maven

maven是基于项目对象模型(POM),可以通过一小段描述信息来管理项目的构建，报告和文档的软件项目管理工具
管理项目的一个仓库

maven源码:
bin目录中包含了mvn运行的脚本
conf是一些配置文件
lib包含了一些类库
boot包含一个类加载框架，maven使用类加载器加载类库

标准的maven-demo结构
```
maven-demo
|– pom.xml
`– src
    |– main
    | `– java
    |         `– org
    |             `– flyne
    |                 `– demo
    |                     `– App.java
    `– test
        `– java
            `– org
                `– flyne
                    `– demo
                        `– AppTest.java
```

POM文件
pom.xml文件是一个Maven项目的核心配置文件，该配置文件包含了大部分的构建一个项目所需要的信息。

maven的常用命令
mvn -v 查看maven版本
compile 编译
test 测试（maven本身并没有什么功能，它的主要功能都是依靠插件来完成的，就像eclipse，如果需要使 eclipse具有开发c/c++能力的话，就要下载cdt和其编译环境，对于junit而言在我们在第一次使用mvn test 命令对项目进行测试时会去中央仓库中自动下载一个名为maven-surefire-plugin的插件，然后在**它会在src/test/java/目录下去查找，是否有Test开头或结束的类，如果找到则默认执行所有测试用例并生成测试报告，如果想指定对单独方法进行测试可以使用类似mvn test -Dtest=HelloTest ,该命令支持通配符，**也就是mvn tes -Dtest=Hello*Test,这样就会运行测试以Hello开头和Test结尾的所有测试用例， 也可以用逗号分隔一次执行多个测试用例， 具体的扩展使用可以查看官方文档。）
package 打包

clean 删除target
install 安装jar包到本地仓库中

创建目录的俩种方式：
1 archetype:generate 按照提示进行选择
2 archetype:generate -DgrounID=组织名，公司网址的反写，项目名
-DartifactId=项目名-模块名
-Dversion=版本号
-Dpackage=代码存在的包

maven中的坐标和仓库
坐标 构件（构件通过坐标作为其唯一标识）

仓库：
本地仓库和远程仓库

仓库：放置各种依赖的地方
C:\apache-maven-3.3.9\lib\maven-model-builder-3.3.9\org\apache\maven\model\pom-4.0.0.xml——Maven为我们提供的超级pom我们所有的pom都会记录这个pom这里记录的有全球的中央仓库，**中央仓库中放置了几乎所有开源项目的资源——jar包**
镜像仓库：Maven的中央服务器都是放置在国外的，有时我们不能访问外网，国内也有他的镜像仓库，这样就能更好的访问了
C:\apache-maven-3.3.9\conf\settings.xml这个文件中就配置着镜像的信息，可以配置国内的镜像文件的路径，这样就能更快更好的访问了

maven从远程中央库下载的资源默认放在了C:\Users\Administrator\.m2\repository这个目录，为了安全性，一般不会选择放在c盘下可以修改本地仓库的目录，指定到更安全的目录下，另外，setting.xml这个文件也需要另存一份，**这是为了更新版本的时候不用重新修改这个配置文件**

我的的默认项目目录是在
D:\WorkSpace\warehouse

maven的生命周期和插件

clean 。compile。test。package。install

完整的项目构建过程包括：
清理，编译，测试，打包，集成测试，验证，部署
maven生命周期
clean 清理项目
default 构建项目
site 生成项目的站点

pre -clean执行清理前的工作
clean清理上一次构建生成的所有文件
post-clean 执行清理后的文件
default 构建项目（最核心） 都属于 compile。test。package。install
site 生成项目站点的
pre -site 在生成项目站点前要完成的工作
site 生成项目的站点文档
post -site 在生成项目站点后要完成的工作
site -deploy 发布生成的站点到服务器上
maven 只是实现java的

pom.xml解析
pom详解
http://blog.csdn.net/ithomer/article/details/9332071

这是基本的坐标信息
<groupId>反写的公司网址+项目名
<artifactId>项目名+模块名
<version>当前项目的版本号(第一个0表示大版本号，第二个0表示分之版本号，第三个0表示小版本号)
SNAPSHOP 快照版
alpha 内部测试
beta 公测
Release 稳定版
GA 正式发布
<packaging> 默认是jar 还可以打包为其他类型 war zip pom

<name>项目描述名
<url>是项目的地址
<description>项目描述
<developers>开发人员列表
<licenses>许可证信息
<organization>组织信息

