* 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（complie，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

* Asynchrou... 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

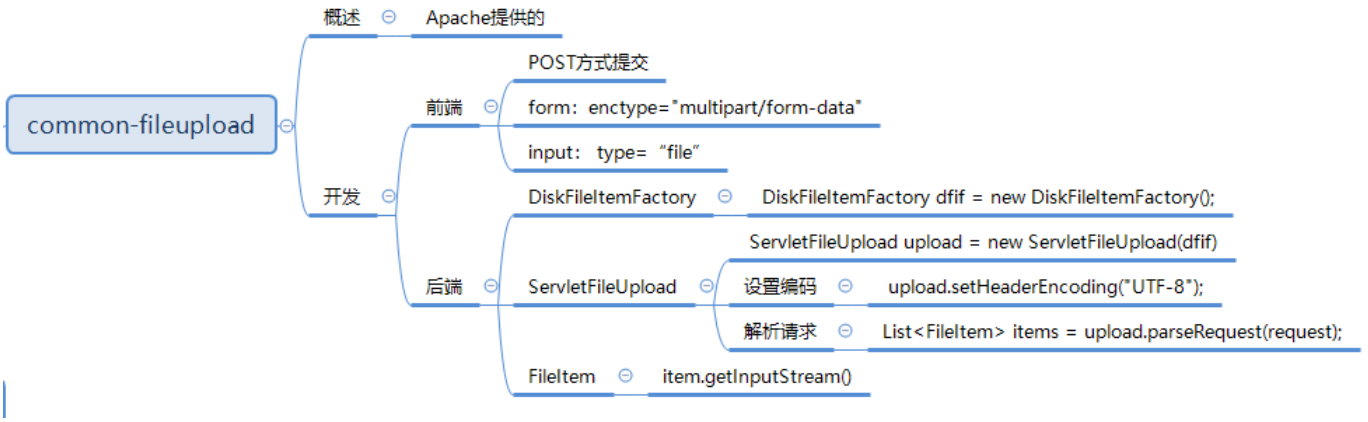
* 客户端

* 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




```
* 开发案例
  * 前端
```

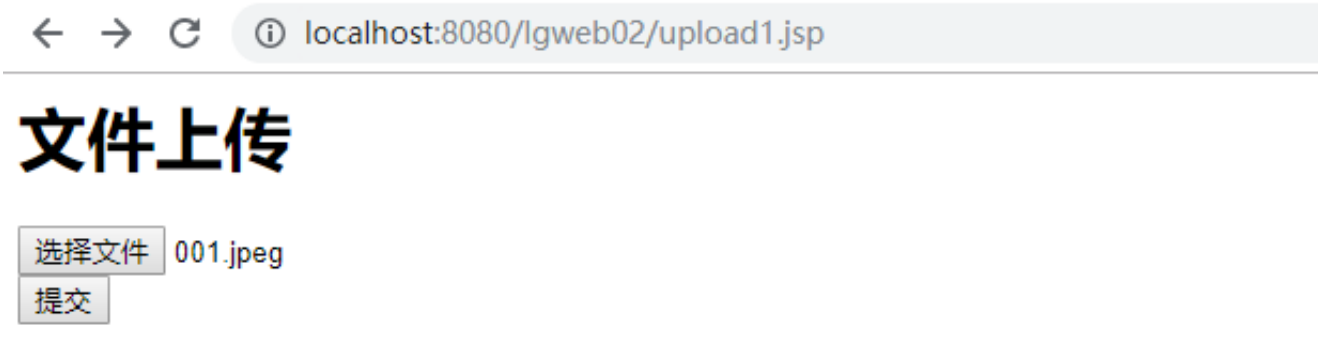
```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>文件上传1</title>
</head>
<body>
<h1>文件上传</h1>
<form action="${pageContext.request.contextPath }/upload" method="post" enctype
    <input type="file" name="file1"/><br>
    <input type="submit" value="提交"/>
</form>
</body>
</html>
```

  * 后端

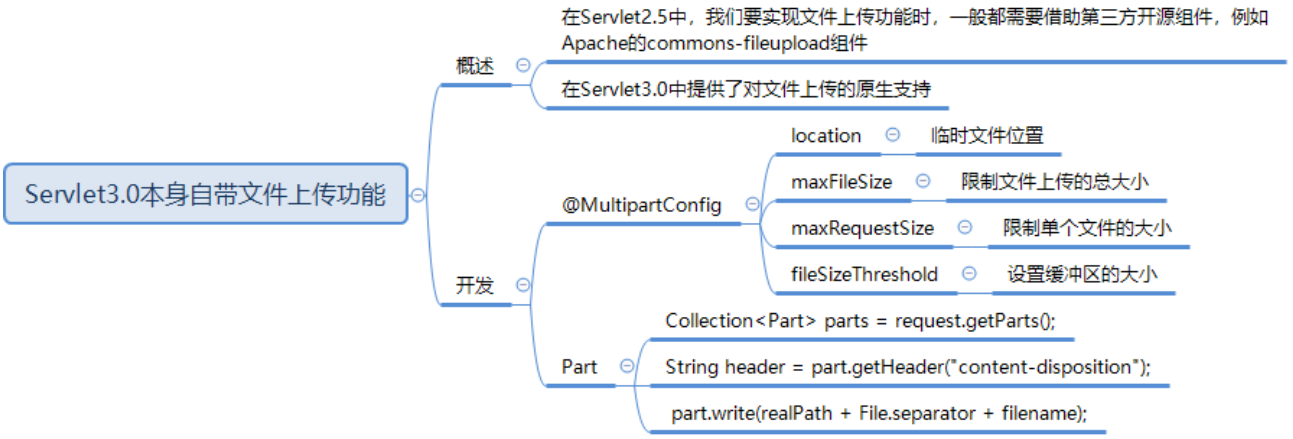
```java
@WebServlet("/upload")
public class UploadServlet1 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
        String realPath = req.getServletContext().getRealPath("/WEB-INF/upload"
        File saveDir = new File(realPath);
        if (!saveDir.exists() || !saveDir.isDirectory()) {
            saveDir.mkdir();
        }
        // 2 构建DiskFileItemFactory
        DiskFileItemFactory dfif = new DiskFileItemFactory();
        // 3 构建ServletFileUpload
        ServletFileUpload upload = new ServletFileUpload(dfif);
        // 4 设置编码
        upload.setHeaderEncoding("UTF-8");
        // 6 通过请求获取FileItem的列表
        InputStream is = null;
        FileOutputStream out = null;
        try {
            List<FileItem> items = upload.parseRequest(req);
            for (FileItem item : items) {
                // 得到上传的文件名称,
                String filename = item.getName();
                System.out.println(filename);
                if (filename == null || filename.trim().equals("")) {
                    continue;
```

```
                }
                // 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
                // 如：  c:\a\b\1.txt，而有些只是单纯的文件名，如：1.txt
                // 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
                filename = filename.substring(filename.lastIndexOf("\\") + 1);
                is = item.getInputStream();
                out = new FileOutputStream(saveDir + "\\" + filename);
                IOUtils.copy(is, out, 1024 * 8);
            }
        } catch (FileUploadException e) {
            e.printStackTrace();
        } finally {
            if (is != null) {
                is.close();
            }
            if (out != null) {
                out.close();
            }
        }

    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
        doGet(req, resp);
    }
}


