

Maven

一.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.

二.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,MyEclipse)--→Ant.

Ant 是 Eclipse 中默认的打包工具.

③.Maven:

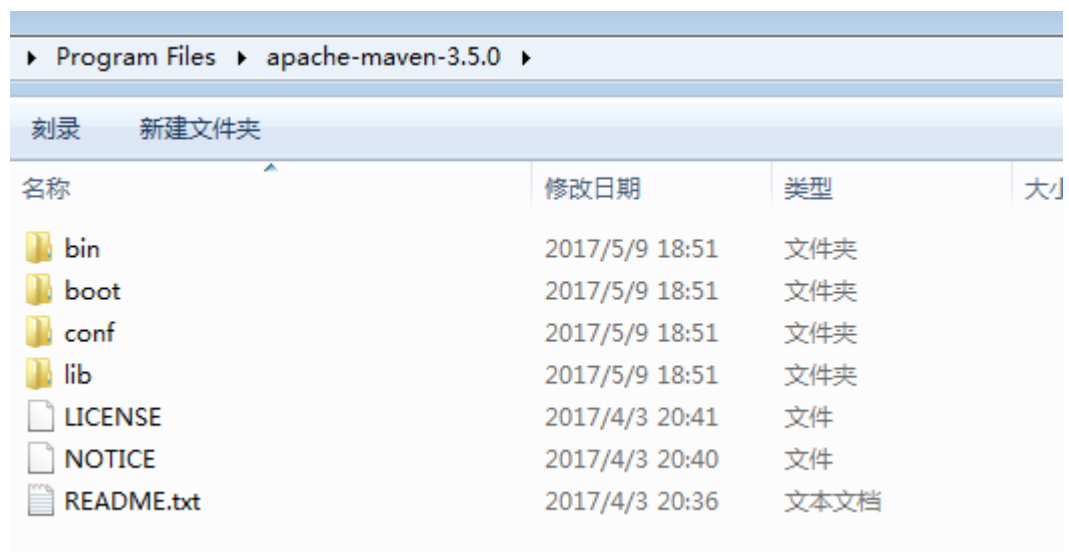
Maven 可以对项目的目录结构进行约束,可以知道项目的源码应该放在哪个地方,也知道 class 字节码应该放在哪个地方.

④. IntelliJ IDEA(Android Studio):

都是基于 Gradle 进行项目的构建和依赖管理.

三.Maven 环境的搭建

1.Maven 工具目录介绍



The screenshot shows the directory structure of Apache Maven 3.5.0. The path is Program Files > apache-maven-3.5.0. The directory contains several subdirectories and files:

名称	修改日期	类型	大小
bin	2017/5/9 18:51	文件夹	
boot	2017/5/9 18:51	文件夹	
conf	2017/5/9 18:51	文件夹	
lib	2017/5/9 18:51	文件夹	
LICENSE	2017/4/3 20:41	文件	
NOTICE	2017/4/3 20:40	文件	
README.txt	2017/4/3 20:36	文本文档	

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.

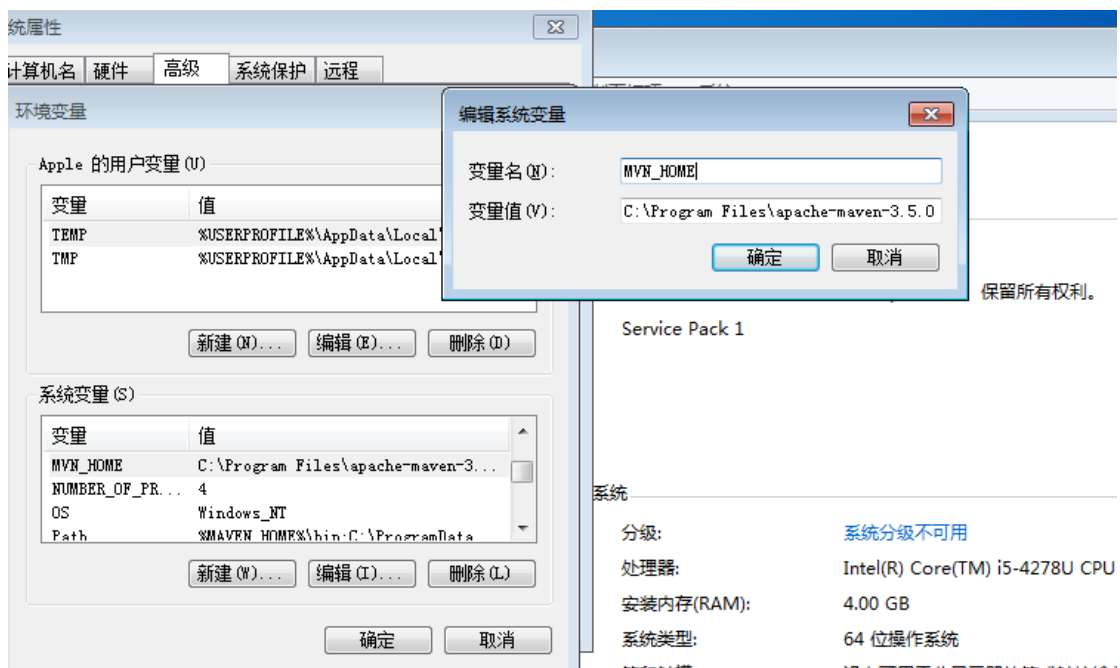
conf:里面存放 maven 的配置文件.里面有个很重要的 settings.xml 文件.

