《Maven配置安装》

Maven项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的软件项目管理工具。

在项目开发中Maven可以对jar包和对工程之间的依赖关系进行管理，集成tomcat插件，maven项目可以自动发布到tomcat下。

maven仓库中存储jar包，可以一次下载，所有项目通用。

1. 安装Maven

1.1 下载Maven

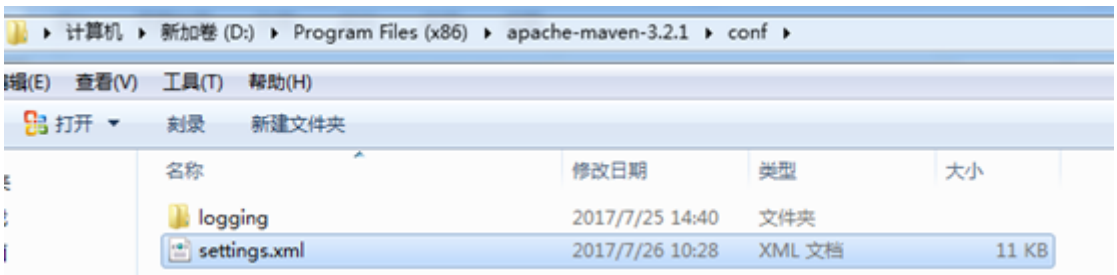
下载maven的官网地址： http://maven.apache.org/download.cgi

在参考资料中给大家提供了压缩文件 apache-maven-3.2.1.zip 。

解压到安装路径即可。

1.2 修改配置文件

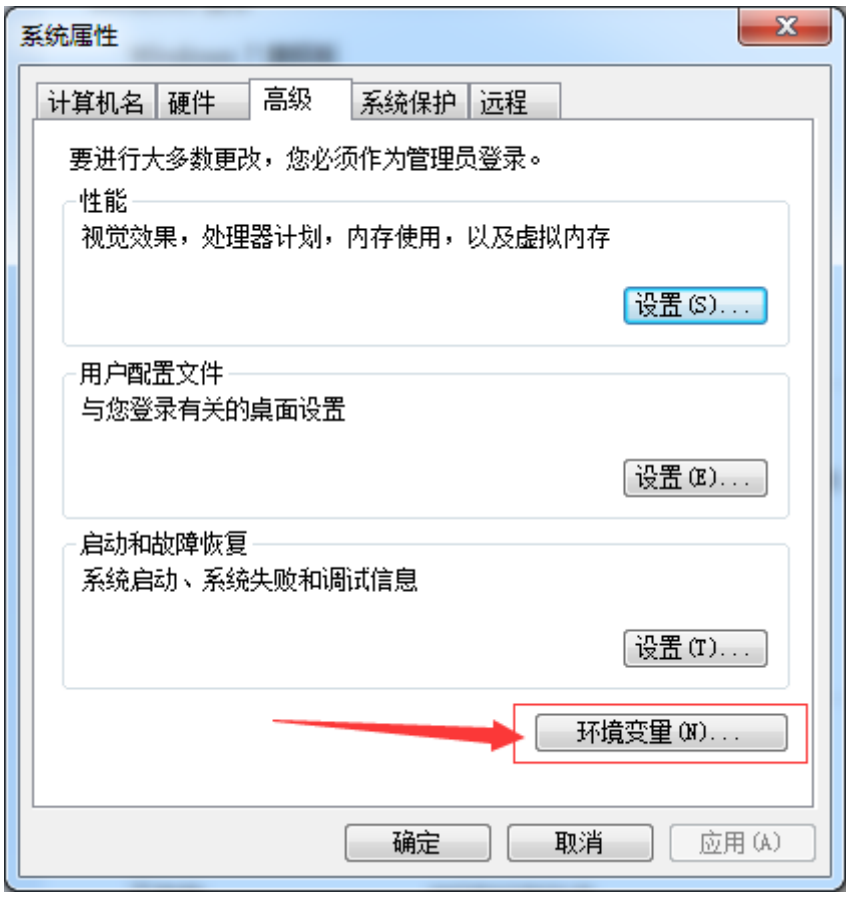
在conf文件夹里有一个settings.xml



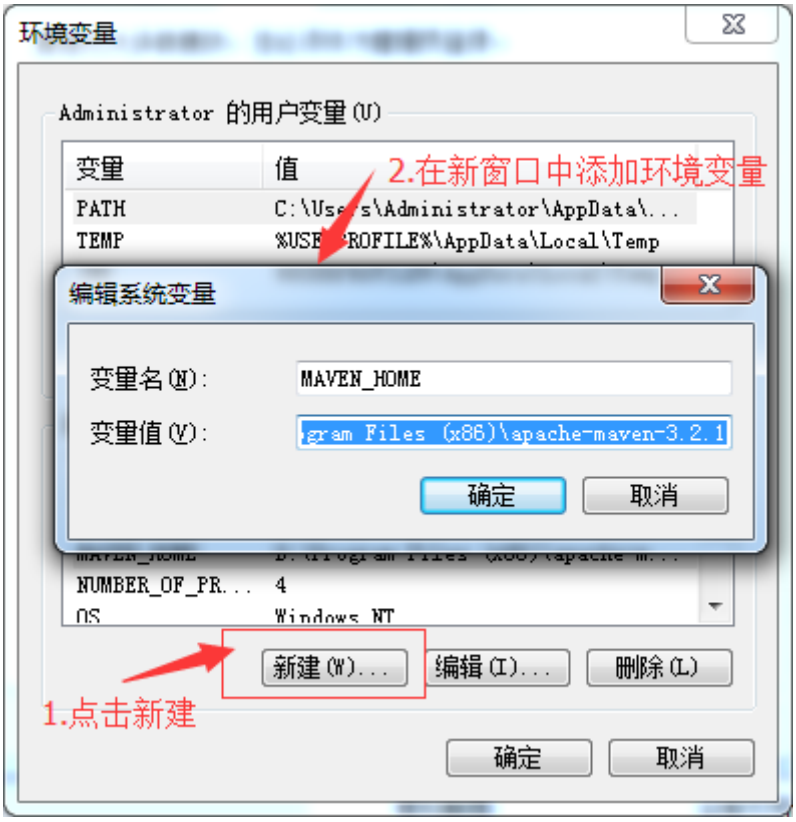
编辑配置文件，指定本地仓库的路径，即下载的jar包存放在哪里。

```
41  | The sections in this sample file are intended to give you a running start
42  | getting the most out of your Maven installation. Where appropriate, the de
43  | values (values used when the setting is not specified) are provided.
44  |
45  |-->
46  <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
47            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
48            xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
          http://maven.apache.org/xsd/settings-1.0.0.xsd">
49    <!-- localRepository
50     | The path to the local repository maven will use to store artifacts.
51     |
52     | Default: ${user.home}/.m2/repository                    指定本机上的仓库路径
53     -->
54    <localRepository>E:\maven\repository</localRepository>
```

