

- \* 学习目标

- \* 能够掌握使用common-fileupload做文件上传

- \* Servlet3.0之前

- \* 导包

- \* 客户端：

- \* post , enctype="multipart/form-data" , input-file

- \* 服务端

- \* DiskFileItemFactory

- \* ServletFileUpload

- \* parseRequest

- \* List<FileItem>

- \* item.getInputStream ---IO开发---IOUtils.copy(is,out);

- \* 能够掌握使用 Servlet3.0 新特性做文件上传

- \* request.getParts()

- \* List<Part>

- \* part.write();

- \* 能够理解maven的概述

- \* apache , 项目管理工具 , 项目构建 , 依赖的管理 , 项目信息

- \* 项目构建：maven-src- main - java

- resources

- target test java

- resources

- pom.xml

- \* pom.xml

- \* project-(modelVersion,groupId,artifactId,version

- \* 其他：package , dependencies ( dependency-scope ( compile , test , provided , runtime ) )

- \* 项目生命周期命令：mvn clean compile test package install deploy

- \* 能够掌握maven的开发

- \* 操作：5个案例

---

- \* 回顾

- \* Web监听器

- \* 8个：

- \* ServletContextListener,HttpSessionListener,ServletRequestListener

- \*

- ServletContextAttributeListener,HttpSessionAttributeListener,ServletRequestAttributeListener

- \* Bean --> HttpSessionBindingListener,序列化

- \* 实现监听器

- \* 配置：文件配置（xml），注解配置

- \* Ajax

- \* Asynchronou... JavaScript And Xml

- \* 与服务端交互数据--->json

- \* 局部刷新

- \* 开发方式

- \* 原生开发

- \* XMLHttpRequest--- open("GET",url,true)---send()---onreadystatechange--4,200

- \* JQ开发

- \* \$.get(url,callback)

- \* \$.post(url,data,callback)

- \* \$.ajax({type:"POST",url:murl.})

- \* 基于BootStrap的分页案例

- \* 服务端

- \* t\_user:id,username,password,批处理插入10007

\* 实体Bean:User,Result:code,msg

PageBean:接受客户端请求数据和响应

\* pageSize,pageNum,totalRecord,totalPage(totalRecord%pageSize==0?  
totalRecord/pageSize:totalRecord/pageSize+1)

\* list,startIndex

\* select \* from t\_user limit 0,10; --- (n=pageNum-1)\*pageSize

\* select \* from t\_user limit 10,10;

\* select \* from t\_user limit 20,10;

\* select \* from t\_user limit startIndex,pageSize

\* Dao

\* Service

\* Controller

\* Postman

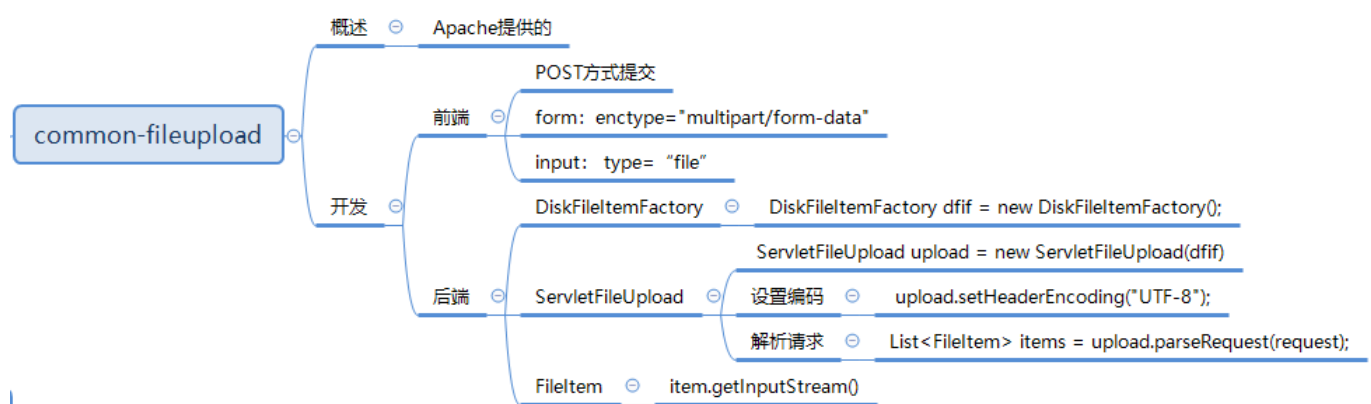
\* 客户端

\* Ajax

\* 动态表格创建

\* 分页插件：Jq\_paginator:不受css框架限定

\* 能够掌握使用common-fileupload做文件上传



1 \* 开发案例

2 \* 前端

```
3 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
4 <html>
5 <head>
6     <title>文件上传1</title>
7 </head>
8 <body>
9 <h1>文件上传</h1>
10 <form action="${pageContext.request.contextPath }/upload" method="post" enctype
11     <input type="file" name="file1"/><br>
12     <input type="submit" value="提交"/>
13 </form>
14 </body>
15 </html>
16 * 后端
17 @WebServlet("/upload")
18 public class UploadServlet1 extends HttpServlet {
19     @Override
20     protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
21         String realPath = req.getServletContext().getRealPath("/WEB-INF/upload"
22         File saveDir = new File(realPath);
23         if (!saveDir.exists() || !saveDir.isDirectory()) {
24             saveDir.mkdir();
25         }
26         // 2 构建DiskFileItemFactory
27         DiskFileItemFactory dfif = new DiskFileItemFactory();
28         // 3 构建ServletFileUpload
29         ServletFileUpload upload = new ServletFileUpload(dfif);
30         // 4 设置编码
31         upload.setHeaderEncoding("UTF-8");
32         // 6 通过请求获取FileItem的列表
33         InputStream is = null;
34         FileOutputStream out = null;
35         try {
36             List<FileItem> items = upload.parseRequest(req);
37             for (FileItem item : items) {
38                 // 得到上传的文件名称,
39                 String filename = item.getName();
40                 System.out.println(filename);
41                 if (filename == null || filename.trim().equals("")) {
42                     continue;
```

```

43         }
44         // 注意：不同的浏览器提交的文件名是不一样的，有些浏览器提交上来的文件
45         // 如： c:\a\b\1.txt，而有些只是单纯的文件名，如： 1.txt
46         // 处理获取到的上传文件的文件名的路径部分，只保留文件名部分
47         filename = filename.substring(filename.lastIndexOf("\\") + 1);
48         is = item.getInputStream();
49         out = new FileOutputStream(saveDir + "\\ " + filename);
50         IOUtils.copy(is, out, 1024 * 8);
51     }
52 } catch (FileUploadException e) {
53     e.printStackTrace();
54 } finally {
55     if (is != null) {
56         is.close();
57     }
58     if (out != null) {
59         out.close();
60     }
61 }
62
63 }
64
65 @Override
66 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
67     doGet(req, resp);
68 }
69 }
70
71