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:

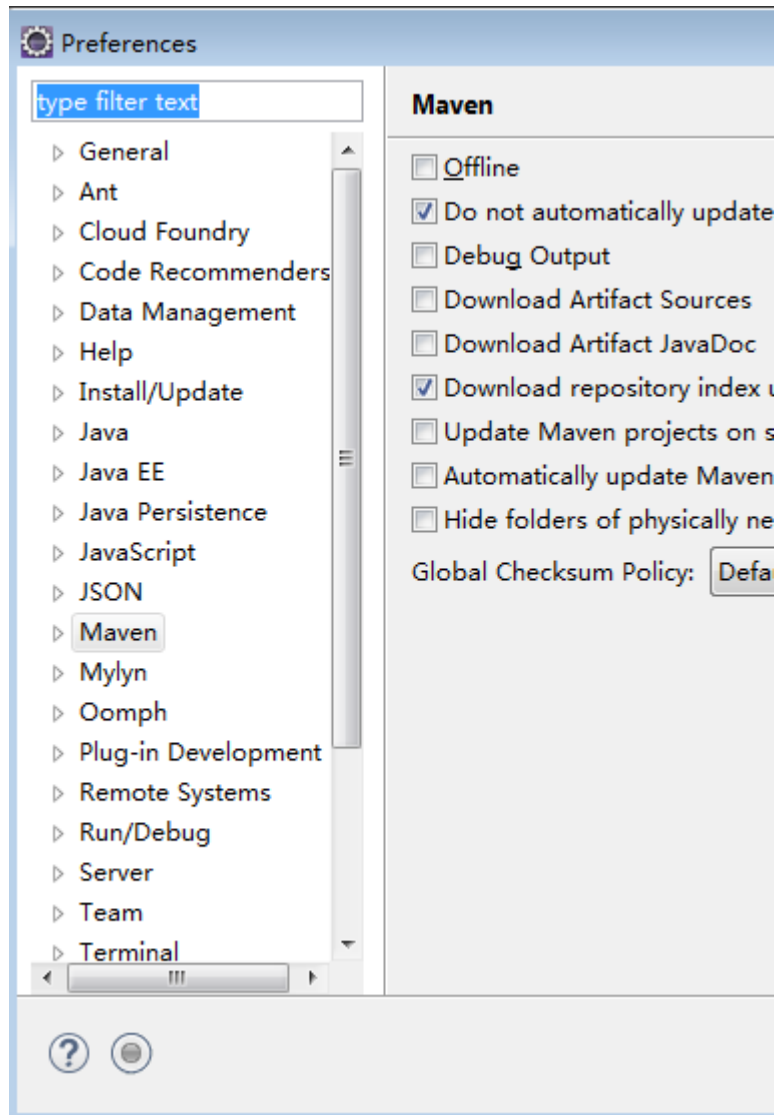


进行测试,在 cmd 中输入 `mvn -v` 命令.

```
C:\Users\Apple>mvn -v
Apache Maven 3.5.0 (ff8f5e7444045639af65f6095c62210b5713f426; 2017-04-04T03:39:06+08:00)
Maven home: C:\Program Files\apache-maven-3.5.0\bin\..
Java version: 1.8.0_121, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_121\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 7", version: "6.1", arch: "amd64", family: "windows"
```