```

* 效果图



* 能够掌握使用 Servlet3.0 新特性做文件上传



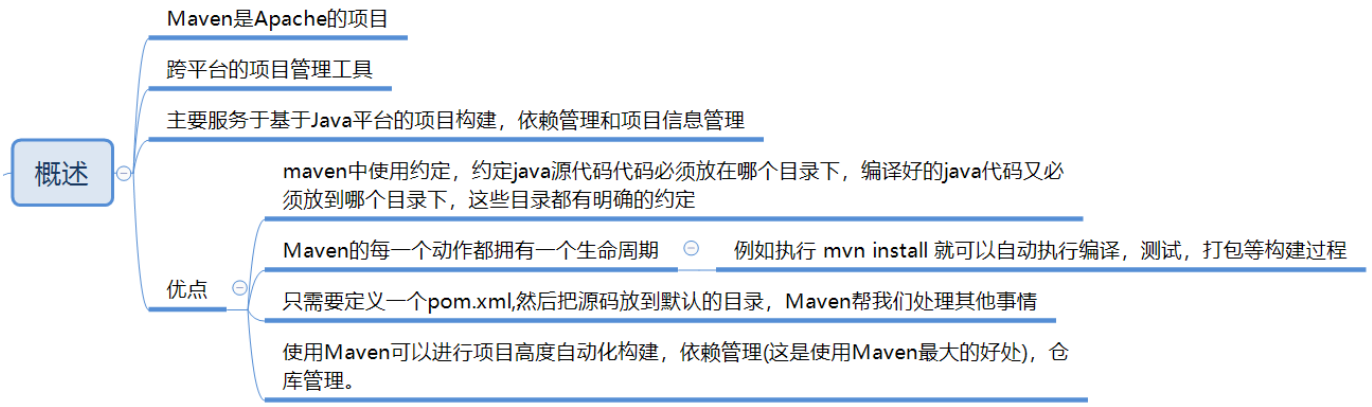

```
* 开发案例
 * 前端页面
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>文件上传1</title>
</head>
<body>
<h1>文件上传</h1>
<form action="${pageContext.request.contextPath }/upload2" method="post" enctyp
    <input type="file" name="file1"/><br>
    <input type="submit" value="提交"/>
```

```
</form>
</body>
</html>
 * 后端
@WebServlet("/upload2")
@MultipartConfig
public class UploadServlet2 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
        String realPath = req.getServletContext().getRealPath("/WEB-INF/upload"
        File saveDir = new File(realPath);
        if (!saveDir.exists() || !saveDir.isDirectory()) {
            saveDir.mkdir();
        }
        Collection<Part> parts = req.getParts();
        for (Part part : parts) {
            String header = part.getHeader("content-disposition");
            System.out.println(header);
            String filename = getFileName(header);
            System.out.println(filename);
            part.write(realPath + File.separator + filename);
        }
        PrintWriter out = resp.getWriter();
        out.println("上传成功");
        out.flush();
        out.close();

    }
    /**
     * 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
     * filename="snmp4j--api.zip" IE浏览器下：form-data; name="file";
     * filename="E:\snmp4j--api.zip"
     *
     * @param header
     *               请求头
     * @return 文件名
     */
    public String getFileName(String header) {
        /**
         * String[] tempArr1 = header.split(";");代码执行完之后，在不同的浏览器下，
```

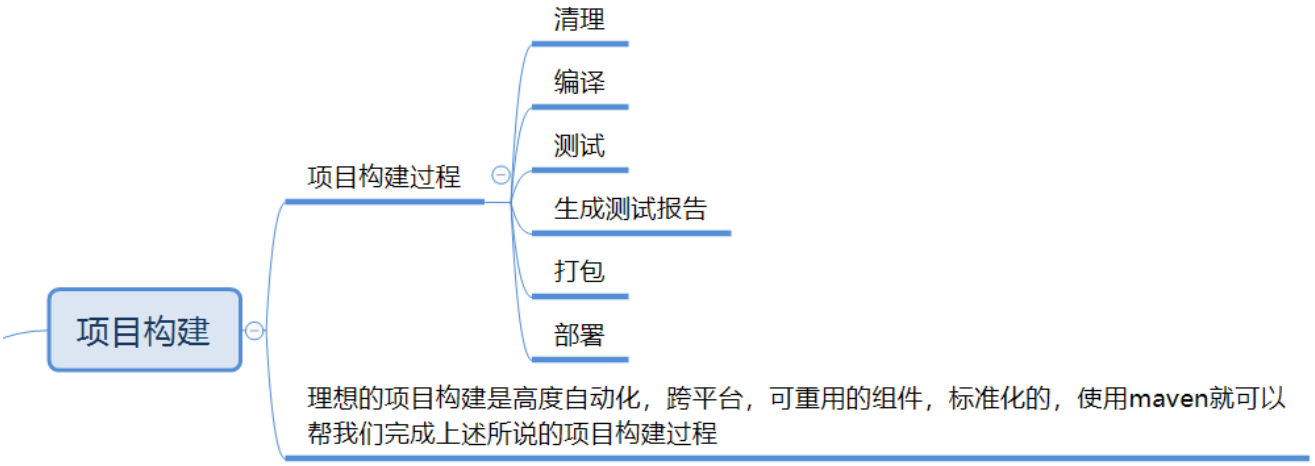
```
53          * 火狐或者google浏览器下：tempArr1={form-data,name="file",filename="snmp
54          * IE浏览器下：tempArr1={form-data,name="file",filename="E:\snmp4j--api.
55          */
56         String[] tempArr1 = header.split(";");
57         /**
58          * 火狐或者google浏览器下：tempArr2={filename,"snmp4j--api.zip"}
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) thr
68         doGet(req, resp);
69     }
70 }
71
72
```

