Maven
多模块管理

北京动力节点教育科技有限公司

动力节点课程讲义

DONGLIJIEDIANKECHENGJIANGYI

www.bjpowernode.com

第1章 Maven 管理多模块应用

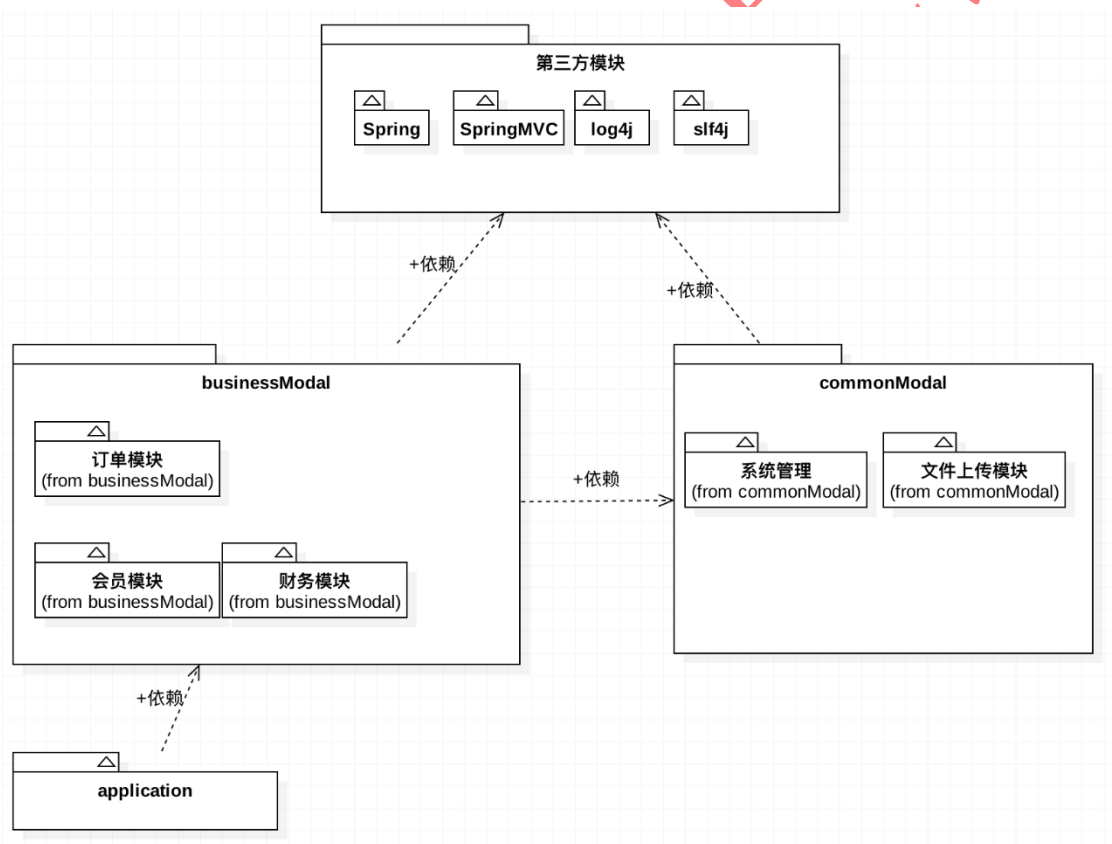
1.1 场景描述

commonModel：提供公共的基础服务，比如工具类、常量类等等；

bussinessModel：业务模块，是系统真正要实现的业务，依赖于 common 模块，比如订单管理、财务统计、会员管理等；

application：可发布的 web 应用，由各个 bussinessModel 组成，最终满足项目整体需求；

第三方模块：包括各类框架，Spring、MyBatis、日志等。整个应用都是依赖它们完成开发的；



第2章 如何使用 Maven 管理以上的结构呢？

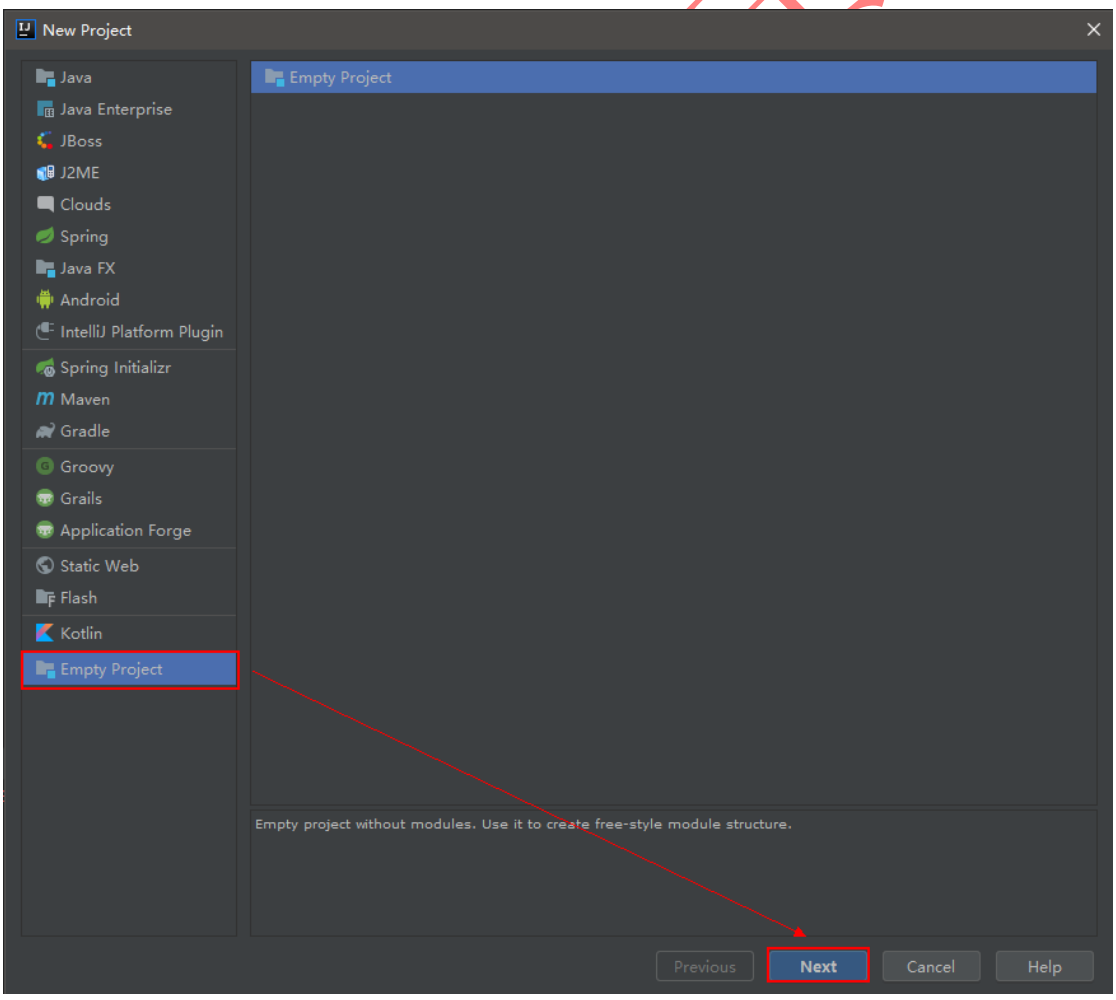
Maven 管理多模块应用的实现是互联网项目中多使用分布式开发，那么每个独立的服务都会使用独立的项目进行维护，那么这样就需要使用多模块应用管理，来实现项目的高度统一。

