

華東理工大學

信息科学与工程学院

《软件工程》 实验报告七

系 别 计算机系

专 业 计算机科学与技术

年 级 2020 级

姓 名 刘子言

指导教师 阮 彤

2022-2023 学年 第 1 学期

# 实验七 基于 Maven Build 的自动构建

## 一、实验目的

- 1、理解什么是自动构建，对比什么是手动构建；
- 2、理解 JUnit 测试框架的用途。

## 二、实验装置

个人 PC 机器，apache-maven-3.8.3，jdk1.8.0\_202。

## 三、实验内容

### 1、配置系统环境变量

在 DOS 命令窗口键入 mvn -version 查看 Maven 及 jdk 版本号，检测环境配置是否成功：

```
选择 管理员: C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19044.2251]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>mvn -version
Apache Maven 3.8.3 (ff8e977a158738155dc465c6a97ffaf31982d739)
Maven home: D:\Java\apache-maven-3.8.3
Java version: 1.8.0_202, vendor: Oracle Corporation, runtime: D:\Java\jdk1.8.0_202\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 10", version: "10.0", arch: "x86", family: "windows"

C:\Users\Administrator>_
```

Maven 版本号：apache-maven-3.8.3

Jdk 版本号：jdk1.8.0\_202

### 2、检察待编译文件的目录结构

1) pom.xml 文件所在目录

E:\《数学+计算机》\软件工程\实验代码\实验七 maven\mvn\_in\_action\_code\ch-3\hello-world

2) src 文件目录

E:\《数学+计算机》\软件工程\实验代码\实验七 maven\mvn\_in\_action\_code\ch-3\hello-world

名称	修改日期	类型	大小
.settings	2009/10/9 1:46	文件夹	
src	2009/10/10 19:34	文件夹	
target	2022/11/10 14:22	文件夹	
.classpath	2009/10/10 23:40	CLASSPATH 文件	1 KB
.project	2009/10/9 1:46	PROJECT 文件	1 KB
pom.xml	2010/8/26 16:46	XML 文档	2 KB

3) .java 源文件所在目录

E:\《数学+计算机》\软件工程\实验代码\实验七 maven\mvn\_in\_action\_code\ch-3\hello-world\src\main\java\com\juvenxu\mvnbook\helloworld



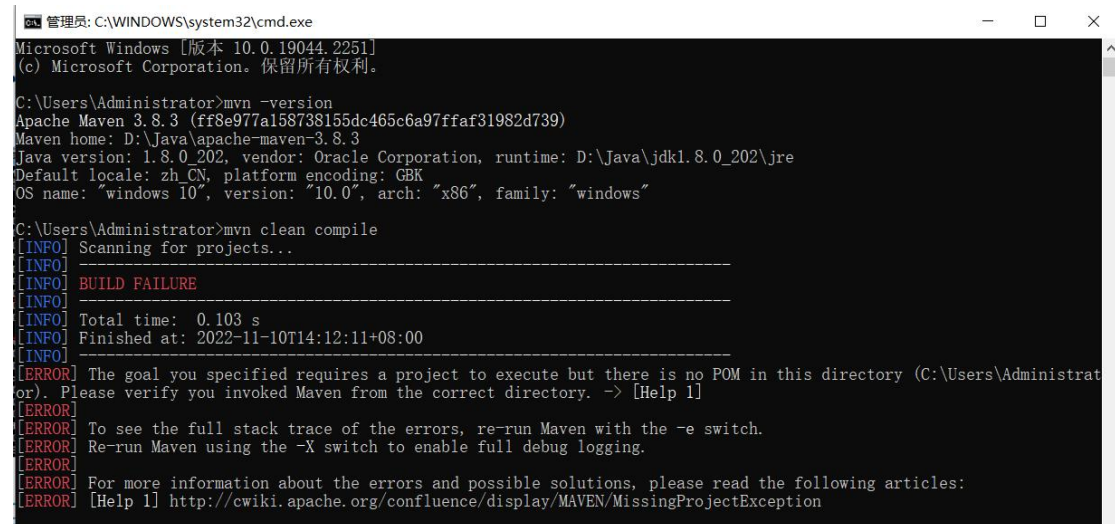
#### 4) Junit 测试文件所在目录

E:\《数学+计算机》\软件工程\实验代码\实验七  
maven\mvn\_in\_action\_code\ch-3\hello-world\src\test\java\com\juvenxu\mvnbook\helloworld



