Maven的基本概念：

Maven是一个项目管理工具，将项目开发和管理过程抽象成一个项目对象模型（POM）

也就是将每一个项目看做一个对象

坐标：在仓库上的定位

坐标组成：groupid 项目隶属组织名称，artifactid 项目名称，version

在mvnrepository.com网站上查找资源，找到坐标之后复制就ok

配置阿里私服镜像：

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<mirrorOf>central</mirrorOf>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

</mirror>

全局setting和用户setting区别：

公共seting和私人

三个仓库：

本地仓库，远程仓库（私服），中央仓库

私服能加快开发效率，并且能在一定范围内共享资源，比如有些资源只能对内部开放，不对外共享

常用maven命令：

compile 编译

clean 清理

test 测试

package 打包

install 安装

下面一个例子测试这些命令的具体作用：

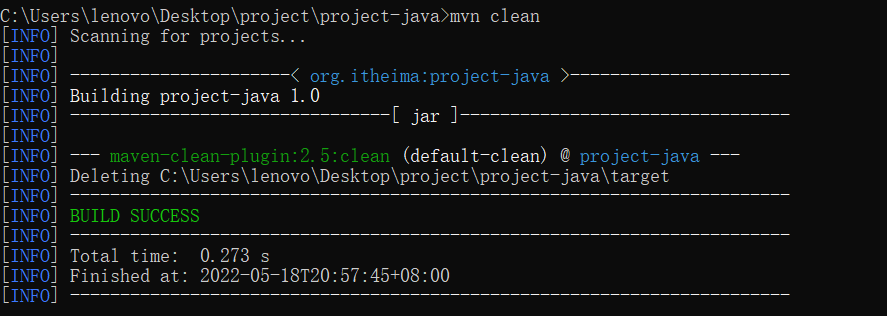
手写一个maven项目：



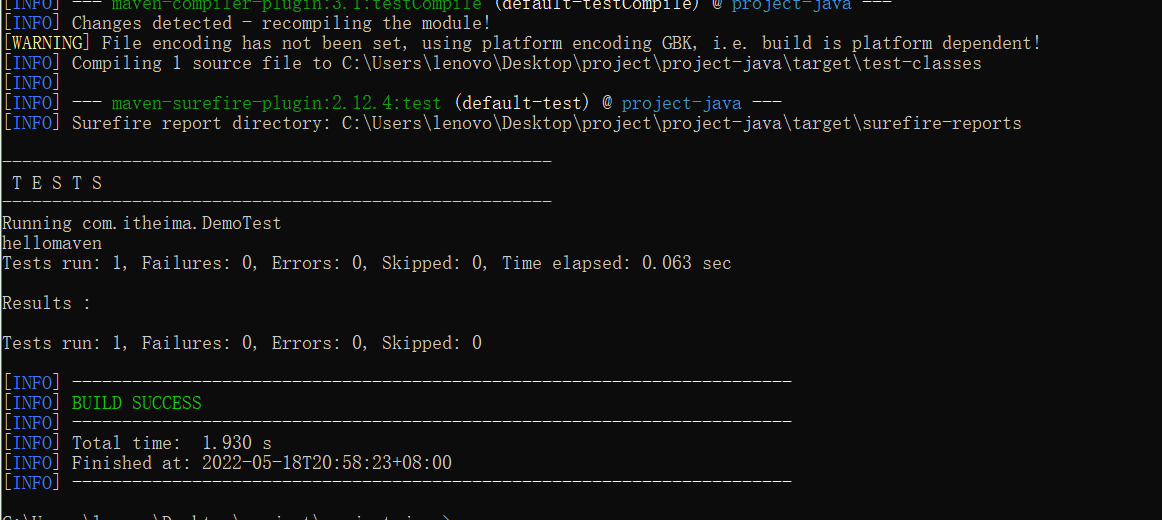
编译：



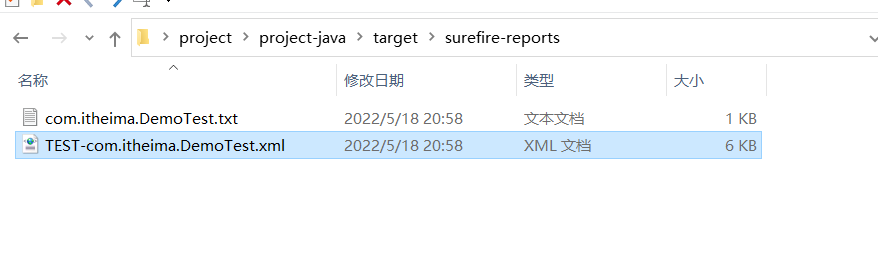
