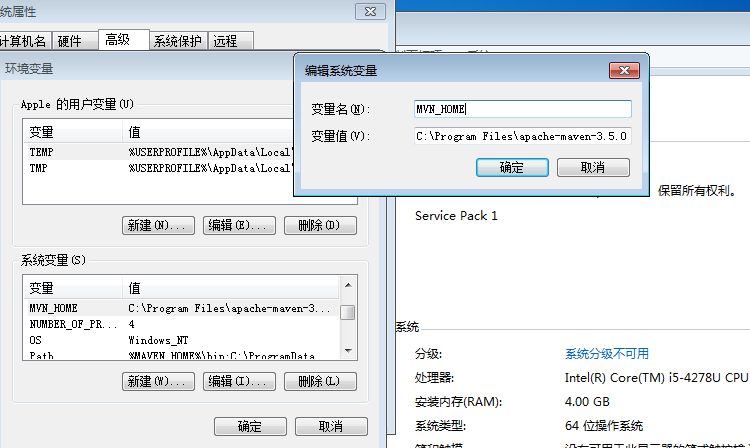
<conf:里面存放maven的配置文件.里面有个很重要的settings.xml>文件.

lib:存放maven自己依赖的jar包.

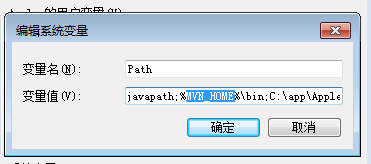
### 2.配置Maven环境变量

MVN\_HOME=C:\Program Files\apache-maven-3.5.0

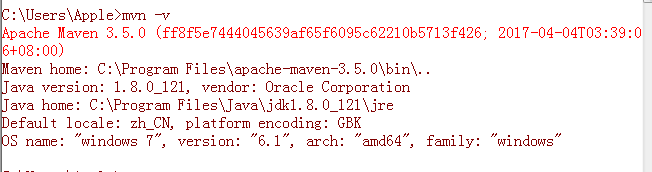
如图:



在path中引用刚才定义好的MVN\_HOME路径:

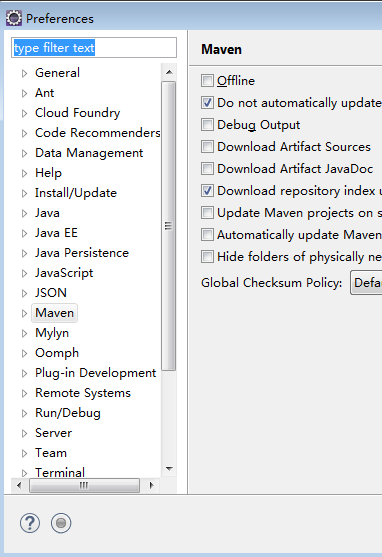


进行测试,在cmd中输入mvn –v命令.