1.3 配置环境变量



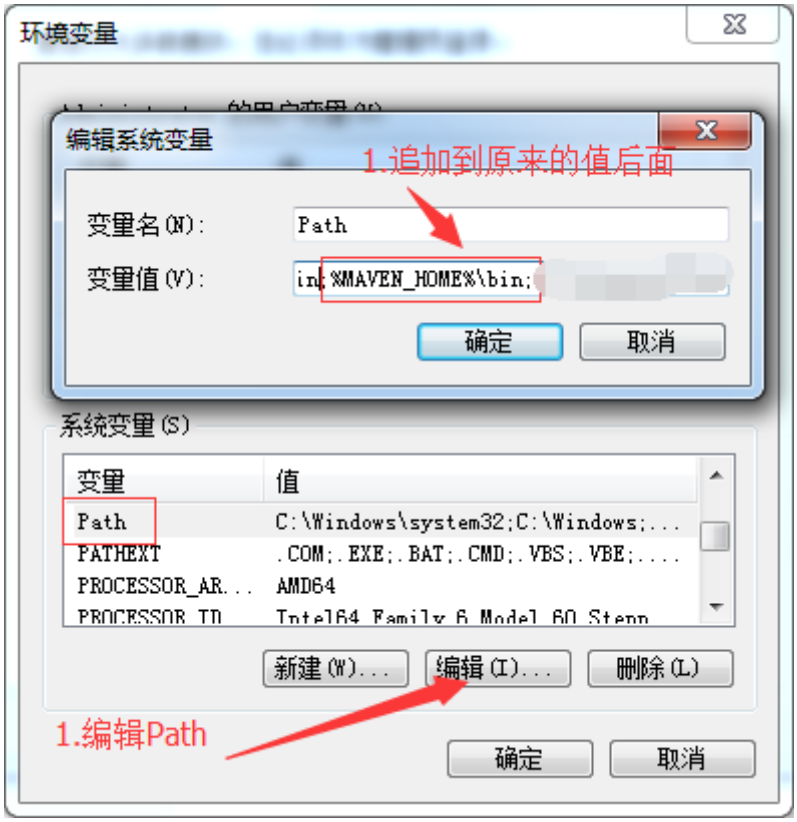
添加变量MAVEN_HOME，变量值为1.1中的解压路径



修改Path变量，在原来的Path后面追加

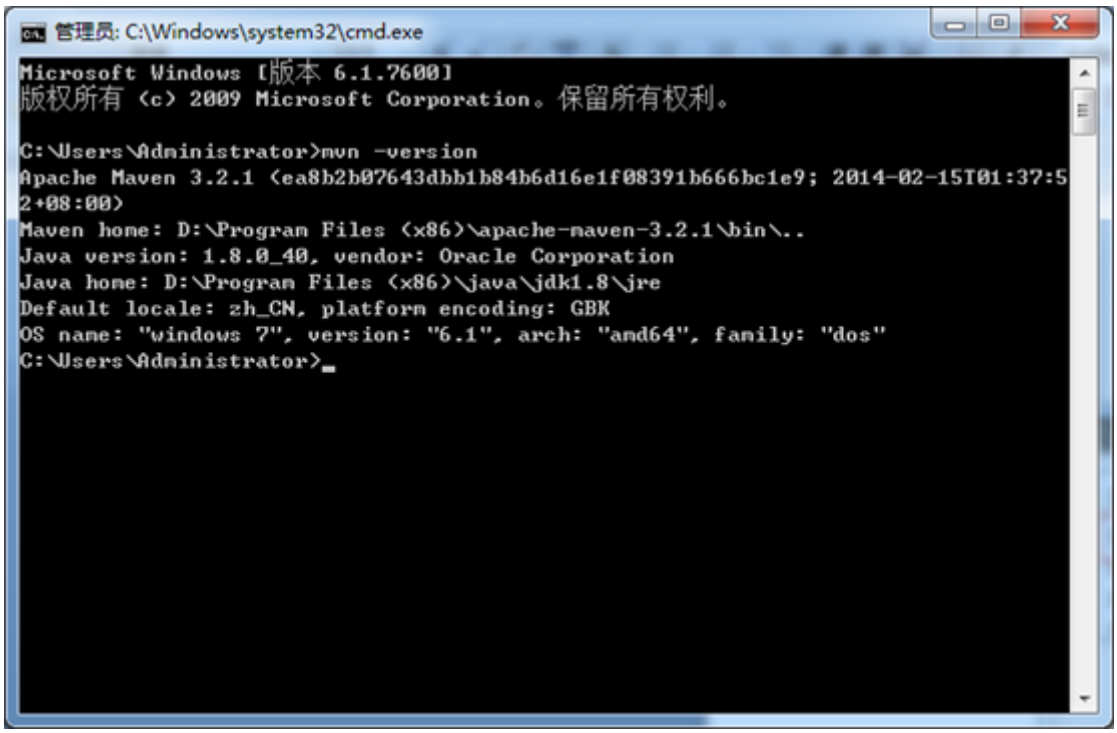
;%MAVEN_HOME%\bin\;

注意前面有个分号;



1.4 测试

打开命令窗口，运行mvn –version



maven安装成功

2. 配置JDK版本

使用Maven创建的java项目，需要可以在配置文件中设置默认使用的jdk版本。

修改Maven安装目录的conf/setting.xml文件,找到配置jdk的节点：

注意不要直接复制，因为不同版本的maven中配置不同，找到你原来的节点，在原来的基础上修改。

```
<profile>
 <id>jdk18</id>
 <activation>
  <activeByDefault>true</activeByDefault>
  <jdk>1.8</jdk>
 </activation>
 <properties>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
 <maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>
 </properties>
</profile>
```

上面的配置指定的JDK版本是1.8

3. 配置镜像

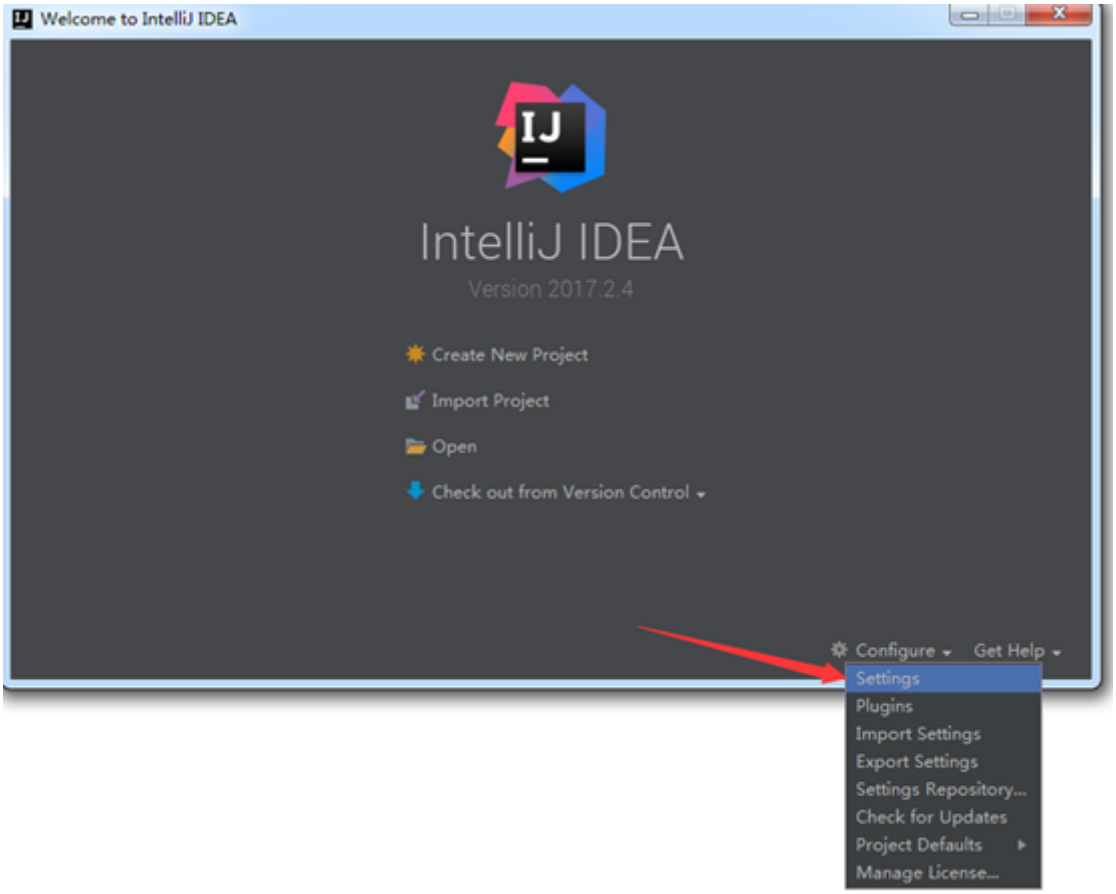
Maven的中央仓库是在国外的，如果网络有限制，会导致jar包下载的很慢或者无法下载。可以在网上找一些国内的镜像：