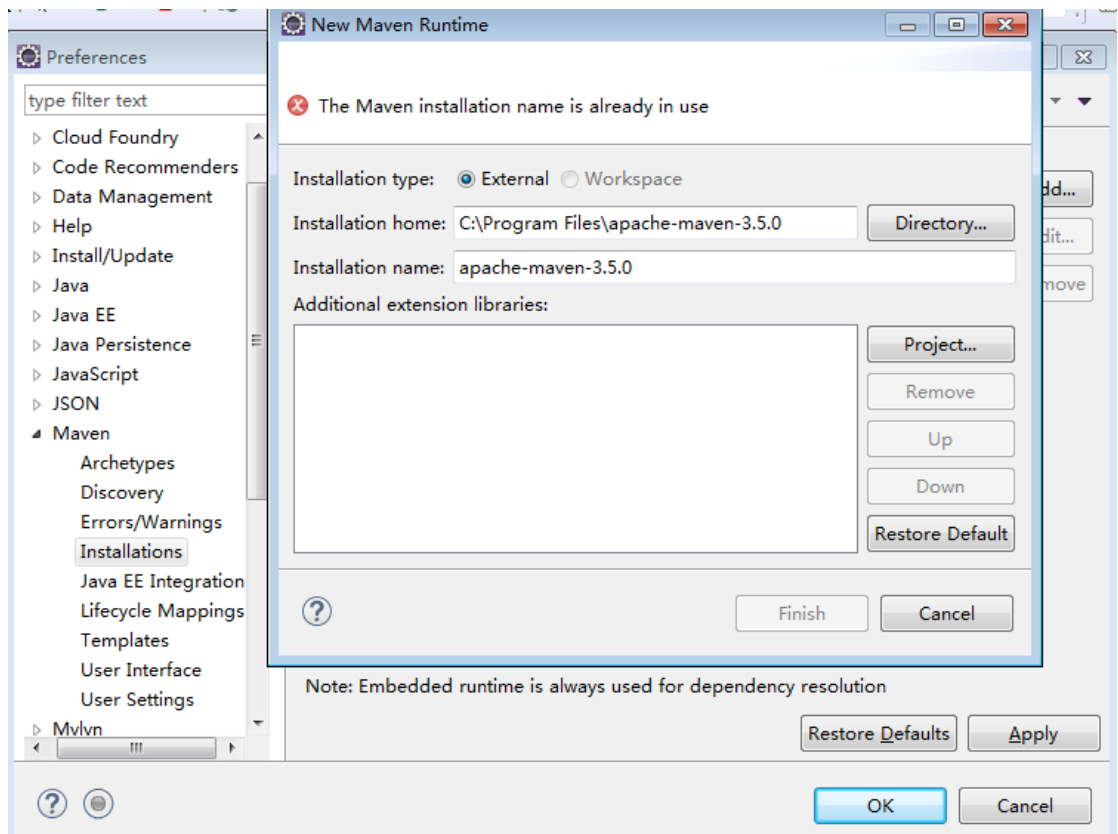
3. 将 Maven 与 Eclipse 整合

①.在 Eclipse 中安装 maven 插件

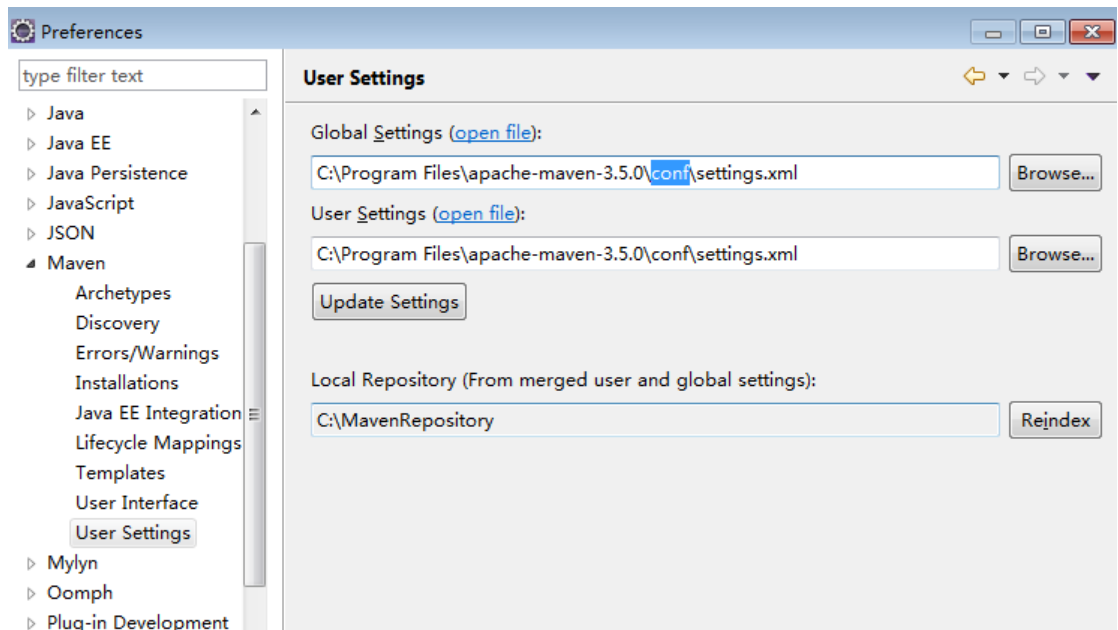


②.在 Maven---installations-->add:

在 eclipse 中关联 maven 软件



③.在 eclipse 中关联 maven 的 settings.xml 配置文件:



sili

④.修改 maven-conf 目录下的 settings.xml 文件.

修改本地仓库的路径.

```
<!--  
settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"  
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
        xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/  
        settings-1.0.0.xsd">  
  <!-- localRepository  
  | The path to the local repository maven will use to store artifacts.  
  |  
  | Default: ${user.home}/.m2/repository  
  -->  
  <localRepository>C:\MavenRepository</localRepository>
```

四.Maven 的仓库

1. 本地仓库:

用来存放从远程仓库或者从私服中下载的 jar 包.

2. 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.

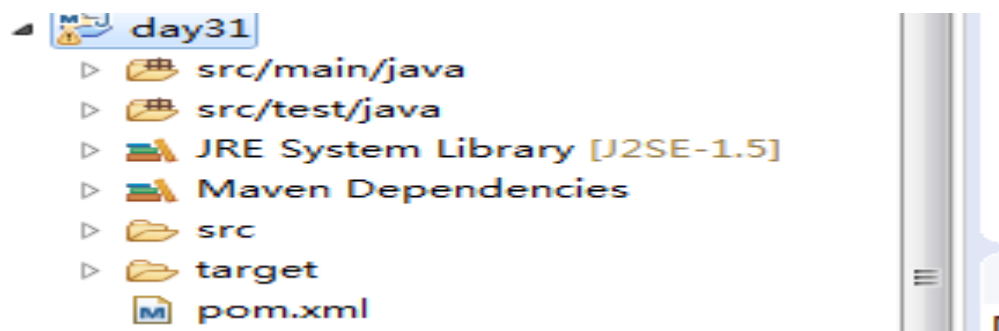
Maven 下载 jar 包的流程(重点):

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.