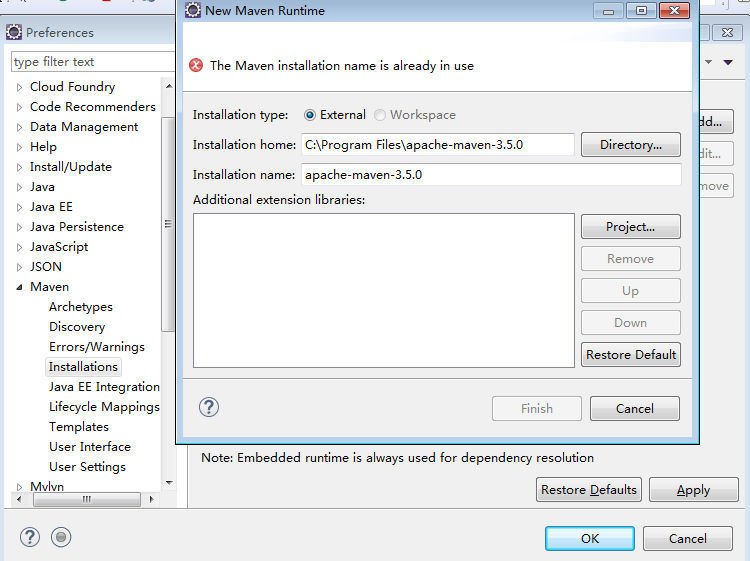
### 将Maven与Eclipse整合

#### .在Eclipse中安装maven插件

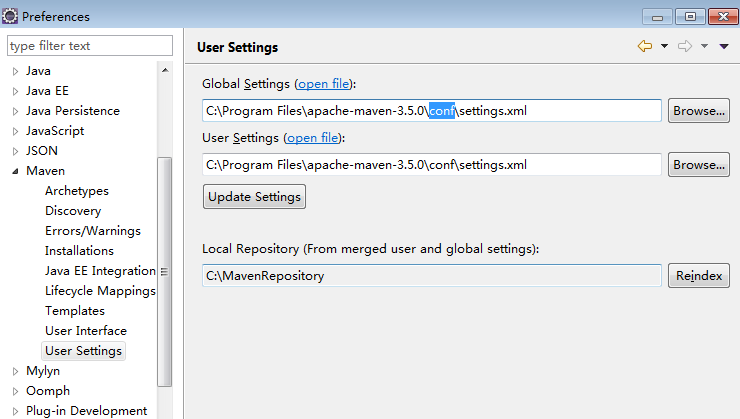


#### .在Maven---installations--🡪add:

在eclipse中关联maven软件

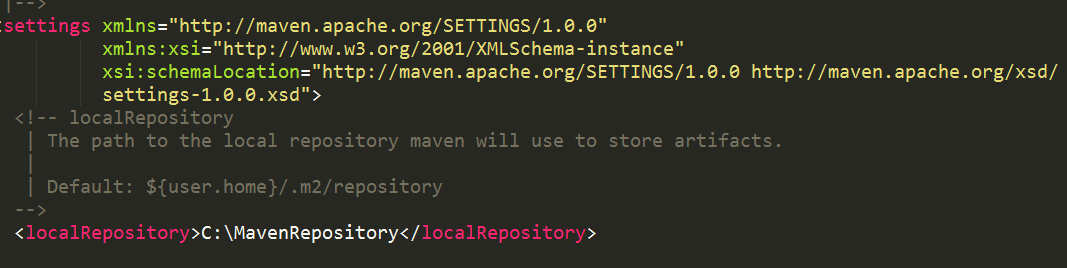


#### .在eclipse中关联maven的settings.xml配置文件:

sili

#### .修改maven-conf目录下的settings.xml文件.

修改本地仓库的路径.



## Maven的仓库

### 本地仓库:

用来存放从远程仓库或者从私服中下载的jar包.

### 远程仓库:

包含中央仓库与其他公司或者组织提供的一种供别人来下载jar的地址.

### 3. 中央仓库:

中央仓库也是远程仓库的一种.

maven官方默认的自己提供一个存放jar的文件夹.

### 4. 私服:Nexus私服

私服是我们自己搭建的一个存放jar的地址.

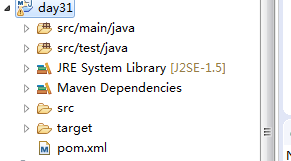
### Maven下载jar包的流程(重点):

某个项目要依赖jar包--🡪本地仓库找-🡪私服中去找-🡪远程和中央仓库中找---->如果都没有,就会报异常-🡪最终要么自己开发jar包要么就不用这个jar包了.

## 在Eclipse中利用Maven进行项目构建

### 构建JavaSE项目

#### .利用Maven提供的骨架(模板)进行创建(联网创建项目):



src/main/java:用来存放java源文件;

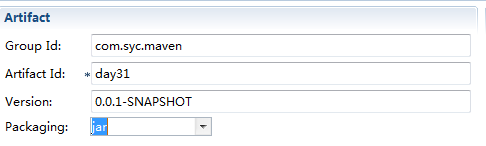
src/test/java:用来存放测试类的源文件.

src目录是上面src目录的映射镜像.

target:存放编译好的字节码文件,还有其他依赖的资源文件.

pom.xml:核心的配置文件,项目对象模型配置文件.

