# maven

## 一、简介

### 1、什么是maven?

maven是apache下面的一个**项目管理**和**构建**工具,它可以帮助我们创建和管理项目

### 2、为什么使用maven?

可以将一个项目按某种规则进行拆分，分成不同的模块，多个模块间存在依赖关系,可以使用maven进行项目的管理和构建

#### 2.1、jar包的管理工具

- 通过仓库管理jar包
- 解决jar包间的冲突和依赖
- 自动下载jar包

#### 2.2、项目的管理工具

#### 2.3、自动化的构建工具

构建过程:编译主代码--->编译测试代码---->执行单元测试---->生成测试报告----->打包---->部署

### 3、术语

#### 3.1、中央仓库

是一个网络仓库, 用于存储jar包和maven插件

http://repo1.maven.apache.org/maven2/

#### 3.2、本地仓库

从中央仓库中下载的jar包在本地的存放位置，也是一个仓库。

#### 3.3、镜像仓库

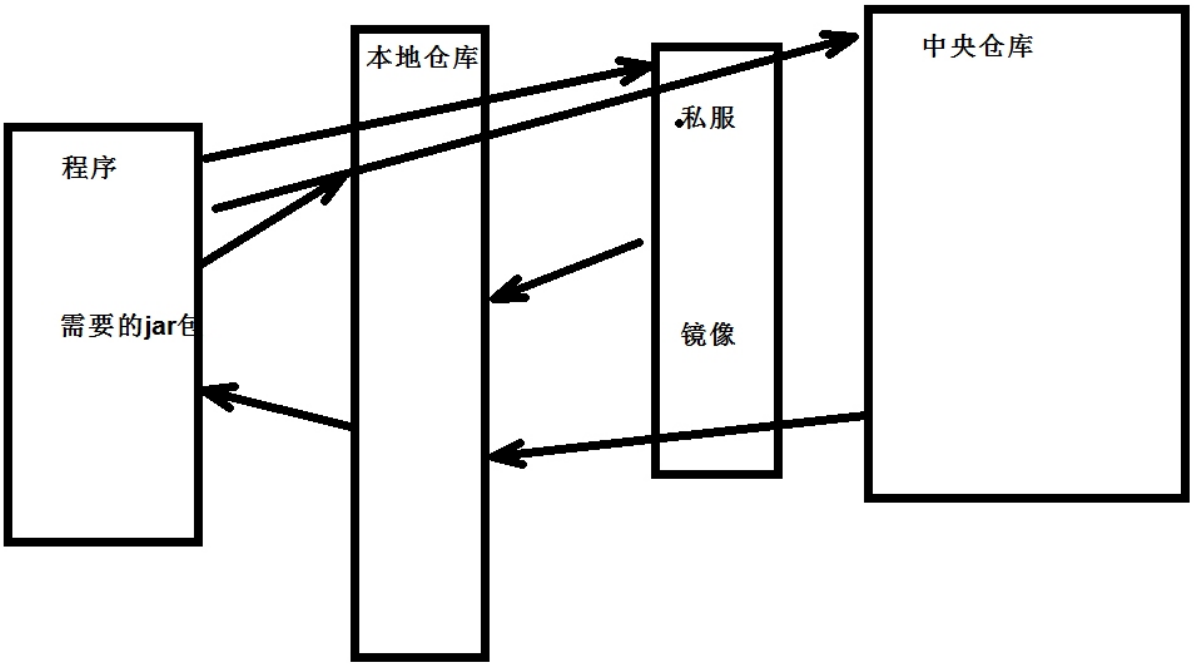
对某一仓库的镜像，

例如：阿里云：http://maven.aliyun.com/nexus/content/groups/public/

#### 3.4、私服

局域网内部搭建的maven服务器

#### 3.5、各个仓库之间的关系




### 4、单词

repository　仓库

archetype　原型、骨架(模板)

artifact　　成品(项目 jar  war.....)

## 二、安装

- 解压
- 配置环境变量

  mevan依赖java_home

```
echo  %java_home%

M2_HOME=maven的安装目录
PATH=原PATH;%M2_HOME%/bin;

在控制台运行命令：mvn  -verison
出现以下信息
Apache Maven 3.2.5 (12a6b3acb947671f09b81f49094c53f426d8cea1; 2014-12-15T01:29:23+08:00)
Maven home: E:\maven\apache-maven-3.2.5
Java version: 1.8.0_171, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_171\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 7", version: "6.1", arch: "amd64", family: "dos"

说明maven安装成功
```

## 三：配置本地仓库

### 3.1、本地仓库的默认位置

```
~\.m2\repository
```

### 3.2、修改本地仓库的默认位置

修改conf/settings.xml

```
<localRepository>D:\maven\maven-repo1</localRepository>
```

### 3.3、使用命令创建maven项目

```
cd maven/maven-projects
mvn archetype:generate #自动生成maven项目
```

## 四：配置镜像库

```
<mirror>
  <id>aliyun</id>
  <mirrorOf>*</mirrorOf><!--所有访问都使用该镜像库-->
  <name>aliyun maven</name>
  <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
</mirror>
```

## 五：配置maven的JDK版本

默认使用maven创建的项目使用的jdk是1.5

修改其默认配置

```
<profile>
  <id>jdk-1.7</id>
  <activation>
    <activeByDefault>true</activeByDefault>
    <jdk>1.7</jdk>
  </activation>
  <properties>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
    <maven.compiler.compilerVersion>1.7</maven.compiler.compilerVersion>
  </properties>
</profile>
```