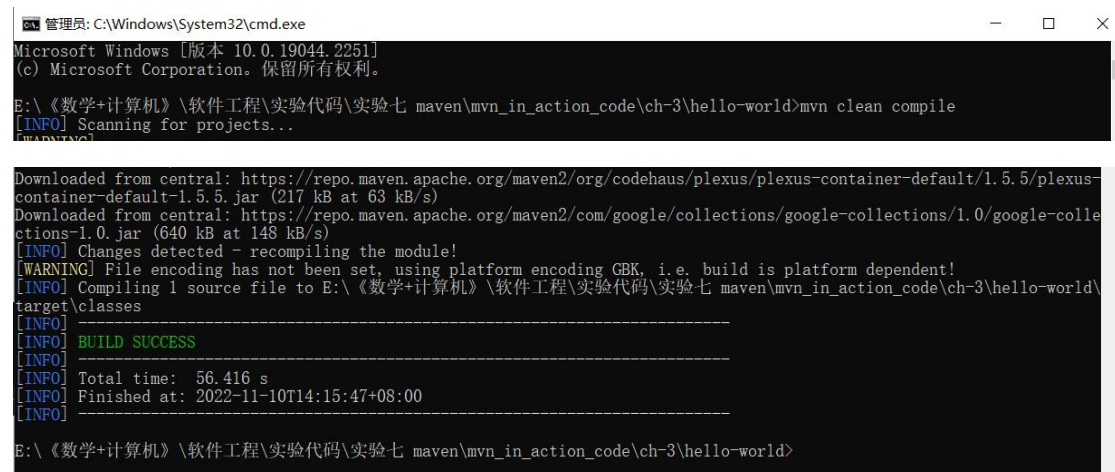
### 3、Maven Build 实践

#### 1) 在系统目录下运行 mvn clean compile, 察看结果



运行后发现 build failure, 因为路径不对, 找不到 pom.xml 文件。

#### 2) 进入 pom.xml 文件的目录, 再次键入 mvn clean compile 运行, 察看结果

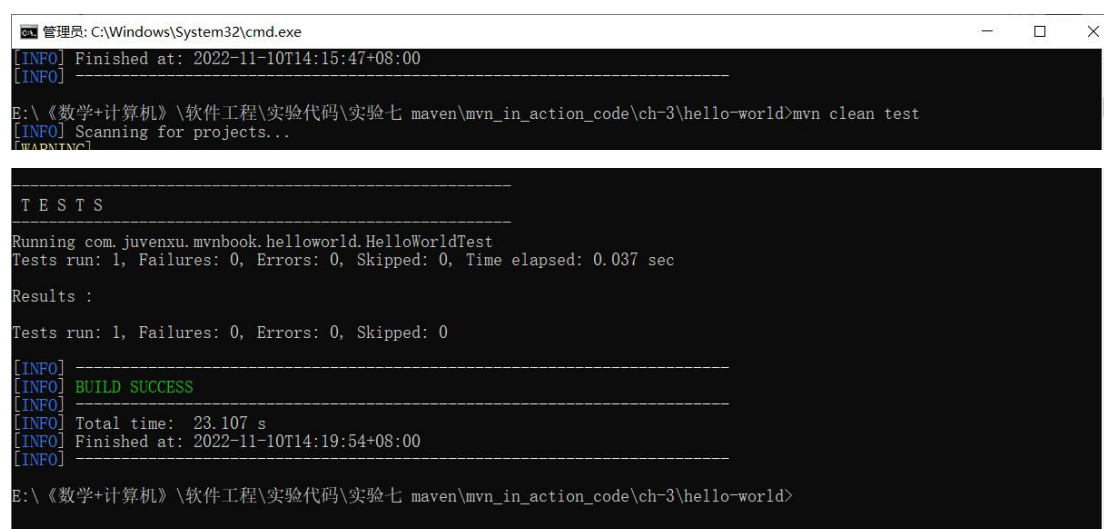


运行后可见编译结果为 **build success**，即表示编译成功。

查看该目录下，生成一个 **target** 文件夹，用于存放编译、打包后的输出文件：



### 3) 继续键入命令 `mvn clean test`，进行单元测试



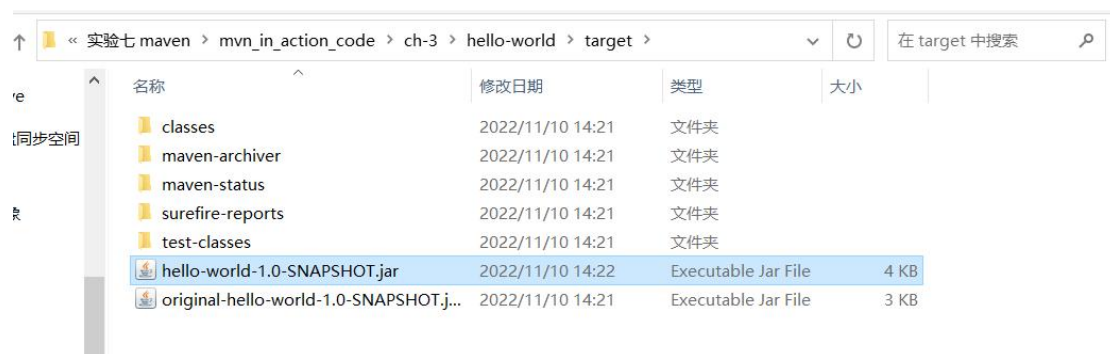
运行结果为 **build success**，即表示测试成功。

### 4) 继续键入命令 `mvn clean package`，把 java 项目打包成 jar 包或 war 包



运行结果为 **build success**，即表示打包成功。

打开 **target** 文件夹，发现文件夹下生成一个 **jar** 包：



5) `cd` 命令进入 `target` 目录下，键入 `java -jar hello-world-1.0-SNAPSHOT.jar`，运行 `hello-world-1.0-SNAPSHOT.jar` 文件

```
E:\《数学+计算机》\软件工程\实验代码\实验七 maven\mvn_in_action_code\ch-3\hello-world>cd target
E:\《数学+计算机》\软件工程\实验代码\实验七 maven\mvn_in_action_code\ch-3\hello-world\target>hello-world-1.0-SNAPSHOT.jar
E:\《数学+计算机》\软件工程\实验代码\实验七 maven\mvn_in_action_code\ch-3\hello-world\target>java -jar hello-world-1.0-SNAPSHOT.jar
Hello Maven
E:\《数学+计算机》\软件工程\实验代码\实验七 maven\mvn_in_action_code\ch-3\hello-world\target>
```

输出结果为 `hello Maven`，表示运行成功。

## 四、实验心得

通过本次实验，我基本掌握了 `Maven` 工具的基本操作与使用，熟悉了其常用的命令，并在此基础上进行了 `maven build` 实践，进一步加深了对 `maven` 命令使用的熟练程度，并对其基本的文件结构有了大致的掌握。

`Maven` 作为 `Java` 热门的自动化构建工具，与手动构建相比，大大地提高了项目构建的效率与便携性，对 `Java` 项目的开发十分友好。`Maven` 工具的学习，也为我后期项目的开发打下了基础，收获颇丰。