五.在 Eclipse 中利用 Maven 进行项目构建

1. 构建 JavaSE 项目

①.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;

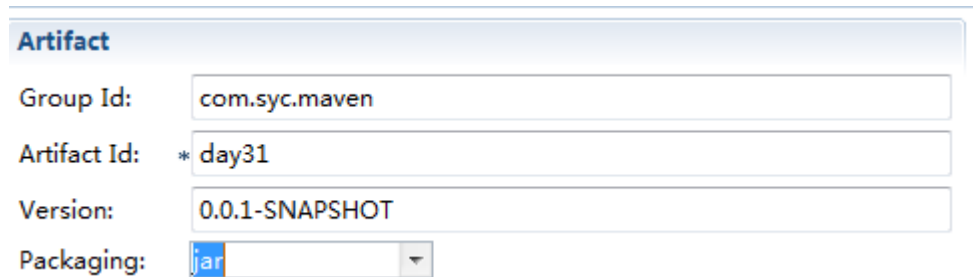
src/test/java:用来存放测试类的源文件.

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



The screenshot shows the 'Artifact' tab in a Maven IDE. It contains four input fields: 'Group Id' with the value 'com.syc.maven', 'Artifact Id' with the value '* day31', 'Version' with the value '0.0.1-SNAPSHOT', and 'Packaging' with a dropdown menu showing 'jar'.

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

```
<!-- 配置多个依赖 -->
<dependencies>
  <!-- 依赖,jar包 -->
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>HTTPClient</groupId>
    <artifactId>HTTPClient</artifactId>
    <version>0.3-3</version>
  </dependency>
</dependencies>
```

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.

```
16         <artifactId>junit</artifactId>
17         <version>3.8.1</version>
18         <scope>test</scope>
19     </dependency>
20     <dependency>
21         <groupId>HTTPClient</groupId>
22         <artifactId>HTTPClient</artifactId>
23         <version>0.3-3</version>
24         <scope>compile</scope>
25     </dependency>
26 </dependencies>
27 <repositories>
28     <repository>
29         <snapshots>
30             <enabled>>false</enabled>
```

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

在 Effective pom 中可以看到配置的默认的中央仓库.

```
<repositories>
  <repository>
    <!-- 是否下载snapshots版本的jar包 -->
    <snapshots>
      <enabled>>false</enabled>
    </snapshots>
    <!-- 仓库的id -->
    <id>central</id>
    <!-- 仓库的名称 -->
    <name>Central Repository</name>
    <!-- 仓库的url -->
    <url>https://repo.maven.apache.org/maven2</url>
  </repository>
</repositories>
```

插件仓库:

```

:!-- 配置插件仓库 -->
:pluginRepositories>
  <pluginRepository>
    <releases>
      <updatePolicy>never</updatePolicy>
    </releases>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
    <id>aliyun</id>
    <name>aliyun Repository</name>
    <url>http://maven.aliyun.com/nexus/content/repositories/c
  </pluginRepository>
:/pluginRepositories>

```

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

```

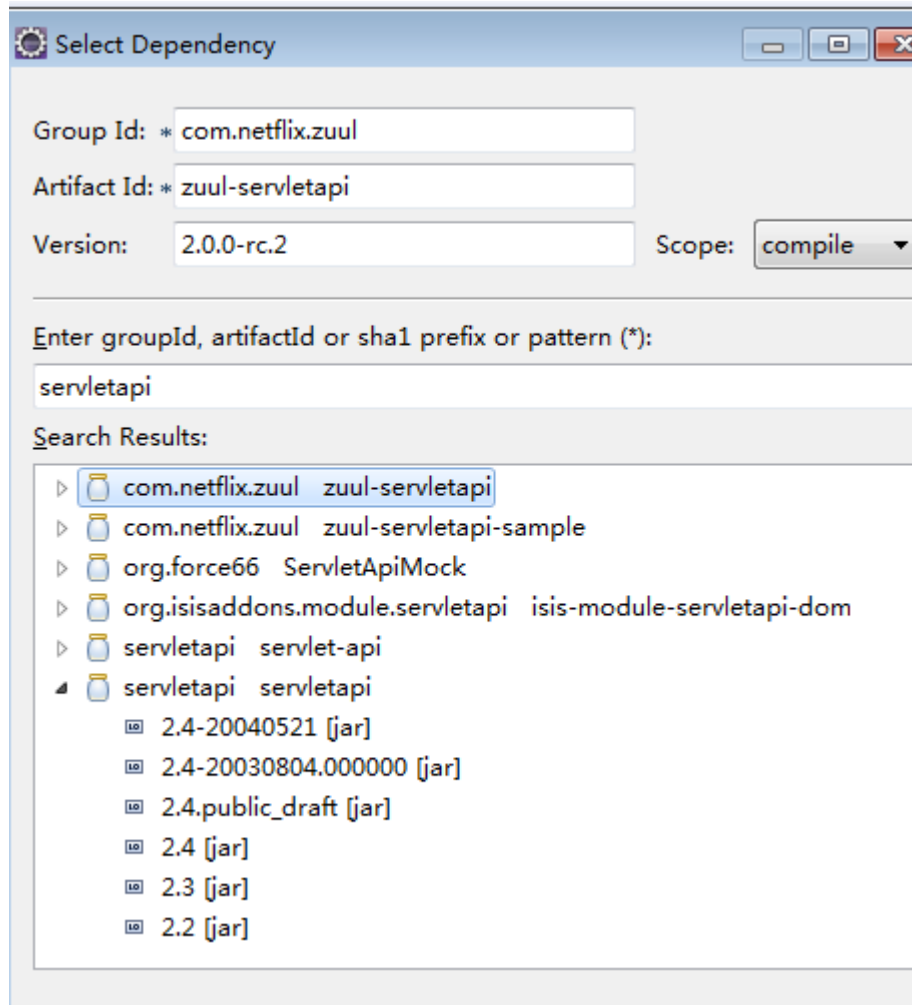
junit : 3.8.1 [test]
HTTPClient : 0.3-3 [compile]