清除：



测试：

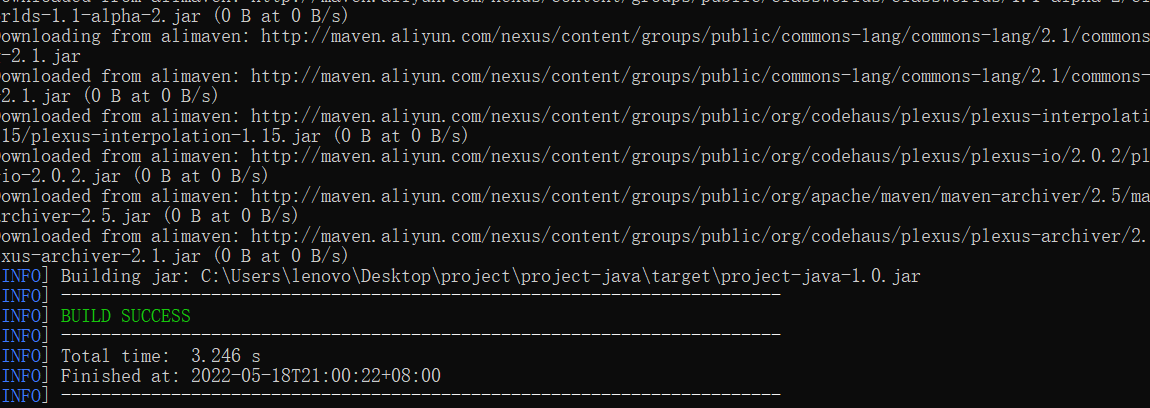


测试报告：



打包：

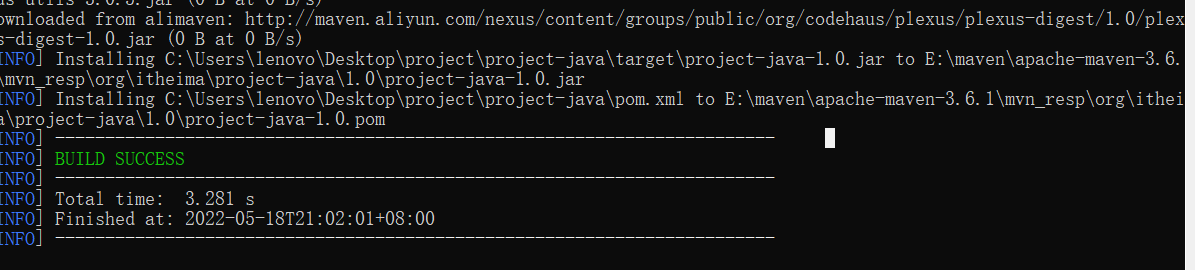
Mvn package



执行mvn install：

把项目打包到本地仓库中去





Idea操作创建Maven：

注意：Idea不支持maven3.6.2以上的版本

快捷导入以前导入的jar包，在pom.xml文件中 按alt + insert 然后找

导入Mysql驱动举例：

<!--Mysql驱动引入-->