```

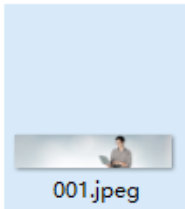
\* 效果图

← → ↺ ⓘ localhost:8080/lgweb02/upload1.jsp

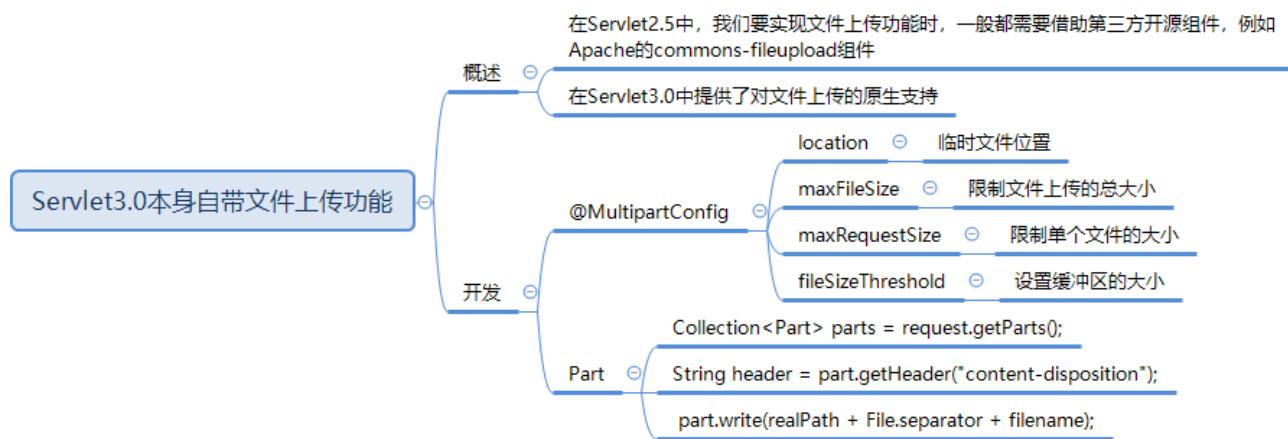
## 文件上传

选择文件 001.jpeg

提交



## \* 能够掌握使用 Servlet3.0 新特性做文件上传



```
1 * 开发案例
2 * 前端页面
3 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
4 <html>
5 <head>
6     <title>文件上传1</title>
7 </head>
8 <body>
9 <h1>文件上传</h1>
10 <form action="{pageContext.request.contextPath }/upload2" method="post" enctype="multipart/form-data">
11     <input type="file" name="file1"/><br>
12     <input type="submit" value="提交"/>
</form>
```

```

13 </form>
14 </body>
15 </html>
16 * 后端
17 @WebServlet("/upload2")
18 @MultipartConfig
19 public class UploadServlet2 extends HttpServlet {
20     @Override
21     protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
22         String realPath = req.getServletContext().getRealPath("/WEB-INF/upload"
23         File saveDir = new File(realPath);
24         if (!saveDir.exists() || !saveDir.isDirectory()) {
25             saveDir.mkdir();
26         }
27         Collection<Part> parts = req.getParts();
28         for (Part part : parts) {
29             String header = part.getHeader("content-disposition");
30             System.out.println(header);
31             String filename = getFileName(header);
32             System.out.println(filename);
33             part.write(realPath + File.separator + filename);
34         }
35         PrintWriter out = resp.getWriter();
36         out.println("上传成功");
37         out.flush();
38         out.close();
39
40     }
41     /**
42      * 根据请求头解析出文件名 请求头的格式：火狐和google浏览器下：form-data; name="f
43      * filename="snmp4j--api.zip" IE浏览器下：form-data; name="file";
44      * filename="E:\snmp4j--api.zip"
45      *
46      * @param header
47      *         请求头
48      * @return 文件名
49      */
50     public String getFileName(String header) {
51         /**
52          * String[] tempArr1 = header.split(";");代码执行完之后，在不同的浏览器下，

```

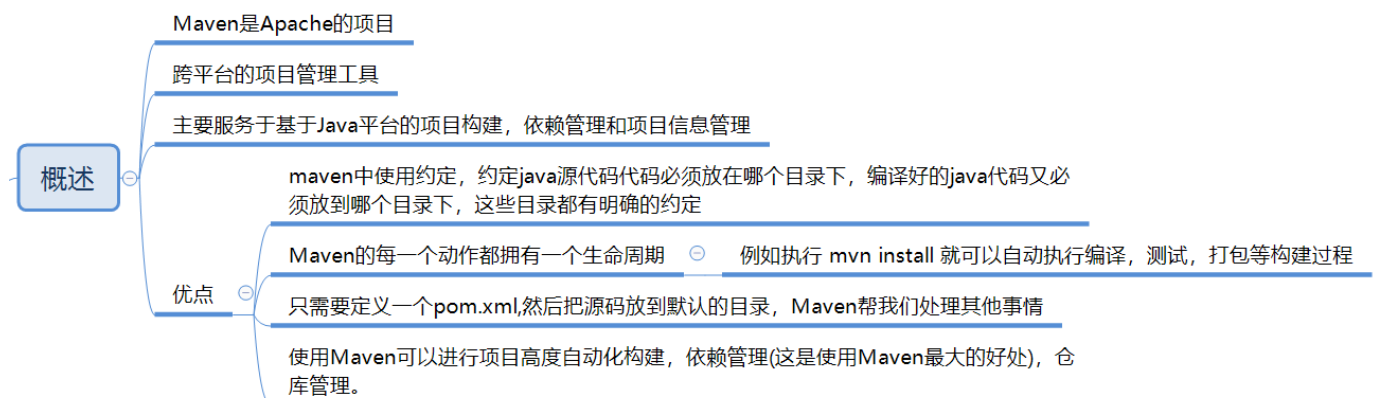
```

53      * 火狐或者google浏览器下: tempArr1={form-data,name="file",filename="snmp
54      * IE浏览器下: tempArr1={form-data,name="file",filename="E:\snmp4j--api.
55      */
56      String[] tempArr1 = header.split(";");
57      /**
58      * 火狐或者google浏览器下: tempArr2={filename,"snmp4j--api.zip"}
59      * IE浏览器下: tempArr2={filename,"E:\snmp4j--api.zip"}
60      */
61      String[] tempArr2 = tempArr1[2].split("=");
62      // 获取文件名, 兼容各种浏览器的写法
63      String fileName = tempArr2[1].substring(tempArr2[1].lastIndexOf("\\") +
64      return fileName;
65  }
66  @Override
67  protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
68      doGet(req, resp);
69  }
70 }
71
72