依赖列表
<dependencies><dependency>
<grounpID>
<artifactId>
<version>
<type>
<scope>
<optional>设置依赖是否可选
<exclusions>排除依赖传递列表

<dependencyManagement>依赖的管理
<build>对我们的行为提供相应的支持
<plugins>插件列表
<plugin>插件
<grounpID>
<artifactId>
</plugins>
</bulid>

<pareent>对父模块的继承

<modules > 在这里定义多个多个模块，可以一起编译<module>

maven经常涉及到一个词汇就是多模块开发 就是涉及到这些父模块的基础以及模块 还有依赖管理

http://www.blogjava.net/fancydeepin/archive/2015/06/27/maven-modules.html

Dependency Scope 6 种范围

依赖范围（classPath：编译，运行，测试）

1.compile：默认范围，编译测试运行都有效
2.provided：在编译和测试时有效
3.runtime：在测试和运行时有效
4.test:只在测试时有效
5.system:在编译和测试时有效，与本机系统关联，可移植性差
6.import:导入的范围，它只使用在dependencyManagement中，表示从其他的pom中导入dependency的配置

依赖有传递性，
A依赖B，B依赖C，那么A也依赖C。如果要A不依赖C，则要用exclusions标签来排除依赖。

依赖冲突
如果A与B采用了相同的构件，但是构件的版本不一样，我们该如何采用哪个构件呢
1 短路优先
A——B——C——X（jar）
A——D——X（jar）
2 先声明先优先
在pom中**依赖列表中，谁先声明就依赖谁，这个是在短路优先不管用的情况下，其次用的**

聚合和继承，是一次执行install（聚合）
如果我们想一次构建多个项目模块，那我们就需要对多个项目模块进行聚合

```
1<modules>2<module>模块一</module>3<module>模块二</module>4<module>模块三</module>5</modules>
```

例如：对项目的Hello、HelloFriend、MakeFriends这三个模块进行聚合

```
1<modules>2<module>../Hello</module>3<module>../HelloFriend</module>4<module>../MakeFriends</module>5</modules>
```

其中module的路径为相对路径。

继承为了消除重复，我们把很多相同的配置提取出来，例如：grouptId，version等 主要用的是dependencyManagement这个标签进行管理，在管理层（就是上一层 pom）：记住package改为：pom

子pom(下一层）通过<parent>标签然后引用父pom的坐标

项目管理利器（Maven）——聚合
1.聚合：如果项目D依赖项目C，项目C依赖项目B，项目B依赖项目A，我们需要一个个安装这项项目，在Maven中有一种方式可以将多个项目一次性安装，这就是聚合的概念。简单讲就是，需要人工多次操作的，只要MAVEN能理解，一次性告诉他，他就能帮我们做这件单调烦人的事情了。——使用<modules></modules>这个标签。
2.继承：多次使用到的依赖，比如：单元测试，没有必要在所有的项目中都引用一下，此时就可以采用继承的方式来实现，先来一个父级的POM.XML然后再继承此POM.XML

用pom.xml来管理我们构建的依赖，还有管理项目构建过程

方便的第三方框架的管理和便捷的项目构建过程可以大大提高效率，这就是构建工具的意义

<localRepository>d:/m2/repository</localRepository>改变默认仓库路径
表示本地仓库的地址为：d:/m2/repository。

maven 日用三板斧

mvn archetype:generate 创建maven项目

mvn package 打包，上面已经介绍过了

mvn package -Prelease打包，并生成部署用的包，比如deploy/*.tgz

mvn install 打包并安装到本地库

mvn eclipse:eclipse 生成eclipse项目文件

mvn eclipse:clean 清除eclipse项目文件

mvn site 生成项目相关信息的网站

maven插件常用参数

mvn -Dwtpversion=2.0 指定maven版本

mvn -Dmaven.test.skip=true 如果命令包含了test phase，则忽略单元测试

mvn -DuserProp=filePath 指定用户自定义配置文件位置

mvn -DdownloadSources=true -Declipse.addVersionToProjectName=true eclipse:eclipse 生成eclipse项目文件，尝试从仓库下载源代码，并且生成的项目包含模块版本（注意如果使用公用POM，上述的开关缺省已打开）

maven简单故障排除

mvn -Dsurefire.useFile=false如果执行单元测试出错，用该命令可以在console输出失败的单元测试及相关信息

set MAVEN_OPTS=-Xmx512m -XX:MaxPermSize=256m 调大jvm内存和持久代，maven/jvm out of memory error

mvn -X maven log level设定为debug在运行

mvndebug 运行jpda允许remote debug

mvn --help 这个就不说了。。