<dependencies>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.46</version>

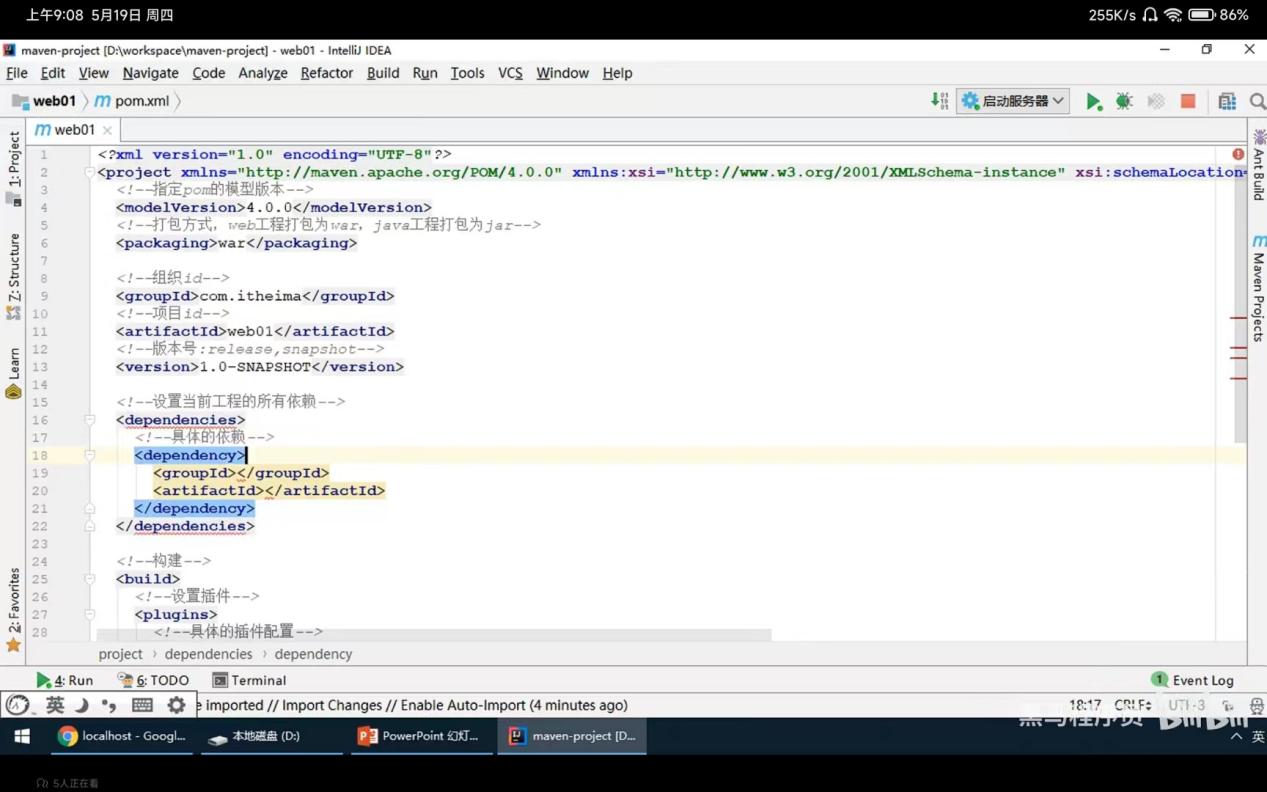
<!--scope标签设置依赖的作用范围-->

<scope>compile</scope>

</dependency>

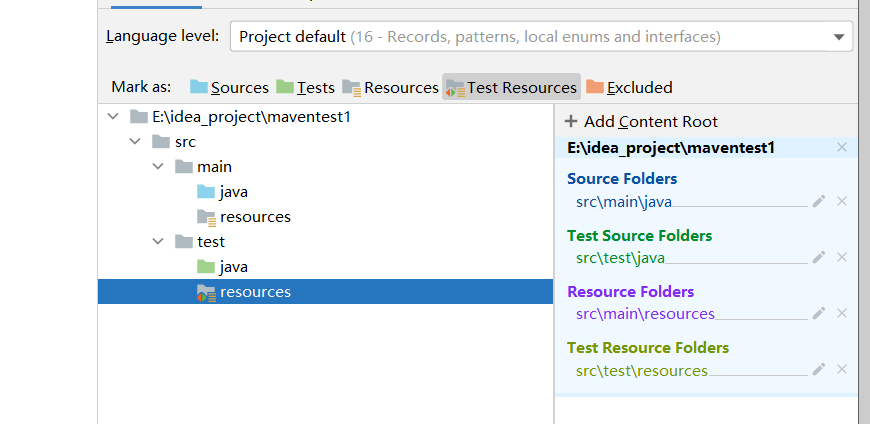
</dependencies>

三个标签的作用：组织，项目，版本（release 发行版，snapshot 开发版）

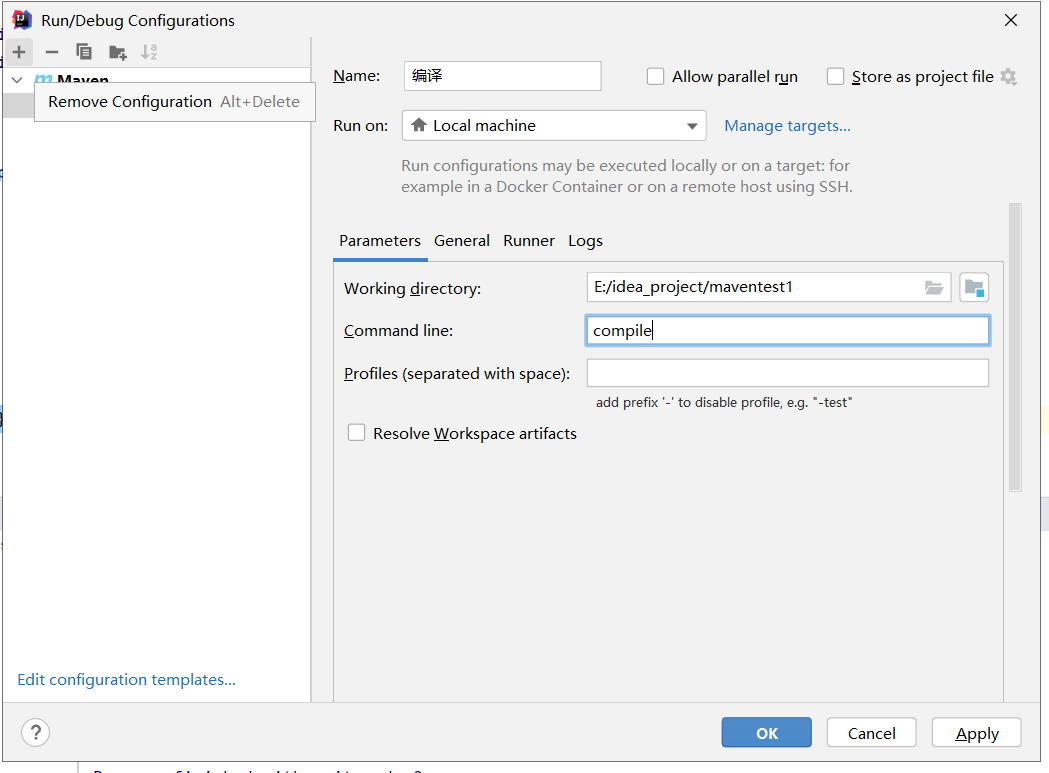


创建步骤：

1. 首先创建一个空项目，然后设置maven版本
