Maven作用：通过pom.xml管理依赖，也可以通过mvn命令来创建(generate)、编译(compile)、测试(test)、清理(clean)、打包(package)、安装(install)、部署(deploy)项目.

Maven是基于项目对象模型(POM project object model)，可以通过一小段描述信息（配置）来管理项目的构建，报告和文档的软件项目管理工具。Maven的核心功能是合理叙述项目间的依赖关系，通过pom.xml文件的配置获取jar包，而不用手动去添加jar包。

Maven项目都会在POM.xml中使用<groupId>、<artifactId>、<version>来标识自己，以让别的项目可以通过上述3个属性找到自己：

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

//-------------------------------------------------------------------------------------

<groupId>com.wuhao.maven.quickstart</groupId> groupId指定包路径名

<artifactId>simple</artifactId> artifactId指定项目名

<packaging>jar</packaging>

<version>1.0</version> version指定版本

//-------------------------------------------------------------------------------------

<name>simple</name>

<url>http://maven.apache.org</url>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

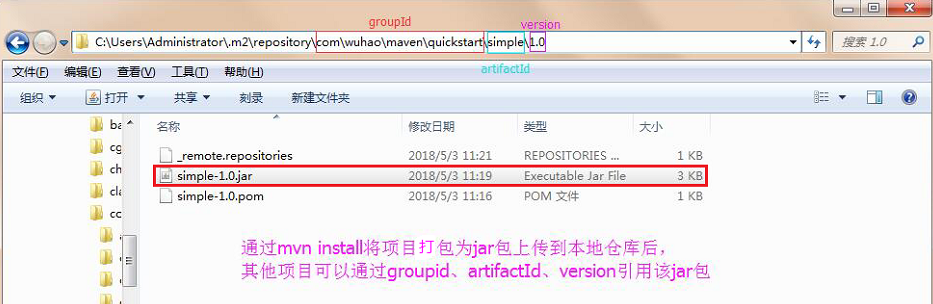
<scope>test</scope>

</dependency>

</dependencies>

</project>

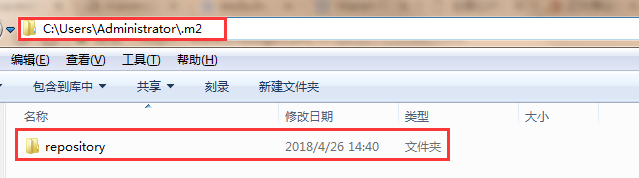
在使用mvn install将项目文件压缩打包上传到本地仓库后，<groupId>、<artifactId>、<version>对应为：



Maven可以用来管理jar依赖，jar包是放在仓库中的，Maven中的仓库分为本地仓库、第三方仓库(私服)、中央仓库.

本地仓库:

Maven会将工程中依赖的构件(Jar包)从远程下载到本机一个目录下管理, 默认在 $user.home/.m2/repository下



可以在Maven的安装目录下的\conf\settings.xml文件中进行修改

-------------------------------------------

<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>/path/to/local/repo</localRepository>

-->

-------------------------------------------

第三方仓库(私服):

一般由公司设立，只对本公司内部共享使用。既可以作为公司内部构件协作和存档，也可作为公用类库镜像缓存，减少从外部访问和下载的频率。使用私服为了减少对中央仓库的访问，私服可以使用局域网，中央仓库必须使用外网。连接私服需要单独配置。如果没有配置私服，默认不使用。

中央仓库：

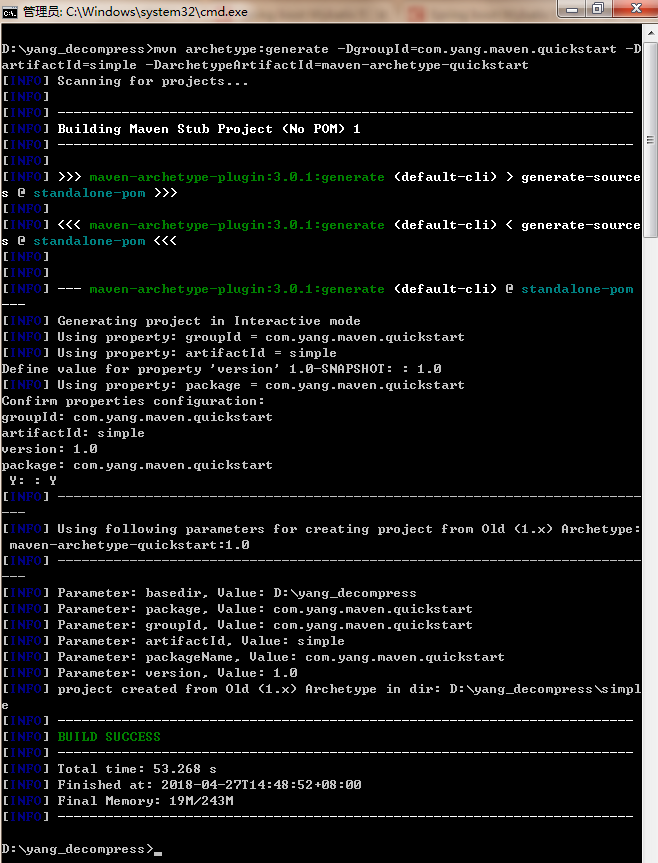
Maven内置了远程公用仓库：<http://repo1.maven.org/maven2,> 该公共仓库是由Maven官方维护，里面有大量的常用类库，并包含了世界上大部分流行的开源项目构件。工程依赖的jar包如果本地仓库没有，默认从中央仓库下载.