2.1 第一种实现方式

项目名称：maven-modules-project
完成功能：使用 IntelliJ IDEA 实现 Maven 管理多模块的应用开发

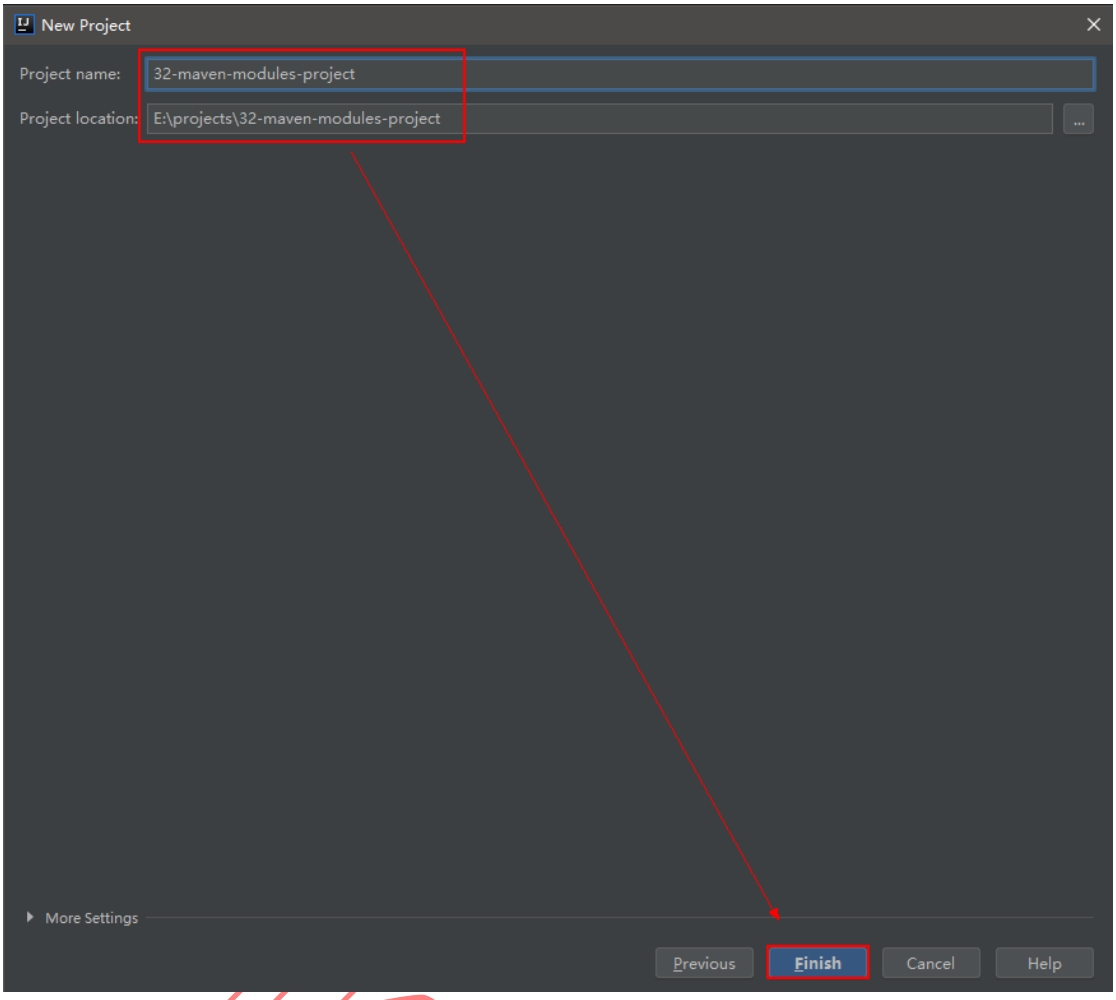
2.1.1 创建 Project 为 Empty Project

点击“Next”下一步



2.1.2 设置项目名称和项目存放位置

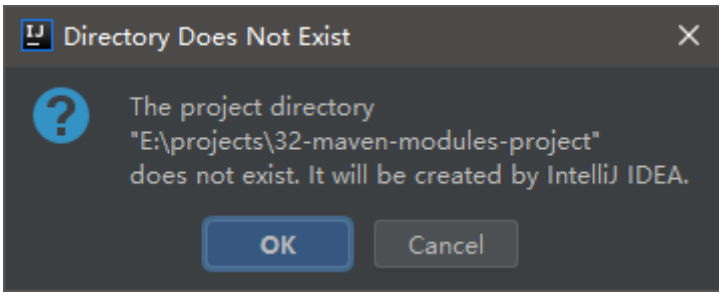
为新创建的项目设置项目名称和项目位置，然后点击“Finish”。



Project name：项目名称
Project location：项目存放位置

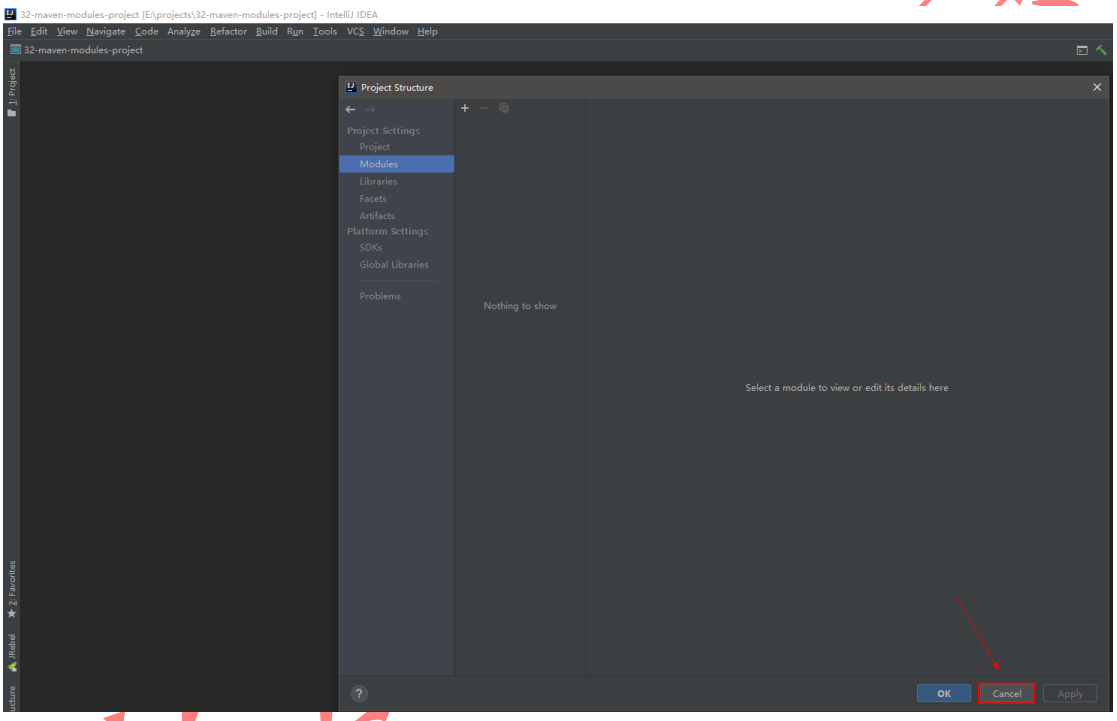
2.1.3 “文件夹不存在”提示框

提示内容如下：
项目目录“E:\项目\32 Maven 模块项目”不存在。它将由 IntellijIDEA 创建。
点击“OK”