2. 然后创建一个新的maven模块，注意为不同文件夹添加颜色：



1. 写Java文件，然后注入依赖，注意要及时刷新，使用maven插件完成编译测试打包等
2. 或者可以自己配置



配置好之后，点击运行就能完成相应的操作

使用骨架完成maven项目模块创建：

新建模块，使用骨架

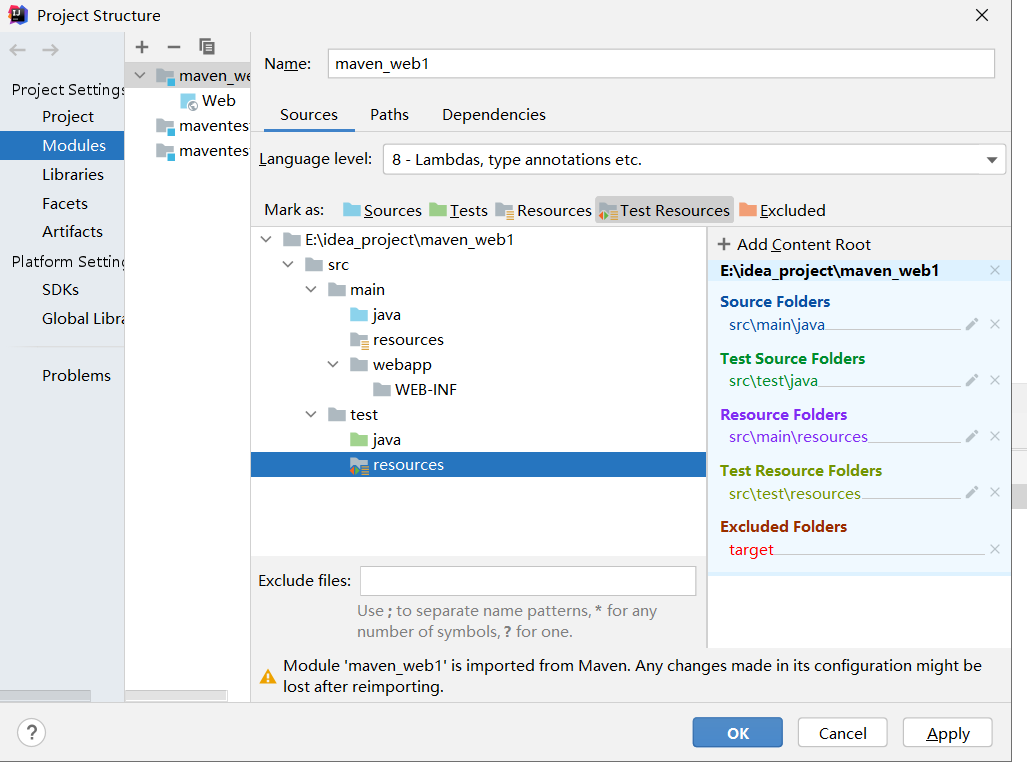


创建之后添加两个resources文件夹，改一下文件夹颜色

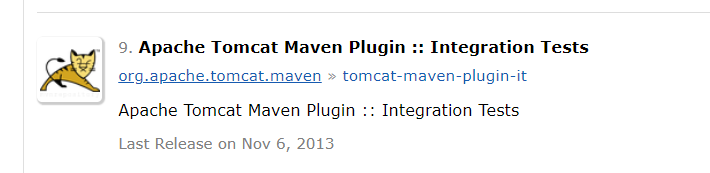
使用骨架创建web项目模块：

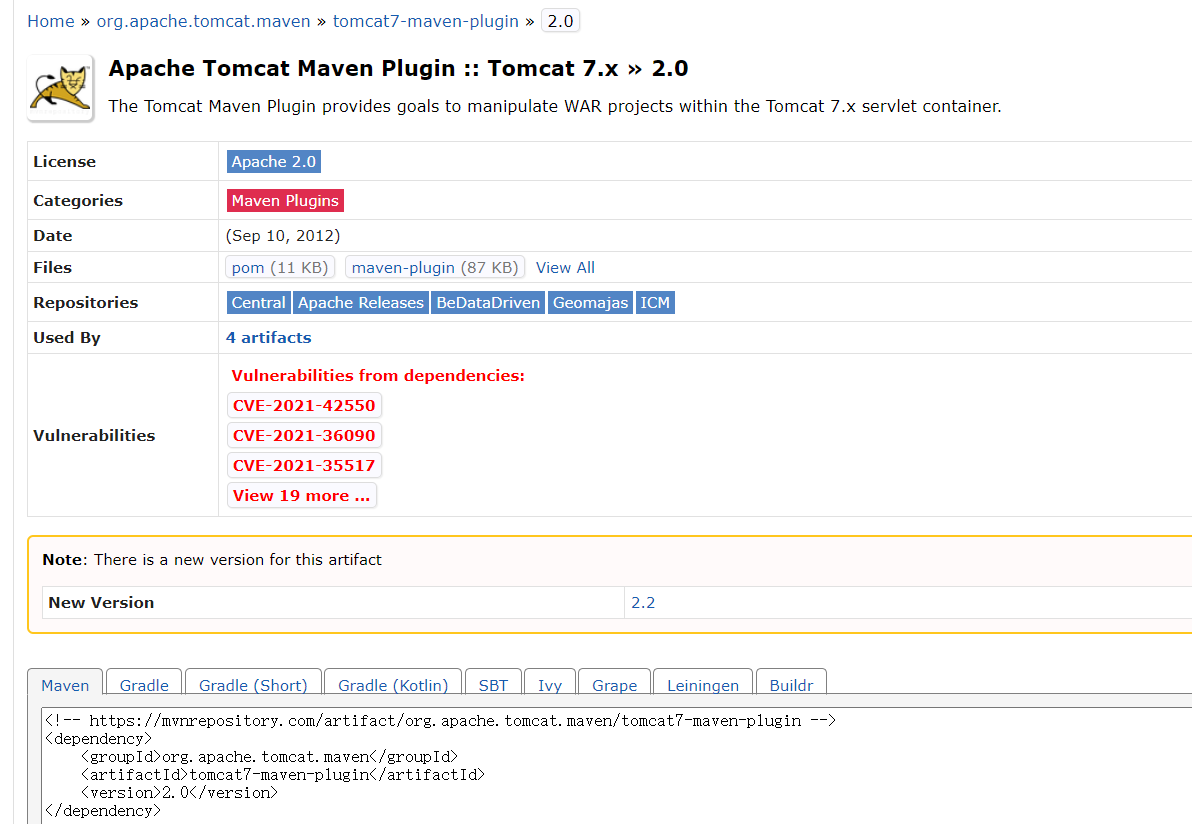


