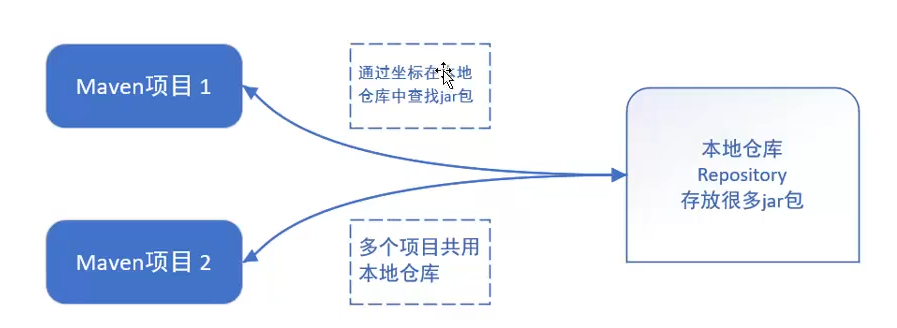
# 依赖管理

jar包管理也叫依赖管理。

## 原理与规范

1）在开发项目时，需要将所需的jar包导入到项目lib目录中，这些jar包会占用项目的体积，甚至有时比项目本身文件量还要大。为此，我们可以通过Maven来将这些jar包存放在一个本地仓库中，在项目中可以通过坐标来引用本地仓库中的jar包。实现了项目与jar包分离，减轻了项目体积。并且本地仓库能够被多个项目共用。



2）项目通过坐标引入本地仓库中的jar包。其坐标的命名规范通常如下:

**公司名/组织名(groupld) + 项目名/子项目名(artifactld) + 版本号**

3）在使用本地仓库时也不必担心效率问题，Maven通过建立本地索引，可以快速的找

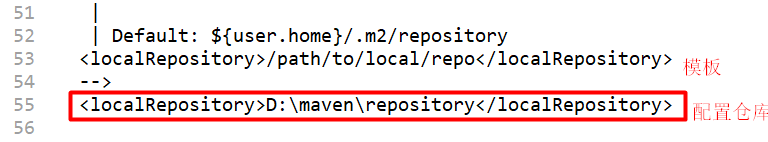
到符合要求的jar包，从而解决效率问题。

## 配置本地仓库

Maven安装后默认的本地仓库位置位于用户家目录.m2目录中的repository目录，如下：



如果需要修改，在 Maven安装目录/conf/settings.xml文件。将localRepository标签中的路径设置为本地仓库的路径。



本地仓库由自己创建，用于存放jar包，供Maven项目使用。

# Maven仓库

## 仓库的基本概念

Maven中有3个仓库：本地仓库、远程仓库、中央仓库。

1. 本地仓库指的是开发者本机电脑上的仓库，由个人维护。

2）远程仓库指的是私服上搭建的仓库，一般是公司在公司服务器搭建的。

3）中央仓库为大型仓库，由第三方团队维护。

国外的如<https://mvnrepository.com/>

国内的如阿里云中央仓库[**https://maven.aliyun.com**](https://maven.aliyun.com)

在配置中央仓库时，如果配置的是国外的，则下载速度有限；所以，在国内一般将远程仓库配置为阿里云的（但阿里云的jar包没有国外的全）。

## 仓库间的协作流程

1. 本地仓库中没有jar包时，如果配置了远程仓库和中央仓库。则首先会从远程仓库中下载jar包到本地仓库；如果远程仓库中也没有jar包，则远程仓库会去中央仓库中下载jar包到远程仓库，然后，本地仓库从远程仓库下载到本地仓库，最后供项目使用。
2. 本地仓库中没有jar包时，如果只单单配置了远程仓库或者是中央仓库。则会直接去远程仓库或中央仓库中下载到本地仓库，最后供项目使用。