```

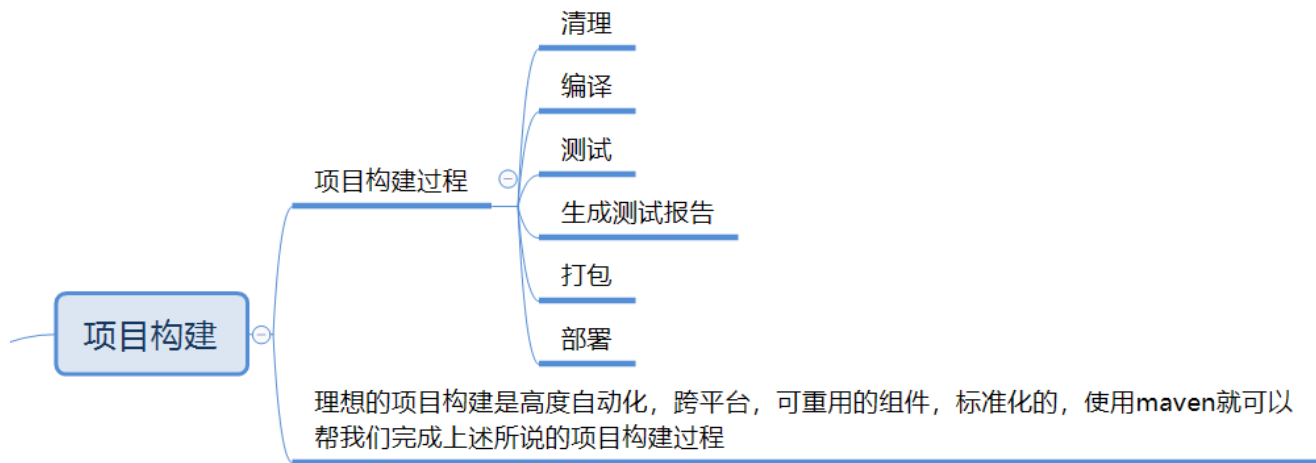
## \* 能够理解maven的概述

### \* 概述

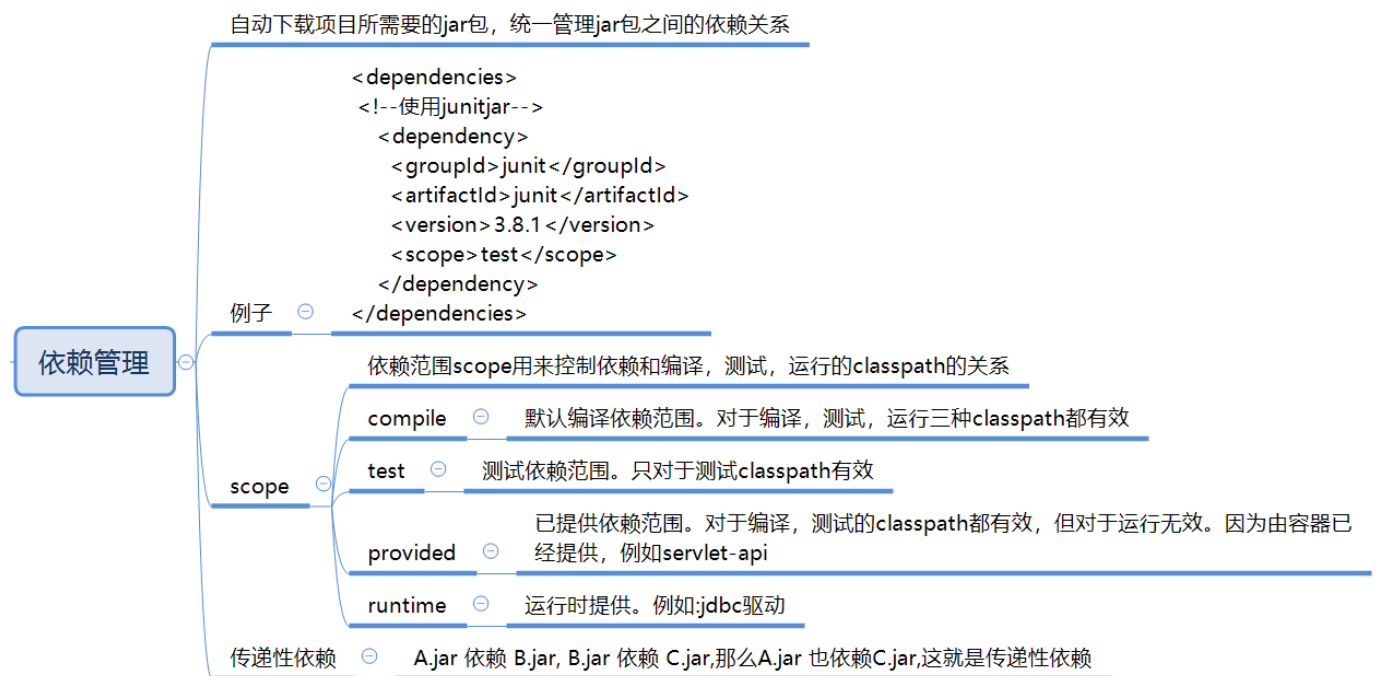


### \* 项目构建





## \* 依赖管理



## \* 能够掌握maven的开发



\* 下载

\* <http://maven.apache.org/download.cgi>

← → ↻ [maven.apache.org/download.cgi#](http://maven.apache.org/download.cgi#) 应用 开服务 网址安全导航 安全绿

Plugin Developer Centre  
Maven Central Repository  
Maven Developer Centre  
Books and Resources  
Security

COMMUNITY  
Community Overview  
Project Roles  
How to Contribute  
Getting Help  
Issue Management  
Getting Maven Source  
The Maven Team

PROJECT  
DOCUMENTATION  
Project Information

**Memory** No minimum requirement

**Disk** Approximately 10MB is required for the Maven installation itself. In addition to that, additional disk space will be used for your local Maven repository. The size of your local repository will vary depending on usage but expect at least 500MB.

**Operating System** No minimum requirement. Start up scripts are included as shell scripts and Windows batch files.

**Files**

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

	Link	Checksums	Signature
Binary tar.gz archive	<a href="#">apache-maven-3.6.0-bin.tar.gz</a>	<a href="#">apache-maven-3.6.0-bin.tar.gz.sha512</a>	<a href="#">apache-maven-3.6.0-bin.tar.gz.asc</a>
Binary zip archive	<a href="#">apache-maven-3.6.0-bin.zip</a>	<a href="#">apache-maven-3.6.0-bin.zip.sha512</a>	<a href="#">apache-maven-3.6.0-bin.zip.asc</a>
Source tar.gz archive	<a href="#">apache-maven-3.6.0-src.tar.gz</a>	<a href="#">apache-maven-3.6.0-src.tar.gz.sha512</a>	<a href="#">apache-maven-3.6.0-src.tar.gz.asc</a>
Source zip archive	<a href="#">apache-maven-3.6.0-src.zip</a>	<a href="#">apache-maven-3.6.0-src.zip.sha512</a>	<a href="#">apache-maven-3.6.0-src.zip.asc</a>