* 能够理解maven的概述

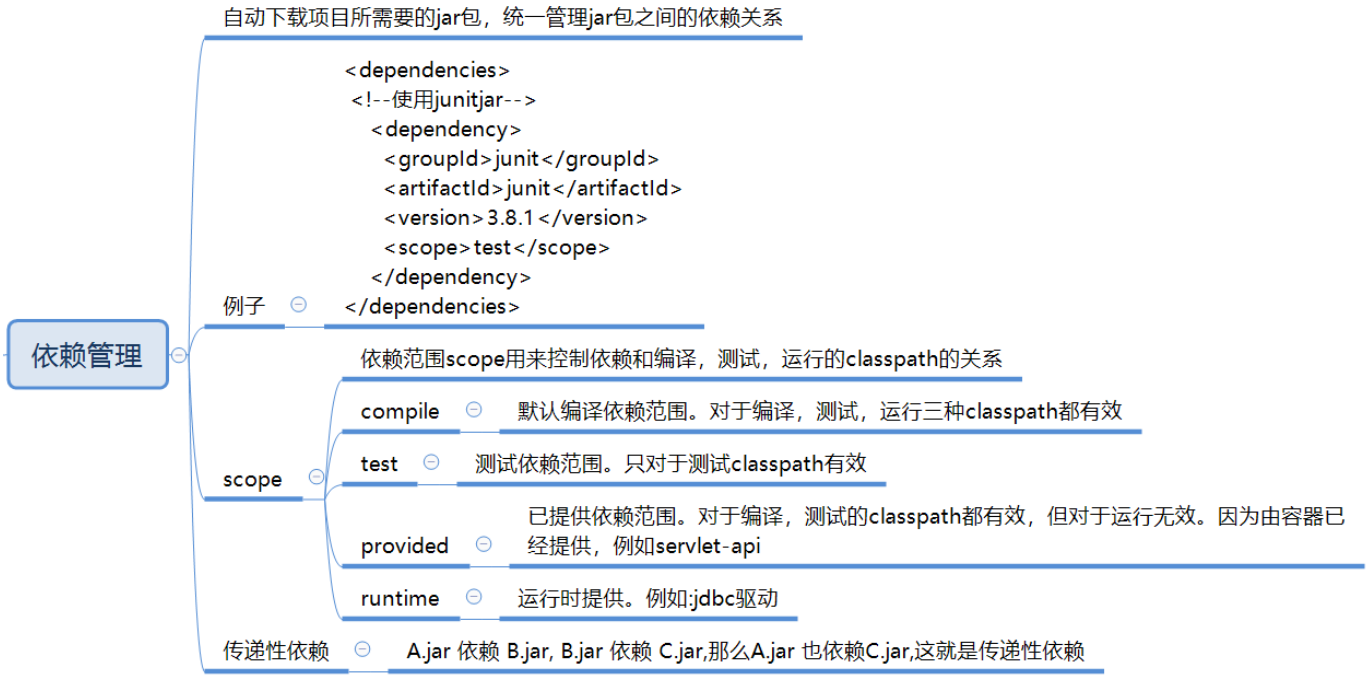
  * 概述



  * 项目构建



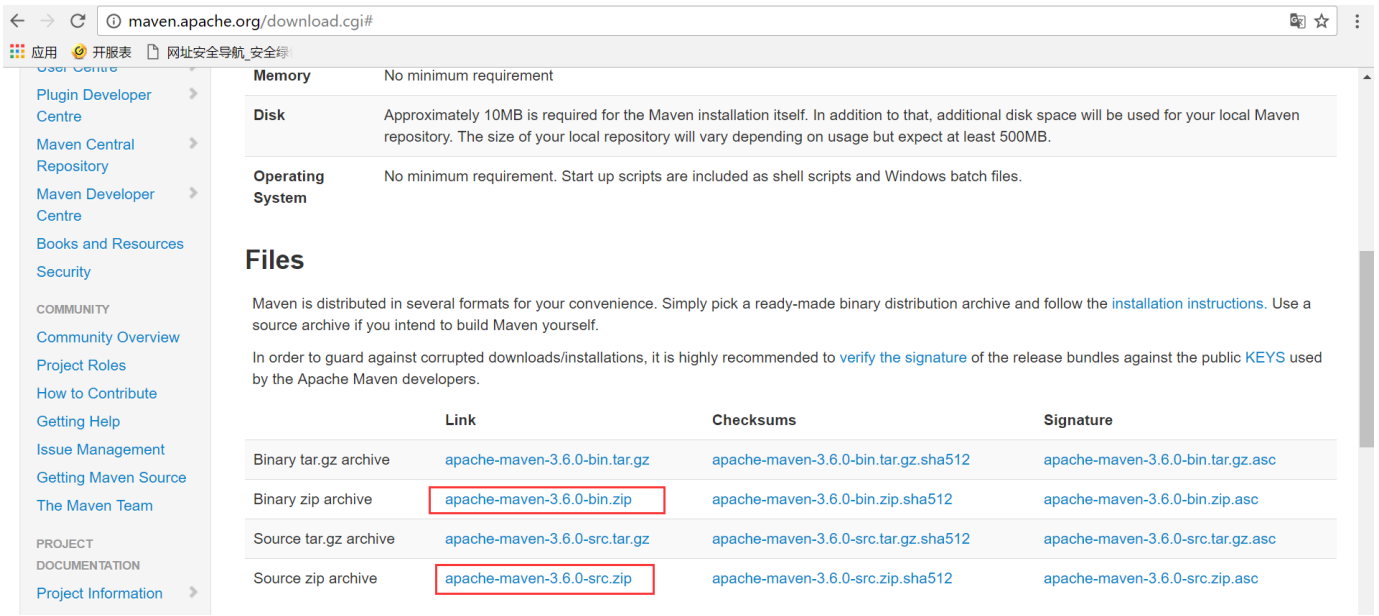
* 依赖管理

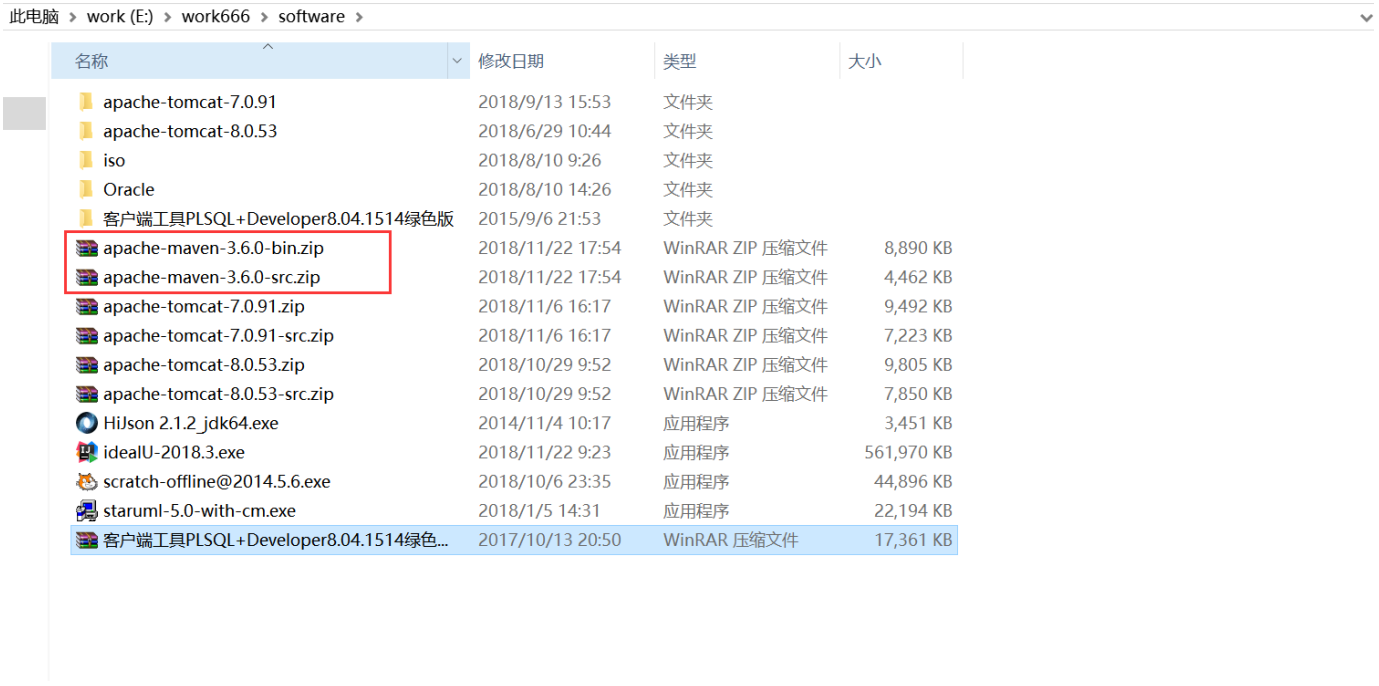


* 能够掌握maven的开发

* 下载

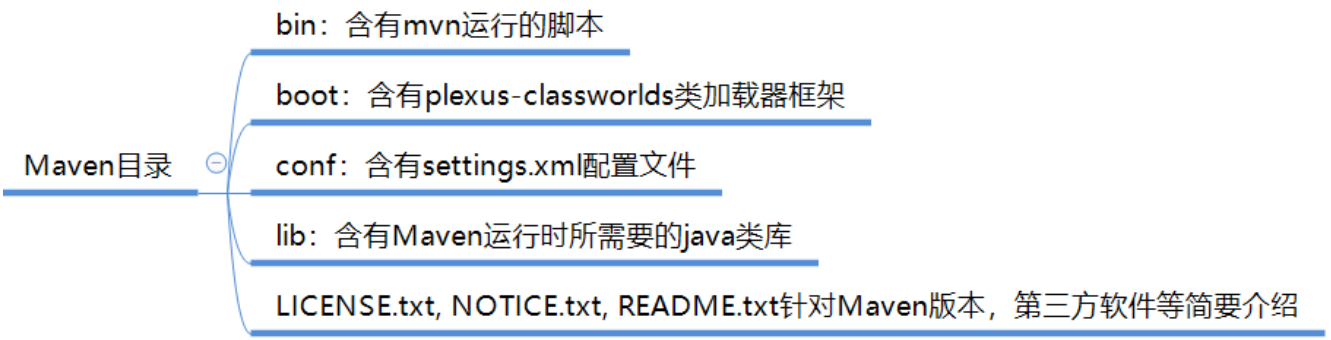
  * http://maven.apache.org/download.cgi



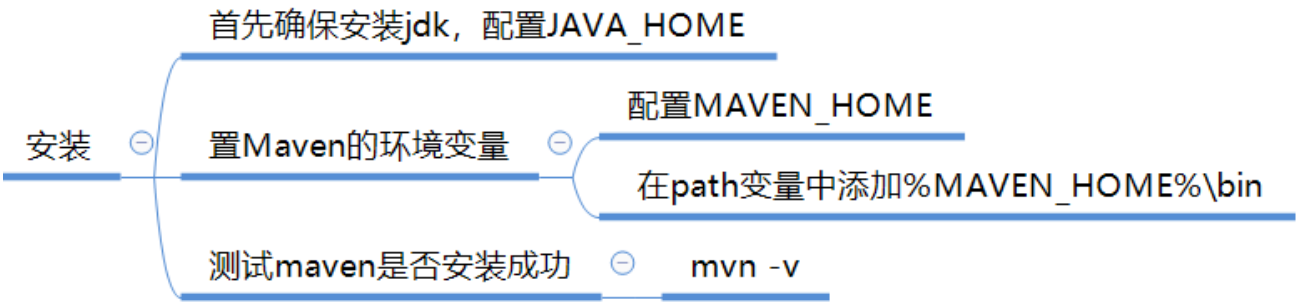

此电脑 › work (E:) › work666 › software ›
名称
修改日期
类型
大小
apache-tomcat-7.0.91 2018/9/13 15:53 文件夹
apache-tomcat-8.0.53 2018/6/29 10:44 文件夹
iso 2018/8/10 9:26 文件夹
Oracle 2018/8/10 14:26 文件夹
客户端工具PLSQL+Developer8.04.1514绿色版 2015/9/6 21:53 文件夹
apache-maven-3.6.0-bin.zip 2018/11/22 17:54 WinRAR ZIP 压缩文件 8,890 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