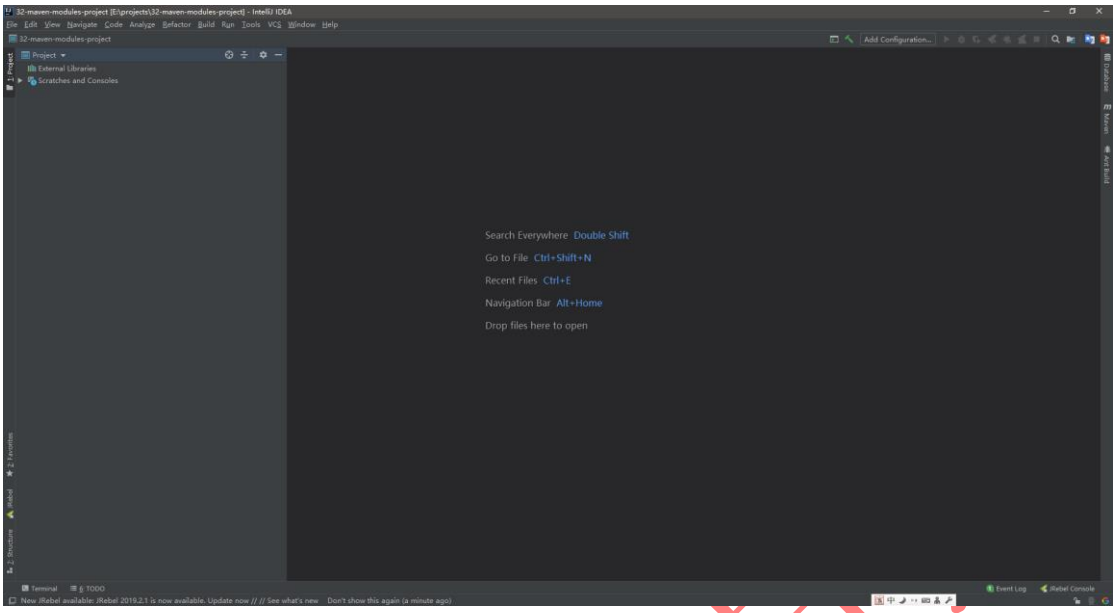


2.1.4 项目结构

选择“Cencel”取消

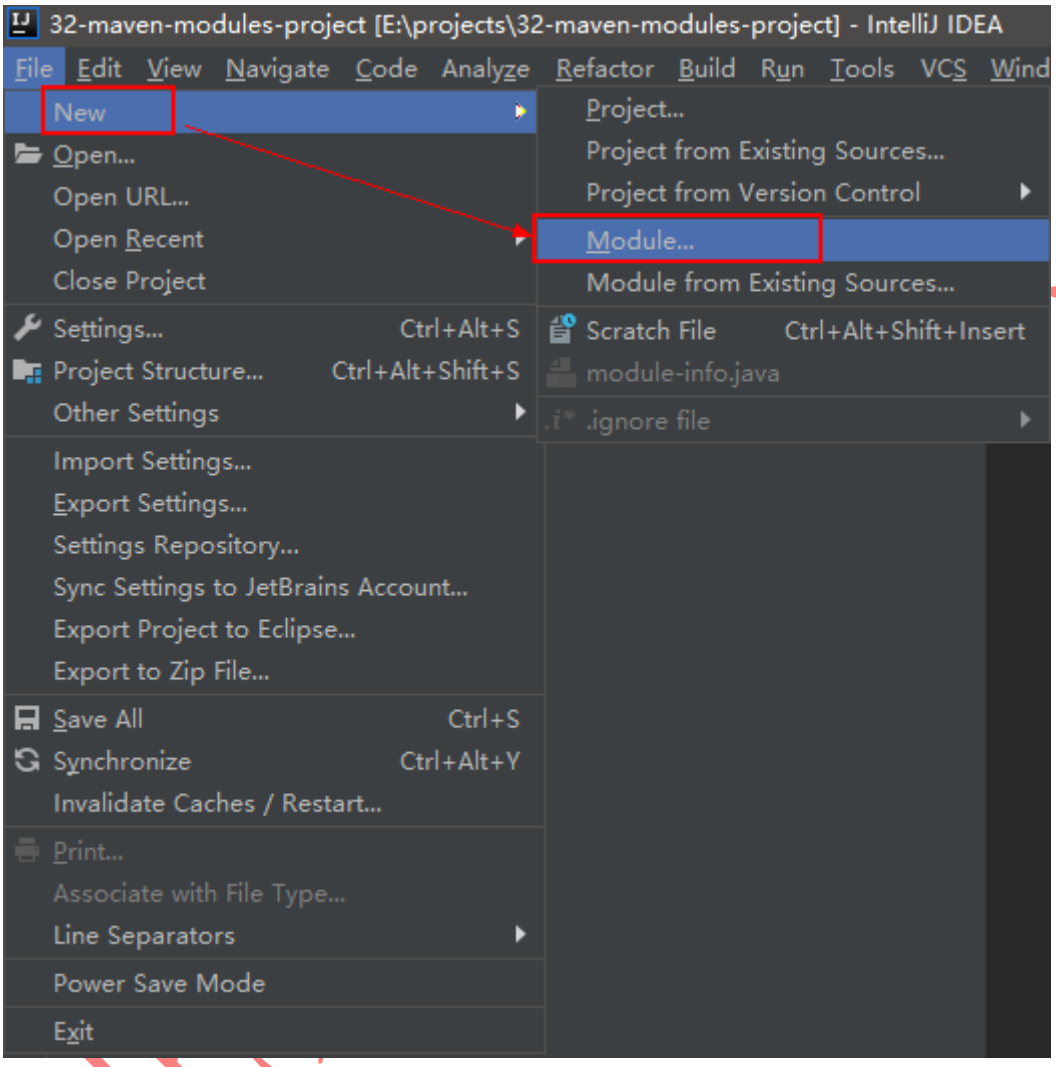


Maven 空项目创建成功，如下图

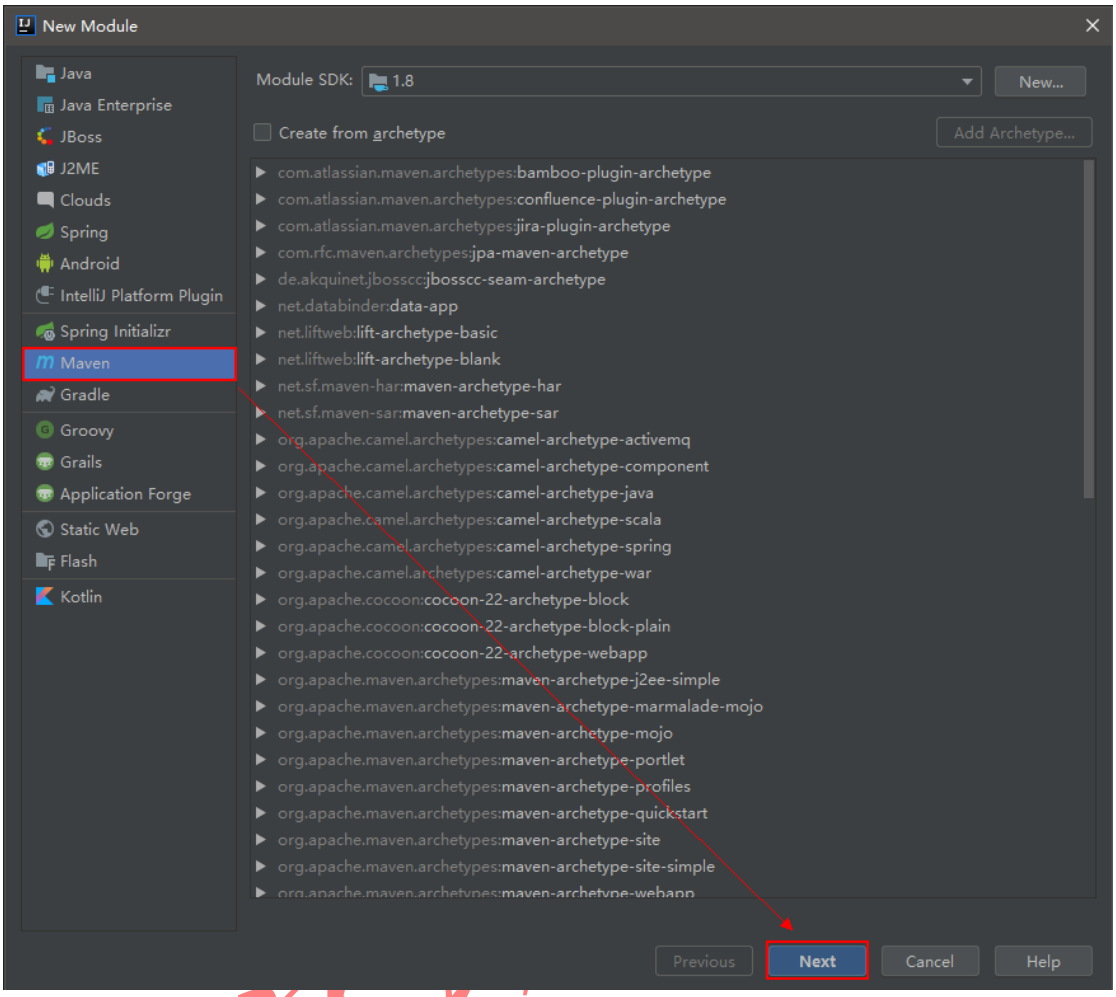


2.1.5 创建 Maven 父工程

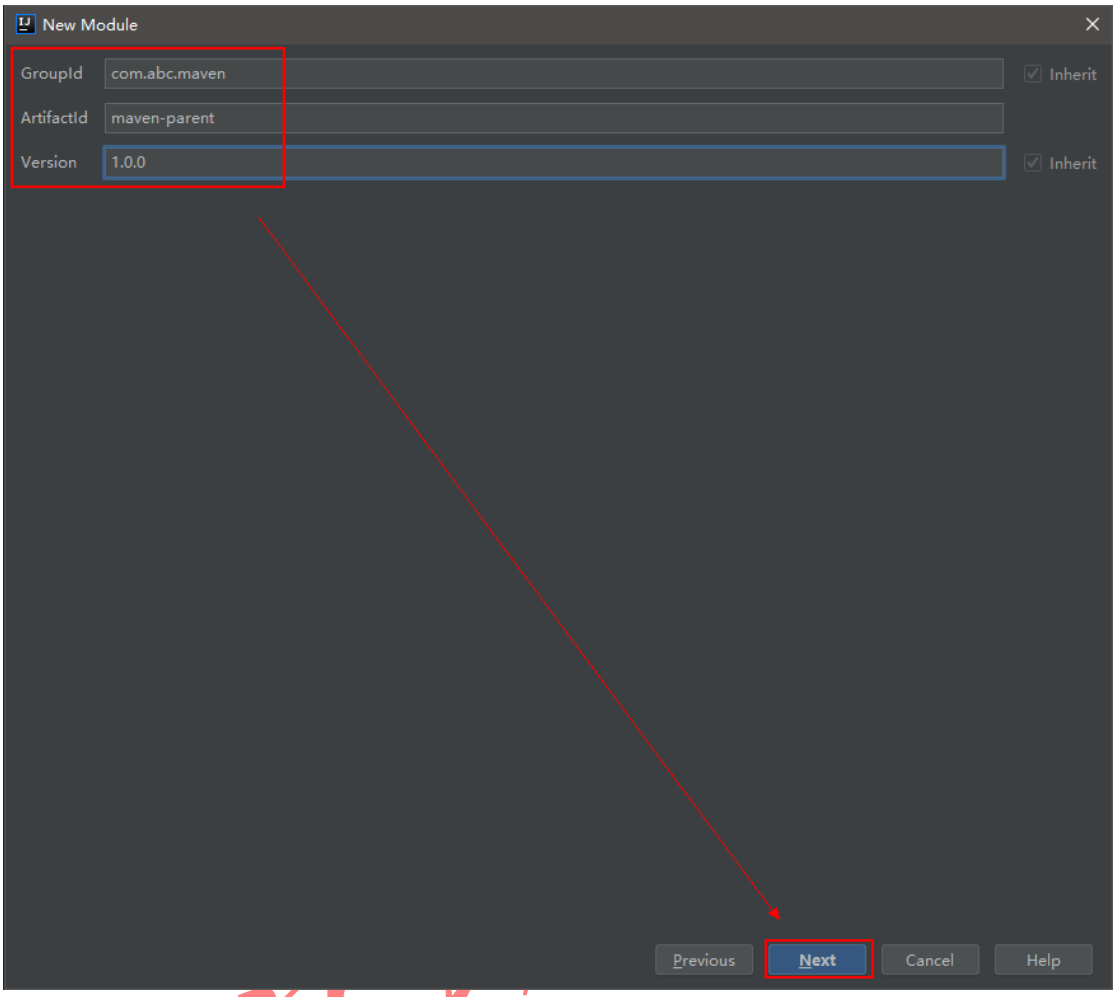
（1） 创建一个 Model 工程



（2） 选择 Maven 工程

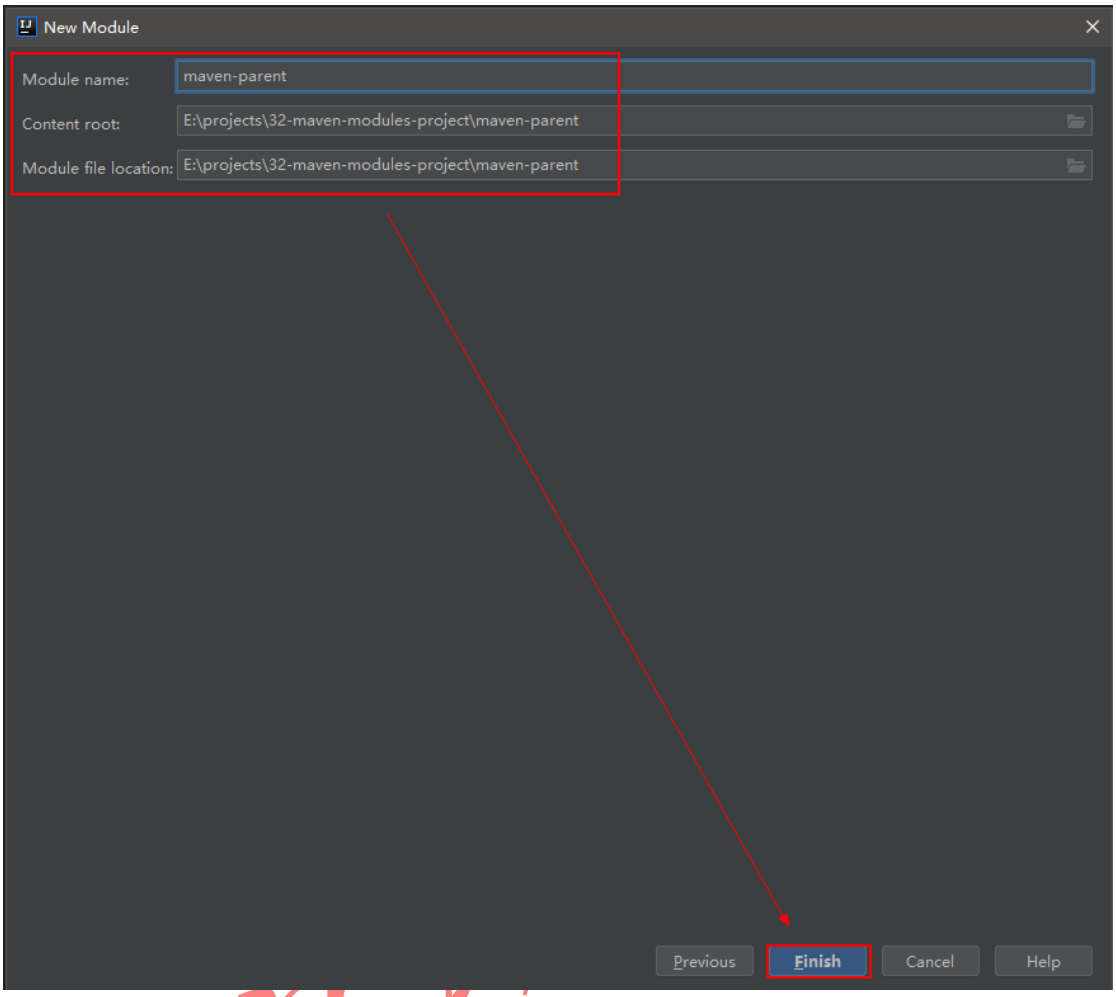


（3） 设置 Module 的 GAV 坐标



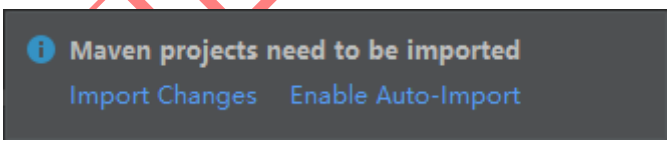
GroupId：公司域名的倒序
ArtifactId：项目或模块名称
Version：项目或模块版本号

（4） 模块内容存放位置



（5） 配置导入设置

Maven 项目被修改后，需要“手动更新”或“自动更新”，通常选择“Enable Auto-Import”



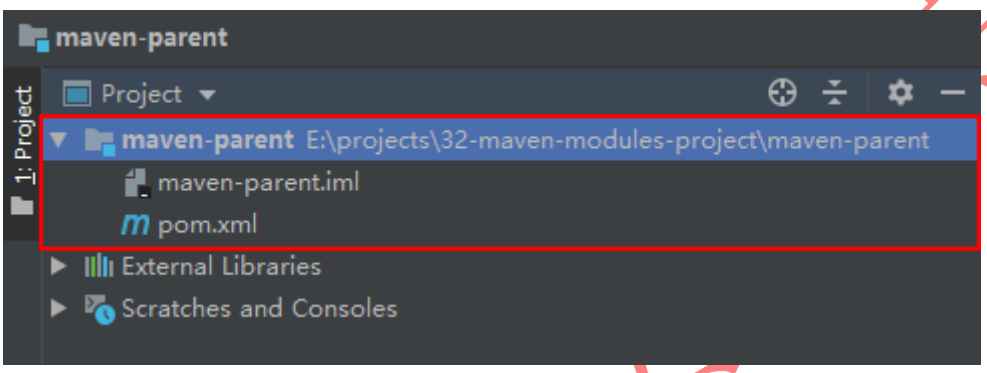
（6） 设置父工程的 pom 文件

父工程的 packaging 标签的文本内容必须设置为 pom。

```xml
<?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>com.abc.maven</groupId>
    <artifactId>maven-parent</artifactId>
    <version>1.0.0</version>

    <packaging>pom</packaging>

</project>
```