```

[erview](#)
[Dependencies](#)
[Dependency Hierarchy](#)
[Effective P](#)

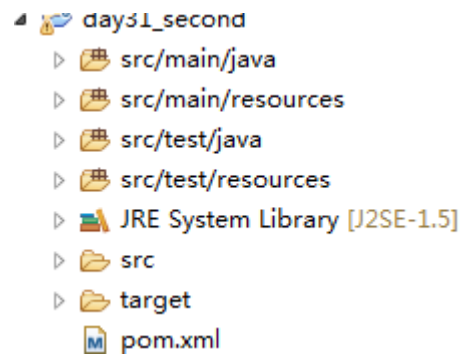
5. Dependencies 视图

搜索需要用的 jar 包



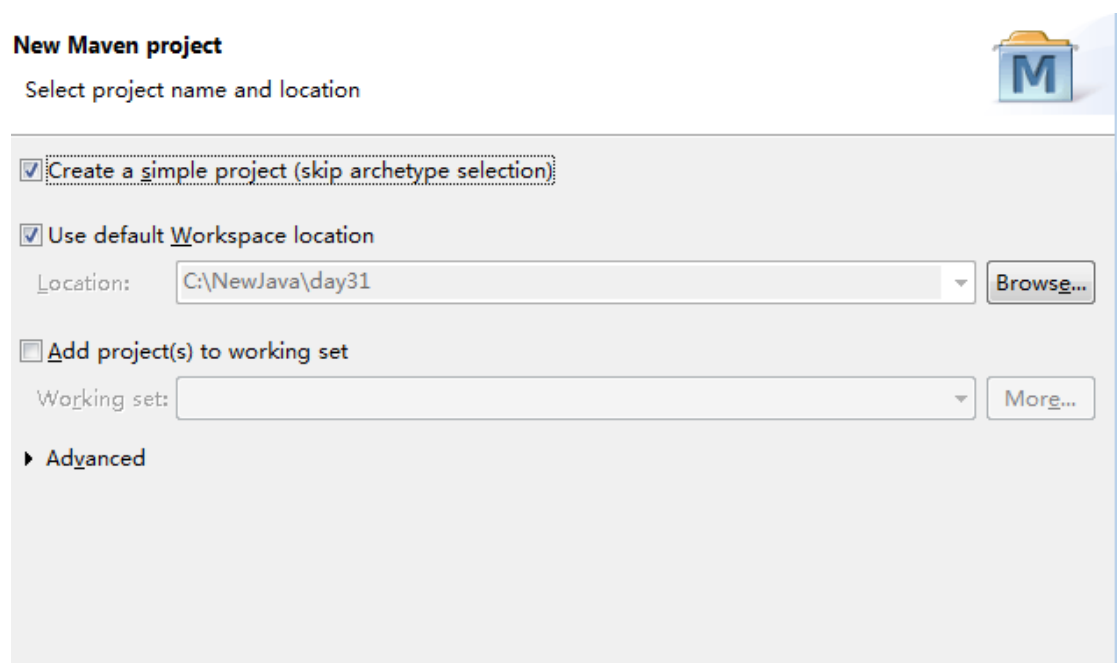
②.不用骨架进行项目的创建(没有联网的):





2. 构建 JavaEE 项目

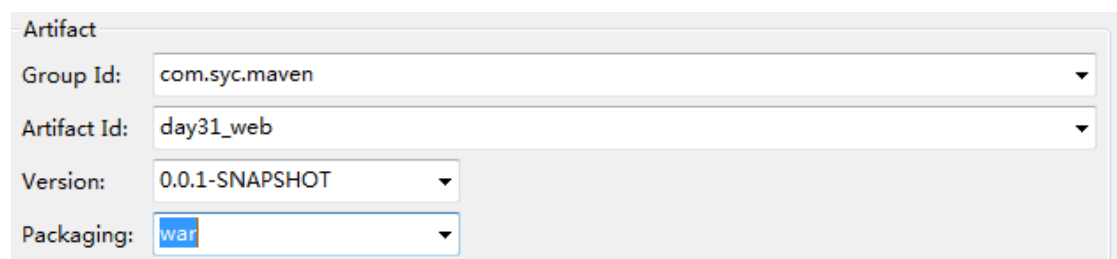
第一步:



注意:

一定要勾选第一个 checkbox!