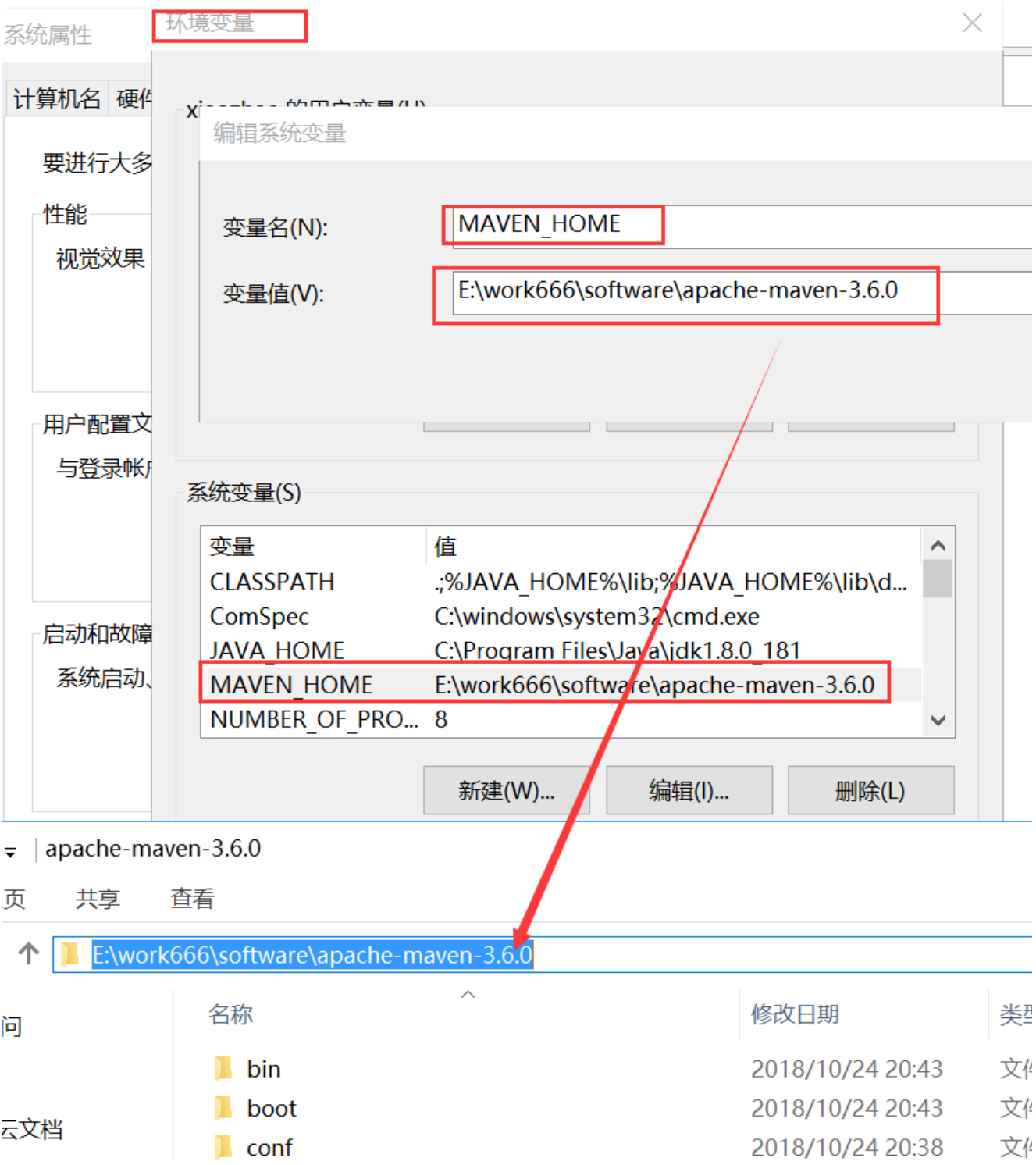


* 首先确保安装jdk，配置JAVA_HOME

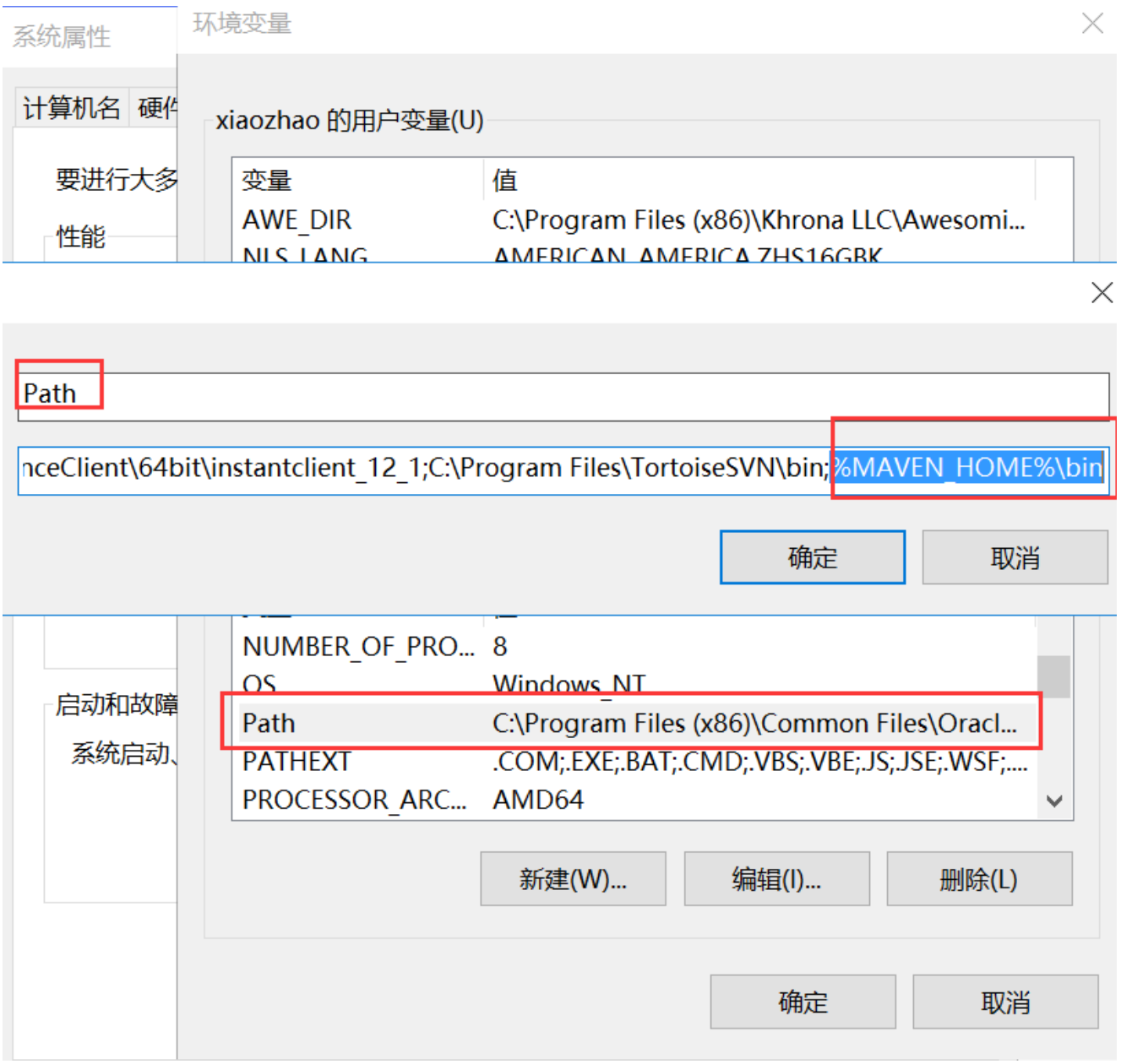
```
C:\Users\xiaozhao>echo %JAVA_HOME%
