发布jar到本地仓库

mvn install:install-file -DgroupId=org.eweb4j -DartifactId=fel -Dversion=0.9 -Dpackaging=jar -Dfile=\downloads\\fel-all-0.9\\fel-all-0.9.jar

maven将工程打包成一个jar，连同依赖的jar包

<plugins>  
<plugin>  
<groupId>org.apache.maven.plugins</groupId>  
<artifactId>maven-assembly-plugin</artifactId>  
<version>2.5.5</version>  
<configuration>  
<archive>  
<manifest>  
<mainClass>gmv.ProductAddToERP</mainClass>  
</manifest>  
</archive>  
<descriptorRefs>  
<descriptorRef>jar-with-dependencies</descriptorRef>  
</descriptorRefs>  
</configuration>  
<executions>  
<execution>  
<id>make-assembly</id>  
<phase>package</phase>  
<goals>  
<goal>single</goal>  
</goals>  
</execution>  
</executions>  
</plugin>  
</plugins>

2、maven 打包javadoc包，和source包

<plugin>  
<groupId>org.apache.maven.plugins</groupId>  
<artifactId>maven-javadoc-plugin</artifactId>  
<version>2.10.4</version>  
<configuration>  
<encoding>UTF-8</encoding>  
<aggregate>true</aggregate>  
<charset>UTF-8</charset>  
<docencoding>UTF-8</docencoding>  
</configuration>  
<executions>  
<execution>  
<id>attach-javadocs</id>  
<goals>  
<goal>jar</goal>  
</goals>  
</execution>  
</executions>  
</plugin>  
<!--配置生成源码包-->  
<plugin>  
<groupId>org.apache.maven.plugins</groupId>  
<artifactId>maven-source-plugin</artifactId>  
<version>3.0.1</version>  
<executions>  
<execution>  
<id>attach-sources</id>  
<goals>  
<goal>jar</goal>  
</goals>  
</execution>  
</executions>  
</plugin>

mvn dependency:tree -Dverbose -Dincludes=commons-collections

查看自定jar的依赖情况用 include参数

mvn dependency:tree -Dincludes=:commons-collections

mvn dependency:tree -Dincludes=:commons-collections,:guava

mvn dependency:tree -Dverbose -Dincludes=asm:asm

3、maven命令默认是使用的C:\Users\WIN7\.m2\settings.xml

也可以如下指定使用的settings.xml文件：

mvn comple --settings C:\Users\WIN7\.m2\settings.xml

4、jar依赖检查插件

|  |
| --- |
| <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-enforcer-plugin</artifactId>  <executions>  <execution>  <id>enforce-versions</id>  <phase>validate</phase>  <goals>  <goal>enforce</goal>  </goals>  <configuration>  <rules>  <!-- 多次重复依赖 -->  <banDuplicatePomDependencyVersions/>  <!-- 禁止引用自定义一些比如不安全、不稳定的版本暂时没有定义-->  <bannedDependencies/>  <!-- 检测是否所有依赖统一版本 -->  <dependencyConvergence/>  <!--检测是否循环依赖 -->  <banCircularDependencies/>  </rules>  <fail>true</fail>  </configuration>  </execution>  </executions>  </plugin> |

跳过测试

-Dmaven.test.skip=true

讲解repository和mirror的关系

https://blog.csdn.net/nimeijian/article/details/65649463 setting文件分析

虽然mirrors可以配置多个子节点，但是它只会使用其中的一个节点，即**默认情况下配置多个mirror的情况下，只有第一个生效，**只有当前一个mirror

无法连接的时候，才会去找后一个；而我们想要的效果是：当a.jar在第一个mirror中不存在的时候，maven会去第二个mirror中查询下载，但是maven不会这样做！

**注意：**

 　　配置多个mirror时，mirrorOf不能配置" \* "，" \* " 的意思就是（根据mirrorOf和repository的id）匹配所有的仓库（repository），

这样就是说如果你需要某个jar，他会从镜像地址去下载这个jar。不管你配置了多少个库，即使这些库的地址不一样，仍然会从镜像地址访问