## 六：创建maven项目

### 6.1、java项目的创建

#### 6.1.1、目录结构

java工程约定的目录结构如下

```
|-项目名称
    |-src  //程序代码
        |-main  //程序主代码
            |-java  //源代码
                |--//用于存放源代码，相当于传统项目的src
            |-resources  //配置文件目录
                |--//用于存放配置文件

        |-test //测试代码
            |-java  //用于存放测试代码
            |-resources
                |--//用于存放测试代码需要的配置文件
    |-pom.xml  //maven的核心配置文件
```

#### 6.1.2、执行maven操作

```
先切换到对应的项目所在的目录，与pom.xml平级的目录
mvn compile
mvn clean compile test
```

### 6.2、web项目的创建

#### 6.2.1、目录结构

```
|-项目名称
    |-src  //程序代码
        |-main  //程序主代码
            |-java  //源代码
                |--//用于存放源代码，相当于传统项目的src
            |-resources  //配置文件目录
                |--//用于存放配置文件
            |-webapp  //网站的根目录，相当于传统项目的webRoot
                |WEB-INF
                    |-web.xml
        |-test //测试代码
            |-java  //用于存放测试代码
            |-resources
                |--//用于存放测试代码需要的配置文件
    |-pom.xml  //maven的核心配置文件
```

## 七：Maven命令

### 1、常用命令

| 命令 | 作用 | 功能描述 |
| --- | --- | --- |
| archetype | 创建maven项目 | 根据模板生成项目骨架 |
| clean | 清理 | 删除target目录 |
| compile | 编译 | 将main中的源代码编译成字节码，存放在target/classes目录下 |
| test | 测试 | 执行测试类(使用junit)，并生成测试报告 |
| package | 打包 | 将java项目打包成jar包，将web工程打包成war包 |
| install | 安装 | 将项目的jar包或者是war包，安装到本地仓库，供其它项目使用 |
| deploy | 发布、安装 | 将项目的jar包或者war包发布到中央仓库，供其他人使用,需要提供账号 |
| site | 生成站点 | 生成关于项目介绍的网页 |

**注意:**

- 所有的命令都要跟在mvn 的后面，如：mvn test
- 所有的命令都必须在某个项目的根目录执行，和pom.xml平级的目录
- 多个命令可以同时执行

### 2、生命周期

maven定义了三套生命周期：clean,default,site

每个生命周期之间是相互独立的，每一生命周期可以包含多个阶段，这些阶段是有顺序的,前面的阶段执行后，才能执行

default生命周期包含：compile,test,package

## 八：在IDEA中使用maven

### 1、 配置maven

### 2、创建maven项目

## 九：pom.xml文件

### 1、简介

pom:project object  model  项目对象模型

pom.xml是maven的核心配置文件

**一个maven项目有且仅有一个pom.xml文件，该文件必须在项目的根目录下**

### 2、坐标

```
<!--
        坐标
        groupId:组织id,表示当前模块所隶属的项目
        artifactId:模块id
        version:当前版本号
        根据这三个坐标指定引用在本地仓库中对应的唯一的位置
        规则：所引用的jar包在本地仓库的位置：本地仓库目录/groupid/artifactid/version/artifactId-version.jar
    -->

<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
</dependency>
```

## 3、dependency

### 3.1、基本配置

如何查找一个jar包的坐标？

http://repo1.maven.apache.org/maven2

https://mvnrepository.com/

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.17</version>
</dependency>
```

### 3.2、scope作用域

> 表示依赖的作用域， 用来配置依赖的jar包可作用的范围

| 取值 | 含义 |
| --- | --- |
| compile | 该依赖可以在整个项目中使用，参与打包部署，默认值 |
| test | 该依赖只能在测试代码中使用，不参与打包部署 |
| provided | 该依赖编写源代码时需要，不参与打包部署,如：servlet.jar |
| runtime | 该依赖在编写代码时不需要，运行时需要，参与打包部署,如，mysql驱动包 |
| system | 表示使用本地系统路径下的jar包，oracle驱动 |

## 4、properties

全局属性，一般情况下用于定义全局jar包的版本信息

```
<mysql-connector-java.version>5.1.38</mysql-connector-java.version>

<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>${mysql-connector-java.version}</version>
  <scope>runtime</scope>
</dependency>
```

### 5、plugins

配置插件，是一种工具

- tomcat7-maven-plugin插件(web项目)

```
<!--tomcat7插件-->
<plugin>
  <groupId>org.apache.tomcat.maven</groupId>
  <artifactId>tomcat7-maven-plugin</artifactId>
  <version>2.2</version>
  <configuration>
    <path>/maven03</path>
    <port>8888</port>
  </configuration>
</plugin>
```

## 十：maven中的三大关系

### 1、继承

```
 <!--引用父项目，指定父项目的坐标-->
 <parent>
   <groupId>com.zte.maven02</groupId>
   <artifactId>parent01</artifactId>
   <version>1.0-SNAPSHOT</version>
   <!--指定该父项目的pom.xml-->
   <relativePath>../parent01/pom.xml</relativePath>

 </parent>
```

### 2、聚合

> 将多个子项目添加到父项目中，可以通过对父项目的操作，从而实现对所有子项目做同样的操作

```
 <!--聚合子项目，指定子项目的根目录-->
 <modules>
   <module>../child01</module>
   <module>../child02</module>

 </modules>
```

### 3、依赖

> 最常见的一种关系，就是说你的项目需要依赖到其它项目
>
> 比如：apache-commons包，spring包.....