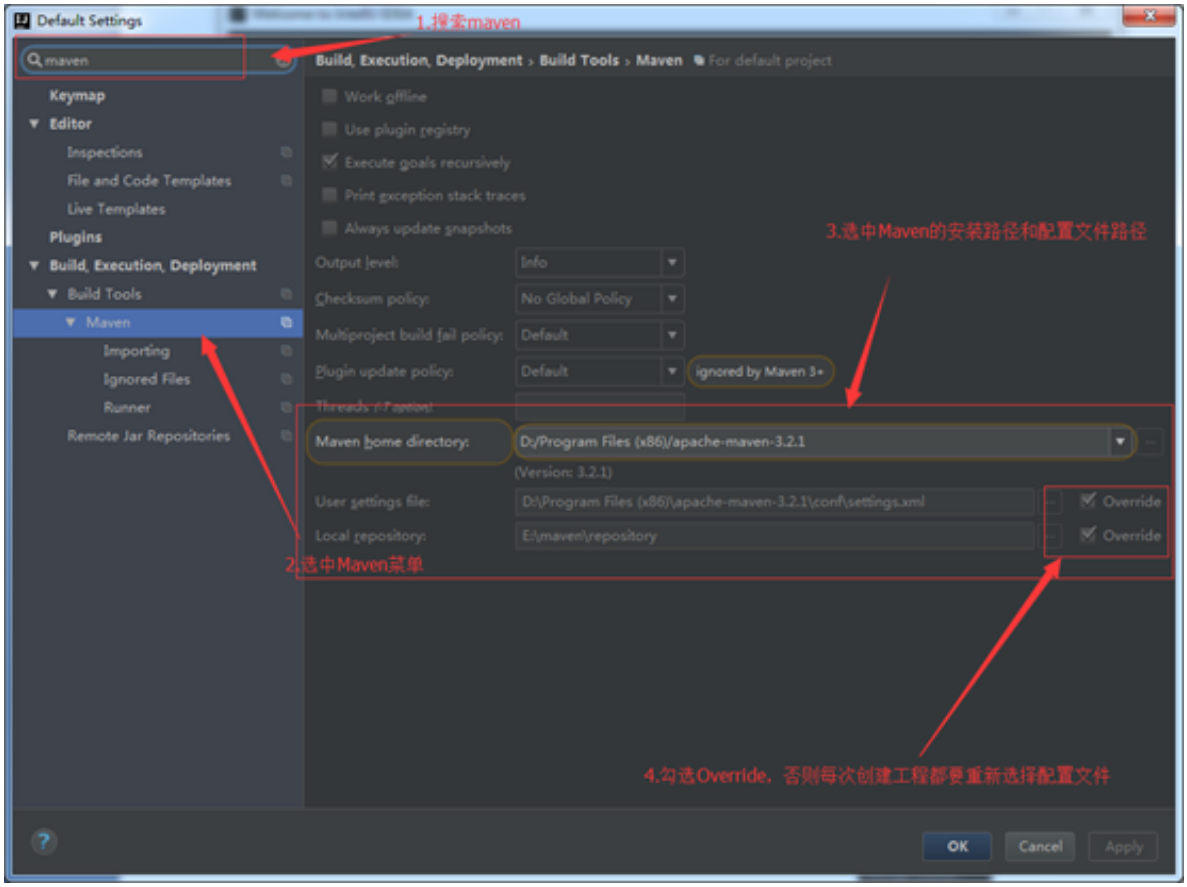
找到setting.xml的节点，加入如下内容：

```
<mirror>
  <id>alimaven</id>
  <name>aliyun maven</name>
<url>http://maven.aliyun.com/nexus/content/groups/public/</url>
  <mirrorOf>central</mirrorOf>
</mirror>
```

上面的配置是在网上找的一个阿里云的镜像，也可以自己搜其它的。

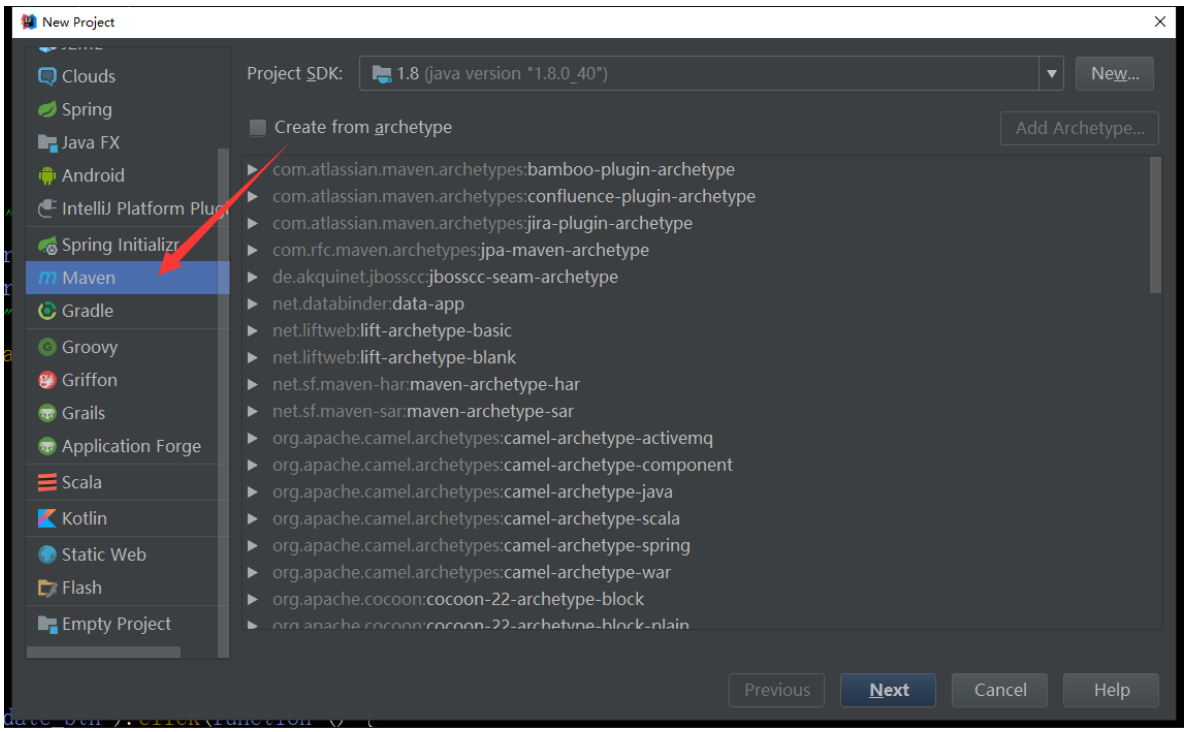
4.IDEA配置


Welcome to IntelliJ IDEA
IntelliJ IDEA
Version 2017.2.4
Create New Project
Import Project
Open
Check out from Version Control
Configure
Get Help
Settings
Plugins
Import Settings
Export Settings
Settings Repository...
Check for Updates
Project Defaults
Manage License...