找到骨架创建之后添加文件夹，修改文件夹颜色：



配置tomcat maven插件，先去网站查询这个插件的配置坐标





然后删除里面的build标签，配置：（注意，坐标要找对，然后build标签在dependencies标签之外）：

<!--构建tomcat maven插件-->

<build>

<plugins>

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

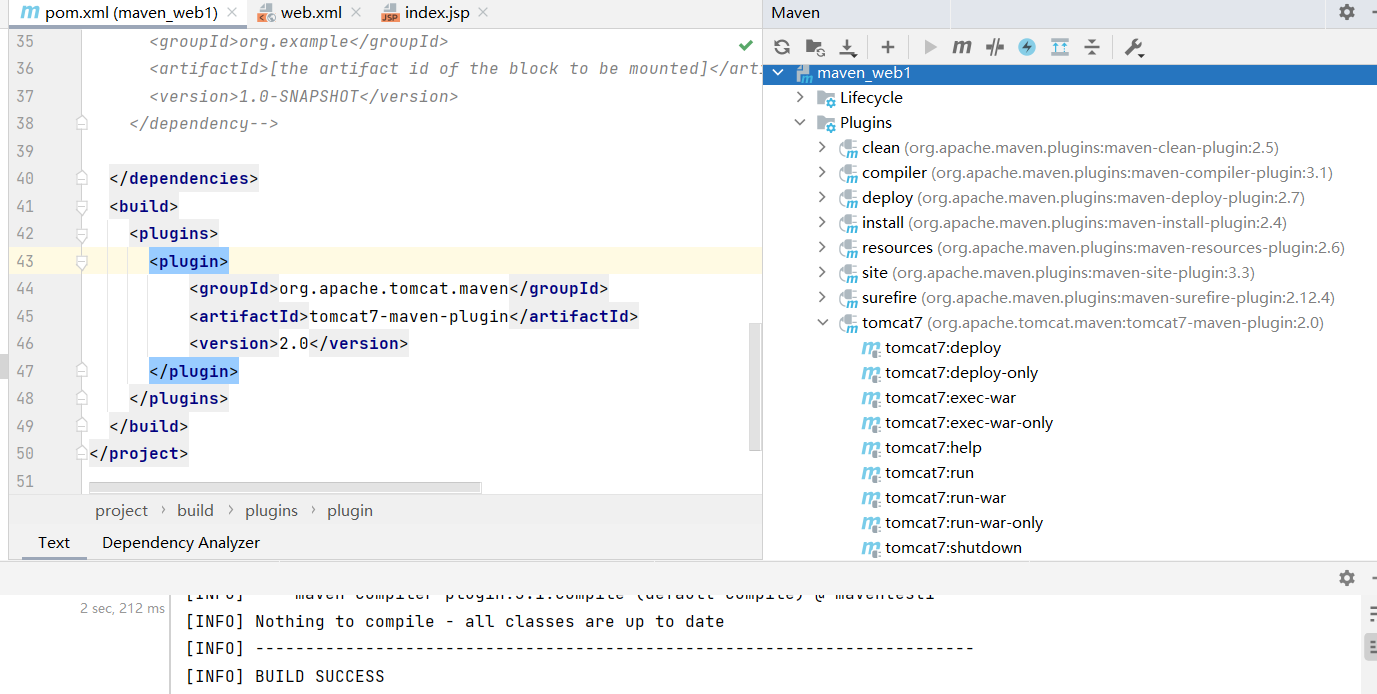
<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.0</version>

