# 1.安装

## 1.将maven的安装目录拷贝到环境变量中去，新建一个路径M2\_HOME，然后将bin目录拷贝到path中，

## 2.测试 ：mvn – v 观察有没有东西出现

# 2、直接cmd编译开发

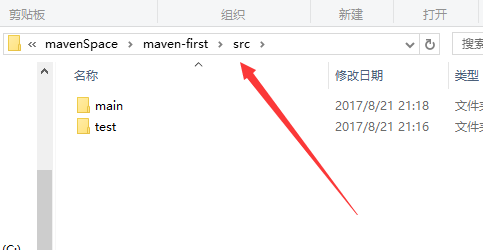
## 1、建立文件夹和pom.xml文件

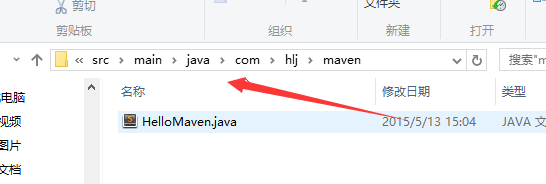
### 1、pom

|  |
| --- |
| <?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">    <modelVersion>4.0.0</modelVersion>  <groupId>maven-fisrtGroupId</groupId>  <artifactId>maven-fristArtifactId</artifactId>  <version>0.0.1-SNAPSHOT</version>  </project> |

### 2、目录结构





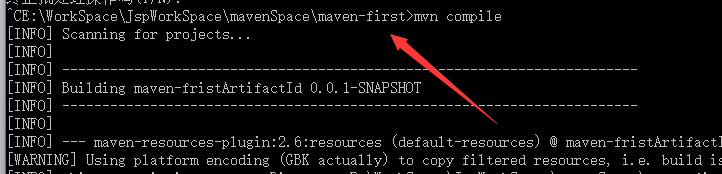


### 3、java文件

|  |
| --- |
| ackage com.hlj.maven;  public class HelloMaven {    public String getName(String name){    return "hello"+name;  }  } |

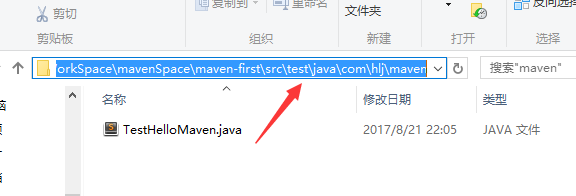
## 4、maven 编译，pom.xml所在项目中

## mvn compile



# 3、使用mvn test

## 1、test文件夹下进行添加测试文件，目录结构和上面的一模一样



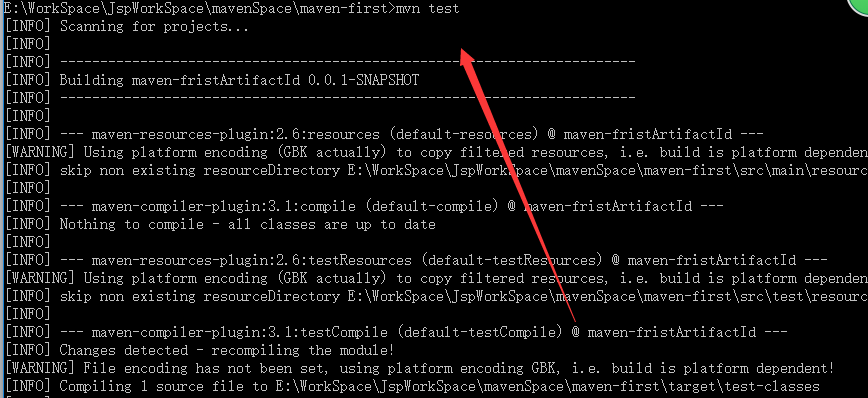
## 2、编辑测试类

|  |
| --- |
| package com.hlj.maven;  import org.junit.\*;  public class TestHelloMaven {    @Test  public void testGetName(){  HelloMaven HelloMaven = new HelloMaven();  HelloMaven.getName("HealerJean");  }  } |