（7） 删除 src 目录

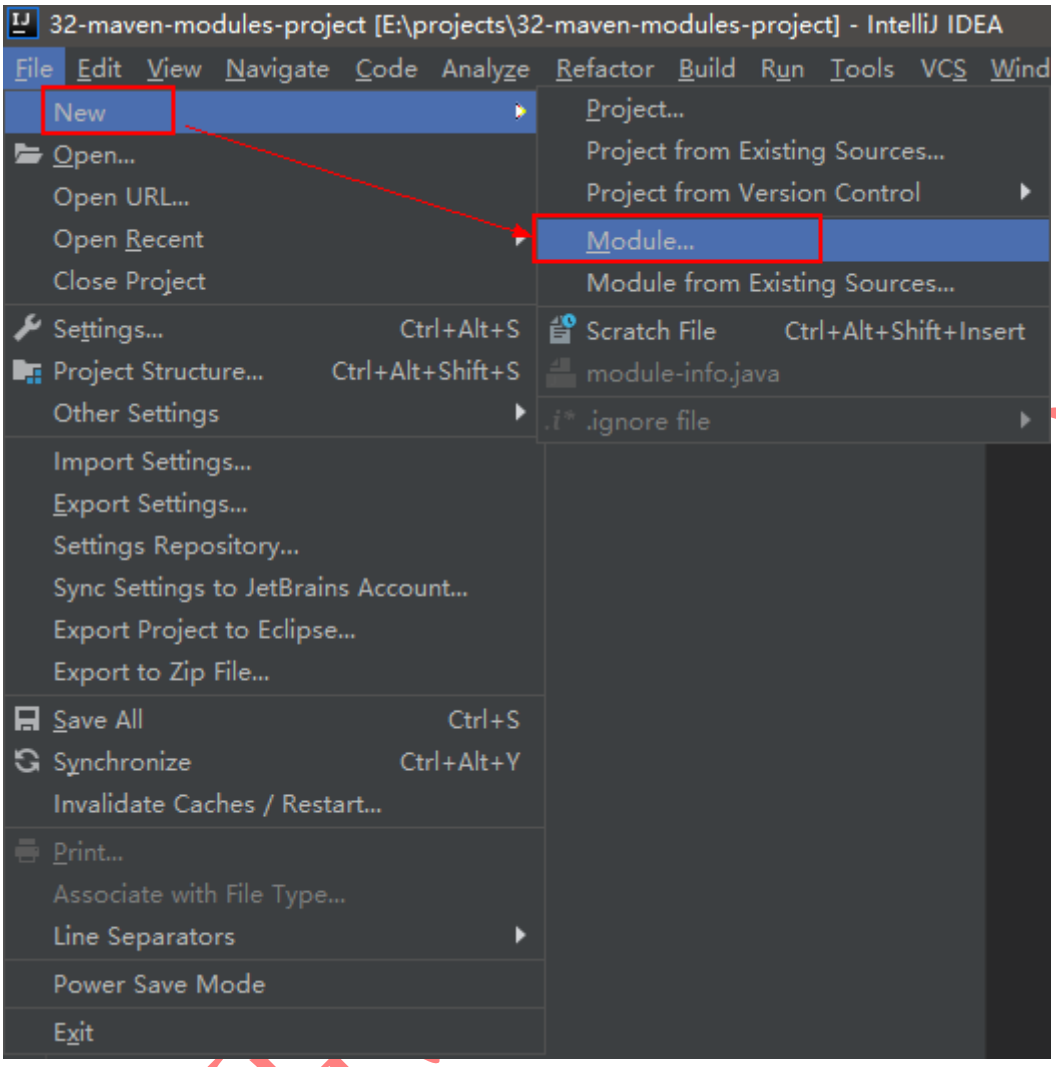
父工程要求 src 目录必须删除掉。



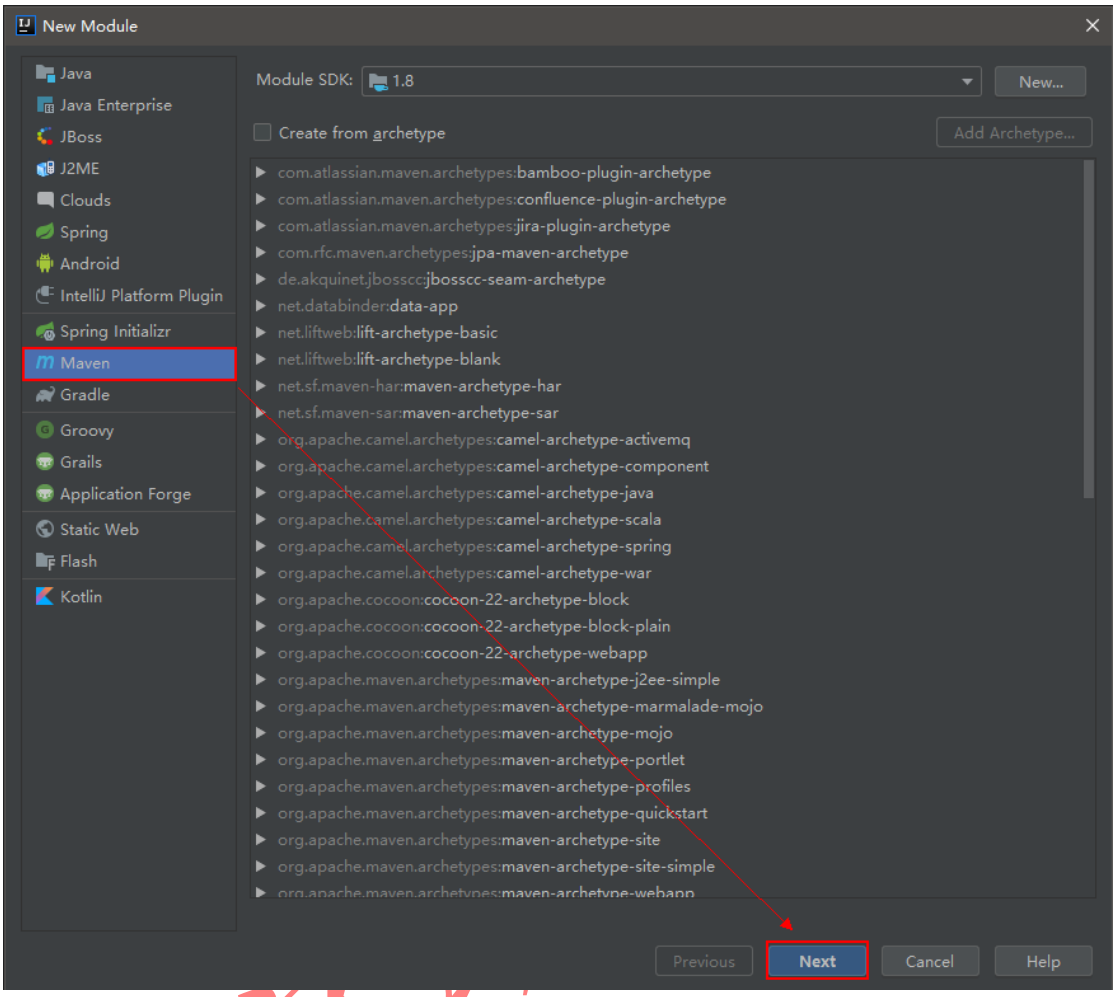
2.1.6 创建子模块

模块名称：maven-java-001，是 maven-parent 父工程的子模块

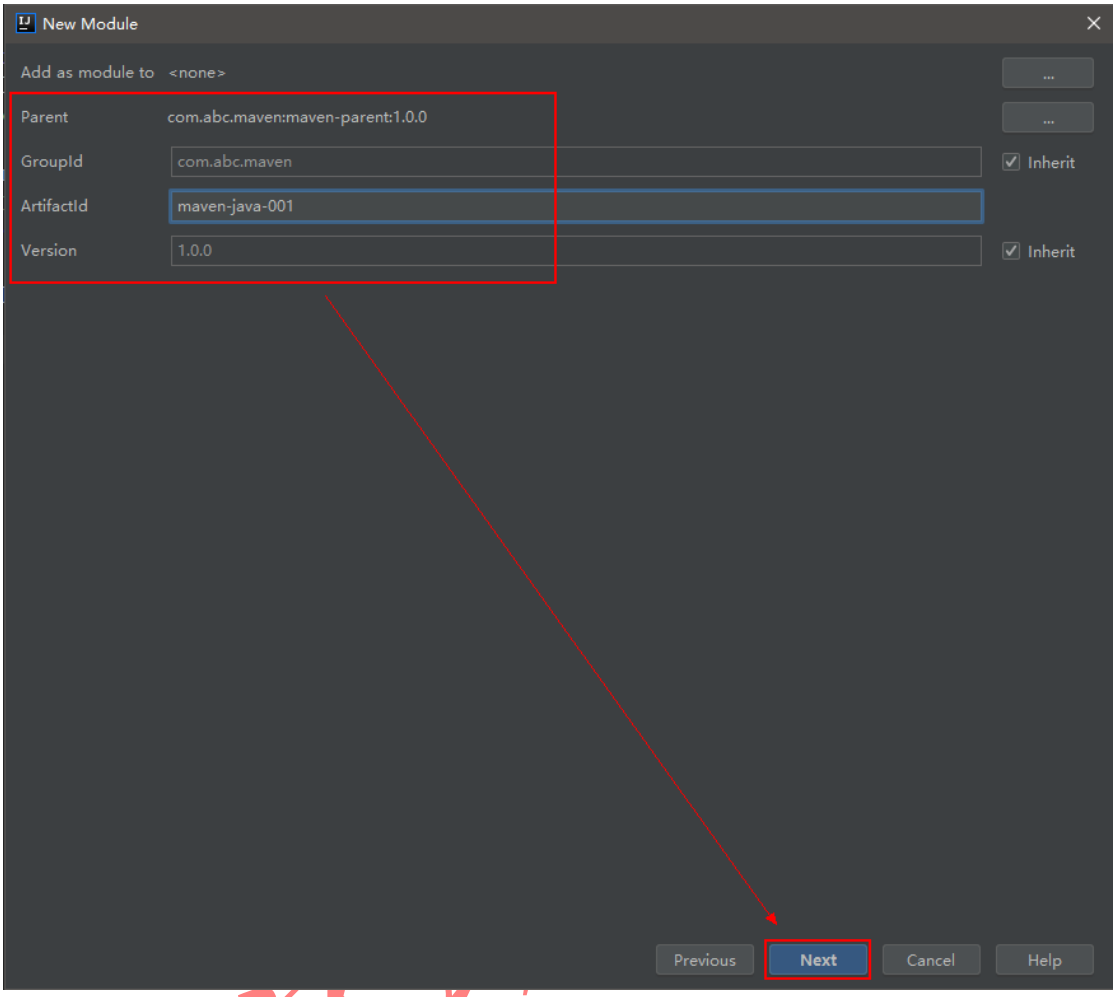
（1） 创建 module 工程



（2） 创建 maven java 工程

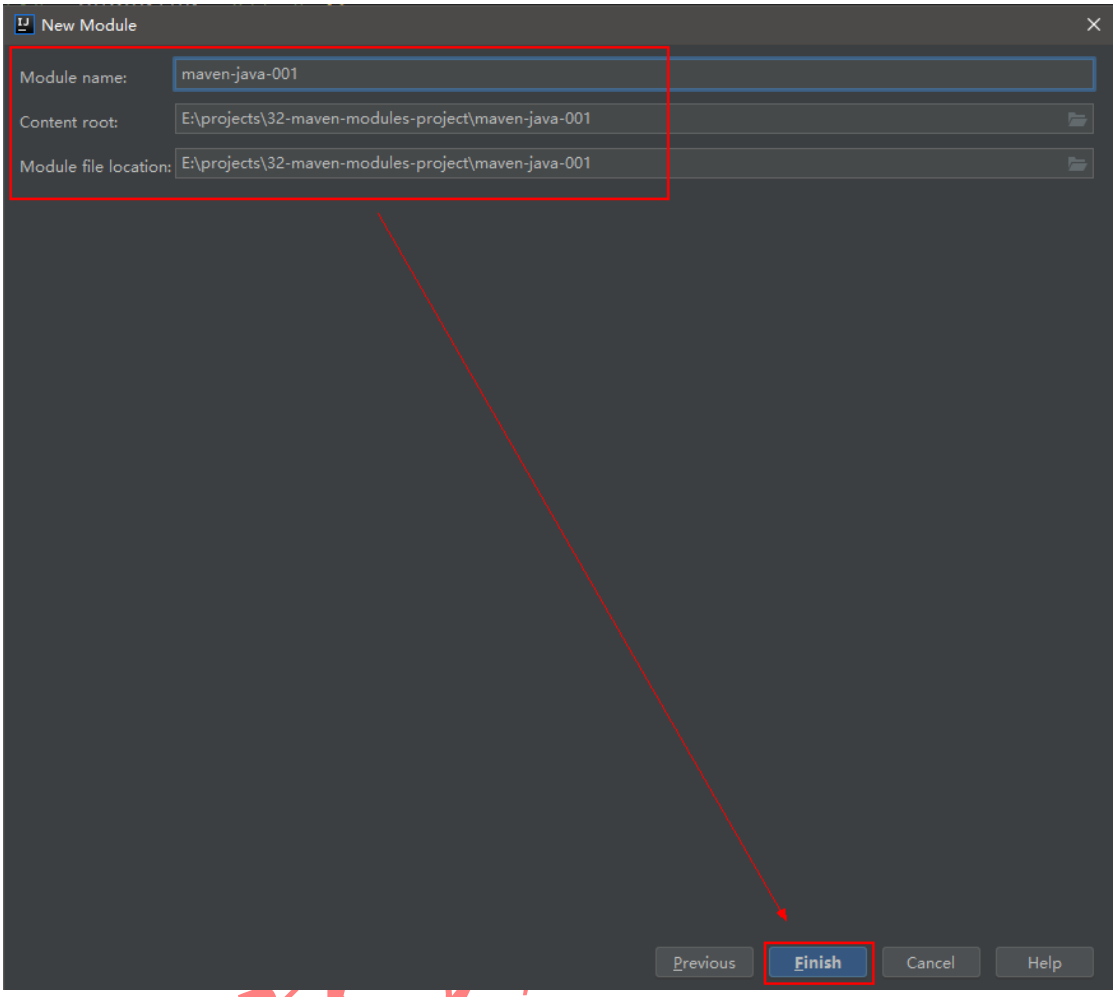


（3） 设置 module 项目基础信息

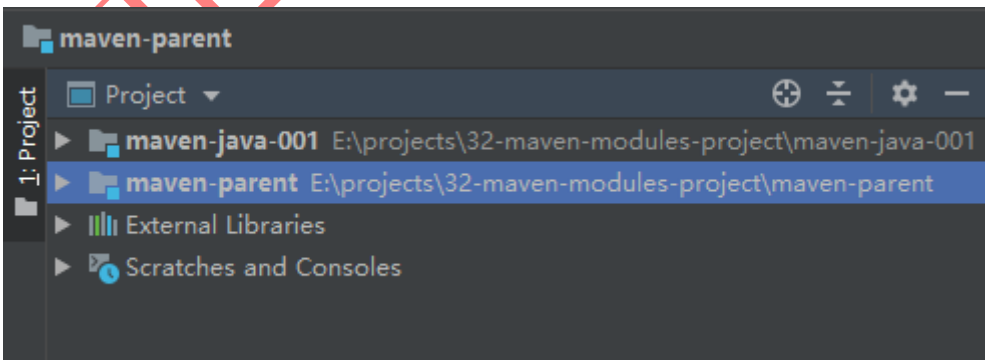


Add as module to：选择将创建的模块添加哪个模块
Parent：选择模块的父工程
GroupId：选择父工程后，默认继承父工程的 GroupId 值
ArtifactId：模块的项目名称
Version：选择父工程后，默认继承父工程的 Version 值

（4） 设置模块存放位置



（5） 项目视图



（6） 子模块项目的 pom 文件

```xml
<?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">
    <parent>
        <artifactId>maven-parent</artifactId>
        <groupId>com.abc.maven</groupId>
        <version>1.0.0</version>
        <relativePath>../maven-parent/pom.xml</relativePath>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>maven-java-001</artifactId>


</project>
```

parent 标签：指向父工程
relativePath 标签：相对路径

2.1.7 设置父工程编译级别

执行效果：

项目中会统一使用 JDK 版本和编译级别，所以项目的编译级别必须统一一致，那么将编译插件添加到父工程，子模块依然会无条件去继承父工程的插件。

（1） 指定编译级别前：

在 File -> Settings -> Build, Execution, Deployment -> Compiler -> Java Compiler 查看

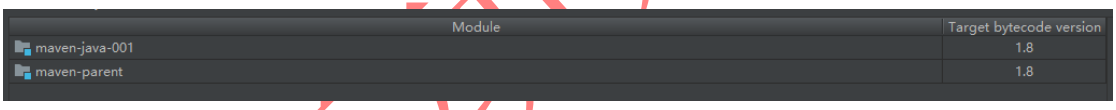
Module	Target bytecode version
maven-java-001	1.5
maven-parent	1.5

（2） 在父工程的 **build -> plugins** 标签中添加编译插件

```
<build>
    <plugins>
        <!-- 编译插件 -->
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <!-- 插件的版本 -->
            <version>3.5.1</version>
            <!-- 编译级别 -->
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
                <!-- 编码格式 -->
                <encoding>UTF-8</encoding>