##### 1.overview视图



group Id:可以理解成是包名,但其实不是.代表项目的唯一标识.

Artifact Id:构建的id. 可以理解成项目名.代表项目的唯一标识.

Version:项目版本号,默认是0.0.1-SNAPSHOT.

group Id与ArtifactId,还有版本号,共同组成这个包的”坐标”.

通过该坐标就可以在全球范围内唯一的确定这个包的位置.

com.syc.maven.day31.0.0.1.SNAPSHOT

SNAPSHOT:快照版本,可以理解成是测试版,功能可能很新,但是很可能不稳定.

RELEASE:正式发布的版本,表示该jar或者该项目是稳定的.

package的类型:

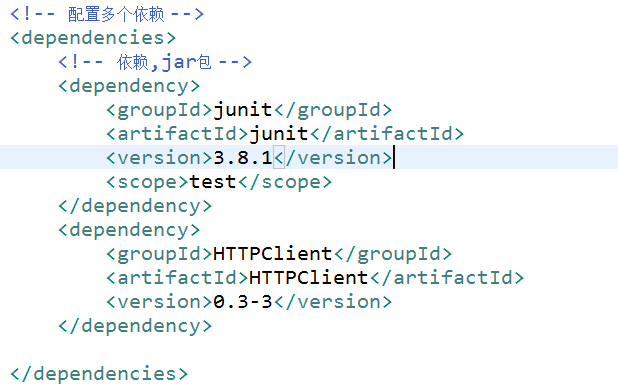
jar:表示是一个JavaSE项目;

war:表示是一个JavaEE项目;

pom:表示该项目是可以被其他项目继承的父项目;

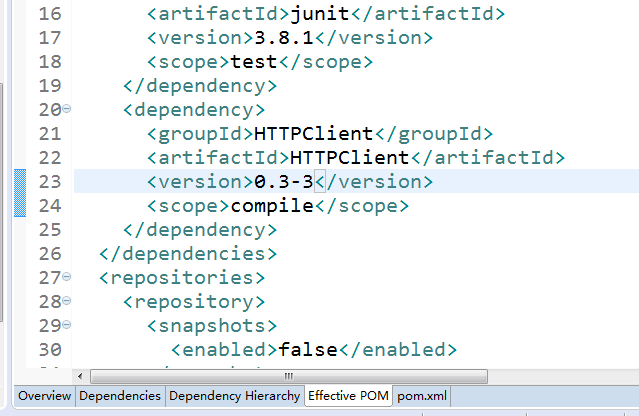
maven-plugin:表示该项目是一个插件形式的项目.

##### 2.pom视图添加jar包依赖



##### 3.Effective pom视图:

展示当前项目的配置信息,但是不能在这个界面中进行修改.



在Effective pom中可以看到配置的默认的中央仓库.



插件仓库:

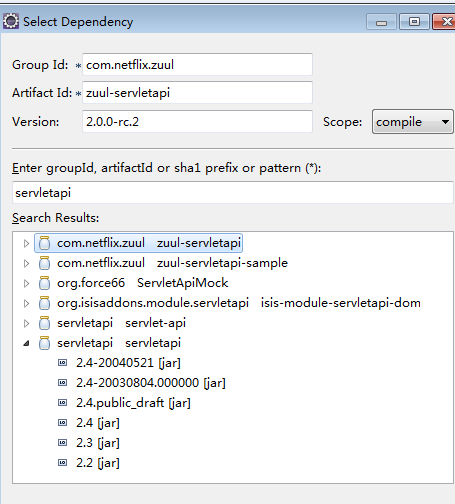


##### 4.Dependance Hierachy,显示当前项目依赖的jar包.

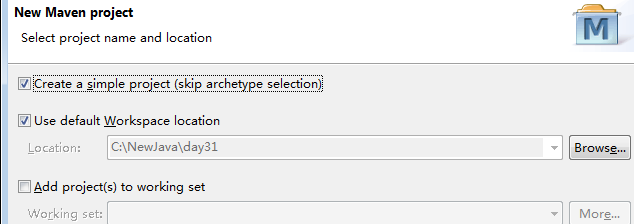


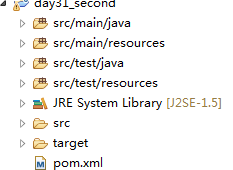
##### 5.Dependencies视图

搜索需要用的jar包



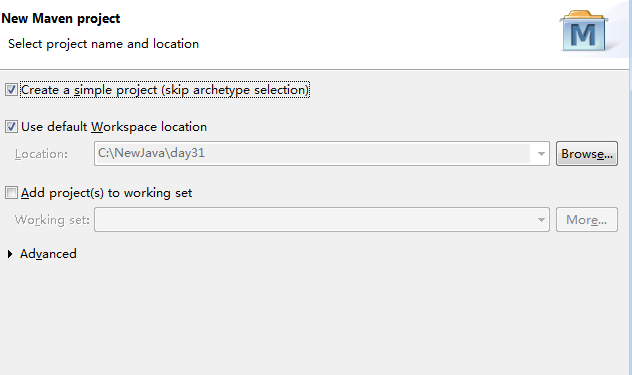
#### .不用骨架进行项目的创建(没有联网的):





### 构建JavaEE项目

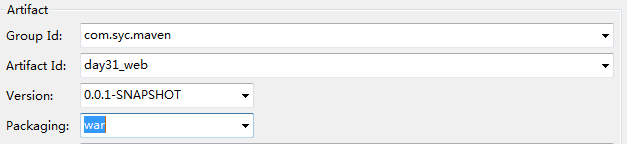
第一步:



注意:

一定要勾选第一个checkbox!

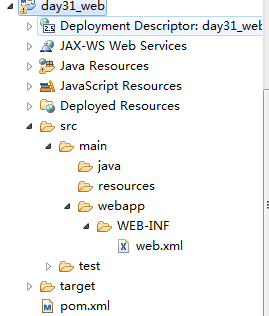
第二步:



注意:

packaging里的类型必须用war!

第三步:

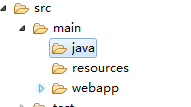


注意:

一开始web项目是有错误的,错误是因为缺少web.xml文件.解决方式是:

右键项目,选择 JavaEE Tools菜单中的第二个菜单.

目录解释:



java:存放java源代码;

resources:存放web项目的配置文件,比如c3p0-config.xml,或者各种框架中的配置文件.

webapp:类似于WebContent目录,里面存放了web.xml文件.

## Maven常用命令

mvn clean:清除项目中的一些原有的冗余信息.

mvn compile:执行编译命令.

mvn test:执行测试命名;

mvn package:将项目进行打包,变成jar或者war.

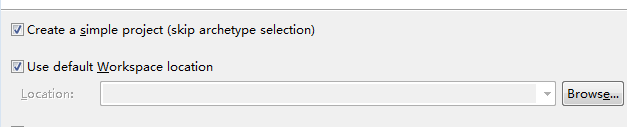
mvn install:将打包好的jar包或者war包存放到我们自己的本地仓库中,方便以后别人引用.

## Maven项目的继承

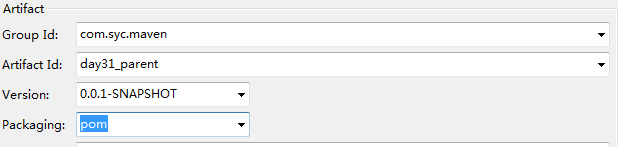
项目的继承:

把一些项目中通用的,重复的代码给提取出来,给每一个子项目进行复用.或者可以把一些公用的jar包,jar包版本号的设置,都在父工程中设置好,然后子项目直接继承.