C:\Program Files\Java\jdk1.8.0_181

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

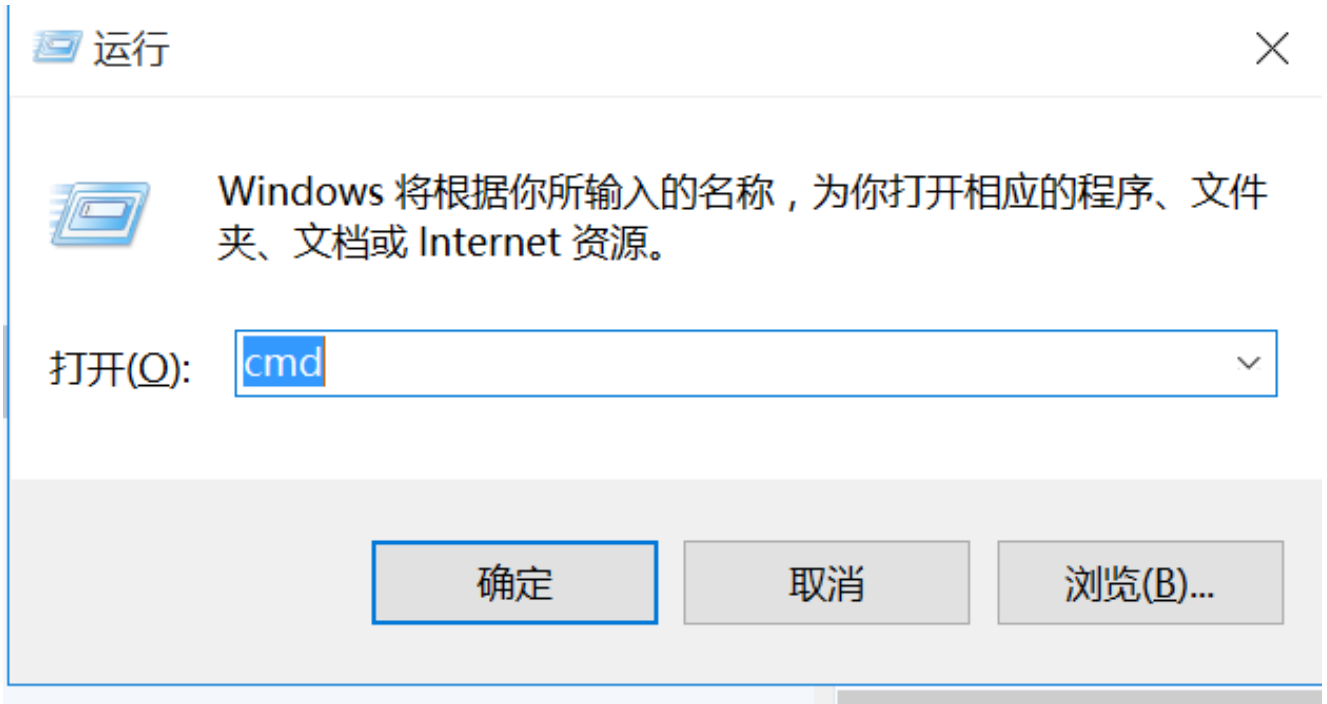