## 3、pom中添加测试依赖junit

|  |
| --- |
| <?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">    <modelVersion>4.0.0</modelVersion>  <groupId>maven-fisrtGroupId</groupId>  <artifactId>maven-fristArtifactId</artifactId>  <version>0.0.1-SNAPSHOT</version>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.10</version>  <scope>test</scope>  </dependency>  </dependencies>  </project> |

## 4、测试mvn test，没有报错，

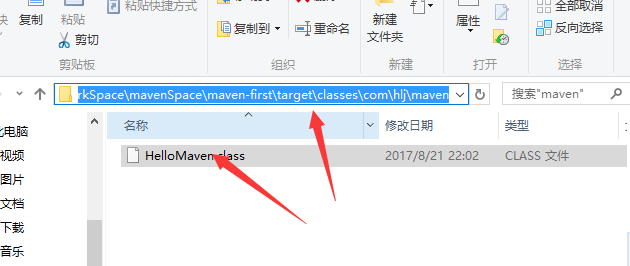


## 5、解释：这里是编译test中的测试类

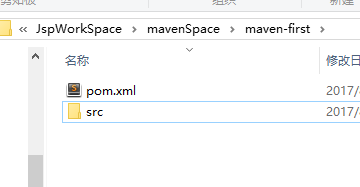
# 4、mvn clean

## 解释：经过上面的编译后会形成targer，并且形成class类，也就是java编译完成的对象，使用这个命令之后会删除这些东西

## 1、经过编译mvn compile 或者mvn test之后

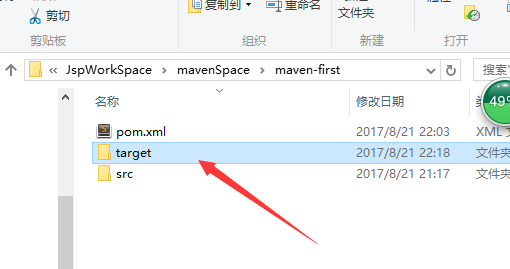


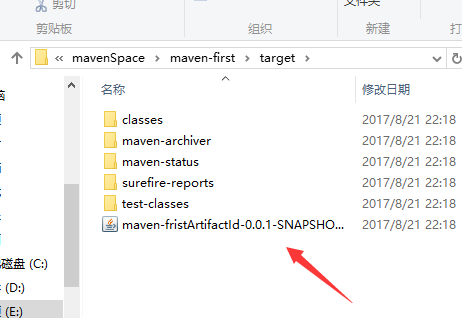
## 2、mvn clean之后，target都不在了，更何况类class



# 5、mvn package打包

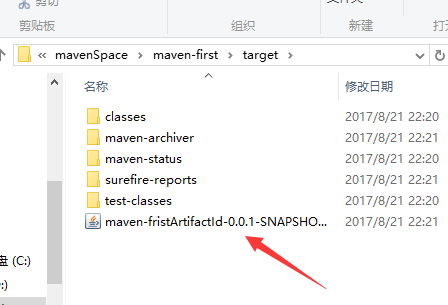
## 1、打包是将项目打包到target目录下面，这里毫无疑问肯定是会执行生成class的。但是这里的打的包不会到m2仓库中，也就是说如果有其他项目依赖于本项目，则打包不会起作用。



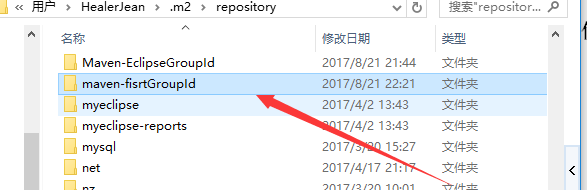


# 6、mvn install

## 1、不仅会打包



## 2.而且会打包到m2仓库中，这样就可以使用其他项目依赖它了



# 7、跳过Test

## mvn -U -DskipTests clean package