此电脑 > work (E:) > work666 > software >

名称	修改日期	类型	大小
apache-tomcat-7.0.91	2018/9/13 15:53	文件夹	
apache-tomcat-8.0.53	2018/6/29 10:44	文件夹	
iso	2018/8/10 9:26	文件夹	
Oracle	2018/8/10 14:26	文件夹	
客户端工具PLSQL+Developer8.04.1514绿色版	2015/9/6 21:53	文件夹	
apache-maven-3.6.0-bin.zip	2018/11/22 17:54	WinRAR ZIP 压缩文件	8,890 KB
apache-maven-3.6.0-src.zip	2018/11/22 17:54	WinRAR ZIP 压缩文件	4,462 KB
apache-tomcat-7.0.91.zip	2018/11/6 16:17	WinRAR ZIP 压缩文件	9,492 KB
apache-tomcat-7.0.91-src.zip	2018/11/6 16:17	WinRAR ZIP 压缩文件	7,223 KB
apache-tomcat-8.0.53.zip	2018/10/29 9:52	WinRAR ZIP 压缩文件	9,805 KB
apache-tomcat-8.0.53-src.zip	2018/10/29 9:52	WinRAR ZIP 压缩文件	7,850 KB
Hijson 2.1.2_jdk64.exe	2014/11/4 10:17	应用程序	3,451 KB
idealU-2018.3.exe	2018/11/22 9:23	应用程序	561,970 KB
scratch-offline@2014.5.6.exe	2018/10/6 23:35	应用程序	44,896 KB
staruml-5.0-with-cm.exe	2018/1/5 14:31	应用程序	22,194 KB
客户端工具PLSQL+Developer8.04.1514绿色...	2017/10/13 20:50	WinRAR 压缩文件	17,361 KB

## \* Maven的目录

此电脑 > work (E:) > work666 > software > apache-maven-3.6.0 >

名称	修改日期	类型	大小
bin	2018/10/24 20:43	文件夹	
boot	2018/10/24 20:43	文件夹	
conf	2018/10/24 20:38	文件夹	
lib	2018/10/24 20:43	文件夹	
LICENSE	2018/10/24 20:43	文件	14 KB
NOTICE	2018/10/24 20:43	文件	1 KB
README.txt	2018/10/24 20:38	文本文档	3 KB

### Maven目录

bin: 含有mvn运行的脚本

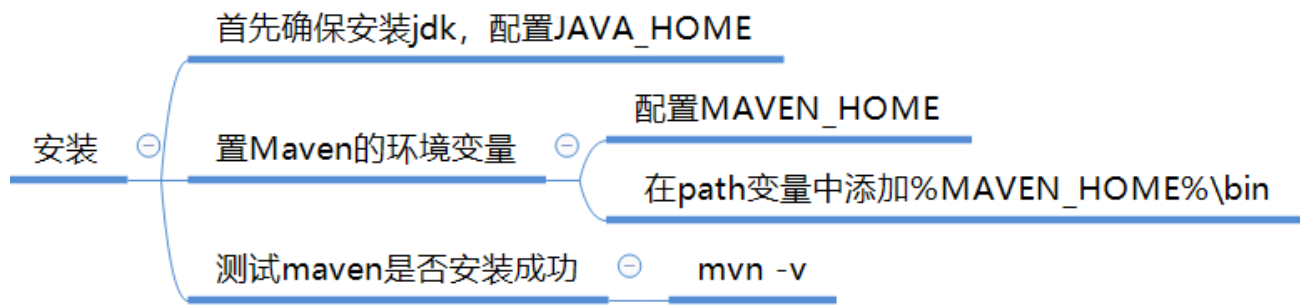
boot: 含有plexus-classworlds类加载器框架

conf: 含有settings.xml配置文件

lib: 含有Maven运行时所需要的java类库

LICENSE.txt, NOTICE.txt, README.txt针对Maven版本, 第三方软件等简要介绍

## \* 安装



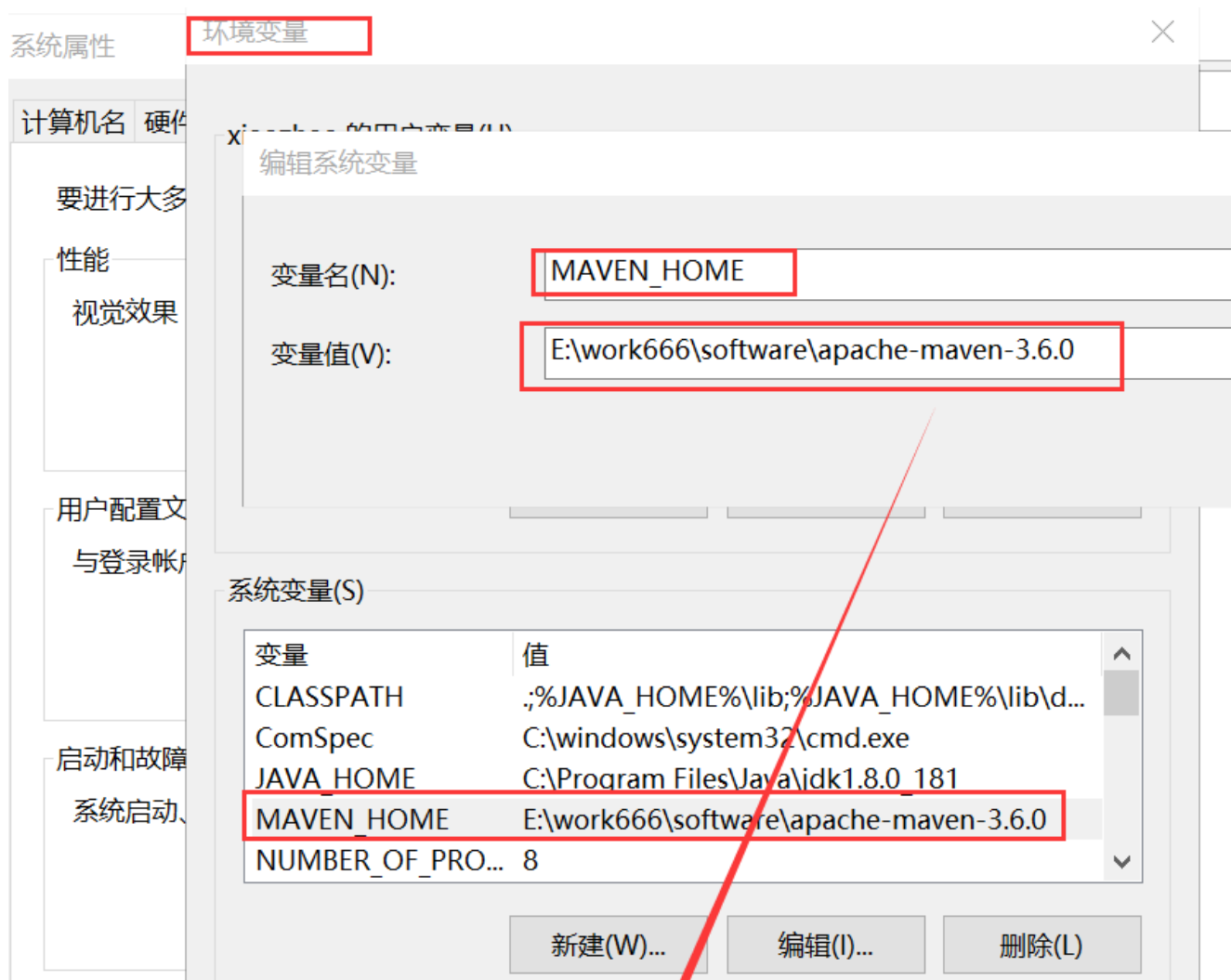
\* 首先确保安装jdk , 配置JAVA\_HOME

```
C:\Users\xiaozhao>echo %JAVA_HOME%
C:\Program Files\Java\jdk1.8.0_181

C:\Users\xiaozhao>java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)
```