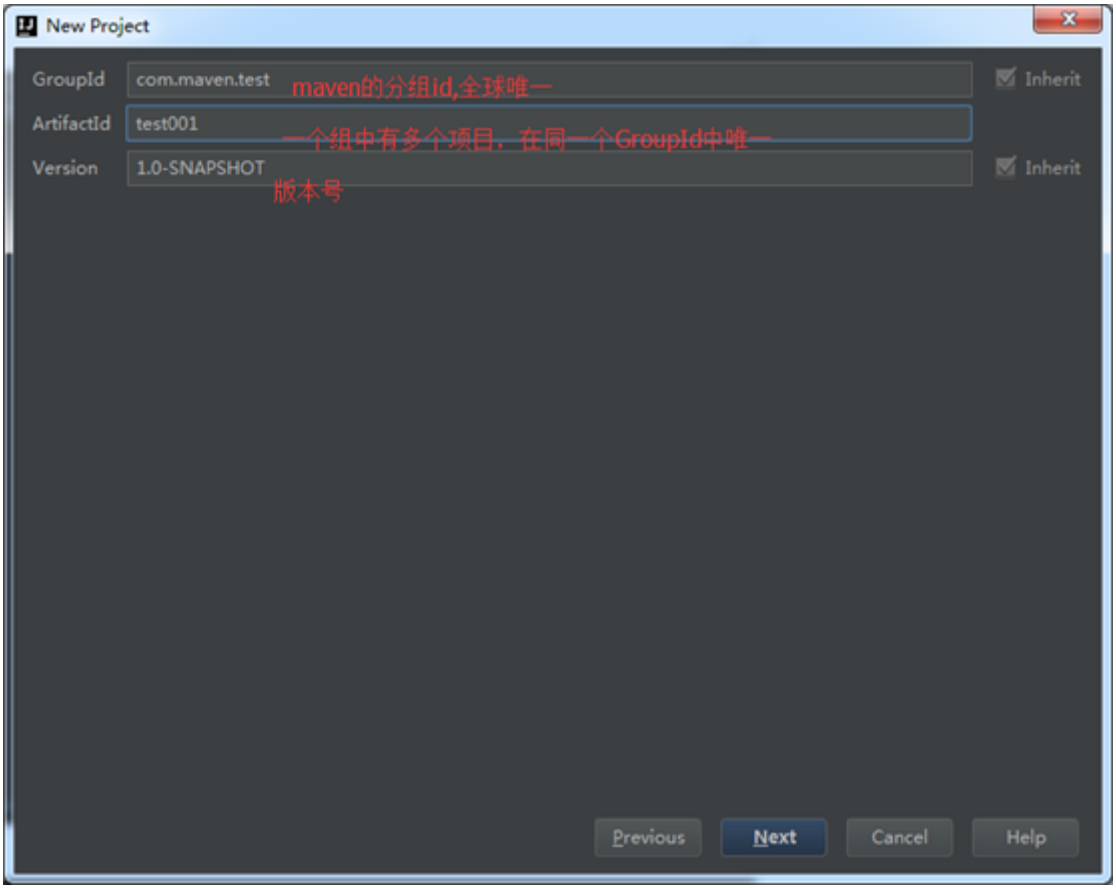

Default Settings
1.搜索maven
maven
Keymap
Editor
Inspections
File and Code Templates
Live Templates
Plugins
Build, Execution, Deployment
Build Tools
Maven
Importing
Ignored Files
Runner
Remote Jar Repositories
Build, Execution, Deployment › Build Tools › Maven
For default project
Work offline
Use plugin registry
Execute goals recursively
Print exception stack traces
Always update snapshots
Output level:
Info
Checksum policy:
No Global Policy
Multiproject build fail policy:
Default
Plugin update policy:
Default
ignored by Maven 3+
3.选中Maven的安装路径和配置文件路径
Maven home directory:
D:/Program Files (x86)/apache-maven-3.2.1
(Version: 3.2.1)
User settings file:
D:\Program Files (x86)\apache-maven-3.2.1\conf\settings.xml
Override
Local repository:
E:\maven\repository
Override
2.选中Maven菜单
4.勾选Override，否则每次创建工程都要重新选择配置文件
OK
Cancel
Apply

