# Maven的工程结构

Project

|-src

| |-main

| | |-java —— 存放项目的.java文件

| | |-resources —— 存放项目资源文件，如spring, hibernate配置文件

| |-test

| |-java ——存放所有测试.java文件，如JUnit测试类

| |-resources —— 测试资源文件

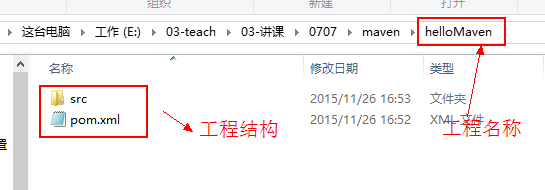
|-target —— 目标文件输出位置例如.class、.jar、.war文件

|-pom.xml ——maven项目核心配置文件

# Maven的工程创建

## 第一步：根据maven的目录结构创建helloMaven工程

target目录会在编译之后自动创建。



## 第二步：创建HelloWorld.java

在src/main/java/cn/itcast/maven目录下新建文件Hello.java

|  |
| --- |
| package cn.itcast.maven;  public class HelloWorld {  public String sayHello(String name){  return "Hello World :" + name + "!";  }  } |

## 第三步：创建TestHelloWorld.java

|  |
| --- |
| package cn.itcast.maven;  import org.junit.Test;  import static junit.framework.Assert.\*;  public class TestHelloWorld{    @Test  public void testSayHello(){  HelloWorld hw = new HelloWorld();  String result = hw.sayHello("zhangsan");  assertEquals("hello zhangsan",result);  }  } |

## 第四步：配置pom.xml

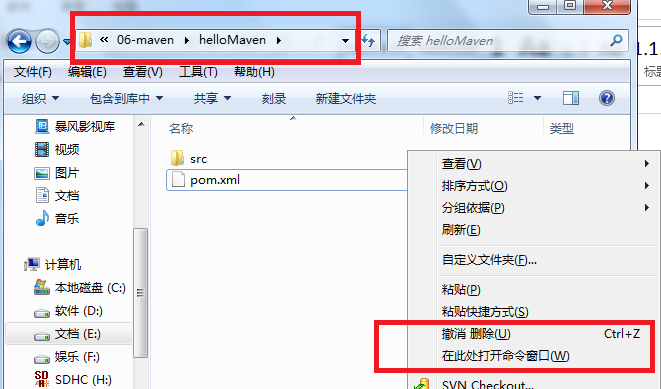


|  |
| --- |
| <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <!-- 版本：4.0.0 -->  <modelVersion>4.0.0</modelVersion>  <!-- 组织名称：暂时使用 组织名称+项目名称 作为组织名称 -->  <!-- 组织名称：实际名称 按照访问路径规范设置，通常以功能作为名称：eg: junit spring -->  <groupId>cn.itcast.maven</groupId>  <!-- 项目名称 -->  <artifactId>HelloWorld</artifactId>  <!-- 当前项目版本号：同一个项目开发过程中可以发布多个版本，此处标示0.0.1版 -->  <!-- 当前项目版本号：每个工程发布后可以发布多个版本，依赖时调取不同的版本，使用不同的版本号 -->  <version>0.0.1</version>  <!-- 名称：可省略 -->  <name>Hello</name>    <!-- 依赖关系 -->  <dependencies>  <!-- 依赖设置 -->  <dependency>  <!-- 依赖组织名称 -->  <groupId>junit</groupId>  <!-- 依赖项目名称 -->  <artifactId>junit</artifactId>  <!-- 依赖版本名称 -->  <version>4.9</version>  <!-- 依赖范围：test包下依赖该设置 -->  <scope>test</scope>  </dependency>    </dependencies>  </project> |