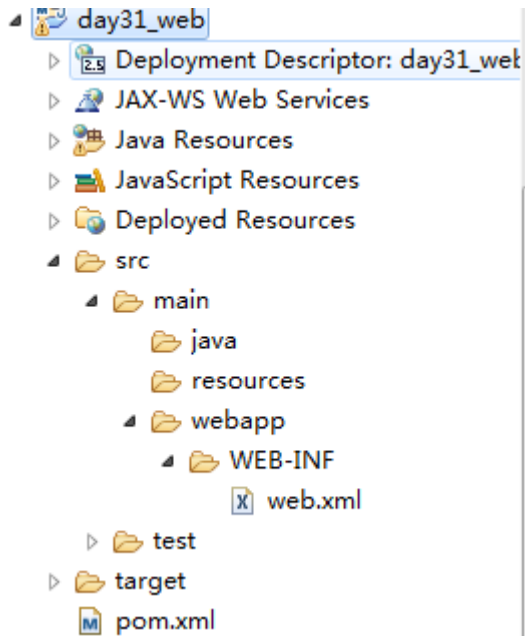
第二步:



注意:

packaging 里的类型必须用 war!

第三步:

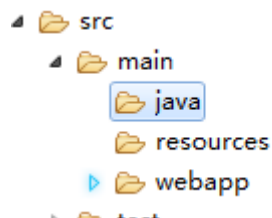


注意:

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.

六.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.

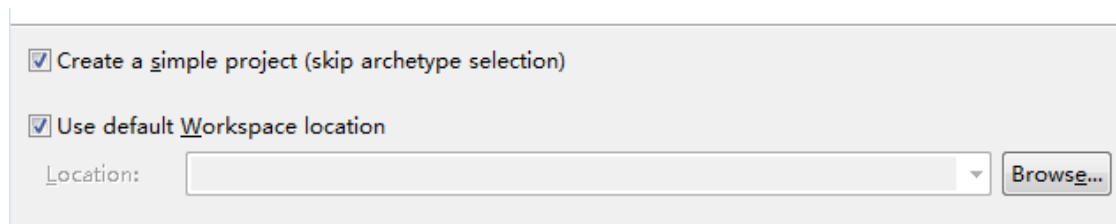
mvn install:将打包好的 jar 包或者 war 包存放到我们自己的本地仓库中,方便以后别人引用.

七.Maven 项目的继承

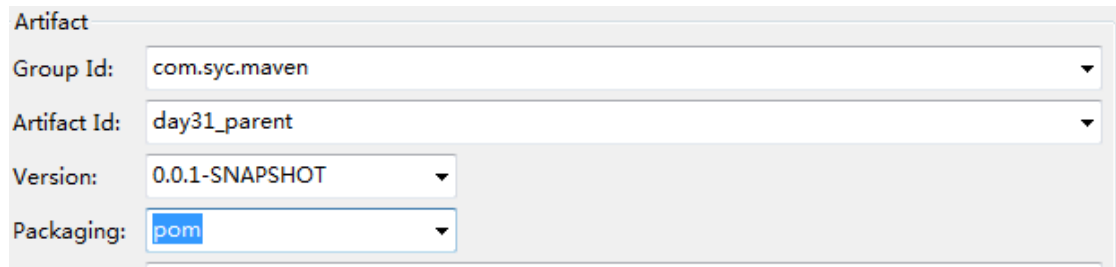
项目的继承:

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.

