初始化配置

> 先配置**JAVA_HOME**，然后配置**MAVEN_HOME**，添加**%MAVEN_HOME%/bin**到PATH中

```
C:\Users\MJ>mvn -v
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: F:\Dev\Java\apache-maven-3.6.3\bin\..
Java version: 14.0.1, vendor: Oracle Corporation, runtime: F:\Dev\Java\jdk-14.0.1
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

> 修改仓库位置：在**%MAVEN_HOME%/conf/settings.xml**的**<settings>**标签中添加1个**<localRepository>**

```
<localRepository>F:\Dev\Java\.m2\repository</localRepository>
```

> 提高仓库的下载速度：在**%MAVEN_HOME%/conf/settings.xml**的**<mirrors>**标签中添加1个**<mirror>**

```
<mirror>
    <id>aliyun</id>
    <name>aliyun</name>
    <url>https://maven.aliyun.com/repository/public</url>
    <mirrorOf>central</mirrorOf>
</mirror>
```

修改Maven项目的JDK版本

方法①

> 在**pom.xml**中添加属性（每个Maven项目都要添加）

```
<properties>
    <maven.compiler.source>14</maven.compiler.source>
    <maven.compiler.target>14</maven.compiler.target>

 <maven.compiler.compilerVersion>14</maven.compiler.compilerVersion>
</properties>
```

方法②

> 在**pom.xml**中添加插件（每个Maven项目都要添加）

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
    <configuration>
        <source>14</source>
        <target>14</target>
    </configuration>
</plugin>
```

方法③

> 在**%MAVEN_HOME%/conf/settings.xml**的**<profiles>**标签中添加1个**<profile>**
>
> 这是一种一劳永逸的办法，不需要去修改每一个Maven项目的**pom.xml**

```
<profile>
    <id>jdk14</id>
    <activation>
        <activeByDefault>true</activeByDefault>
        <jdk>14</jdk>
    </activation>
    <properties>
        <maven.compiler.source>14</maven.compiler.source>
        <maven.compiler.target>14</maven.compiler.target>

 <maven.compiler.compilerVersion>14</maven.compiler.compilerVersion>

    </properties>
</profile>
```

命令行新建Maven项目

方法①

> 在命令行输入：**mvn archetype:generate**，会让我们选择项目类型
>
> 默认值是**7**，**maven-archetype-quickstart**，是一个普通的Java项目，如果希望使用默认值，直接敲回车即可
>
> 如果希望创建一个web项目，应该输入**10**，**maven-archetype-webapp**

```
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[WARNING] No archetype found in remote catalog. Defaulting to internal catalog
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.archetypes:maven-archetype-quickstart:1.
0)
Choose archetype:
1: internal -> org.apache.maven.archetypes:maven-archetype-archetype (An archetype which contains a sample archetype.)
2: internal -> org.apache.maven.archetypes:maven-archetype-j2ee-simple (An archetype which contains a simplifed sample J
2EE application.)
3: internal -> org.apache.maven.archetypes:maven-archetype-plugin (An archetype which contains a sample Maven plugin.)
4: internal -> org.apache.maven.archetypes:maven-archetype-plugin-site (An archetype which contains a sample Maven plugi
n site.
      This archetype can be layered upon an existing Maven plugin project.)
5: internal -> org.apache.maven.archetypes:maven-archetype-portlet (An archetype which contains a sample JSR-268 Portlet
.)
6: internal -> org.apache.maven.archetypes:maven-archetype-profiles ()
7: internal -> org.apache.maven.archetypes:maven-archetype-quickstart (An archetype which contains a sample Maven projec
t.)
8: internal -> org.apache.maven.archetypes:maven-archetype-site (An archetype which contains a sample Maven site which d
emonstrates
      some of the supported document types like APT, XDoc, and FML and demonstrates how
      to i18n your site. This archetype can be layered upon an existing Maven project.)
9: internal -> org.apache.maven.archetypes:maven-archetype-site-simple (An archetype which contains a sample Maven site.
)
10: internal -> org.apache.maven.archetypes:maven-archetype-webapp (An archetype which contains a sample Maven Webapp pr
oject.)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 7:
```