到此maven工程即创建成功。

# Maven的命令

需要在pom.xml所在目录中执行以下命令。



## 基本

### 编译 mvn compile

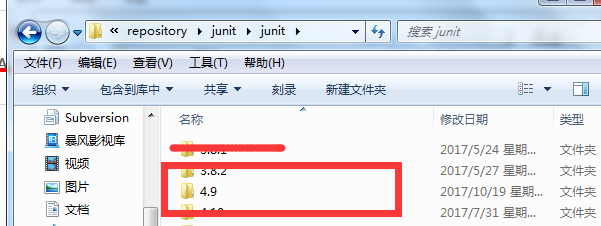
执行 mvn compile命令，完成编译操作

执行完毕后，从网上下载，会生成target目录，该目录中存放了编译后的字节码文件。





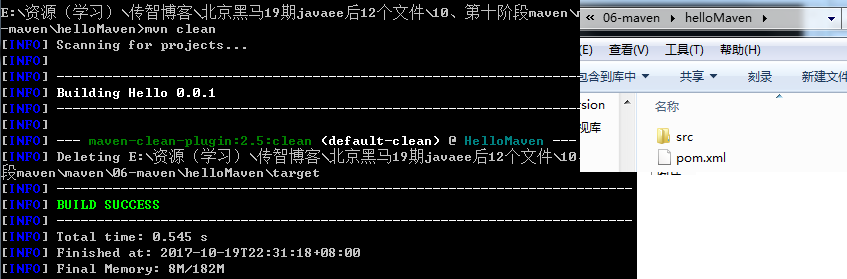
仓库下载



### 清除 mvn clean

执行 mvn clean命令

执行完毕后，会将target目录删除，清除命令，清除已经编译好的class文件。



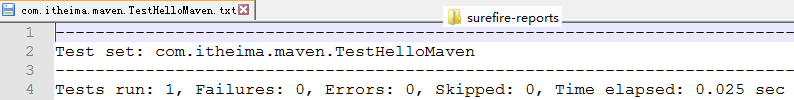
### 测试 mvn test

执行 mvn test命令，完成单元测试操作；注意会有编译执行

执行完毕后，会在target目录中生成三个文件夹：surefire、surefire-reports（测试报告）、test-classes（测试的字节码文件）



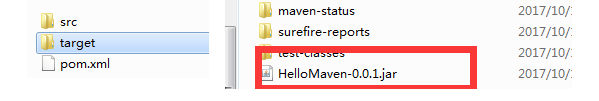
查看错误日志



### 打包 mvn package

执行 mvn package命令，完成打包操作，（默认是jar）

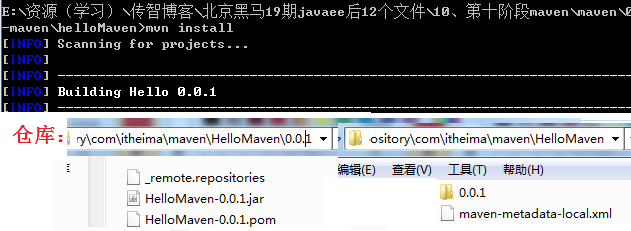
执行完毕后，会在target目录中生成一个文件，该文件可能是jar、war、pom



### 安装 mvn install

执行 mvn install命令，完成将打好的jar包安装到本地仓库的操作

执行完毕后，会在本地仓库中出现安装后的jar包，方便其他工程引用



## Clean组合—对应上面，实际中使用

### mvn clean compile命令

cmd 中录入 mvn clean compile命令

组合指令，先执行clean，再执行compile，通常应用于上线前执行，清除测试类

### mvn clean test命令

cmd 中录入 mvn clean test命令

组合指令，先执行clean，再执行test，通常应用于测试环节

### mvn clean package命令

cmd 中录入 mvn clean package命令

组合指令，先执行clean，再执行package，将项目打包，通常应用于发布前

执行过程：

清理————清空环境

编译————编译源码

测试————测试源码

打包————将编译的非测试类打包

### mvn clean install命令

cmd 中录入 mvn clean install 查看仓库，当前项目被发布到仓库中

组合指令，先执行clean，再执行install，将项目打包，通常应用于发布前

执行过程：

清理————清空环境

编译————编译源码

测试————测试源码

打包————将编译的非测试类打包

部署————将打好的包发布到资源仓库中