    * 配置MAVEN_HOME



* 在path变量中添加%MAVEN_HOME%\bin

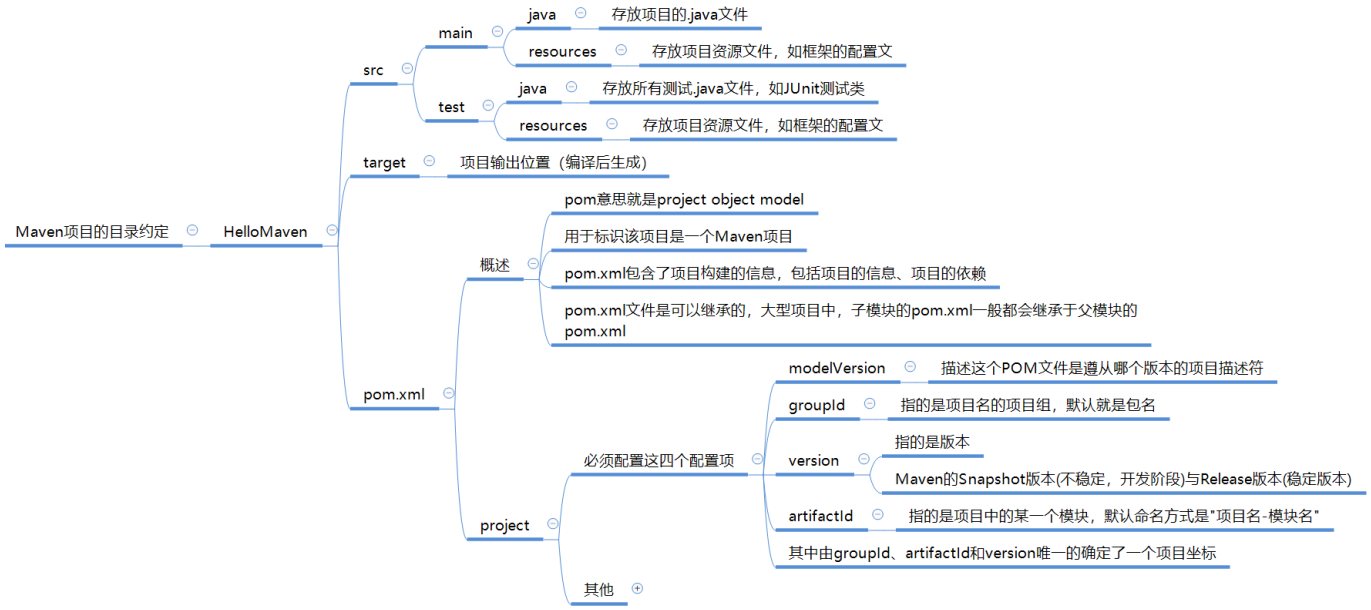


* 测试maven是否安装成功

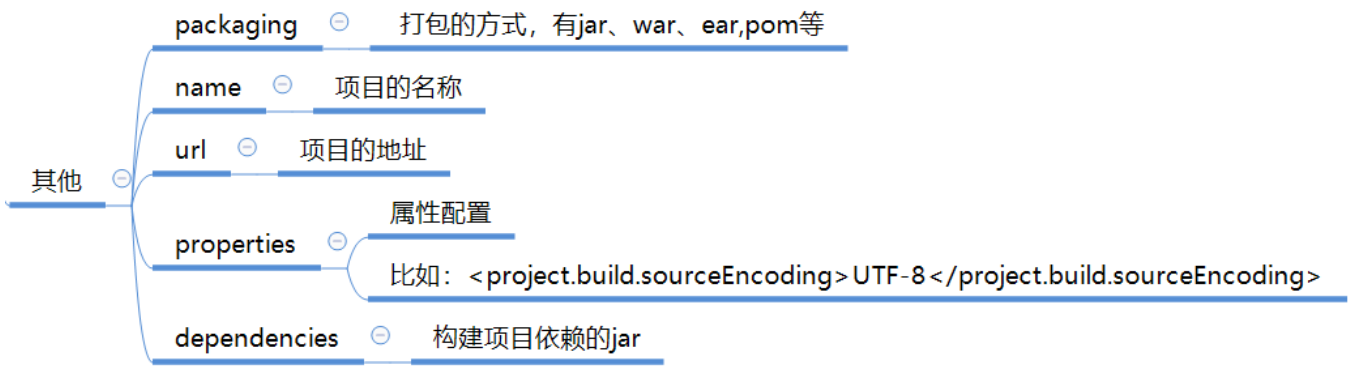