输入**groupId**、**artifactId**、**version**、**package**

如果希望**version**、**package**使用默认值，直接敲回车即可

```
Define value for property 'groupId': com.mj.maven
Define value for property 'artifactId': helloworld
Define value for property 'version' 1.0-SNAPSHOT: :
Define value for property 'package' com.mj.maven: :
```

最后输入**y**表示确认，项目就创建完毕了

```
Confirm properties configuration:
groupId: com.mj.maven
artifactId: helloworld
version: 1.0-SNAPSHOT
package: com.mj.maven
Y: : y
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.1
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: F:\test
[INFO] Parameter: package, Value: com.mj.maven
[INFO] Parameter: groupId, Value: com.mj.maven
[INFO] Parameter: artifactId, Value: helloworld
[INFO] Parameter: packageName, Value: com.mj.maven
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: F:\test\helloworld
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  04:56 min
[INFO] Finished at: 2020-05-31T13:14:12+08:00
[INFO] ------------------------------------------------------------------------
```

方法②

在命令行输入

```
mvn archetype:generate -DgroupId=com.mj.maven -
DartifactId=helloworld -Dversion=1.0-RELEASE -
DarchetypeGroupId=org.apache.maven.archetypes -
DarchetypeArtifactId=maven-archetype-quickstart
```