            </configuration>
        </plugin>
    </plugins>
</build>
```

（3） 指定编译级别后：

在 File -> Settings -> Build, Execution, Deployment -> Compiler -> Java Compiler 再次查看



（4） **JDK1.8 编译插件：**

```
<!-- 编译插件 -->
<plugin>
    <artifactId>maven-compiler-plugin</artifactId>
    <!-- 插件的版本 -->
    <version>3.5.1</version>
    <!-- 编译级别 -->
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
        <!-- 编码格式 -->
        <encoding>UTF-8</encoding>
```

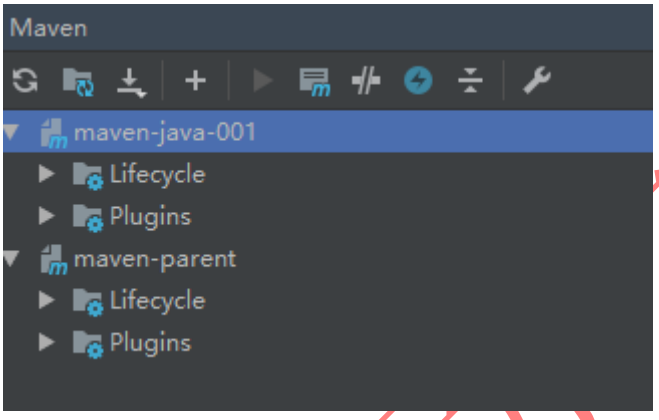
```
    </configuration>
</plugin>
```

2.1.8 父工程添加依赖

执行后效果：

在父工程 dependencies 标签中添加 MySQL 依赖，子模块会无条件继承父工程所有依赖。

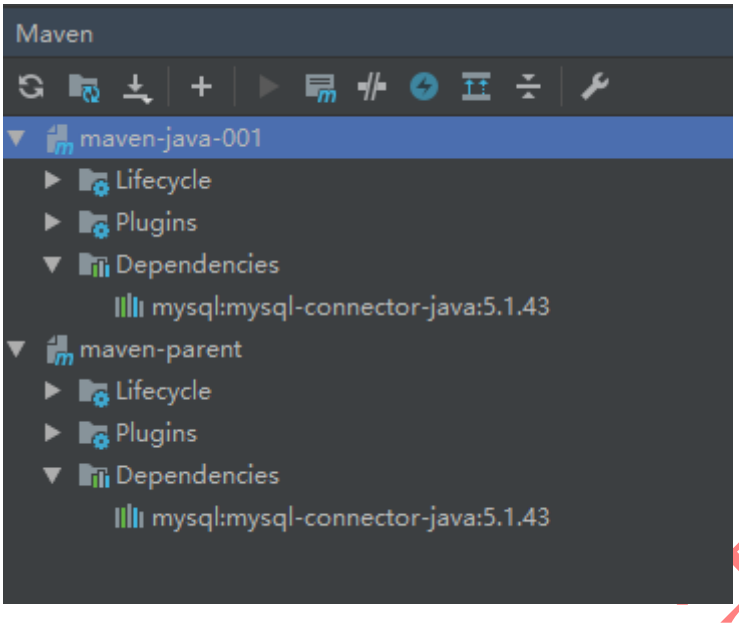
（1） 添加 MySQL 依赖前



（2） 添加 MySQL 依赖（父工程 pom 文件）：

```
<dependencies>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.43</version>
    </dependency>

</dependencies>
```

（3） 添加 MySQL 依赖后



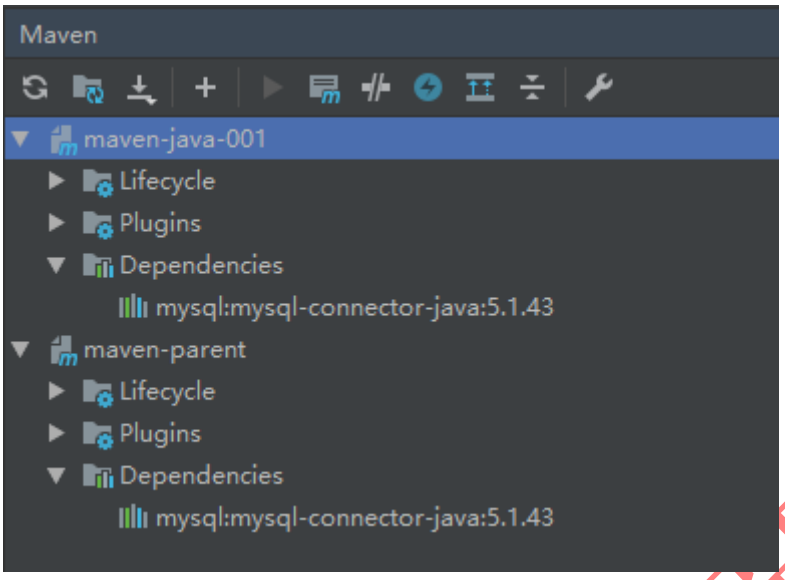
2.1.9 父工程管理依赖版本号

以上写做法，子模块会无条件继承父工程的所有依赖，导致的问题是，本不需要的继承的依赖也会被继承，这就大大增加了项目模块最终打包的大小，也可能未上线埋下了隐患。

也就是说，父工程管理的是所有项目模块的依赖，而不是某一个项目模块的依赖，所以某一个项目模块不需要继承父工程中的所有依赖，这就需要子项目模块向父工程声明需要的依赖即可（声明式依赖）。而此时，父工程实际只需要管理依赖的版本号即可。

实现方式如下：

（1） 父工程添加 dependencyManagement 前

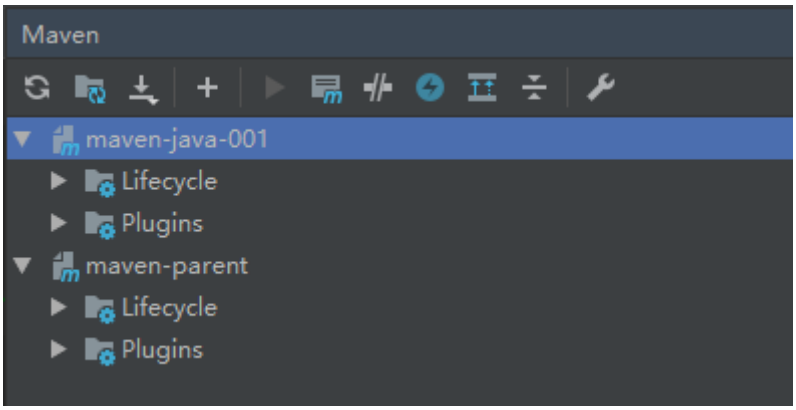


（2） 父工程添加 dependencyManagement 标签管理依赖

```
<!--管理依赖-->
<dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.43</version>
        </dependency>
    </dependencies>

</dependencyManagement>
```

（3） 父工程添加 dependencyManagement 前

子模块项目之前继承的依赖消失，由于父工程通过 dependencyManagement 标签管理依赖，那么之前子模块无条件继承的依赖就全部消失。



（4） 父工程添加 properties 管理版本号

在 properties 标签中，可以自定义标签名称来管理依赖的版本号，通常自定义的标签名称由“项目名称”+version 英文单词构成。被管理的依赖版本号由“${算定标签名称}”来代替。