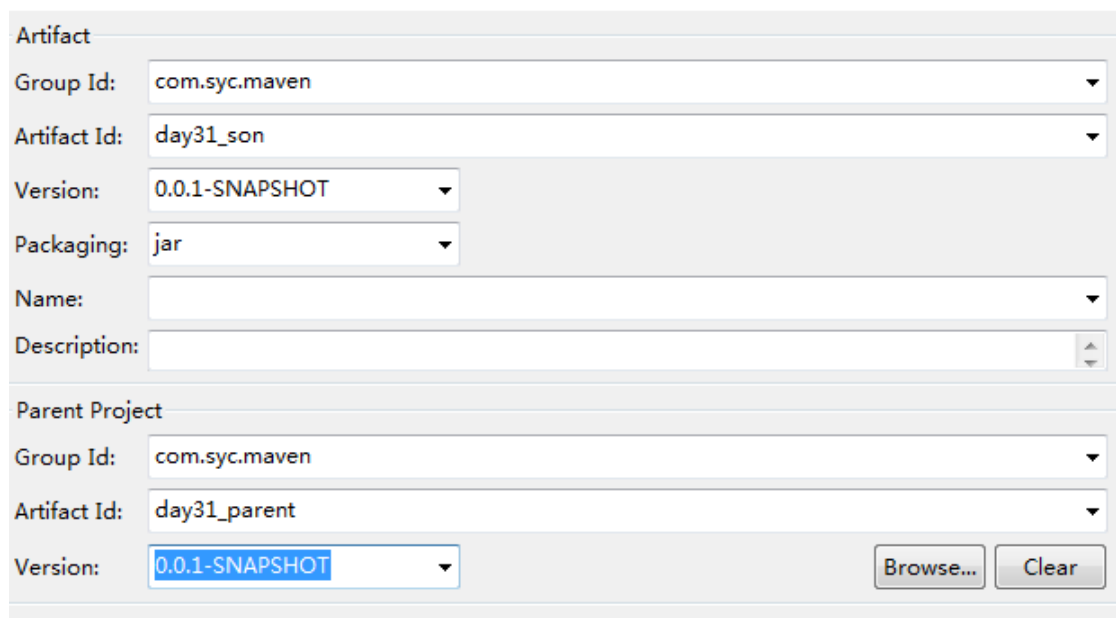


Artifact configuration form showing the following values:

- Group Id: com.syc.maven
- Artifact Id: day31_parent
- Version: 0.0.1-SNAPSHOT
- Packaging: pom

父项目的 packaging 类型必须是 pom!

3.第三步:创建子项目



Artifact configuration form for a child project showing the following values:

- Group Id: com.syc.maven
- Artifact Id: day31_son
- Version: 0.0.1-SNAPSHOT
- Packaging: jar
- Name: (empty)
- Description: (empty)

Parent Project configuration section showing the following values:

- Group Id: com.syc.maven
- Artifact Id: day31_parent
- Version: 0.0.1-SNAPSHOT

Buttons: Browse... Clear

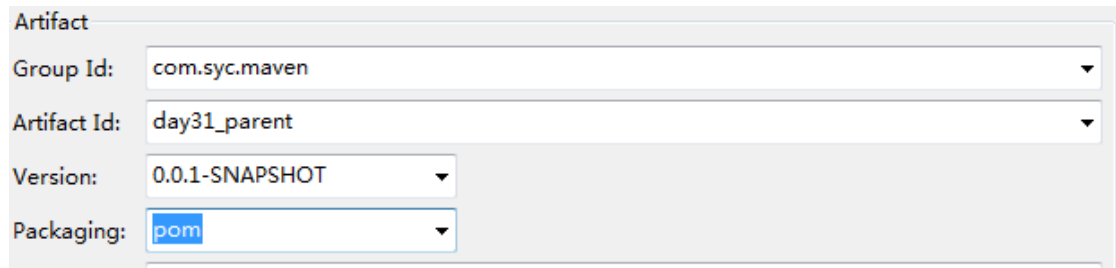
注意:子项目中必须设置 Parent Project!

八.项目的聚合

以三层架构为例,将整个项目拆分成 4 个部分.

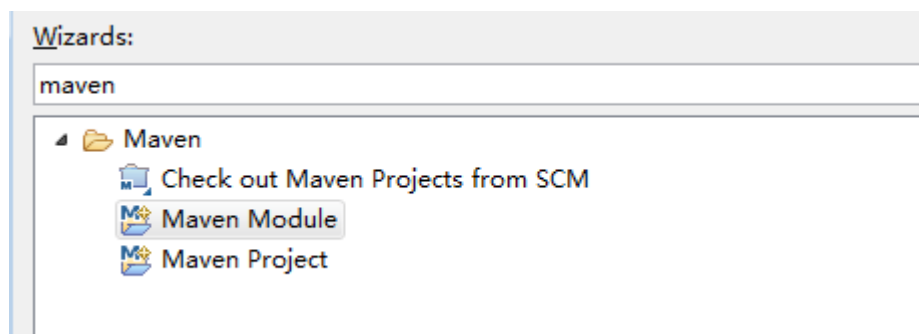
分别是:maven_parent,maven_dao,maven_service,maven_web.

第一步:首先建立一个 **package** 为 **pom** 类型的父项目!



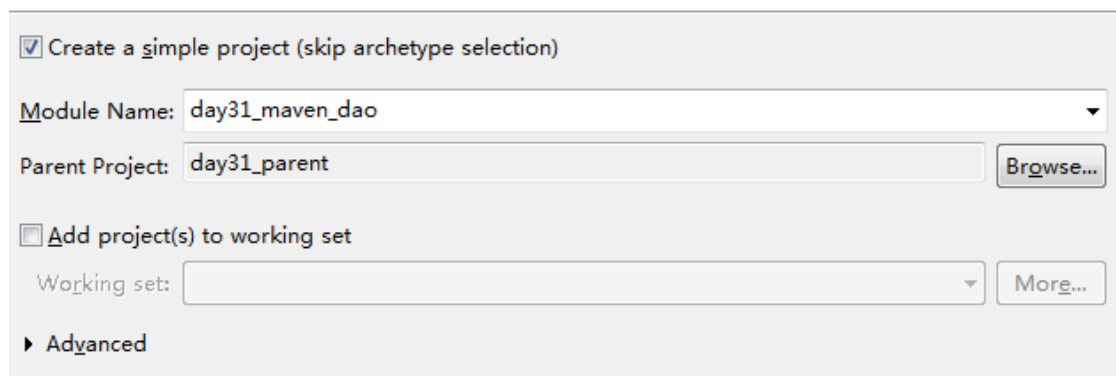
The screenshot shows the 'Artifact' configuration window in an IDE. It contains four fields with dropdown menus: 'Group Id' is set to 'com.syc.maven', 'Artifact Id' is set to 'day31_parent', 'Version' is set to '0.0.1-SNAPSHOT', and 'Packaging' is set to 'pom'.