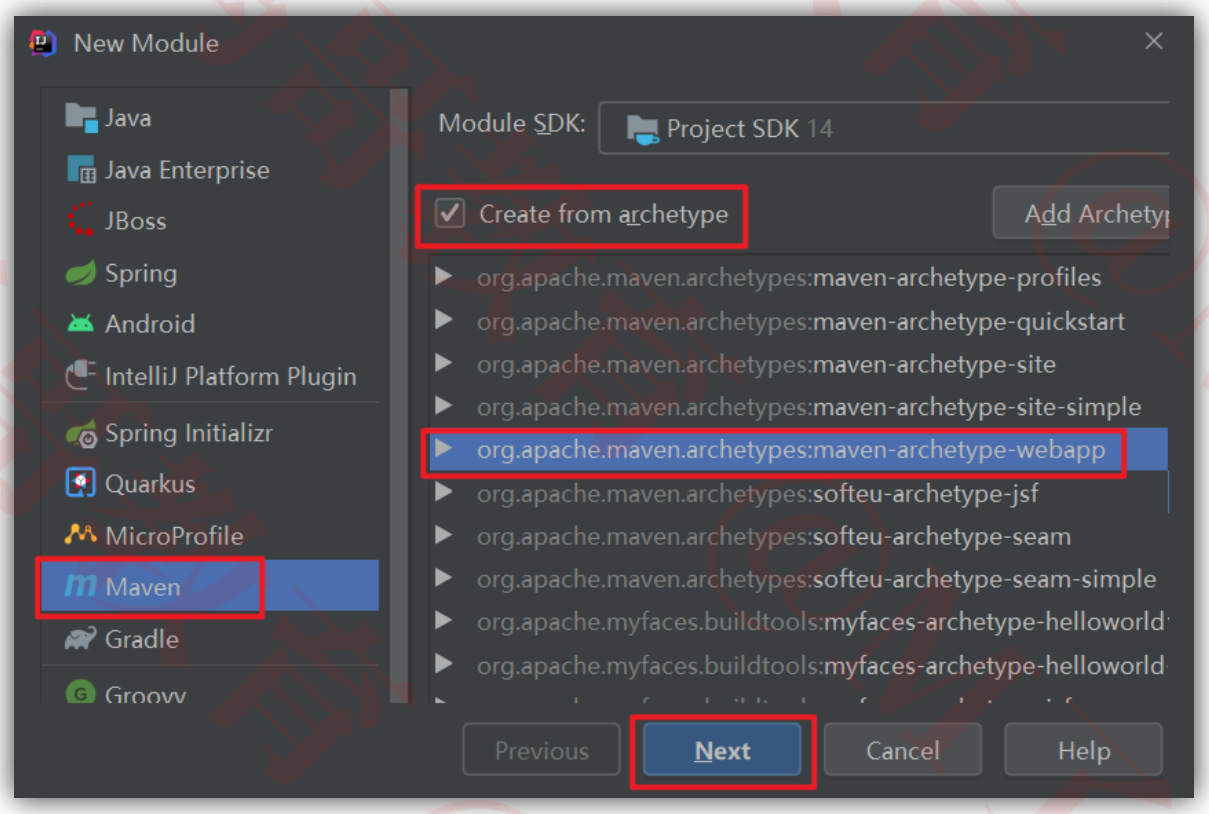
IDEA导入Maven项目

> 选择`pom.xml`进行导入

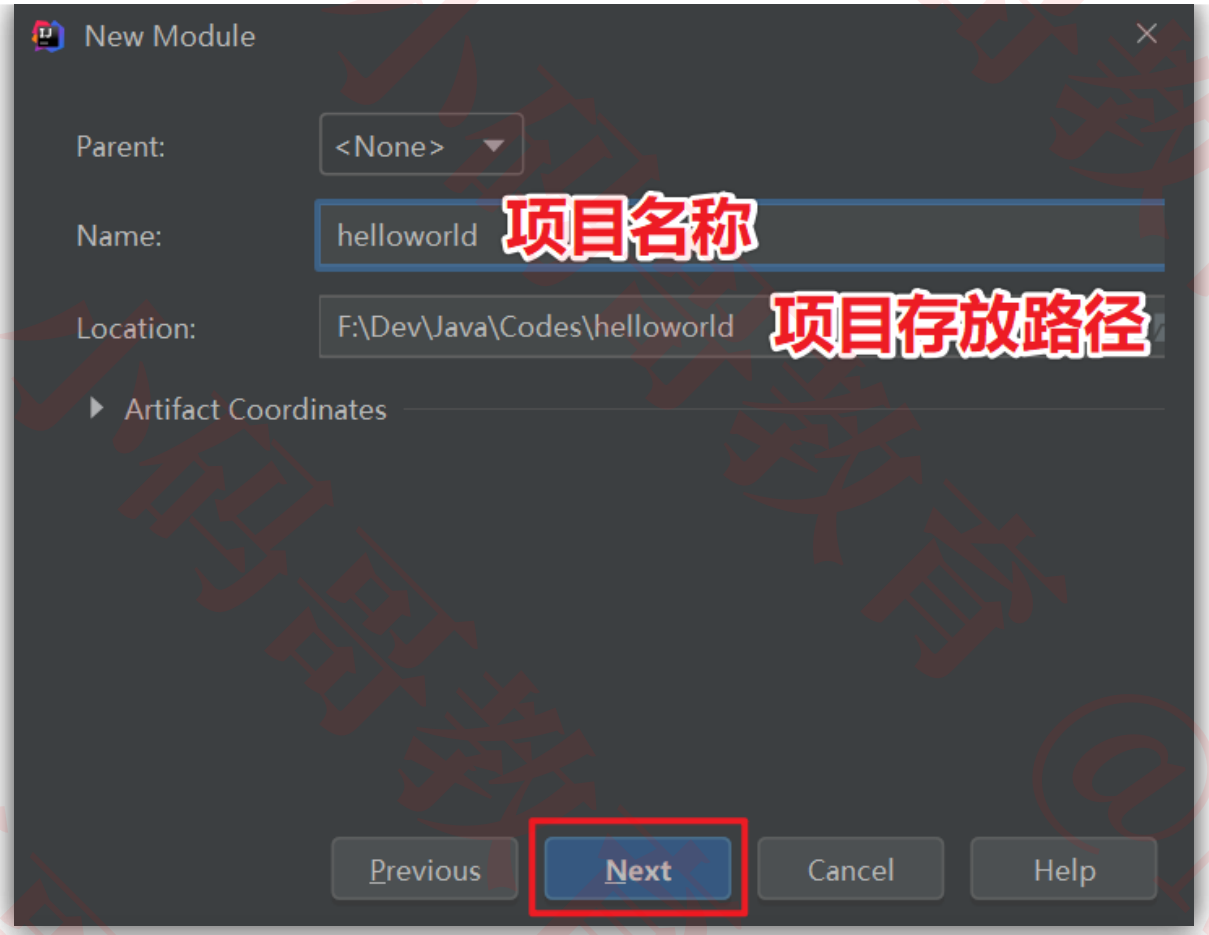