\* 配置Maven的环境变量

\* 配置MAVEN\_HOME



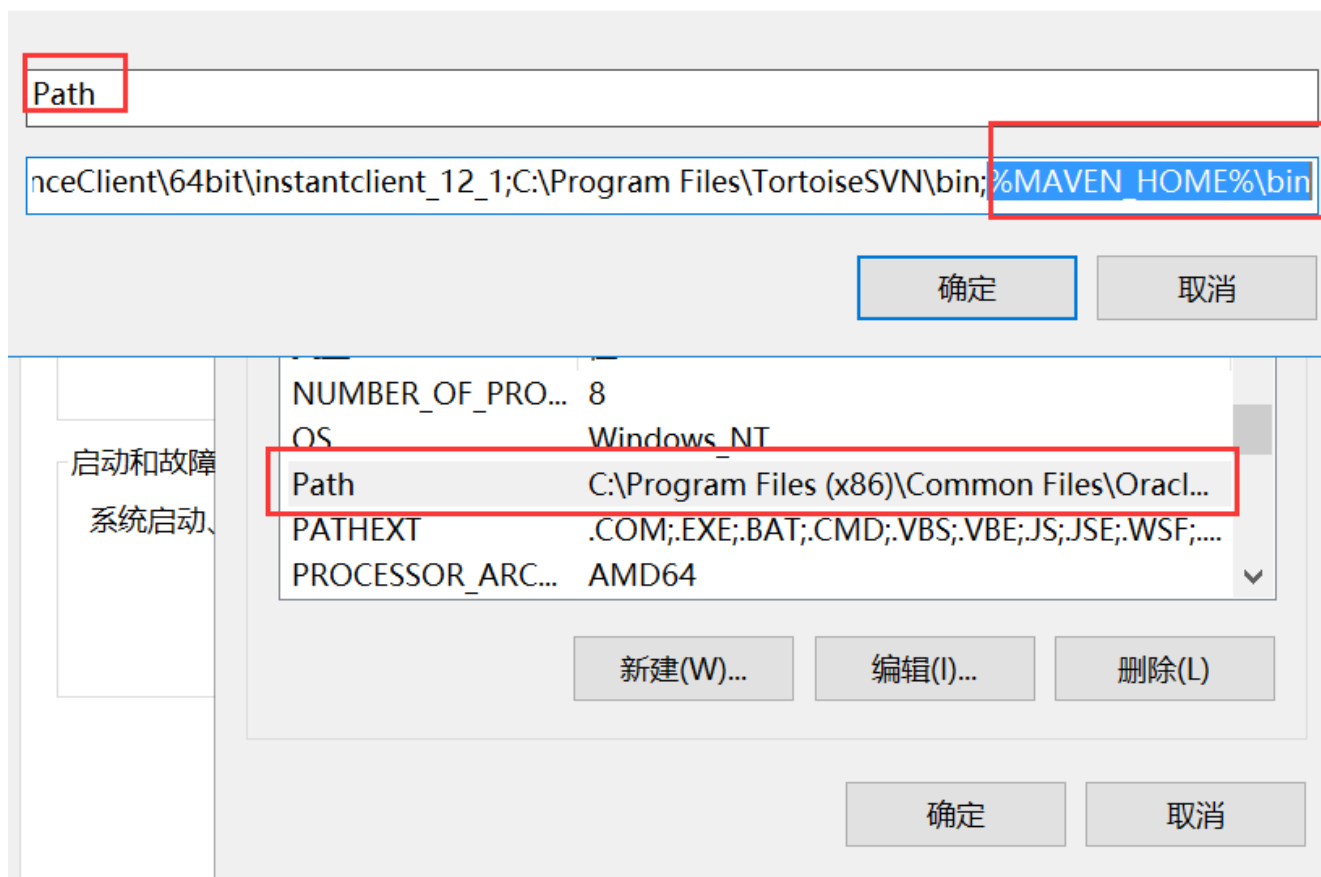
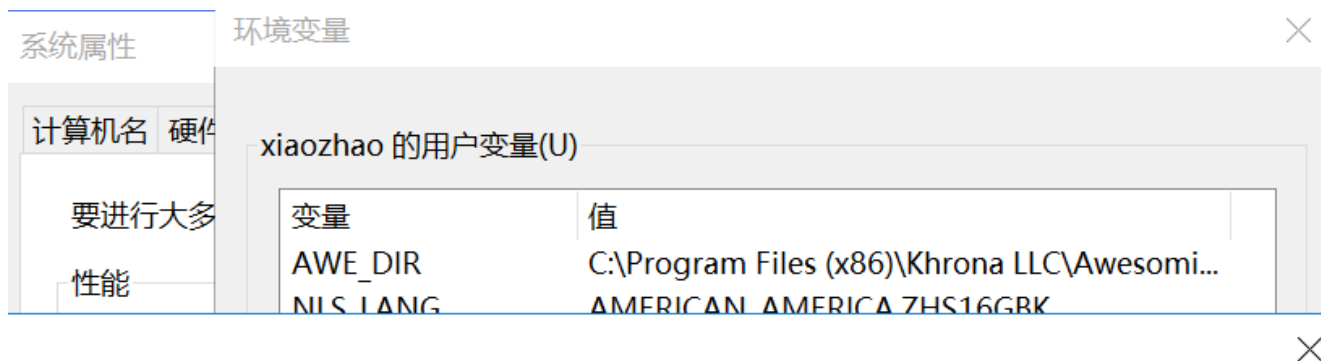
apache-maven-3.6.0