```
C:\Users\xiaozhao>mvn -v
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5ffb20918da4f719f3; 2018-10-25T02:41:47+08:00)
Maven home: E:\work666\software\apache-maven-3.6.0\bin\..
Java version: 1.8.0_181, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0_181\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

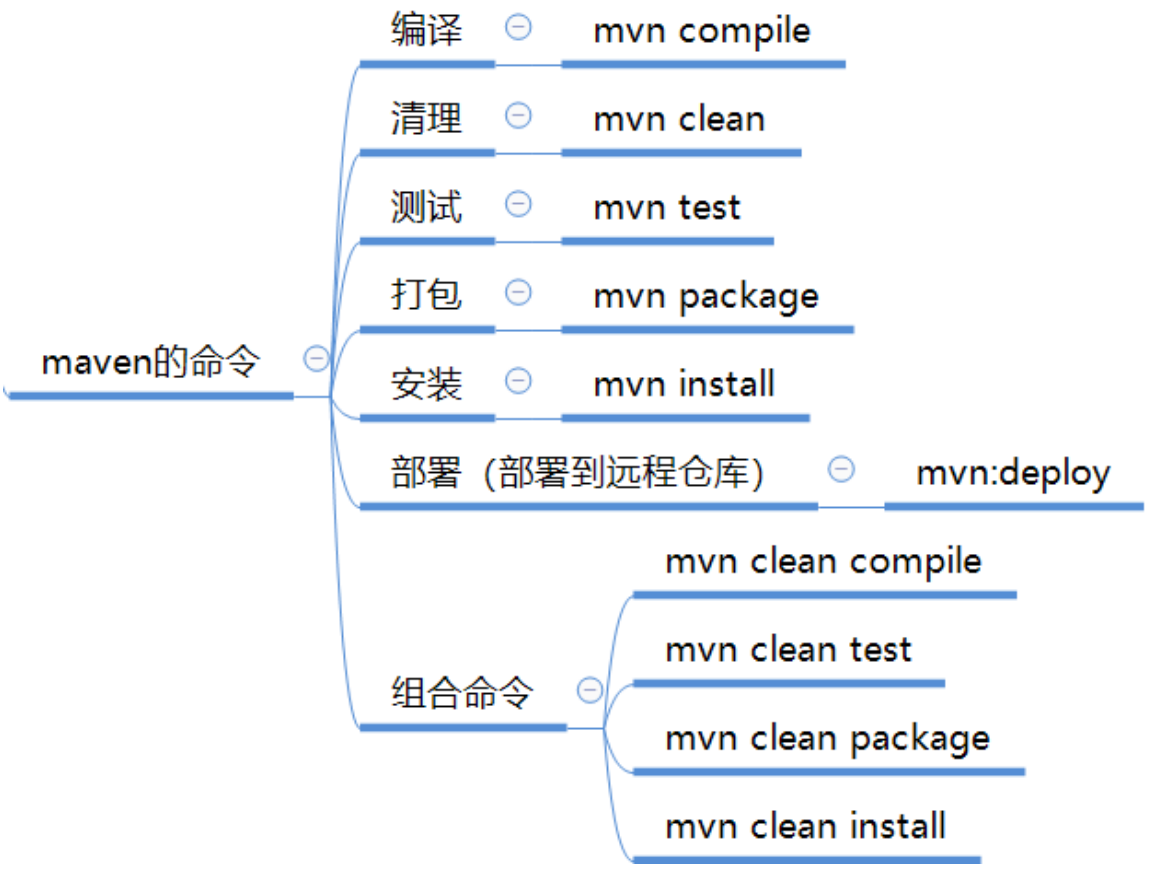
* Maven项目约定的目录



* Project 其他配置项



* maven的命令



* 构建项目

* 开发Maven项目

```
* 案例一
 * 创建项目根文件夹，例如maven1
 * 创建约定的目录
 * 创建“pom.xml”文件
 <?xml version="1.0" encoding="UTF-8"?>
<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/xsd/maven-4.0.0.xsd">
  <!--所有的Maven项目都必须配置这四个配置项-->
  <!--modelVersion 描述这个POM文件是遵从哪个版本的项目描述符-->
  <modelVersion>4.0.0</modelVersion>
  <!--groupId指的是项目名的项目组，默认就是包名-->
  <groupId>com.lg.maven.hello</groupId>
  <!--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"-->
  <artifactId>maven1</artifactId>
  <!--version指的是版本，这里使用的是Maven的快照版本-->
  <!--Maven的Snapshot版本(不稳定，开发阶段)与Release版本(稳定版本)-->
  <version>SNAPSHOT-0.0.1</version>
</project>
  * 在maven规范的目录下（src/main/java），创建HelloWorld.java文件
 public class HelloWorld{
    public static void main(String[] args){
        System.out.println("HelloWorld");
    }
}
 * 编译
    * 在cmd中进入maven项目
    * mvn compile
  * 观察：是否构建成功，是否生成target，默认仓库是否自动下载jar

* 案例二（使用单元测试）
  * 建立maven2项目，同时建立Maven约定的目录结构和pom.xml文件
  * pom.xml
  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.c
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/x
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.lg.maven.hello</groupId>
  <artifactId>maven2</artifactId>
  <version>0.0.1-SNAPSHOT</version>
```

```
  <name>hellomaven</name>
  <!--添加依赖的jar包-->
 <dependencies>
        <!--项目要使用到junit的jar包，所以在这里添加junit的jar包的依赖-->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.9</version>
            <scope>compile</scope>
        </dependency>
   </dependencies>
</project>
 * 代码
  * 放到D:\wook666\java\maven2\src\main\java
public class Person {
    public String sayHello(String name){
        return "Hello "+name+"!";
    }
}
 * 放到D:\wook666\java\maven2\src\test\java
import org.junit.Test;
public class PersonTest {
    @Test
    public void testHello(){
        Person person = new Person();
        String results = person.sayHello("xiaohei");
        System.out.println(results);
    }
}
 * 在测试前，更换仓库的位置
    * D:\wook666\software\apache-maven-3.6.0\conf
    * 打开settings.xml修改位置本地仓库的位置
     * <localRepository>D:\repository</localRepository>
    * 替换原来maven远程仓库（下载比较慢）
     <mirror>
      <id>alimaven</id>
      <name>aliyun maven</name>
      <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
      <mirrorOf>central</mirrorOf>
     </mirror>
```

```
    * 观察下载jar包
 * 测试(每个命令分别查看效果):
   * mvn compile
      * 在target会生成相应的class文件
   * mvn clean
      * 发现删除了target目录
   * mvn test
      * 在cmd中可以发现自动执行单元测试
   * mvn package
     * 在target目录：D:\wook666\java\maven2\target
       * 可以发现jar的生成
   * mvn install
     * 在本地仓库目录D:\repository\com\lg\maven\hello\maven2\0.0.1-SNAPSHOT
        * 可以发现jar的生成
 * 组合命令测试(每个命令分别查看效果):
   * mvn clean compile
   * mvn clean test
   * mvn clean package
   * mvn clean install

* 案例三（使用自己在maven生成的jar）
  * 建立maven3项目，同时建立Maven约定的目录结构和pom.xml文件
  * pom.xml 文件
  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.c
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/x
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.lg.maven.hello3</groupId>
  <artifactId>maven3</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>maven3</name>

    <!--添加依赖的jar包-->
    <dependencies>
        <!--项目要使用到junit的jar包，所以在这里添加junit的jar包的依赖-->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.9</version>
            <scope>compile</scope>
        </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.getMyName();
135         return str;
136     }
137     public String getMyName(){
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("xiaohei");
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 -Darche
157     mvn archetype:generate
158    -DgroupId=com.lg.maven.shopping
159    -DartifactId=shopping
160    -DarchetypeArtifactId=maven-archetype-quickstart
```

```
   -DinteractiveMode=false
  * 测试
    * mvn test
    * mvn clean package
      * java -cp D:\wook666\java\maven4\shopping\target\shopping-1.0-SNAPSHOT.j

* 案例五（自动构建java web项目目录）
  * mvn archetype:generate -DgroupId=com.lg.web -DartifactId=helloweb -Darchety
   * Generating project in Batch mode：在这里需要等待很久，等待下载需要的文件
     * 之前命令加上参数 -X
     * 发现在网络查找该文件:Searching for remote catalog: https://repo.maven.apac
       * 这个文件有8.5M，由于访问网络网速就几KB
     * 解决方案：
       * 下载到本地放到D:\repository\org\apache\maven\archetype\archetype-catalo
       * 执行的时候参数 -DarchetypeCatalog=local
  * mvn archetype:generate -DgroupId=com.lg.web -DartifactId=helloweb -Darchety

  * 测试
   * mvn package
   * 在target 复制出war包，放到tomcat中
   * 访问：http://localhost:8080/helloweb/index.jsp
```