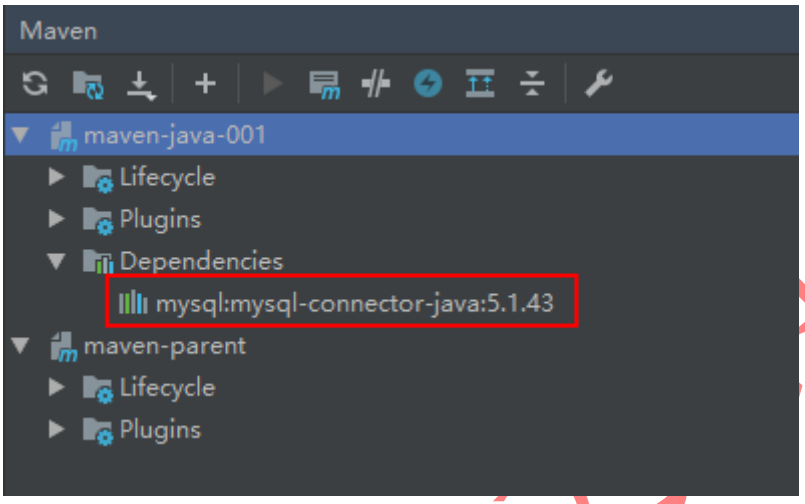
```
<!--管理依赖的版本号-->
<properties>
    <!--自定义标签-->
    <mysql-connector-java-version>5.1.43</mysql-connector-java-version>
</properties>
```

```
<!--管理依赖-->
<dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>${mysql-connector-java-version}</version>
        </dependency>
    </dependencies>

</dependencyManagement>
```

（5） 子模块声明式添加依赖

由于父工程管理依赖的版本号，那么子模块要想继承依赖，只能通过声明式来添加依赖，实际上，子模块中的依赖是继承父工程依赖的版本号；如果子模块已定义依赖版本号，那么以子模块定义的版本号为准。

```xml
<dependencies>
    <!--实际上继承父工程依赖版本号-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
</dependencies>
```

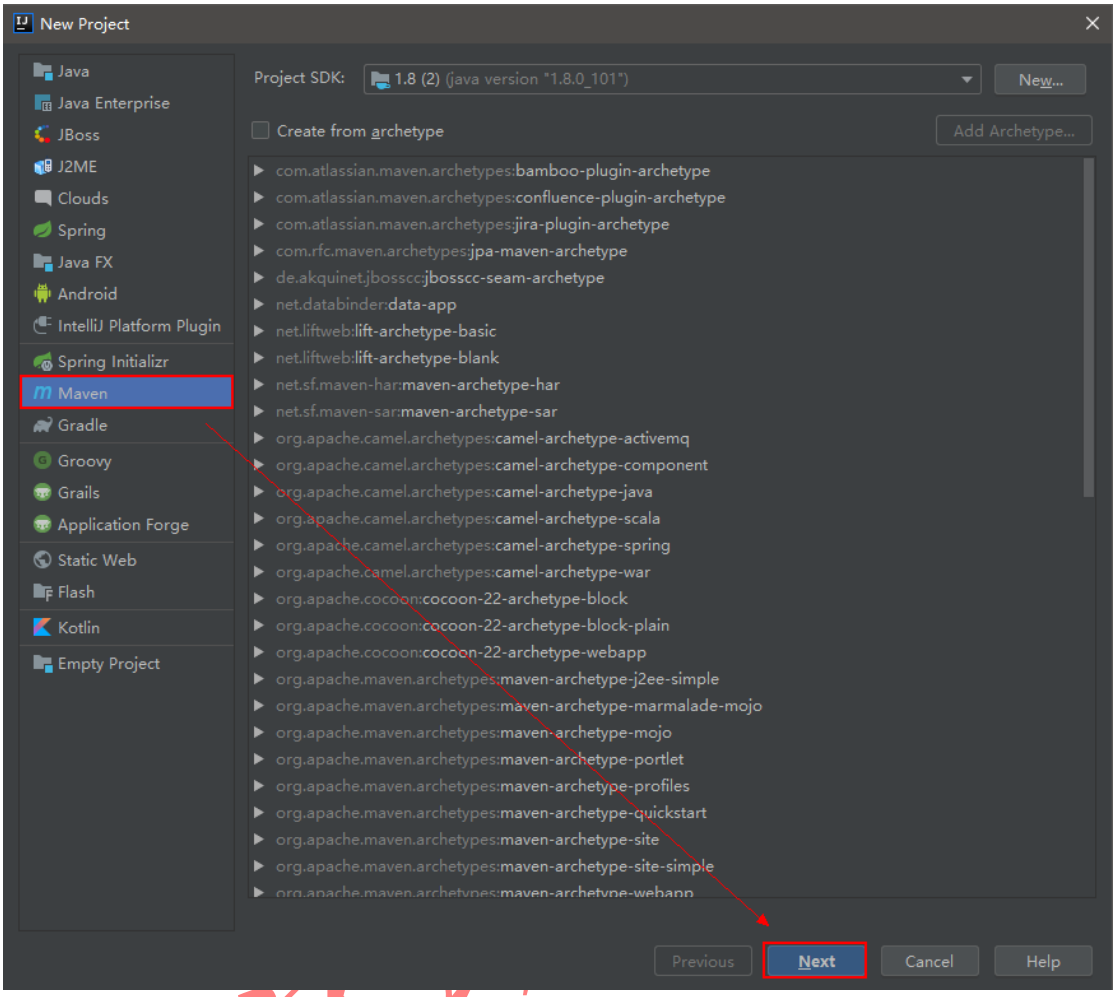


2.2 第二种实现方式

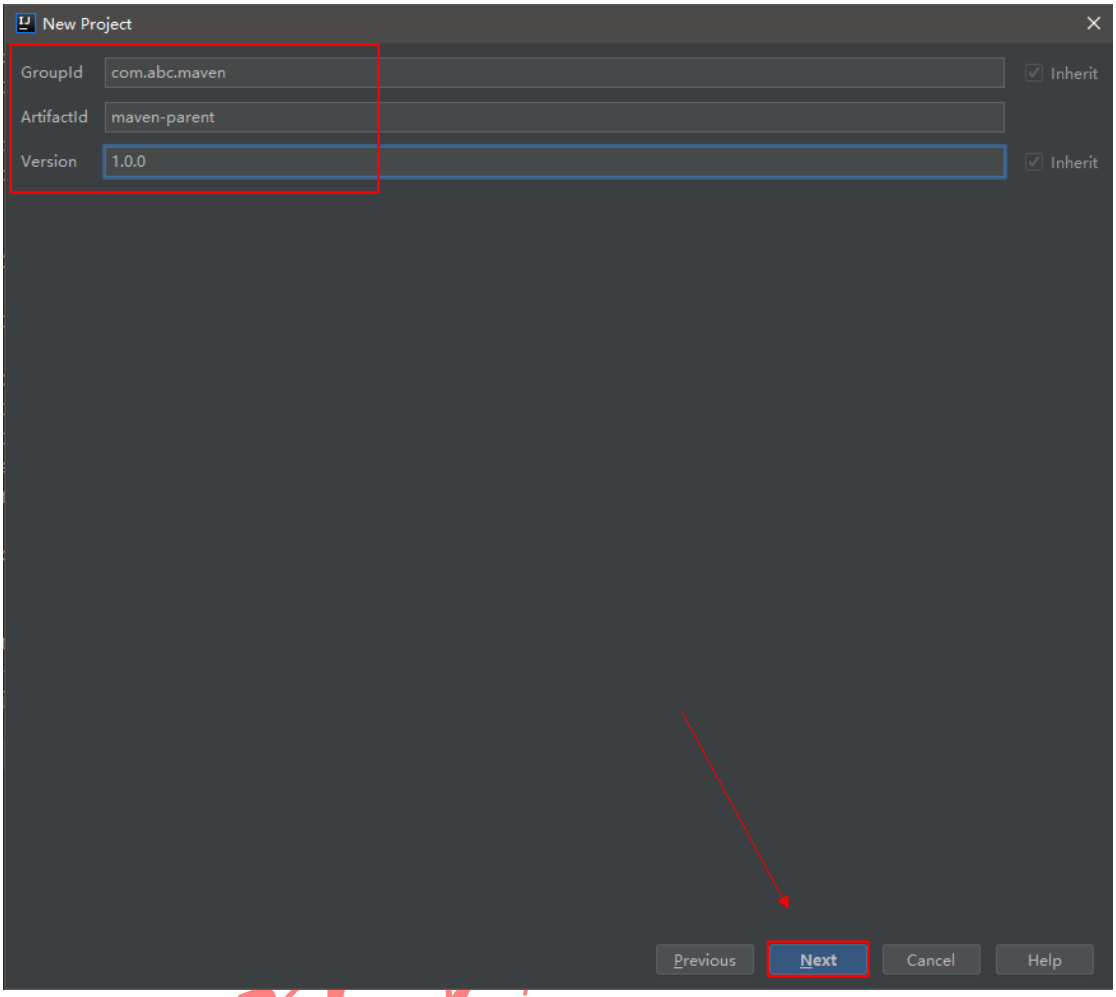
项目名称：maven-parent
完成功能：使用 IntelliJ IDEA 实现 Maven 管理多模块的应用开发

2.2.1 创建 Maven 工程

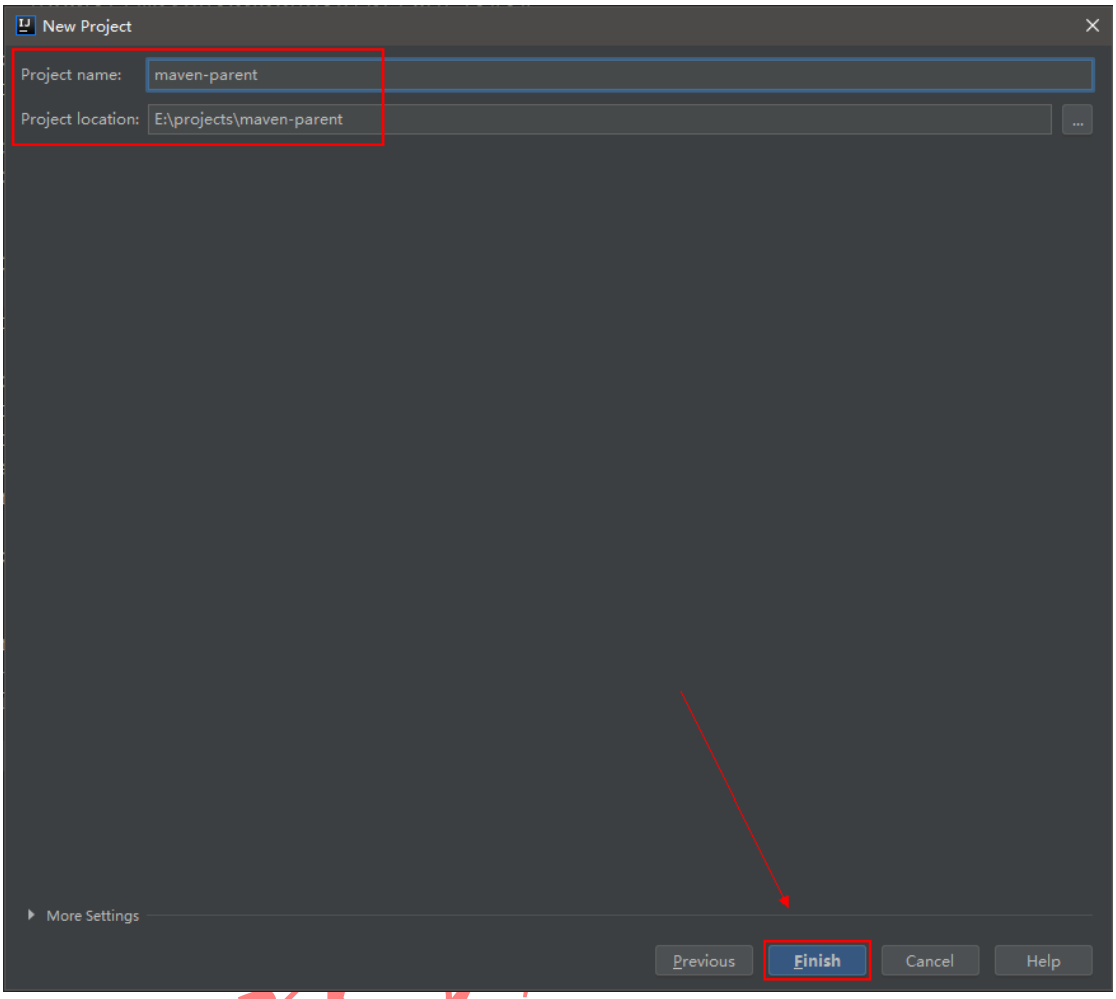
（1） 选择 Maven，点击“Next”下一步



（2） 设置项目坐标，点击“Next”下一步

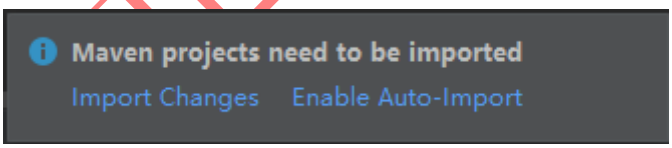


（3） 设置项目名称和项目存放位置，点击“Finish”完成。



（4） 配置导入设置

Maven 项目被修改后，需要“手动更新”或“自动更新”，通常选择“Enable Auto-Import”