页 共享 查看

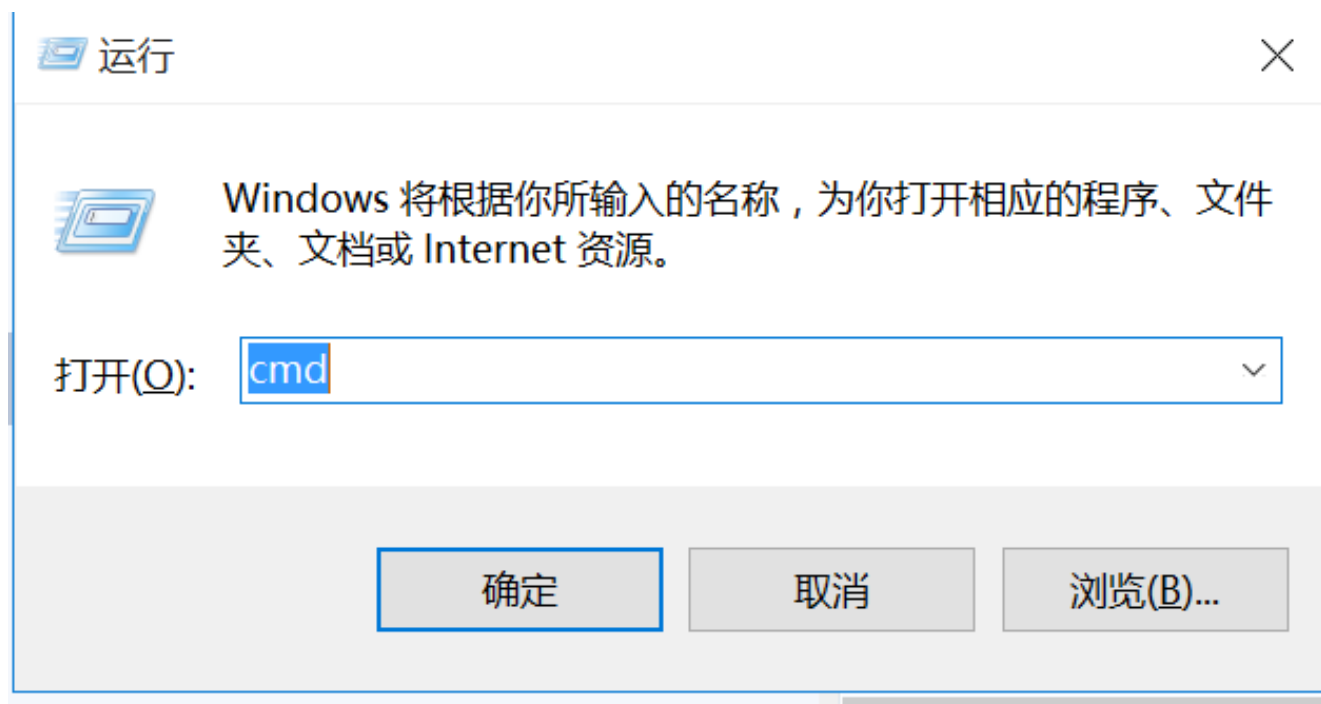
↑ E:\work666\software\apache-maven-3.6.0

名称	修改日期	类型
bin	2018/10/24 20:43	文件
boot	2018/10/24 20:43	文件
conf	2018/10/24 20:38	文件

\* 在path变量中添加%MAVEN\_HOME%\bin



\* 测试maven是否安装成功

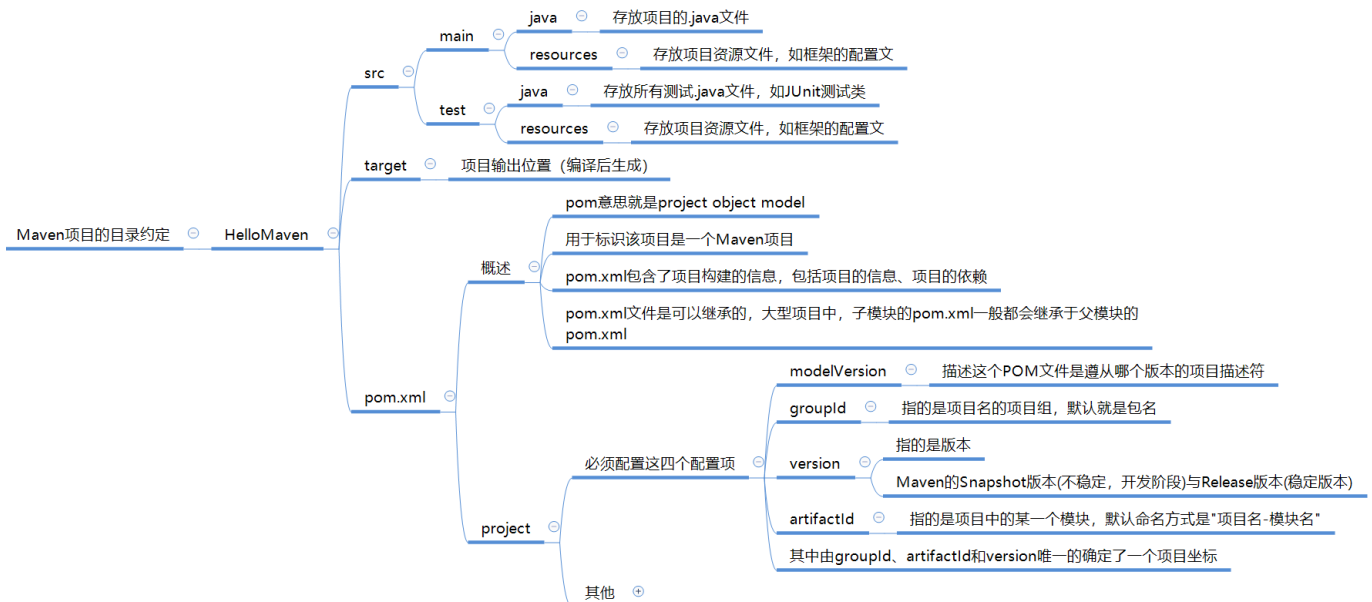


```

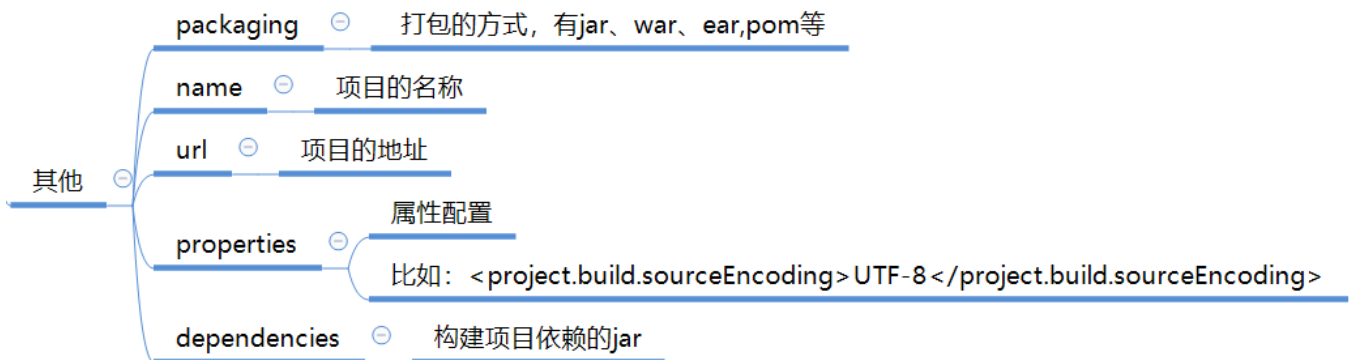
C:\Users\xiaozhao>mvn -v
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5fffb20918da4f719f3; 2018-10-25T02:41:47+08:00)
Maven home: E:\work666\software\apache-maven-3.6.0\bin\..
Java version: 1.8.0_181, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0_181\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