### 1.第一步:创建父工程

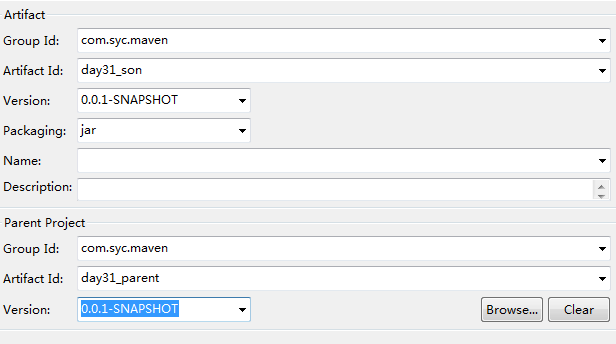


### 2.第二步:父工程包类型必须是pom类型.



父项目的packaging类型必须是pom!

### 3.第三步:创建子项目



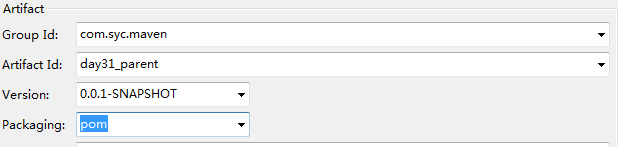
注意:子项目中必须设置Parent Project!

## 项目的聚合

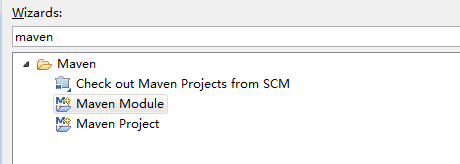
以三层架构为例,将整个项目拆分成4个部分.

分别是:maven\_parent,maven\_dao,maven\_service,maven\_web.

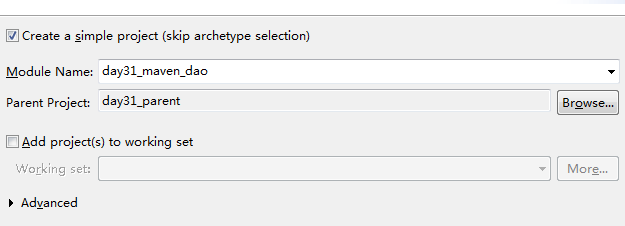
### 第一步:首先建立一个package为pom类型的父项目!

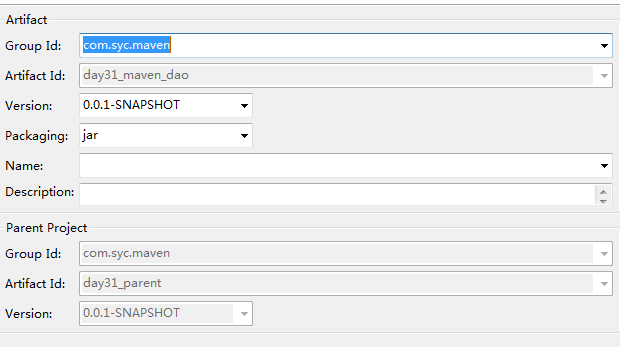


### 第二步:选择父项目,建立maven module类型的项目模块.

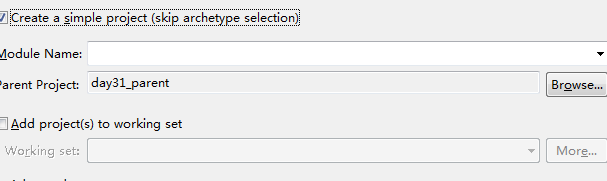


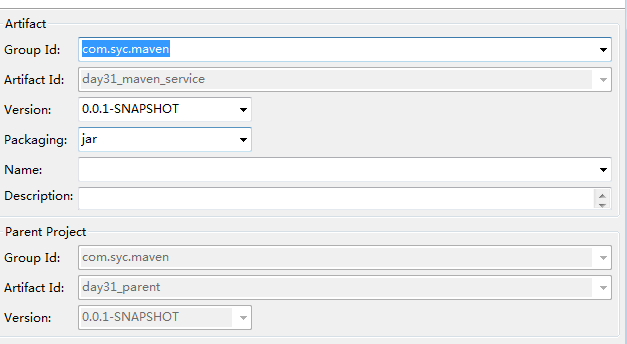
### 第三步:创建dao子模块.