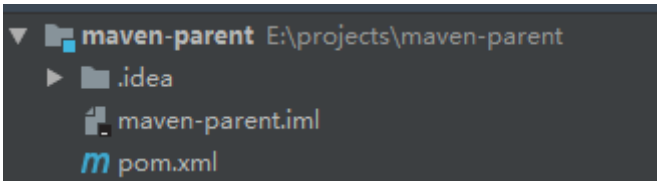


2.2.2 将 maven 工程修改为父工程

（1） 设置 packaging 标签的文本内容

```
<packaging>pom</packaging>
```

（2） 删除 src 目录



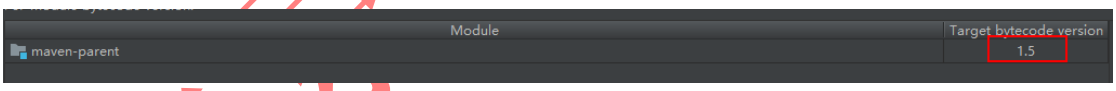
2.2.3 父工程添加编译插件

执行效果：

项目中会统一使用 JDK 版本和编译级别，所以项目的编译级别必须统一一致，那么将编译插件添加到父工程，子模块依然会无条件去继承父工程的插件。

➢ 添加编译插件前

在 File -> Settings -> Build, Execution, Deployment -> Compiler -> Java Compiler



➢ 添加编译插件后

在pom文件的build -> plugins标签中添加插件

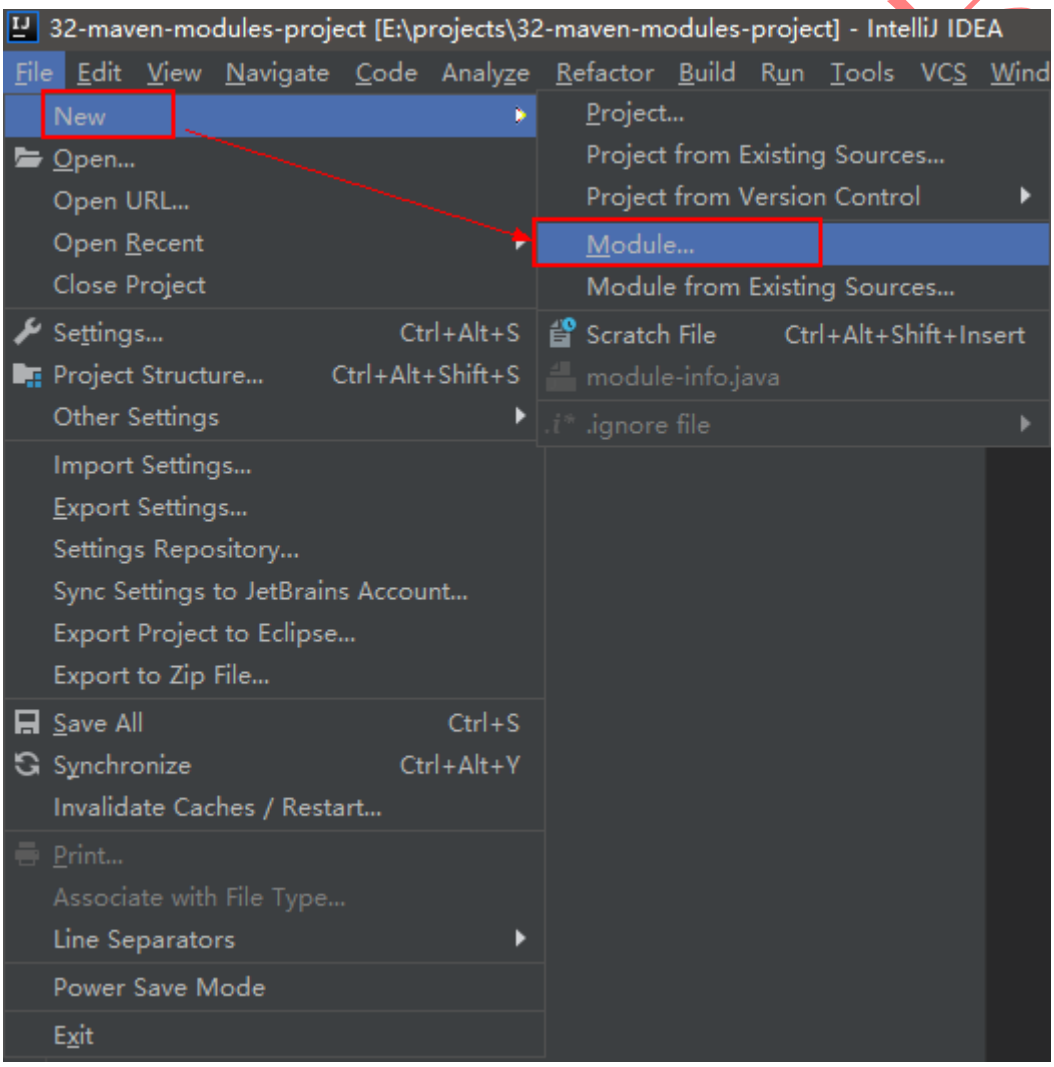
```
<!-- 编译插件 -->
<plugin>
    <artifactId>maven-compiler-plugin</artifactId>
    <!-- 插件的版本 -->
    <version>3.5.1</version>
    <!-- 编译级别 -->
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
        <!-- 编码格式 -->
```

```
        <encoding>UTF-8</encoding>
    </configuration>
</plugin>
```