**Maven创建、编译、测试、清理、打包、安装、部署项目：**

Maven不仅可以管理jar包依赖，也可以用来创建项目：

创建java项目：

mvn archetype:generate -DgroupId=com.yang.maven.quickstart -DartifactId=Sample -DarchetypeArtifactId=maven-archetype-quickstart



创建Web项目：

mvn archetype:create -DgroupId=com.yang.maven.quickstart -DartifactId=myWebApp

-DarchetypeArtifactId=maven-archetype-webapp -Dversion=0.0.1-snapshot

Maven对项目的其他操作(下述操作都需要在有pom.xml的目录下运行):

编译：mvn compile　　--src/main/java目录java源码编译生成class（target目录下）

　 测试：mvn test　　　　--src/test/java 目录编译

　 清理：mvn clean　　　 --删除target目录，也就是将class文件等删除

　 打包：mvn package　　--生成压缩文件：java项目#jar包; web项目#war包, 放在target目录下

　 安装：mvn install　　　--将压缩文件(jar或者war)上传到本地仓库

　 部署|发布：mvn deploy　　--将压缩文件上传私服

Maven的依赖<dependency>子标签：

<dependency>的子标签一共有9个，重要的有7个，分别为：

<dependency>

<groupId></groupId>

<artifactId></artifactId>

<version></version>

<type>jar</type>

<optional>false</optional>

<exclusions>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

</exclusion>

</exclusions>

<scope>compile</scope>

</dependency>

**type：**依赖的类型，比如是jar包还是war包，默认为jar，表示依赖的jar包

<type>pom.lastUpdated</type>

lastUpdated的意思是表示使用更新描述信息，起占位符作用，通俗点讲，选择该类型，jar包不会被加载进来，只是将该jar包的一些描述信息加载进来，使别的jar包在引用他时，能够看到一些相关的提示信息，仅此而已，所以说它只是个占位符，只要记住它的jar包不会被加载进来.

**optional：**标记依赖是否可选。默认值false

比如struts2中内置了log4j这个记录日志的功能，就是将log4j内嵌入struts2的jar包中，而struts2有没有log4j都没关系，有它，提示的信息更多，没它，也能够运行，只是提示的信息相对而言少一些。这个时候就可以对它进行可选操作，想要它就将optionl设置为true，不想要就设置为false。

**exclusions：**排除传递依赖，解决jar冲突问题

依赖传递的意思就是，A项目依赖B项目，B项目依赖C项目，当使用A项目时，会把B也加载进来，这是传递依赖，依次类推，C也会因此给加载进来。

依赖传递有好处，也有坏处，坏处就是jar包的冲突问题，比如，A 依赖 B(B的版本为1)，C 依赖 B(B的版本为2)，如果一个项目同时需要A和C，那么A,C都会传递依赖将B给加载进来，问题就在这里，两个B的版本不一样，将两个都加载进去就会引起冲突，这时候就需要使用exclusions这个属性配置了。maven也会有一个机制避免两个都加载进去，**maven 默认配置在前面的优先使用**，但还是需要使用exclusions来配置更合理。

<exclusions>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

</exclusion>

</exclusions>

上面就表示在引入一个jar包时，将该jar包依赖的其他jar包排除.

**scope：**依赖范围，意思就是通过pom.xml加载进来的jar包，来什么范围内使用生效，范围包括编译时、运行时、测试时。默认值是compile，表示在编译、测试、运行时都有效.



**依赖调节原则：**

maven解决传递依赖时jar包冲突问题的方法：

1、第一原则：路径近者优先原则

A-->B-->C-->D-->X(1.6)

E-->D-->X(2.0)

将使用X(2.0)，因为其路径更近

2、第二原则：第一声明优先原则。就是如果路径相同，maven默认配置在前面的优先使用

A-->B --> X(1.6)

C-->D --> X(2.0)

这样就是路径相同，那么如果A在前面，C在后面，则使用X(1.6)，maven会先根据第一原则进行选择，第一原则不成，则按第二原则处理。