

- 1、javac -d . com/ljq/Hello.java 编译java程序 (-d指定文件目录, .代表当前目录)
- 2、jar -cvfm hello.jar main/resources/MANIFEST.MF(jar命令打包)
- 3、

使用 maven 命令生成项目 (idea 和 eclipse 生成项目最终也是依赖 maven 插件生成的) 不同的项目用不同的插件生成

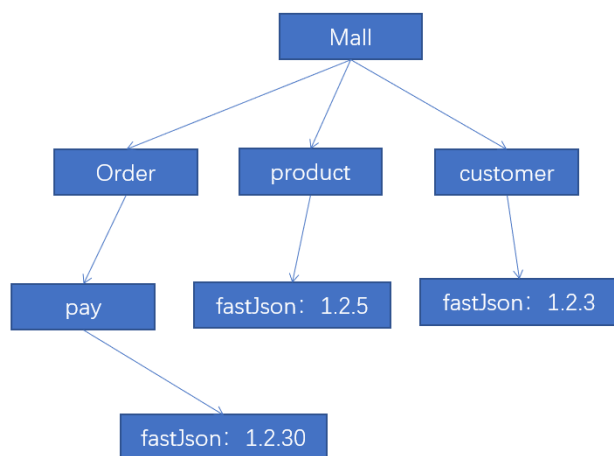
```
mvn archetype:generate -DgroupId=enjoy -DartifactId=simple -DarchetypeArtifactId=maven-archetype-quickstart -Dversion=1.0
```

```
mvn archetype:generate -DgroupId=enjoy -DartifactId=simple-web -DarchetypeArtifactId=maven-archetype-webapp -Dversion=1.0
```

- 4、exclusions排除冲突, execution指定插件运行目标
- 5、每个命令都与插件想关联。
- 6、执行插件mvn com.enjoy:enjoy-plugin:1.0:log -Dlog.name=ljq (可以通过-D传参)。log为插件目标, 一个插件中可以有多个AbstractMojo接口的实现, 每个实现就是不同的目标, 所以执行插件的时候要指定目标。如图:



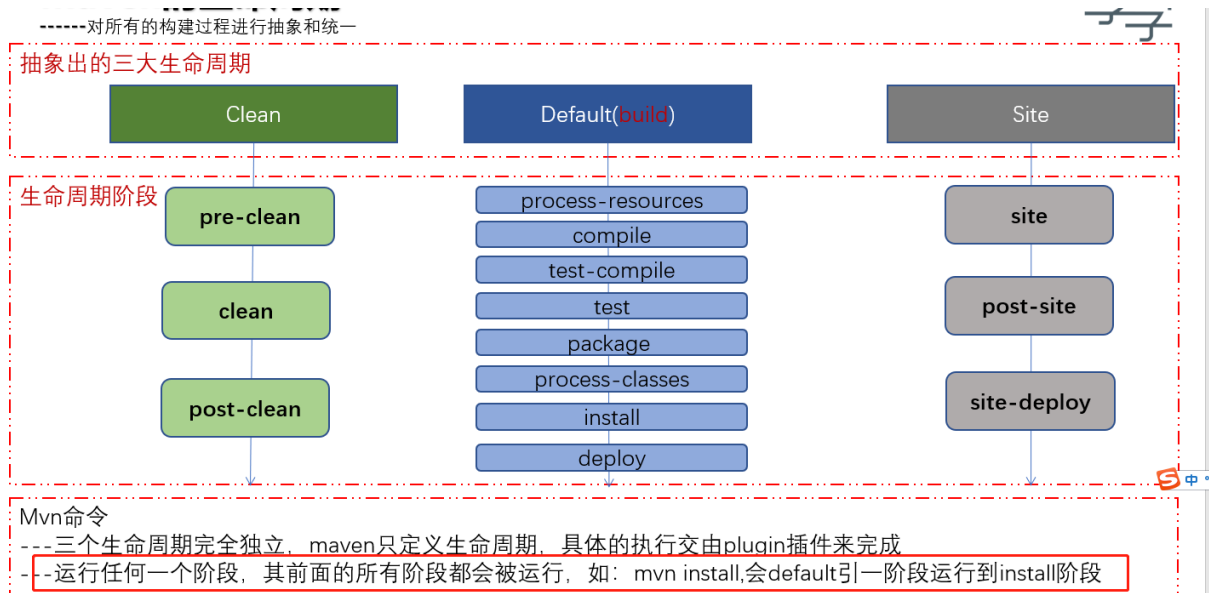
- 7、jar包冲突其实底层是jvm在作祟, 因为ClassLoader对同一个类只能加载一次, 当引入多个相同的jar时里边的class类会冲突, jvm加载时会不知所措。
- 8、maven的依赖传递解决措施如图:



Mall项目应该使用哪个fastjson包呢?

- 1、路径最短原则: product和customer里的引用路径较短
- 2、同路径长度下, 谁先声明谁优先: 看product依赖和customer依赖在pom中声明位置, 谁先用谁的
- 3、使用exclusions排除, 声明不要product或者customer不要fastjson依赖。

- 9、mvn dependency:tree会将pom最终决策出来的依赖拉出来, 树形显示
- 10、maven的生命周期。执行某个命令会默认执行它前面的阶段, 如图:



11、

· dependencyManagement 使用

· 作版本号管理使用

- 1、dependencyManagement 是声明而不引入依赖-----因此子项目需要显式的声明需要用的依赖（不需要版本号）
- 2、子项目中引用一个依赖而无版本号时，会沿着父子层次向上走，直到找到一个拥有 dependencyManagement 元素的项目，使用其版本号

· 用来实现多继承

- 1、maven 只能单继承，即一个项目只能使用 parent 标签定一个父级项目
- 2、使用 dependencyManagement 并设置其 scope=import，即表示继承此项目为父项目

在多继承的时候它只能import其他pom里边的dependencyManagement里边的依赖，继承以后再子pom的dependencies中还是需要手动引入依赖(和子pom的dependencyManagement没啥区别)

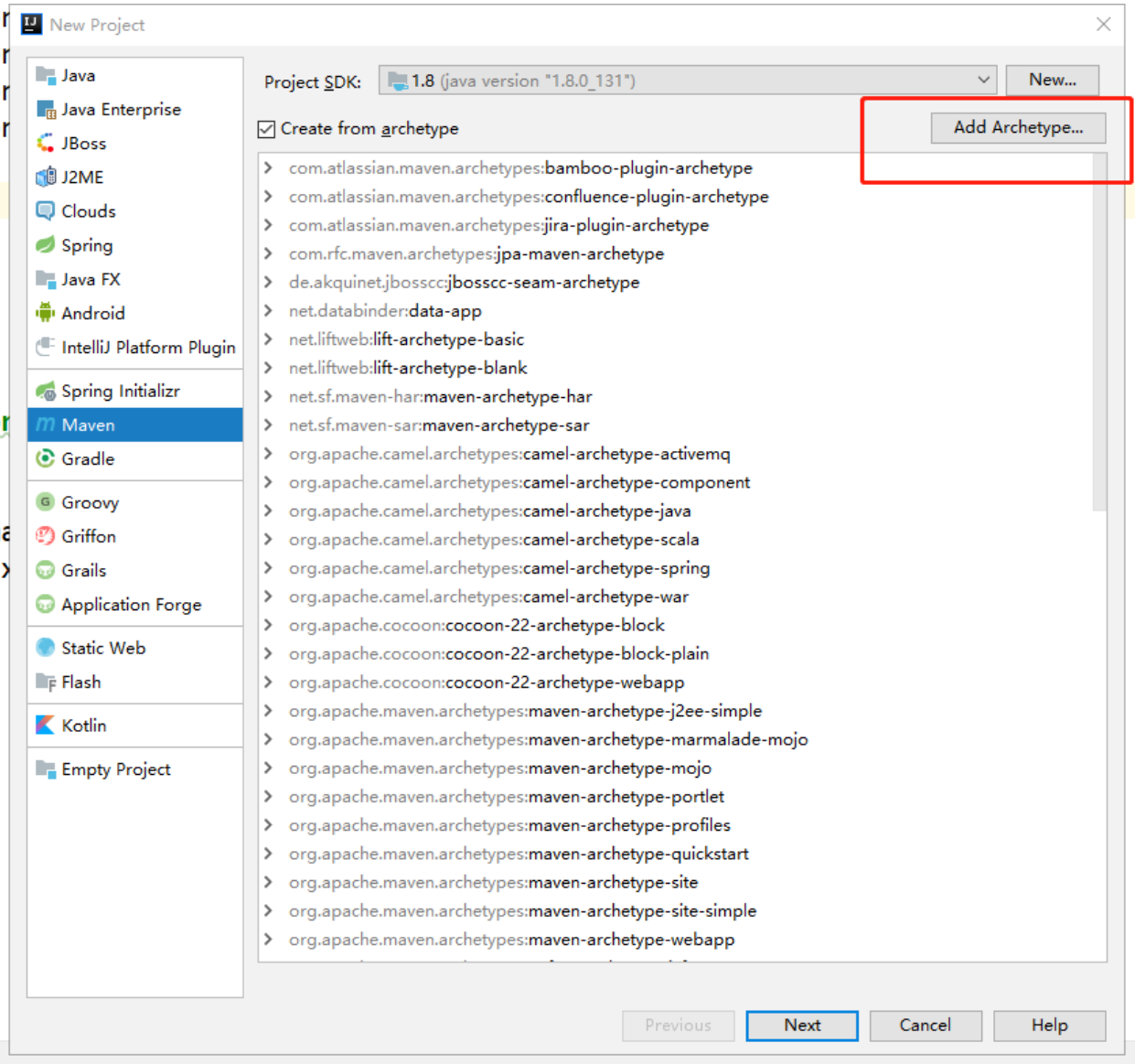
```

<!--再继承一个pom-->
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>enjoy</groupId>
      <artifactId>test</artifactId>
      <version>1.0</version>
      <type>pom</type>
      <!--引入进来-->
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

```

12、release(public)为稳定版本，snapshot为开发版本，格式：<version>1.0-snapshot</version>。

13、定义自己的项目骨架，打包到远程仓库。然后用idea创建项目的时候选择骨架所有人员就可以统一了，如图：



14、group、proxy、hosted的区别如图：

我们在nexus私服或其他maven库看仓库类型有hosted、proxy、group，有点懵比，今天查了一下

项目	具体说明
hosted	本地存储。像官方仓库一样提供本地私库功能
proxy	提供代理其它仓库的类型
group	组类型，能够组合多个仓库为一个地址提供服务

创建group，可以添加多个hosted、proxy，将它们组合起来对外服务：

Repositories / **Select Recipe** / **Create Repository: maven2 (group)**

Name: A unique identifier for this repository

This field is required

Online: ☒ If checked, the repository accepts incoming requests

Storage

Blob store:
Blob store used to store repository contents
default

Strict Content Type Validation:
☒ Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format

Group

Member repositories:
Select and order the repositories that are part of this group

Available

Filter

maven-central
third
maven-public
two

Members

maven-snapshots
maven-releases

创建proxy，需要添加远程仓库的url，比如阿里云demaven仓库地址：

Name: A unique identifier for this repository

This field is required

Online: ☒ If checked, the repository accepts incoming requests

Maven 2

Version policy:
What type of artifacts does this repository store?
Release

Layout policy:
Validate that all paths are maven artifact or metadata paths
Strict

Proxy

Remote storage:
Location of the remote repository being proxied, e.g. `https://repo1.maven.org/maven2/`

This field is required

Blocked:
☐ Block outbound connections on the repository

Auto blocking enabled:
☒ Auto-block outbound connections on the repository if remote peer is detected as unreachable/unresponsive

Maximum component age:
How long (in minutes) to cache
-1

Maximum metadata age:
How long (in minutes) to cache
-1