# 一.Maven基础

## 0.jar包

jar 是 Java ARchive 的缩写，是一种基本 zip 格式的文件格式。目标是将 java 生成的类文件、资源文件、管理文件等按照特定的结构打包成一个独立的文件，方便程序的发布或网络的下载。

## 1.什么是Maven

Maven:项目构建工具，打包工具

### （1）.Maven项目文件结构：

**src\**

**main\**

**java\ ----** 项目开发的java代码

**resources\ --**项目开发的资源文件夹

**test\**

**java\ ---**单元测试代码（开发人员自测）

**pom.xml --** 当前项目的配置文件

基于pom.xml配置，使用maven命令，完成项目构建工作

Maven常用配置

（1）.依赖配置：指定使用哪个依赖包

（2）.指定一些通用配置：jdk版本、编码等

Maven的作用

项目构建：项目依赖，项目打包

## 2.Maven的配置文件——pom.xml

**注意：更改完xml文件以后，一定要刷新一下** 

Xml简单的内容：标签，标签属性

标签的定义方式有以下两种：



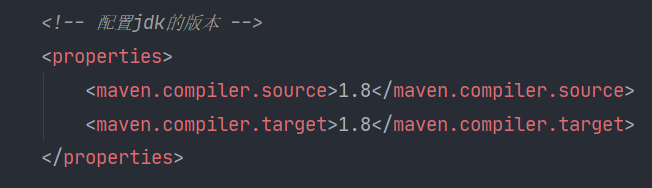
modelversion（pom的版本）之前的不用变



然后下面的groupId,artifactId,version对应的就是当前项目的组织号，产品号和版本号，我们自己修改



接下来就可以配置一些别的东西了，比如jdk的版本，依赖包等



## 3.依赖管理

（1）.maven仓库

maven 仓库是一个类似手机上的 App Store 的东西，上面会有全世界的人上传的各种第三方的 jar 包供我们使用，当我们的项目需要用到其中的内容，可以像手机安装 app 一样方便的进行查找和安装。

[maven 仓库](https://search.maven.org/)

其中我们的项目用到了另一个项目，叫做依赖关系。

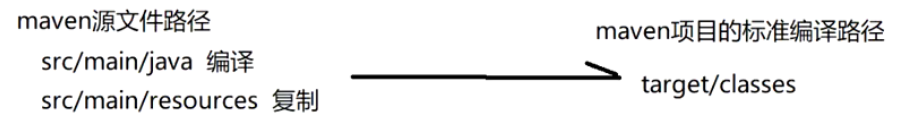
而一个项目中往往需要很多的依赖，所以诞生了依赖管理的概念。

（2）.在pom.xml文件中配置依赖



添加依赖的时候也要写明groupId,artifactId,version，如果在本地找不到，idea会自动开始下载

## 4.生命周期及其相关命令

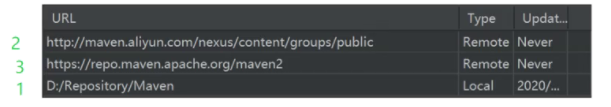


* clean:清楚target目录
* compile:编译当前文件，会在target目录下生成相关编译文件
* package:将当前项目中的类打成jar包供他人使用
* install:将target里面的jar包安装到本地仓库（了解）
* deploy:将target里面的jar包提交到公司的远程仓库给团队其他人员使用（了解）

## 4.Maven加载依赖包的顺序

Maven加载依赖包的顺序为：

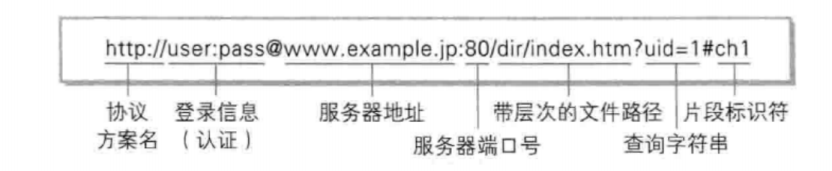
本地仓库——配置的远程仓库（阿里云）——官方远程仓库

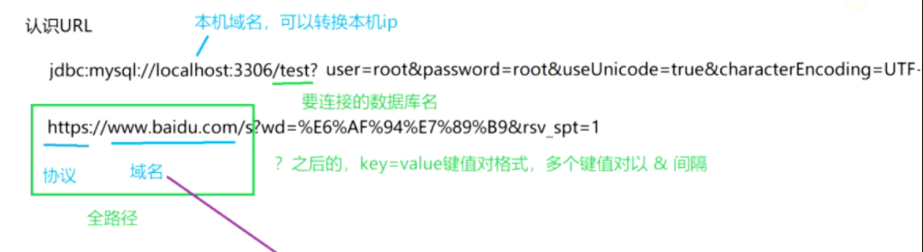


# 二.网络原理之Http

## 1.Http原理

### （1）.URL及协议





* url中的请求数据：请求路径?key1=value&key2=value2

1）.网络数据传输，都需要使用同样的协议，双方约定好的统一规范（封装和解析数据格式规范）

协议：数据格式的约定

目标：学习http协议里面的格式，结合理论实操，进行程序调试

http协议格式本身+http数据（数据中包含自己的数据格式）

2）.再次回顾URL

* 域名：基于DNS解析为IP
* IP:网络中定位主机的地址（逻辑地址）
* PORT:端口号，定位某个主机中唯一的进程（应用程序）
* http请求的访问路径：

服务路径定位资源

* + - url是全路径（绝对路径）
    - uri包含了url这种全路径，还包含相对路径

特殊的注意事项：输入域名进行访问，其实是访问/这个路径的资源

浏览器中，不输入端口号，是因为http协议的默认端口是80

### （2）.urlencode和urldecode

像 / ? : 等这样的字符, 已经被url当做特殊意义理解了. 因此这些字符不能随意出现。

比如, 某个参数中需要带有这些特殊字符, 就必须先对特殊字符进行转义.

转义的规则如下:

将需要转码的字符转为16进制，然后从右到左，取4位(不足4位直接处理)，每2位做一位，前面加上%，编码成%XY格式



"+" 被转义成了 "%2B"

urldecode就是urlencode的逆过程

**注：有时候显示的是中文，但是实际上都是%XY的形式的**