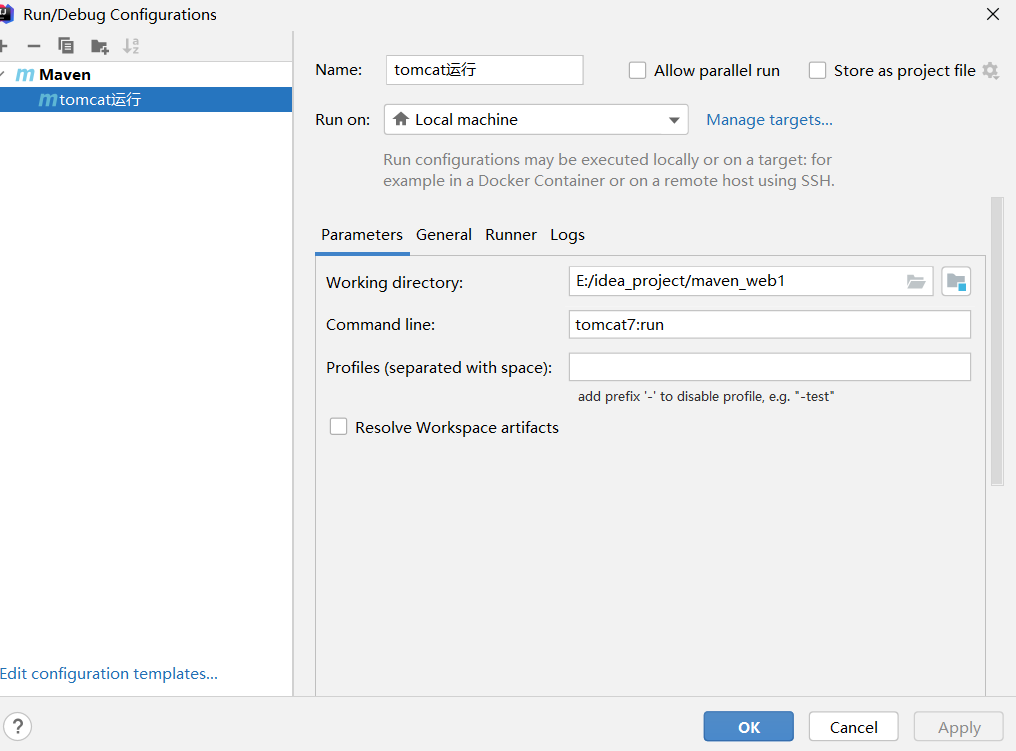
</plugin>

</plugins>

</build>



然后配置操作。。。



依赖管理：

依赖传递：

直接依赖：直接使用坐标注入的依赖

间接依赖：直接依赖之前就使用过的依赖也会随直接依赖传递（出现依赖传递冲突问题）



可选依赖：控制某些依赖不被看到

项目B引入项目A的坐标，但不想看到A中的junit依赖，可以在A中添加<optional>true</optional>标签，让B看不到

排除依赖：主动断开不想要的依赖

项目B不想用到A的junit依赖下的某个依赖就不能修改A，应该去修改junit标签（很难），所以可以在B里面主动排除依赖，在引入A的依赖标签dependency下添加标签如下：

<dependency>

<groupId>comitheima</groupId>

<artifactId>project03</artifactId>

<version>1.0-SNAPSHOT</version>

<exclusions>

<exclusion>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

</exclusion>

</exclusions>

</dependency>

依赖范围：

scope标签设置依赖的作用范围：

compile：编译，测试，运行

test：测试。。。典型的就是Junit的默认值

provided：编译，测试

runtime：测试，运行

system：编译，测试

依赖范围传递性：

项目A引入项目B，项目Acompile 项目B runtime 最终就是runtime



生命周期与插件：

编译 测试 打包 安装

Maven把项目的生命周期划分为3套：

Clean 清理工作

Default 核心工作，编译 测试 打包等等

Site 产生报告，发布站点等

插件和生命周期相匹配，表示在相应的生命周期应该执行的操作

去官网找到对应的插件，找到坐标，配置，然后执行插件功能观察结果

Maven高级：

SSM整合之后开始

1. 分模块开发与设计：