所以，三种仓库的优先级是本地仓库--->远程仓库--->中央仓库。

## 配置中央仓库

在前面已经讲解了如何配置本地仓库。

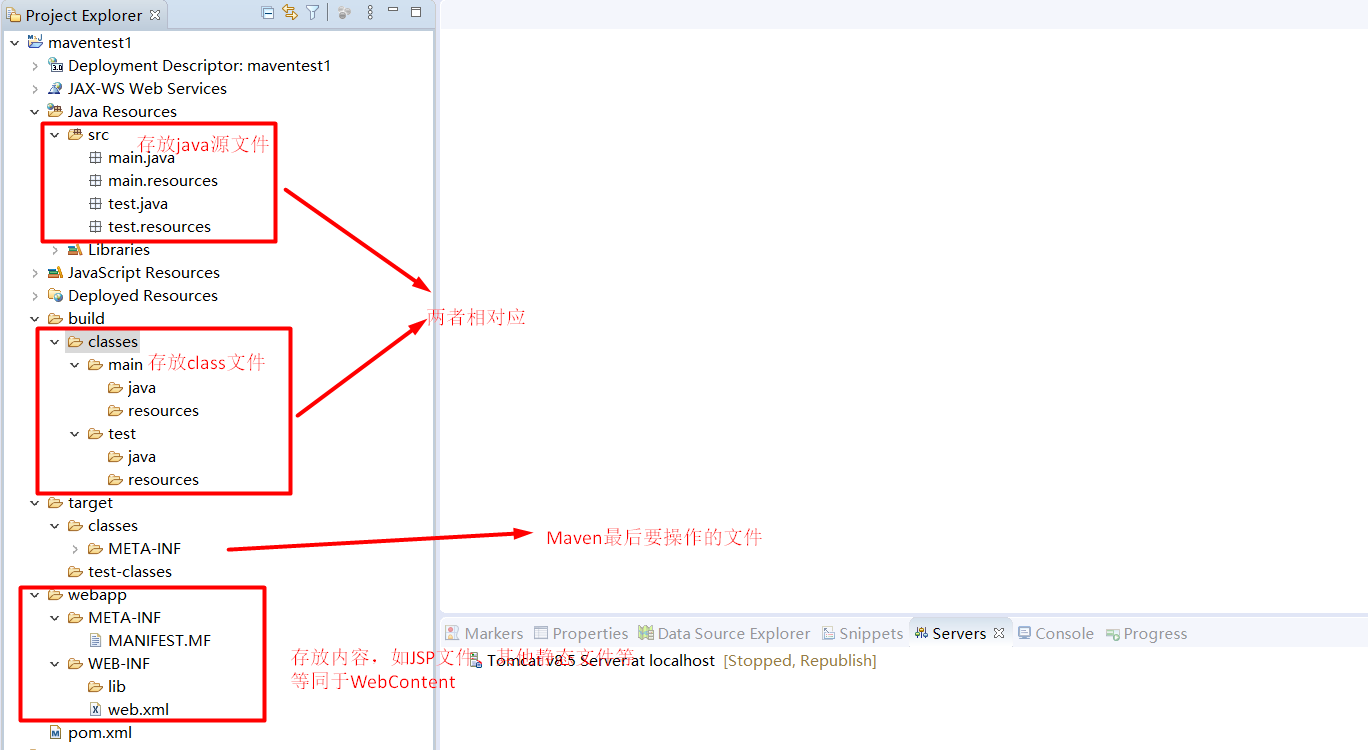
在Maven根目录--->conf--->settings.xml文件中配置中央仓库。如下