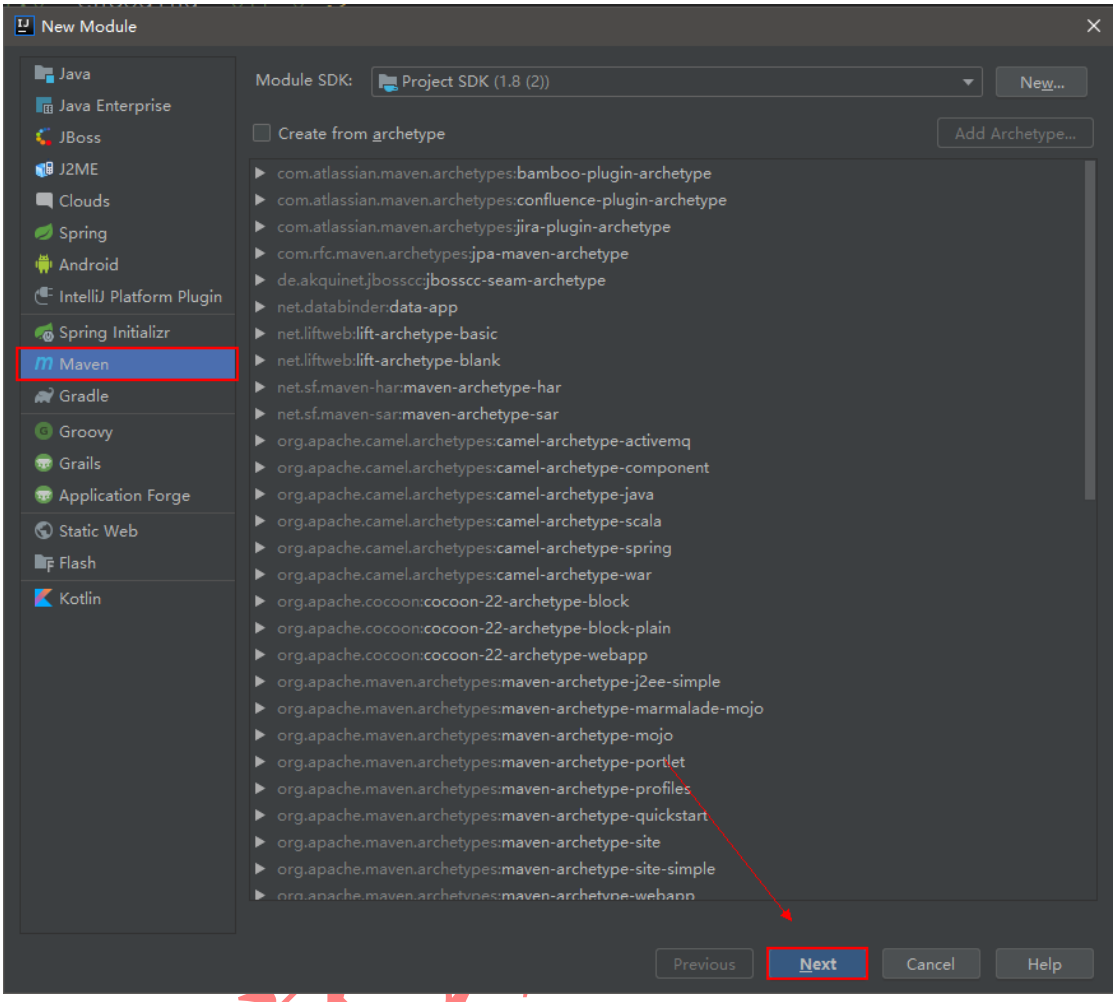
2.2.4 创建子模块

模块名称：maven-java-001

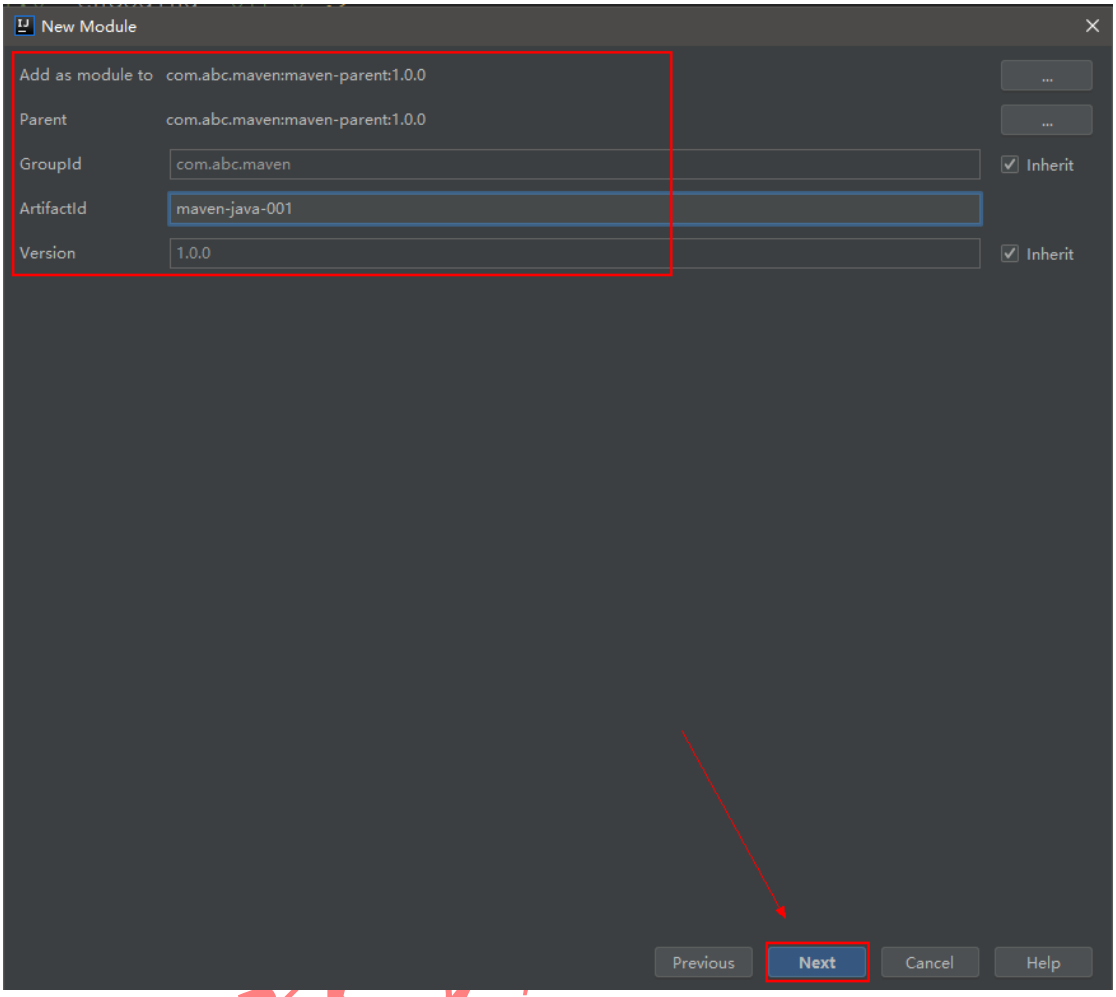
（1） 选择 New Module



（2） 选择 maven 项目

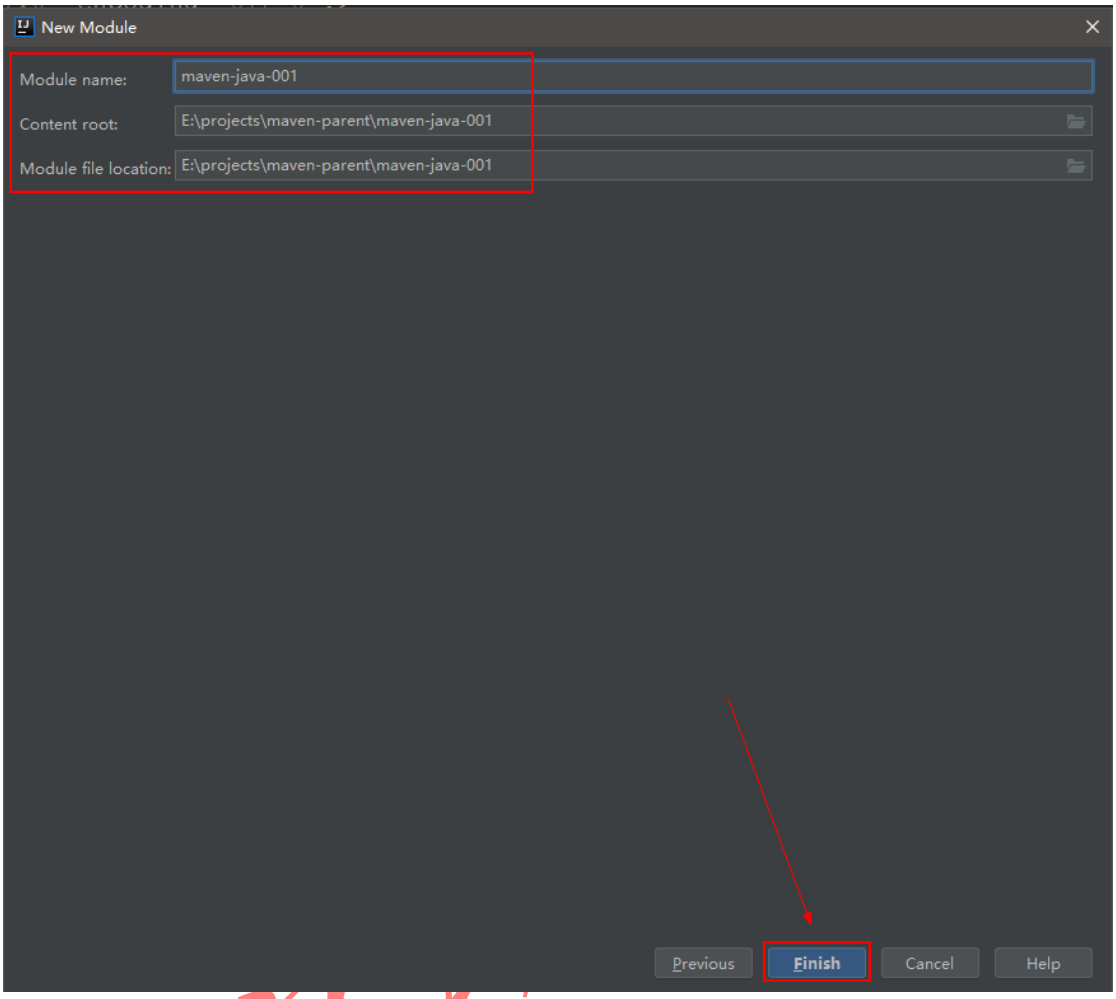


（3） 设置子模块坐标及父工程

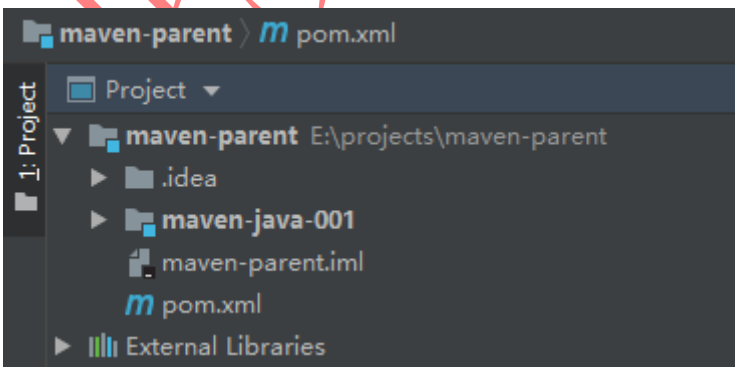


Add as module to：选择将创建的模块添加哪个模块
Parent：选择模块的父工程
GroupId：选择父工程后，默认继承父工程的 GroupId 值
ArtifactId：模块的项目名称
Version：选择父工程后，默认继承父工程的 Version 值

（4） 设置模块名称及存放位置



（5） 项目视图



2.2.5 子模块 pom 文件

```
<?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">
    <parent>
        <artifactId>maven-parent</artifactId>
        <groupId>com.abc.maven</groupId>
        <version>1.0.0</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>maven-java-001</artifactId>
</project>
```

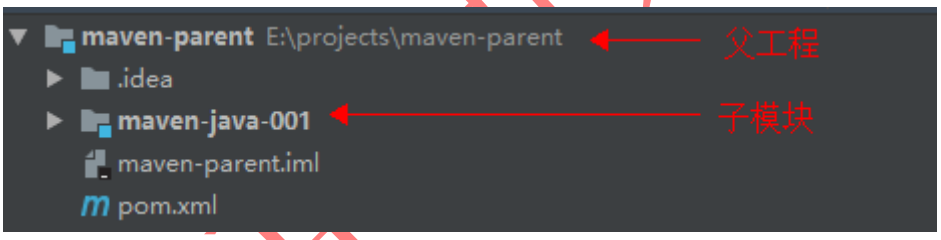
2.2.6 父工程 pom 文件

```
<?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>com.abc.maven</groupId>
    <artifactId>maven-parent</artifactId>
    <version>1.0.0</version>

    <!--父工程包含的所有模块名称-->
    <modules>
        <module>maven-java-001</module>
    </modules>

    <packaging>pom</packaging>

    <build>
        <plugins>
```

```
            <!-- 编译插件 -->
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <!-- 插件的版本 -->
                <version>3.5.1</version>
                <!-- 编译级别 -->
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <!-- 编码格式 -->
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

modules：父工程包含所有的子模块名称

2.2.7 项目结构

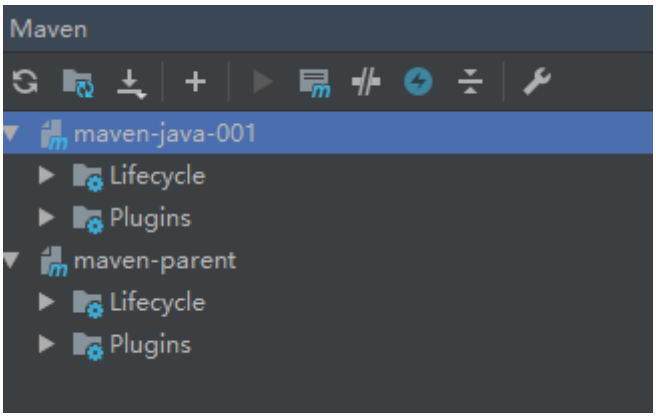


2.2.8 添加依赖

执行后效果：

在父工程的 dependencies 标签里添加 MySQL 依赖，子模块会无条件继承父工程所有依赖。

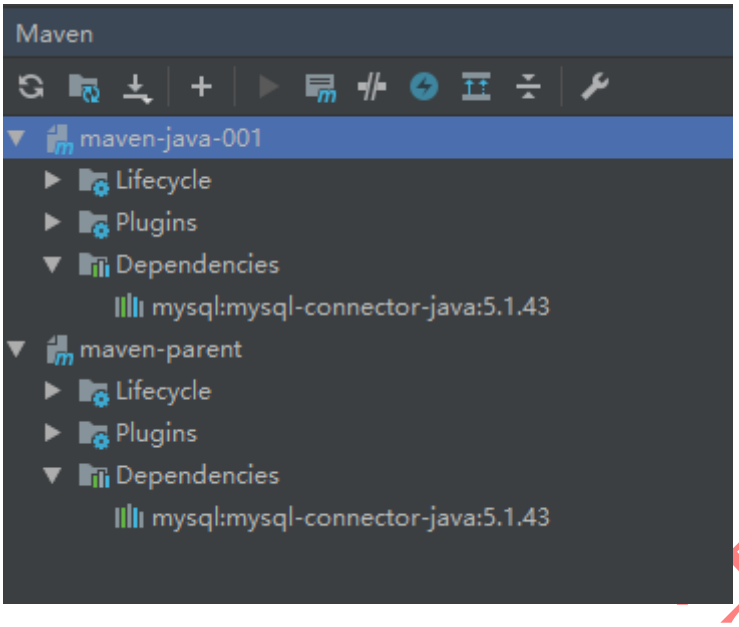
（1） 添加 MySQL 依赖前



（2） 添加 MySQL 依赖（父工程 pom 文件）

```
<dependencies>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.43</version>
    </dependency>

</dependencies>
```

（3） 添加 MySQL 依赖后



2.2.9 父工程管理依赖版本号

同 2.1.9