第二步:选择父项目,建立 **maven module** 类型的项目模块.



The screenshot shows the 'Wizards' dialog box. Under the 'maven' category, the 'Maven Module' option is highlighted with a mouse cursor. Other options visible are 'Check out Maven Projects from SCM' and 'Maven Project'.

第三步:创建 **dao** 子模块.



The screenshot shows the 'Create a simple project (skip archetype selection)' dialog. The 'Module Name' field is set to 'day31_maven_dao'. The 'Parent Project' field is set to 'day31_parent', with a 'Browse...' button next to it. There is an unchecked checkbox for 'Add project(s) to working set' and a 'Working set:' dropdown menu with a 'More...' button. At the bottom, there is a collapsed 'Advanced' section.

Artifact

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:

第四步,创建 **service** 模块,过程与 **dao** 模块一样.

Create a simple project (skip archetype selection)

Module Name:

Parent Project:

☐ Add project(s) to working set

Working set:

Artifact

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:

第五步:创建 web 层模块.

The image shows two screenshots from an IDE's Maven configuration interface.

The top screenshot is the 'Wizards' dialog. The 'maven' filter is applied. Under the 'Maven' folder, the 'Maven Module' option is selected. The 'Create a simple project (skip archetype selection)' checkbox is checked. The 'Module Name' is 'day31_maven_web'. The 'Parent Project' field is empty with a 'Browse...' button. The 'Add project(s) to working set' checkbox is unchecked. The 'Working set' dropdown is empty with a 'More...' button.

The bottom screenshot is the 'Artifact' configuration dialog. The 'Artifact' section has the following values: 'Group Id' is 'com.sys.maven', 'Artifact Id' is 'day31_maven_web', 'Version' is '0.0.1-SNAPSHOT', and 'Packaging' is 'war'. The 'Name' and 'Description' fields are empty. The 'Parent Project' section has the following values: 'Group Id' is 'com.sys.maven', 'Artifact Id' is 'day31_parent', and 'Version' is '0.0.1-SNAPSHOT'.

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,

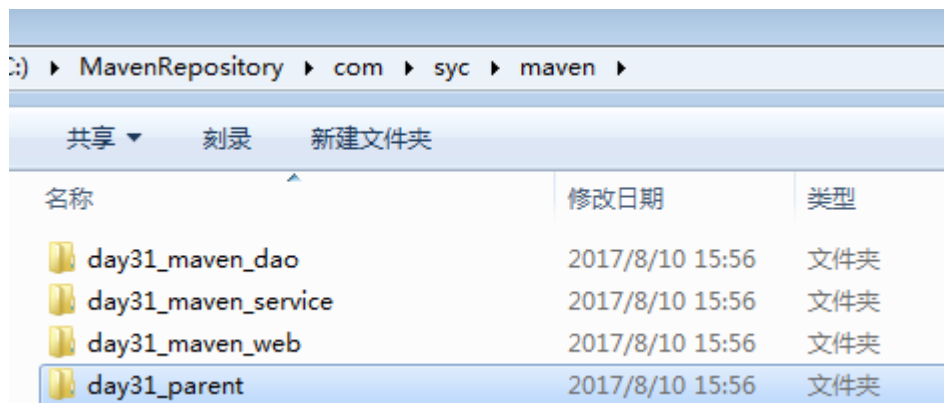
```
<!-- dependencies -->
<modules>
  <module>day31_maven_dao</module>
  <module>day31_maven_web</module>
  <module>day31_maven_service</module>
</modules>
```

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.



MavenRepository > com > syc > maven		
共享 ▾ 刻录 新建文件夹		
名称	修改日期	类型
day31_maven_dao	2017/8/10 15:56	文件夹
day31_maven_service	2017/8/10 15:56	文件夹
day31_maven_web	2017/8/10 15:56	文件夹
day31_parent	2017/8/10 15:56	文件夹

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:

```

<!-- 模块聚合的时候,模块之间要添加依赖。 -->
<dependencies>
  <dependency>
    <groupId>com.syc.maven</groupId>
    <artifactId>day31_maven_dao</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </dependency>
</dependencies>

```

在 web 层 pom.xml 文件中设置,web 层模块依赖 service 模块:

```

<dependencies>
  <dependency>
    <groupId>servletapi</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.4</version>
  </dependency>
  <dependency>
    <groupId>com.syc.maven</groupId>
    <artifactId>day31_maven_service</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </dependency>
</dependencies>

```

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.

```

▶ day31_maven_dao
▶ day31_maven_service
▶ day31_maven_web
▶ day31_parent

```