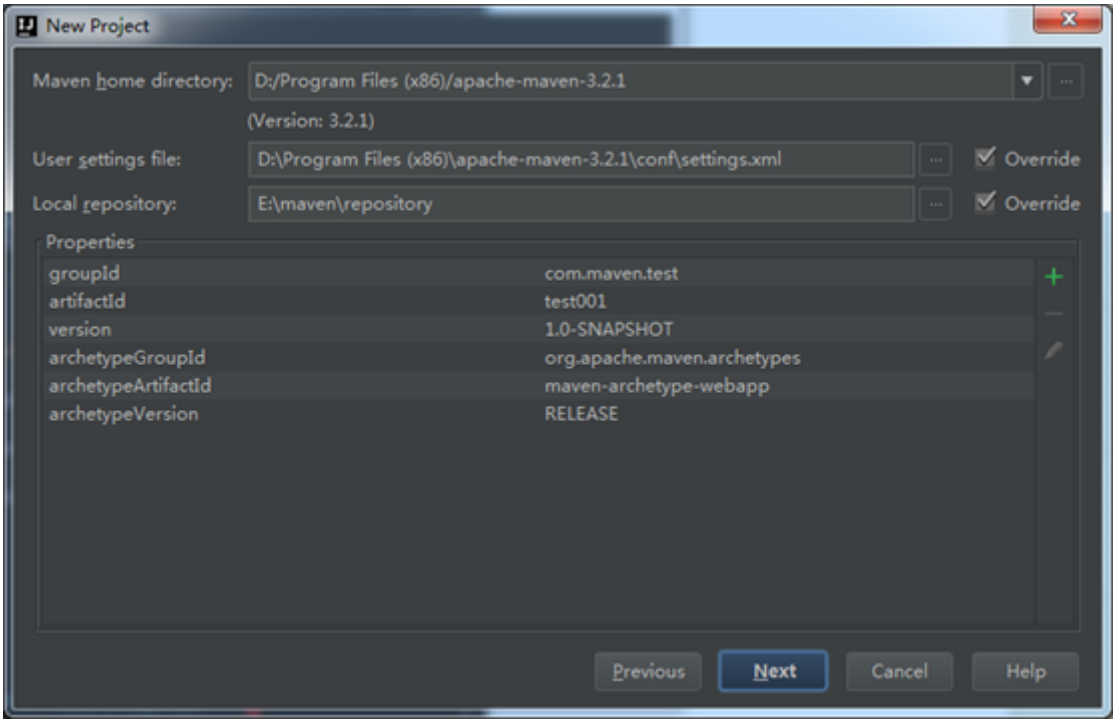
5. 创建工程

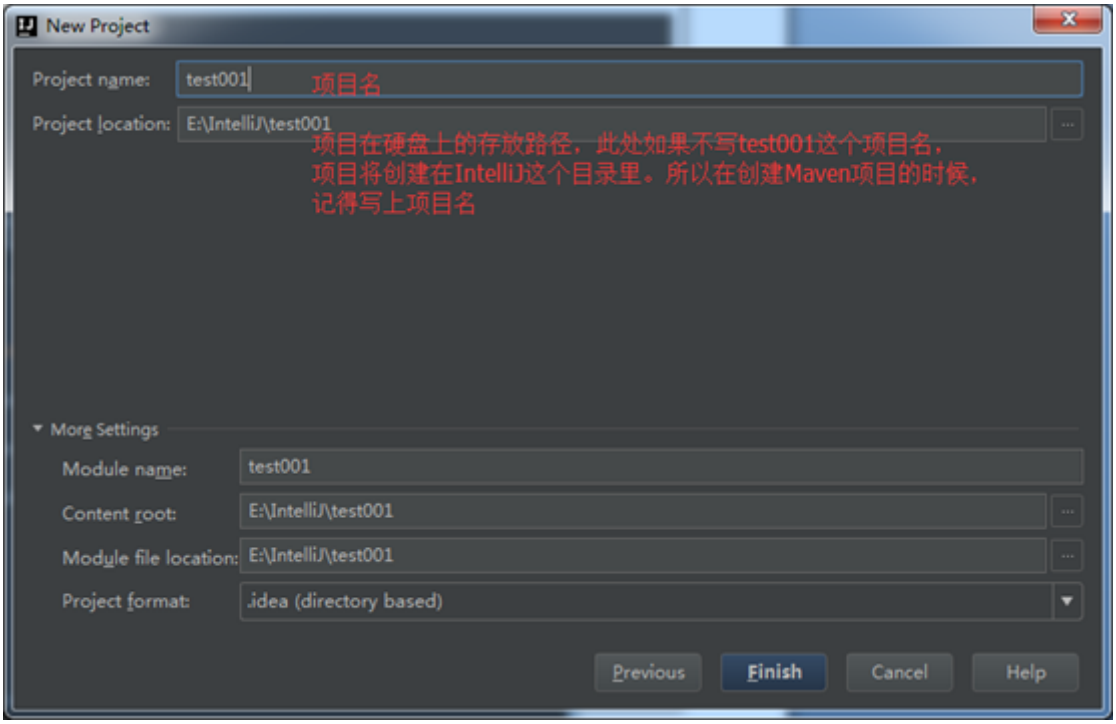
5.1 maven工程

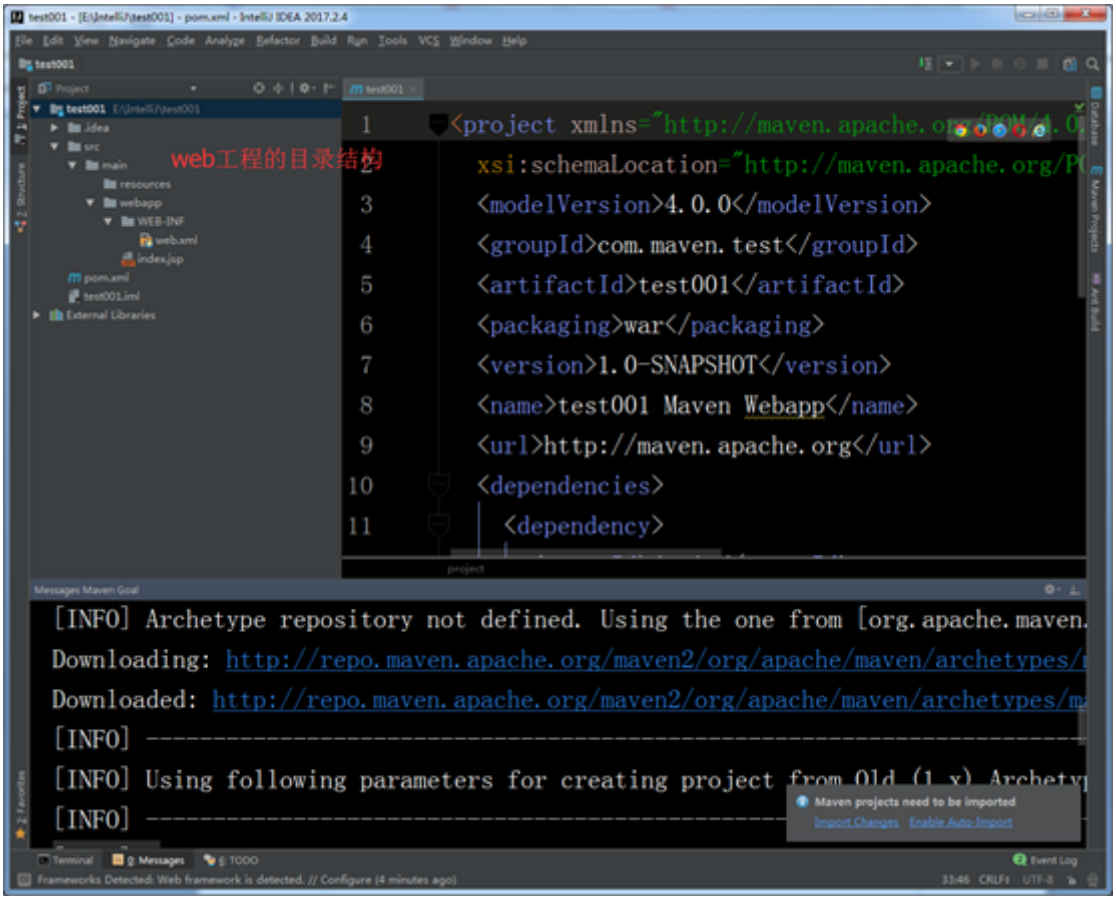


点击Next




New Project
Maven home directory:
D:/Program Files (x86)/apache-maven-3.2.1
(Version: 3.2.1)
User settings file:
D:\Program Files (x86)\apache-maven-3.2.1\conf\settings.xml
Override
Local repository:
E:\maven\repository
Override
Properties
groupId
com.maven.test
artifactId
test001
version
1.0-SNAPSHOT
archetypeGroupId
org.apache.maven.archetypes
archetypeArtifactId
maven-archetype-webapp
archetypeVersion
RELEASE
Previous
Next
Cancel
Help


New Project
Project name:
test001
项目名
Project location:
E:\IntelliJ\test001
项目在硬盘上的存放路径，此处如果不写test001这个项目名，
项目将创建在IntelliJ这个目录里。所以在创建Maven项目的时候，
记得写上项目名
More Settings
Module name:
test001
Content root:
E:\IntelliJ\test001
Module file location:
E:\IntelliJ\test001
Project format:
.idea (directory based)
Previous
Finish
Cancel
Help