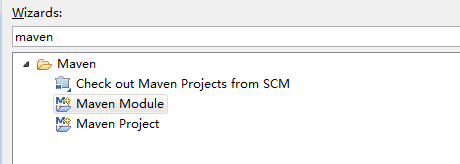


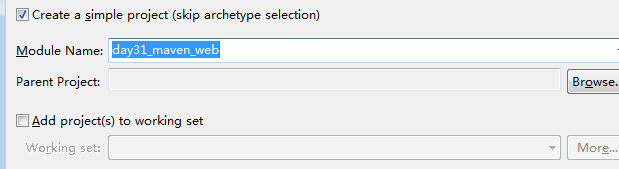
### 第四步,创建service模块,过程与dao模块一样.

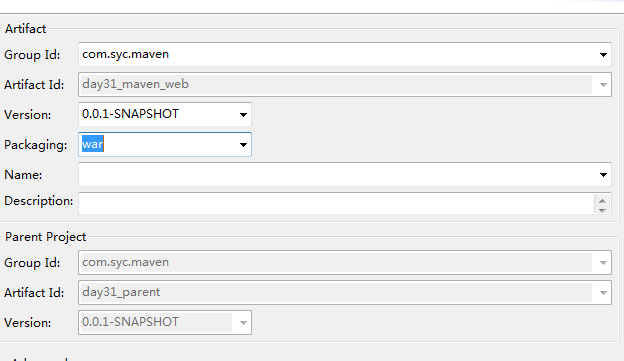




### 第五步:创建web层模块.



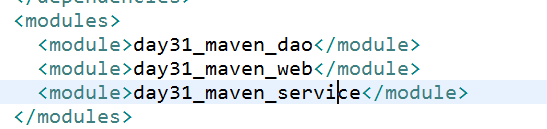




注意:package为war!

这4部分创建好之后,最终父项目中的pom.xml文件中,配置如下:

在父项目中的pom.xml中,

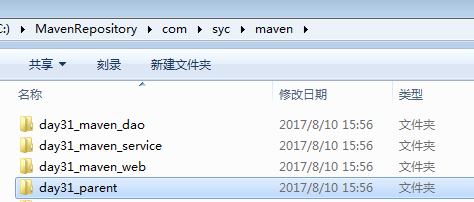


用来整合其他模块.

### 第六步: 把父模块安装到本地仓库.

右键父项目--🡪run as--🡪Maven install;

本地仓库中,出现如下项目目录结构.

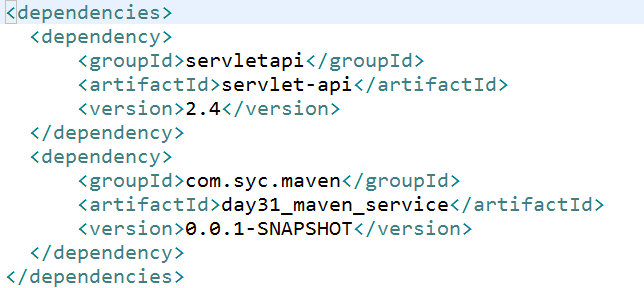


### 第七步:建立模块之间的依赖关系

在service的pom.xml文件中设置,service层模块依赖dao模块:



在web层pom.xml文件中设置,web层模块依赖service模块:



最终,如下图示,这4部分之间形成一种项目的聚合关系,形成一个完整的项目,一般适用于大型项目开发.