```

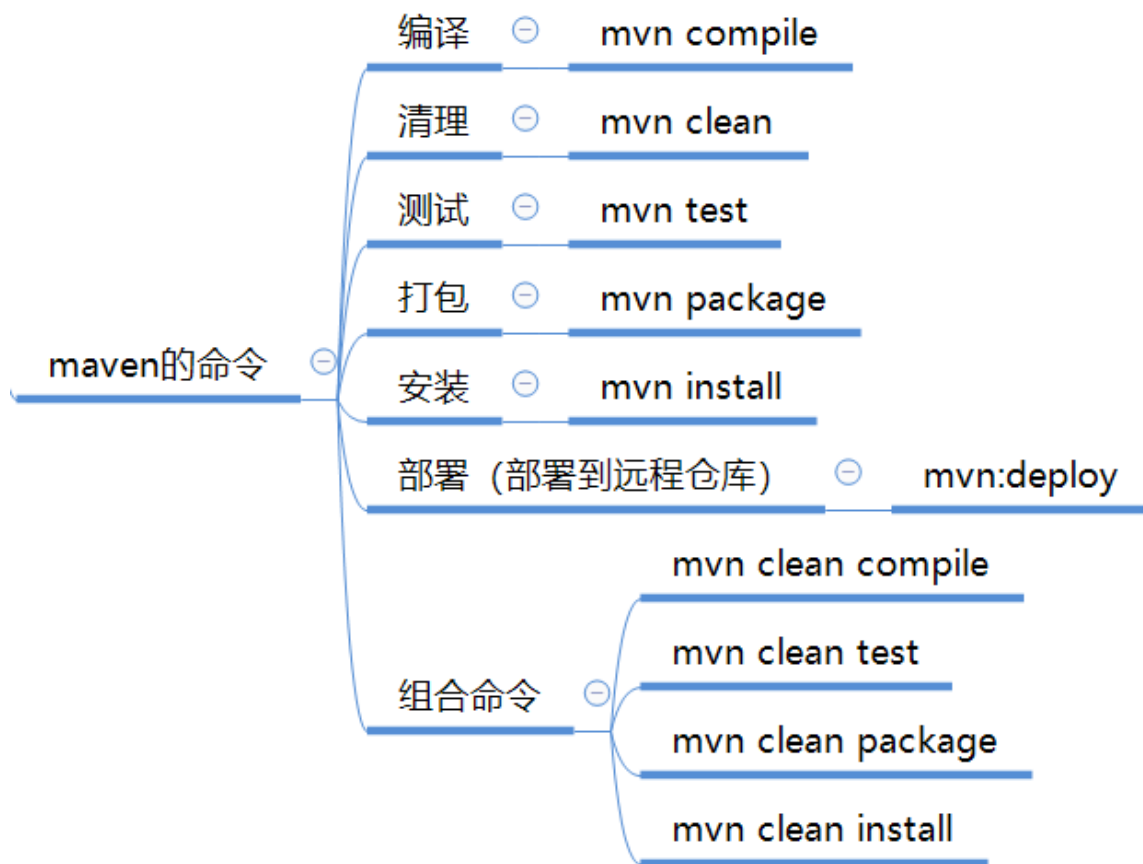
## \* Maven项目约定的目录



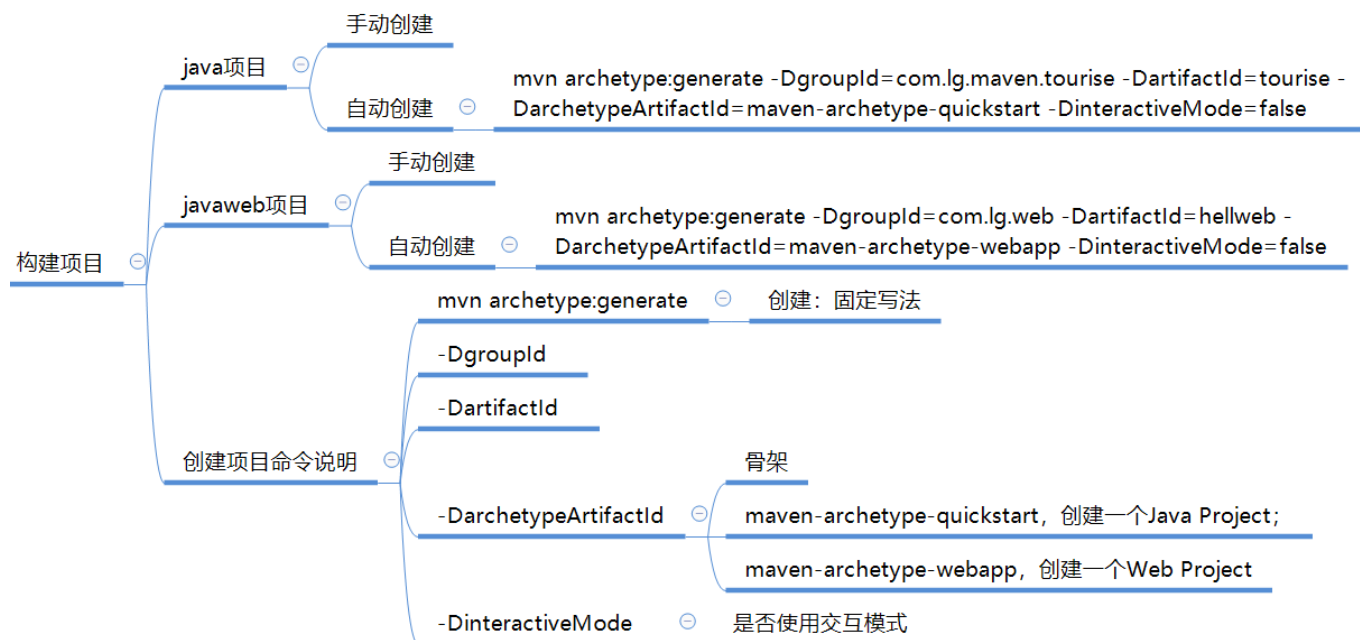
## \* Project 其他配置项



## \* maven的命令



## \* 构建项目



## \* 开发Maven项目



```
1 * 案例一
2 * 创建项目根文件夹，例如maven1
3 * 创建约定的目录
4 * 创建“pom.xml”文件
5 <?xml version="1.0" encoding="UTF-8"?>
6 <project xmlns="http://maven.apache.org/POM/4.0.0"
7   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
8   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
9     http://maven.apache.org/xsd/maven-4.0.0.xsd">
10   <!--所有的Maven项目都必须配置这四个配置项-->
11   <!--modelVersion 描述这个POM文件是遵从哪个版本的项目描述符-->
12   <modelVersion>4.0.0</modelVersion>
13   <!--groupId指的是项目名的项目组，默认就是包名-->
14   <groupId>com.lg.maven.hello</groupId>
15   <!--artifactId指的是项目中的某一个模块，默认命名方式是"项目名-模块名"-->
16   <artifactId>maven1</artifactId>
17   <!--version指的是版本，这里使用的是Maven的快照版本-->
18   <!--Maven的Snapshot版本(不稳定，开发阶段)与Release版本(稳定版本)-->
19   <version>SNAPSHOT-0.0.1</version>
20 </project>
21 * 在maven规范的目录下（src/main/java），创建HelloWorld.java文件
22 public class HelloWorld{
23     public static void main(String[] args){
24         System.out.println("HelloWorld");
25     }
26 }
27 * 编译
28     * 在cmd中进入maven项目
29     * mvn compile
30 * 观察：是否构建成功，是否生成target，默认仓库是否自动下载jar
31
32 * 案例二（使用单元测试）
33 * 建立maven2项目，同时建立Maven约定的目录结构和pom.xml文件
34 * pom.xml
35 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.c
36 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/>
37   <modelVersion>4.0.0</modelVersion>
38   <groupId>com.lg.maven.hello</groupId>
39   <artifactId>maven2</artifactId>
40   <version>0.0.1-SNAPSHOT</version>
```

```
41 <name>hellomaven</name>
42 <!--添加依赖的jar包-->
43 <dependencies>
44     <!--项目要使用到junit的jar包，所以在这里添加junit的jar包的依赖-->
45     <dependency>
46         <groupId>junit</groupId>
47         <artifactId>junit</artifactId>
48         <version>4.9</version>
49         <scope>compile</scope>
50     </dependency>
51 </dependencies>
52 </project>
53 * 代码
54 * 放到D:\wook666\java\maven2\src\main\java
55 public class Person {
56     public String sayHello(String name){
57         return "Hello "+name+"!";
58     }
59 }
60 * 放到D:\wook666\java\maven2\src\test\java
61 import org.junit.Test;
62 public class PersonTest {
63     @Test
64     public void testHello(){
65         Person person = new Person();
66         String results = person.sayHello("xiaohei");
67         System.out.println(results);
68     }
69 }
70 * 在测试前，更换仓库的位置
71 * D:\wook666\software\apache-maven-3.6.0\conf
72 * 打开settings.xml修改位置本地仓库的位置
73 * <localRepository>D:\repository</localRepository>
74 * 替换原来maven远程仓库（下载比较慢）
75 <mirror>
76     <id>alimaven</id>
77     <name>aliyun maven</name>
78     <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
79     <mirrorOf>central</mirrorOf>
80 </mirror>
```

```
81      * 观察下载jar包
82 * 测试(每个命令分别查看效果):
83      * mvn compile
84          * 在target会生成相应的class文件
85      * mvn clean
86          * 发现删除了target目录
87      * mvn test
88          * 在cmd中可以发现自动执行单元测试
89      * mvn package
90          * 在target目录: D:\wook666\java\maven2\target
91          * 可以发现jar的生成
92      * mvn install