# Maven项目结构

## 认识Maven项目结构

创建Maven项目后，其项目结构如下：

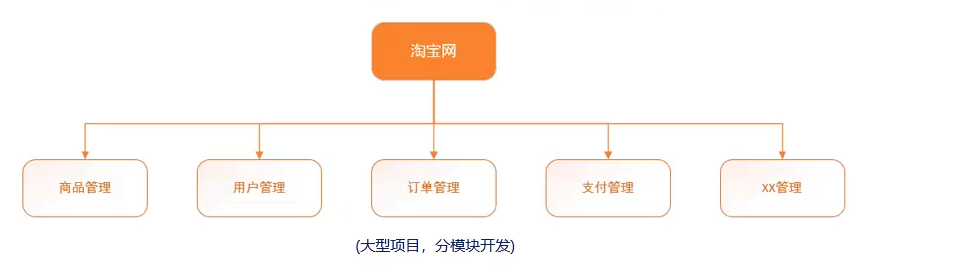


1. src目录存放全部的Java源文件，它对应着build/class文件夹，在build/class文件夹中保存了src编译后的class文件。其中main/java存放编译后的类文件，main/resources存放项目的资源，如SSM配置文件、properties等。test/java存放项目的测试类文件，test/resources存放测试用的资源。（test目录在实际开发中用的不多）
2. webapp目录表示web应用文件目录，web项目的信息，例如web.xml、jsp视图等。等同于WebContent目录。
3. target目录是maven的输出目录。存放项目构建后的文件和目录，如jar包、war包、编译的class文件等。是maven中clean、compile等命令操作的目录。

## 项目分工

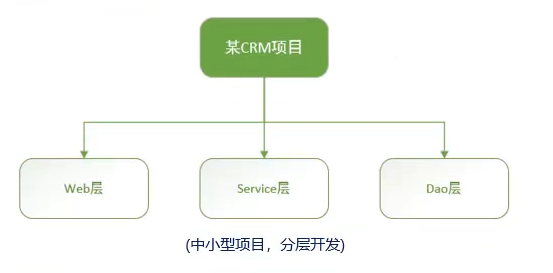
Maven可以将一个项目按照模块分为若干个工程，不同的团队或个人开发各自的工程，可大大的提高开发效率。但模块与模块之间需要进行通信，这时我们会将该模块变为jar包，放入远程仓库中，供有需要的模块进行调用。且不会影响各自模块的源码，因为jar包的形式本身就是.class文件的形式，不需要再次产生Java源文件，所以也不会污染到模块中存放class文件的目录。

1. 对于一个大型项目，项目分工一般以团队为单位，团队内可再次分工。例如淘宝：



将商品管理模块交给A团队、用户管理模块交给B团队，以团队进行开发。在团队中，可再次进行细分。将某模块交给张三、将某模块交给李四进行开发。

1. 对于中小型项目，项目分工一般以个体为单位，依照员工自身能力进行分工。



将项目进行分层，将每层交给对应的人员去开发。例如张三擅长业务层，则将业务层交给张三去开发。

# Maven常用命令与生命周期

## Maven的常用命令

1. **clear**：删除target目录中的东西；
2. **compile**：编译src/main目录中的java源文件（如果target目录不存在，则生成）；
3. **test-compile**：编译src/test目录中的java源文件；
4. **test**：执行test目录中的class文件，如果不存在，则会对src/test目录进行编译；
5. **package**：打包命令，在项目根目录执行此命令，并存入target目录中。包文件名为项目名+版本号.打包文件格式（可以是war或jar）
6. **install**：发布项目到本地仓库，常用在打jar包上，打成jar后可以供其他项目使用
7. **tomcat:run**：一键运行项目
8. **site**：生成项目的描述文档（html形式）。在target根目录中生成site目录，site中

为该项目的描述文档（html形式）

## Maven命令的生命周期

Maven命令有很多，从命令的功能上出发，可分为3个生命周期：

1. Clean生命周期：clean
2. Default生命周期：compile、test-compile、test、package、install
3. Site生命周期：site

不同生命周期的命令执行顺序可以不分先后执行，例如先clean再compile或先compile后clean；但是同一生命周期的命令必须遵循执行顺序。如Defaulst生命周期的执行顺序为：

**compile < test-compile < test < package < install**

例如：install命令的功能是将项目存入（通常是jar包的形式）本地仓库中，但它同时还具有前面4个命令的全部功能，在执行install命令时也会执行前面4个命令。再比如package命令用于将项目进行打包，同时它还具有前面3个命令的全部功能。以此类推。所以，命令的执行顺序指的是命令功能的包含问题。即后面的命令具有前面所有命令的功能。执行它时，也会执行排在它前面的命令。你可以从命令行中依次这些执行命令，观察命令行输出的内容。