如果创建成功，会在IntelliJ中看到完整的目录结构。

如果创建失败，可以在控制台中查看EventLog

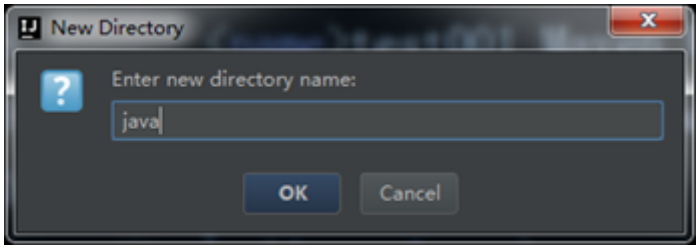
有些资源Maven需要联网下载，如果download失败，项目可能创建失败。

如果创建同名的工程也会报错，如下：

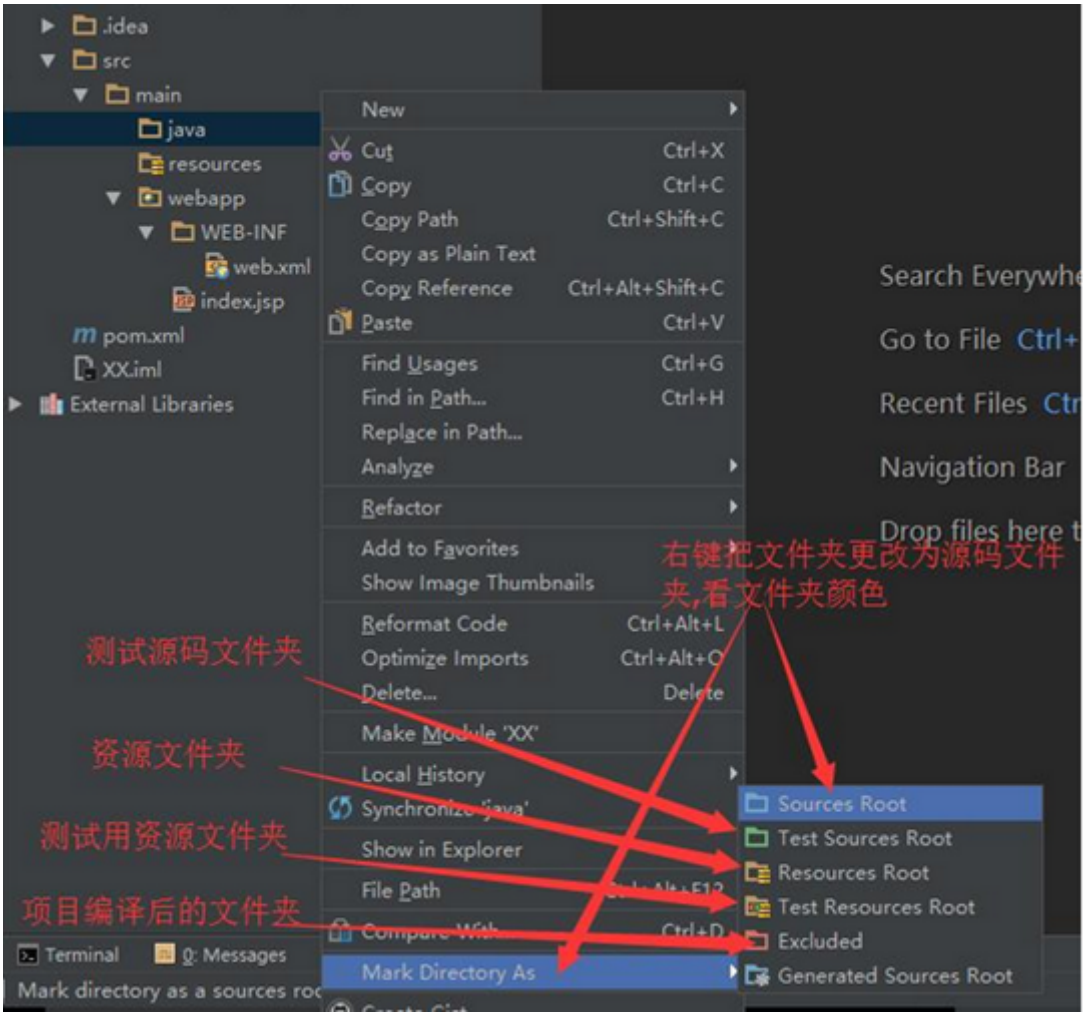
此时需要删除工程目录下对应的文件夹，并删除回收站中对应的文件。

5.2 创建目录

右键单击main->New->Directory



右键java文件夹->Mark Directory As ->Sources Root将文件夹更改成源码文件

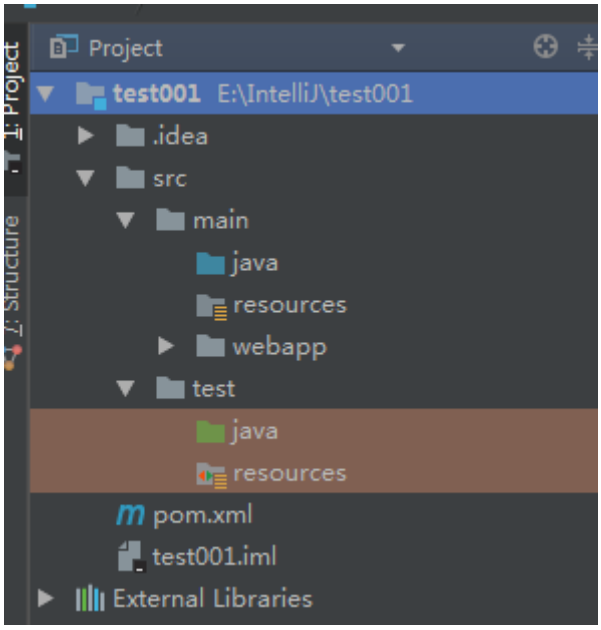


右键resources->Mark Directory As –>Resources Root

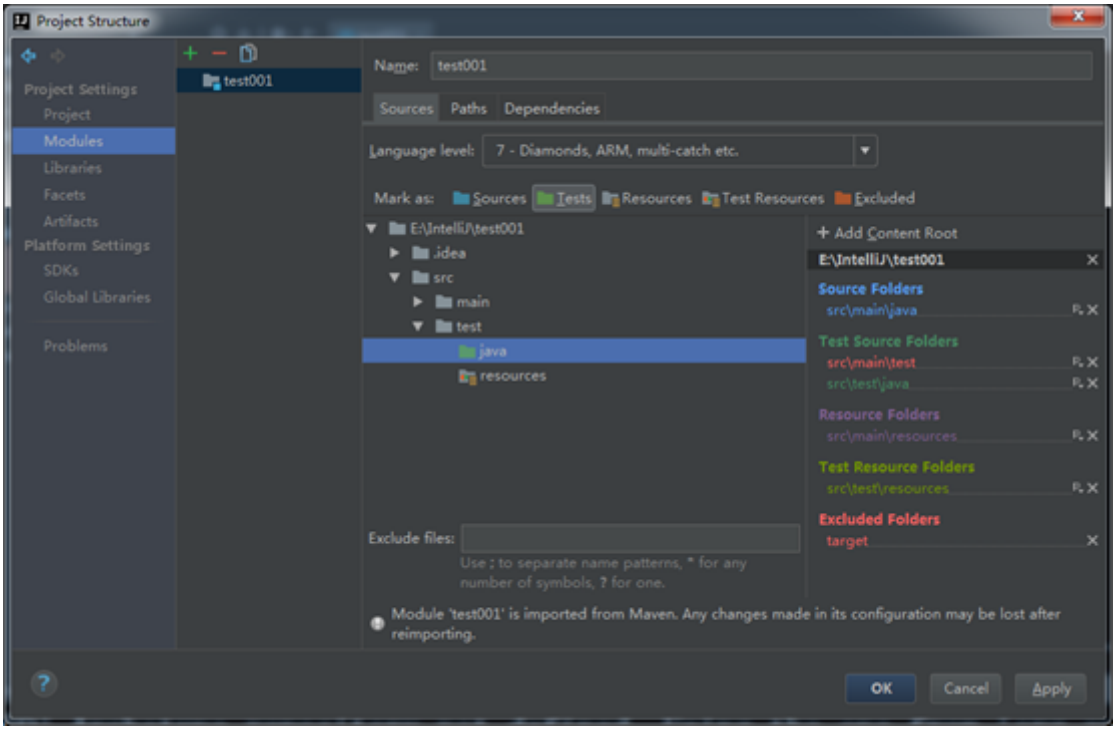
java文件夹下用于存放package，相当于eclipse普通工程中的src目录。