93          * 在本地仓库目录D:\repository\com\lg\maven\hello\maven2\0.0.1-SNAPSHOT
94          * 可以发现jar的生成
95 * 组合命令测试(每个命令分别查看效果):
96      * mvn clean compile
97      * mvn clean test
98      * mvn clean package
99      * mvn clean install
100
101 * 案例三(使用自己在maven生成的jar)
102      * 建立maven3项目, 同时建立Maven约定的目录结构和pom.xml文件
103      * pom.xml 文件
104      <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.c
105 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/>
106      <modelVersion>4.0.0</modelVersion>
107      <groupId>com.lg.maven.hello3</groupId>
108      <artifactId>maven3</artifactId>
109      <version>0.0.1-SNAPSHOT</version>
110      <name>maven3</name>
111
112      <!--添加依赖的jar包-->
113      <dependencies>
114          <!--项目要使用到junit的jar包, 所以在这里添加junit的jar包的依赖-->
115          <dependency>
116              <groupId>junit</groupId>
117              <artifactId>junit</artifactId>
118              <version>4.9</version>
119              <scope>compile</scope>
120          </dependency>
```

```
121
122     <dependency>
123         <groupId>com.lg.maven.hello</groupId>
124         <artifactId>maven2</artifactId>
125         <version>0.0.1-SNAPSHOT</version>
126         <scope>compile</scope>
127     </dependency>
128 </dependencies>
129 </project>
130 * 代码
131 public class HelloPerson {
132     public String sayHelloToPerson(String name){
133         Person person = new Person();
134         String str = person.sayHello(name)+" I am "+this.getName();
135         return str;
136     }
137     public String getName(){
138         return "John";
139     }
140 }
141
142 import org.junit.Test;
143 public class HelloPersonTest {
144     @Test
145     public void testHelloPerson(){
146         HelloPerson helloPerson = new HelloPerson();
147         String results = helloPerson.sayHelloToPerson("xiaohai");
148         System.out.println(results);
149     }
150 }
151 * 测试
152     * mvn test 观察效果
153
154 * 案例四（自动构建java项目目录）
155     * 新建maven4目录
156     * mvn archetype:generate -DgroupId=com.lg.maven -DartifactId=shopping -DarchetypeGroupId=com.lg.maven.archetype -DarchetypeArtifactId=maven-archetype-quickstart
157     mvn archetype:generate
158     -DgroupId=com.lg.maven.shopping
159     -DartifactId=shopping
160     -DarchetypeArtifactId=maven-archetype-quickstart
```

```
161 -DinteractiveMode=false
162 * 测试
163 * mvn test
164 * mvn clean package
165 * java -cp D:\work666\java\maven4\shopping\target\shopping-1.0-SNAPSHOT.jar
166
167 * 案例五（自动构建java web项目目录）
168 * mvn archetype:generate -DgroupId=com.lg.web -DartifactId=helloweb -Darchetype
169 * Generating project in Batch mode: 在这里需要等待很久，等待下载需要的文件
170 * 之前命令加上参数 -X
171 * 发现在网络查找该文件:Searching for remote catalog: https://repo.maven.apache.org/maven2
172 * 这个文件有8.5M，由于访问网络网速就几KB
173 * 解决方案：
174 * 下载到本地放到D:\repository\org\apache\maven\archetype\archetype-catalog
175 * 执行的时候参数 -DarchetypeCatalog=local
176 * mvn archetype:generate -DgroupId=com.lg.web -DartifactId=helloweb -Darchetype
177
178 * 测试
179 * mvn package
180 * 在target 复制出war包，放到tomcat中
181 * 访问: http://localhost:8080/helloweb/index.jsp
```