resources中用于存放配置文件和属性文件。

还可以在src下创建test文件夹，用于junit测试。test下同样创建java和resources



也可以右键项目-选择Open Module Settings打开项目配置页面更改



5.3 maven配置文件

pom.xml是maven的核心配置文件，在pom.xml中可以定义工程的依赖关系，和引用jar包的信息。

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

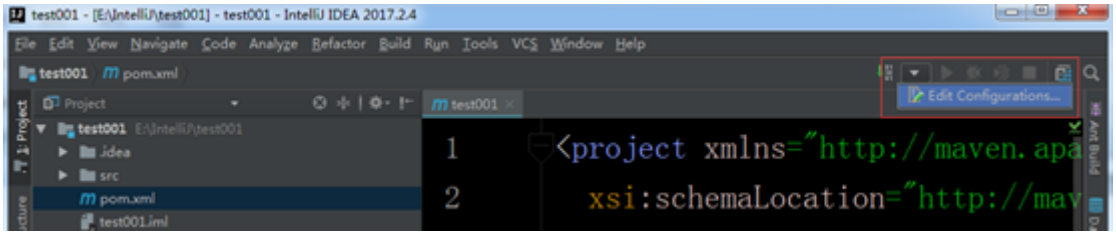
这里引用了junit的jar包，版本是4.12

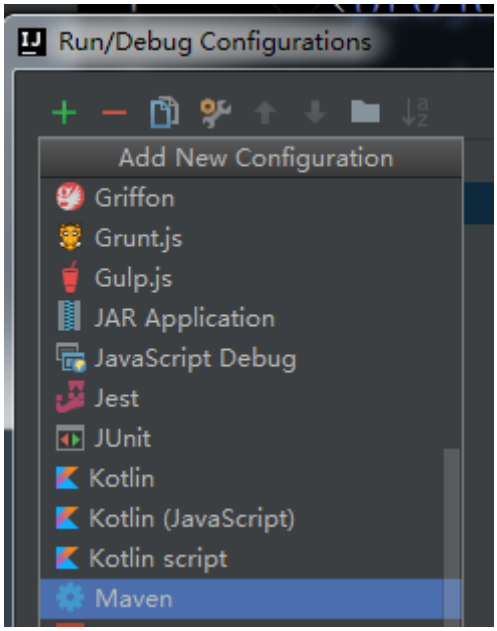
5.4 配置插件

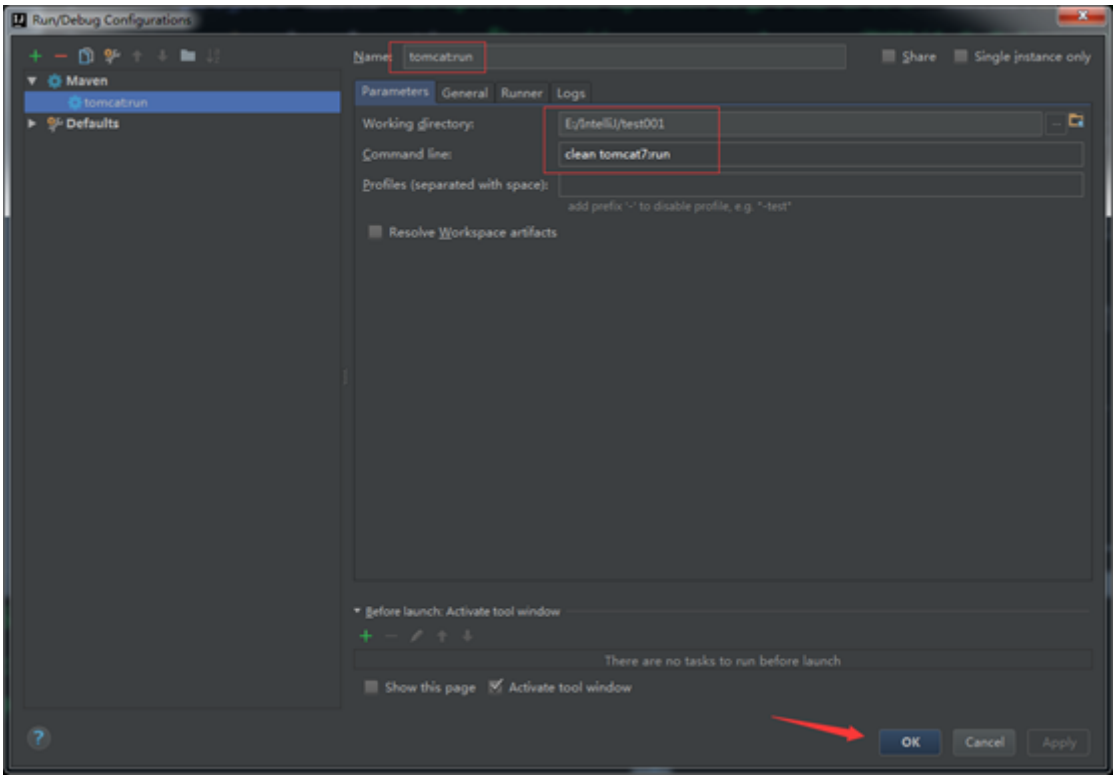
```
<build>
  <finalName>test001</finalName>
  <plugins>
    <plugin>
      <groupId>org.apache.tomcat.maven</groupId>
      <artifactId>tomcat7-maven-plugin</artifactId>
      <configuration>
        <path>/</path>
        <port>8080</port>
      </configuration>
    </plugin>
  </plugins>
</build>
```

pligins用于定义插件，此处引用了tomcat的插件。path是访问路径，如果我们同时运行多个项目，访问路径都是根路径，可以通过修改port端口号来访问项目。

5.5 测试

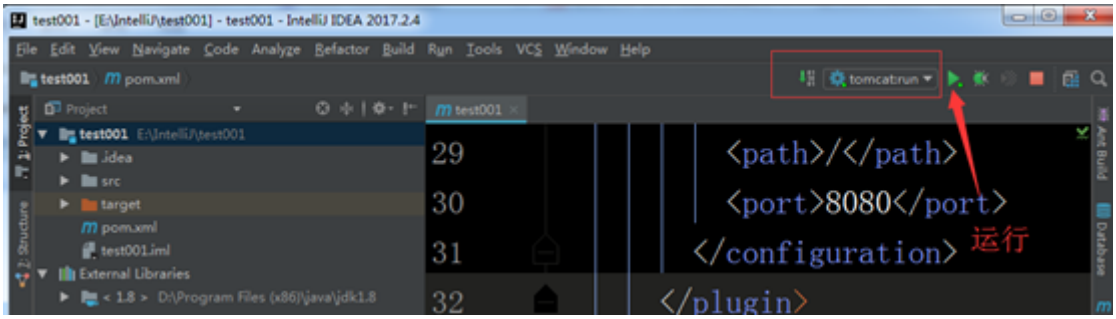






命令clean tomcat7:run

配置成功后可以在工具栏里看到刚才配置的命令，点击绿色的三角

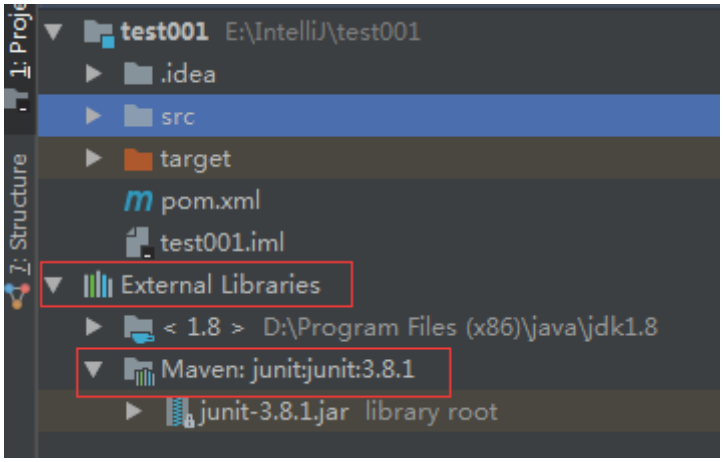


在浏览器里可以看到：

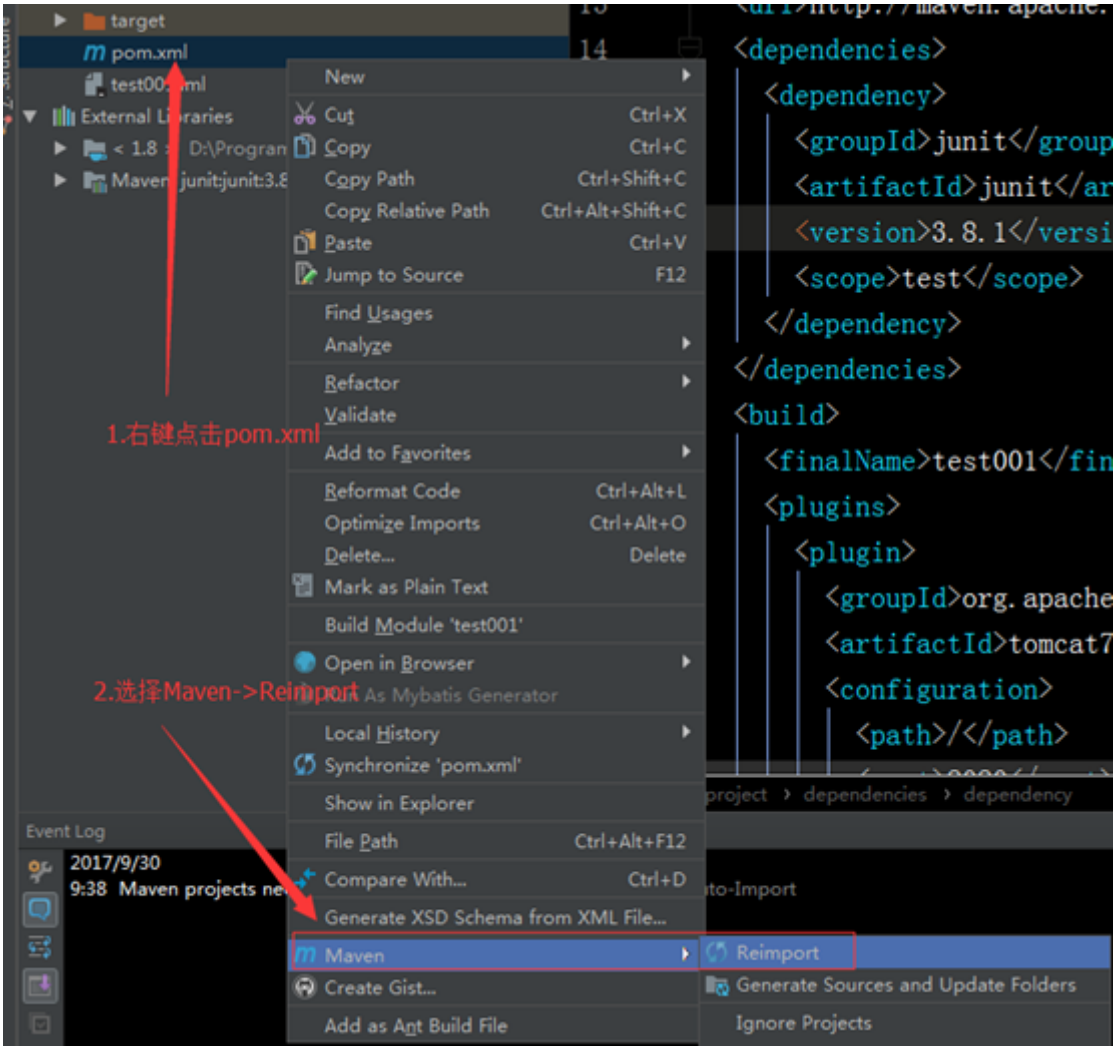
5.6 查看jar包

通常，在pom.xml中加入后maven会自动下载jar包

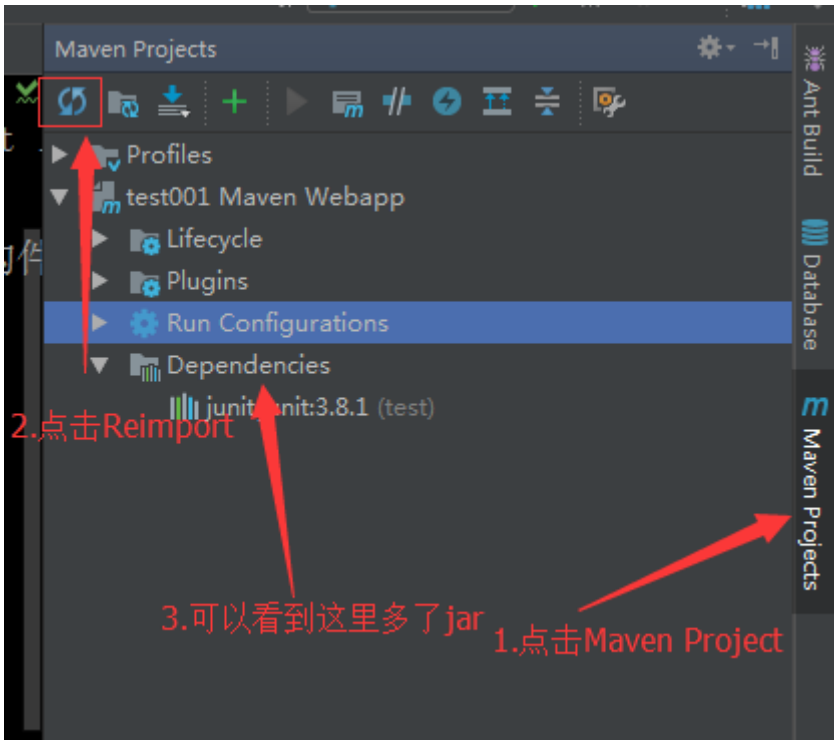
在项目的External Libraries中可以看到maven下载的jar:



如果在pom.xml中加入后，在左侧看不到jar包，可以右键点击pom.xml，选择Maven->Reimport



或者在IDEA右侧点击Maven Project：



jar包没有正确引入的时候，pom.xml中会出现错误提示：

未正确引用的依赖，颜色是不一样的。具体提示的颜色与IntelliJ的主题有关。

5.7 排除jar冲突

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <!-- 排除冲突jar包 -->
    <exclusions>
        <exclusion>
            <groupId>org.hamcrest</groupId>
            <artifactId>hamcrest-core</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

6.Maven继承

6.1 pom工程

pom工程中只有一个pom.xml文件，不包含java和resource目录，它只用来保存一些依赖信息。

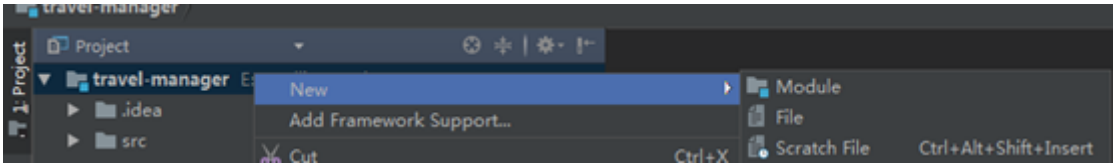
在pom.xml中添加依赖信息信息，如junit、mysql等，也可以定义插件信息：

```
<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>com.maven.test</groupId>
    <artifactId>test-parent</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>pom</packaging>
    <!-- 自定义属性 -->
    <properties>
        <junit.version>4.12</junit.version>
        <mysql.version>5.1.32</mysql.version>
    </properties>
    <!--继承自该项目的所有子项目的默认依赖信息。这部分的依赖信息不会被立即解析,而是当子项目声明一个依赖（必须描述group ID和artifact ID信息），如果group ID和artifact ID以外的一些信息没有描述，则通过group ID和artifact ID匹配到这里的依赖，并使用这里的依赖信息。-->
    <dependencyManagement>
        <dependencies>
            <!--用于junit测试 -->
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>${junit.version}</version>
            </dependency>
            <!-- mysql驱动包 -->
            <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <version>${mysql.version}</version>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <build>
        <!--产生的构件的文件名，默认值是${artifactId}-${version}。-->
        <finalName>${project.artifactId}</finalName>
         <!--使用的插件列表 。-->
        <plugins>
            <!-- 资源文件拷贝插件 -->
            <plugin> <!--plugin元素包含描述插件所需要的信息。-->
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <version>2.7</version>
                <!--作为DOM对象的配置-->
                <configuration>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
            <!-- java编译插件 -->
            <plugin>
```

```
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.2</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
        </plugins>
        <!--子项目可以引用的默认插件信息。该插件配置项直到被引用时才会被解析或绑定到生命周期。给定插件的任何本地配置都会覆盖这里的配置-->
        <pluginManagement>
            <plugins>
                <!-- 配置Tomcat插件 -->
                <plugin>
                    <groupId>org.apache.tomcat.maven</groupId>
                    <artifactId>tomcat7-maven-plugin</artifactId>
                    <version>2.2</version>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>
</project>
```

6.2 创建子工程

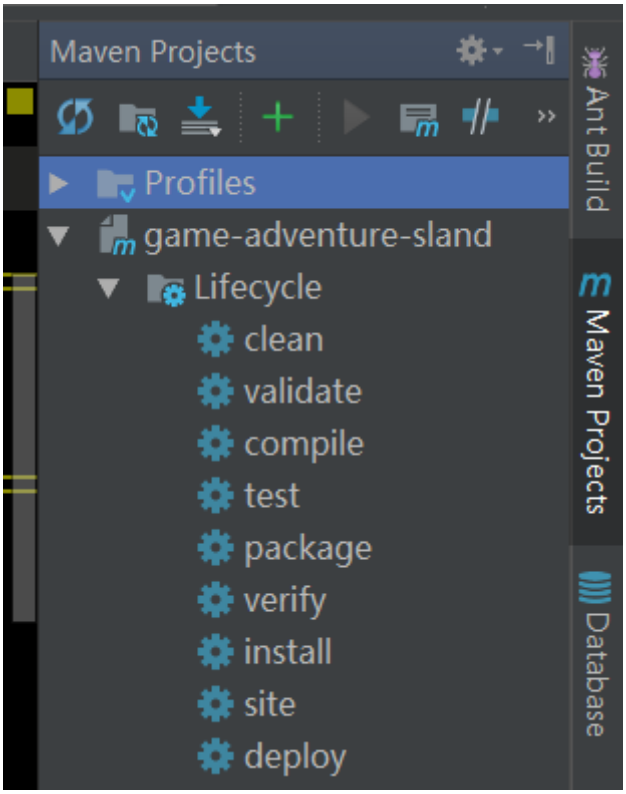


在子工程中添加父工程里定义过的依赖：

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
    </dependency>
</dependencies>
```

可以不用写版本号了，它会自动继承父工程里定义的版本。

7.打包



clean 清除编译结果

package 打包

install 安装到本地仓库