IDEA新建Maven项目（Web项目）

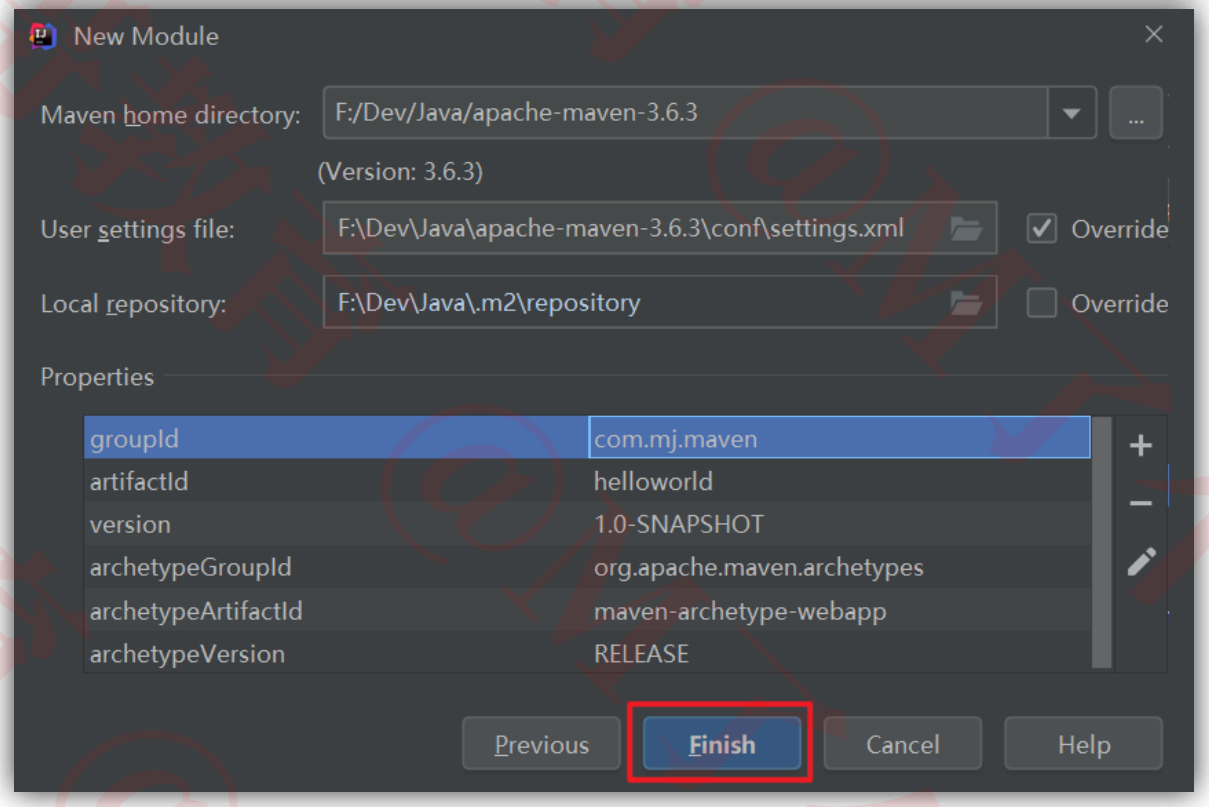
方法①

> 使用`archetype`



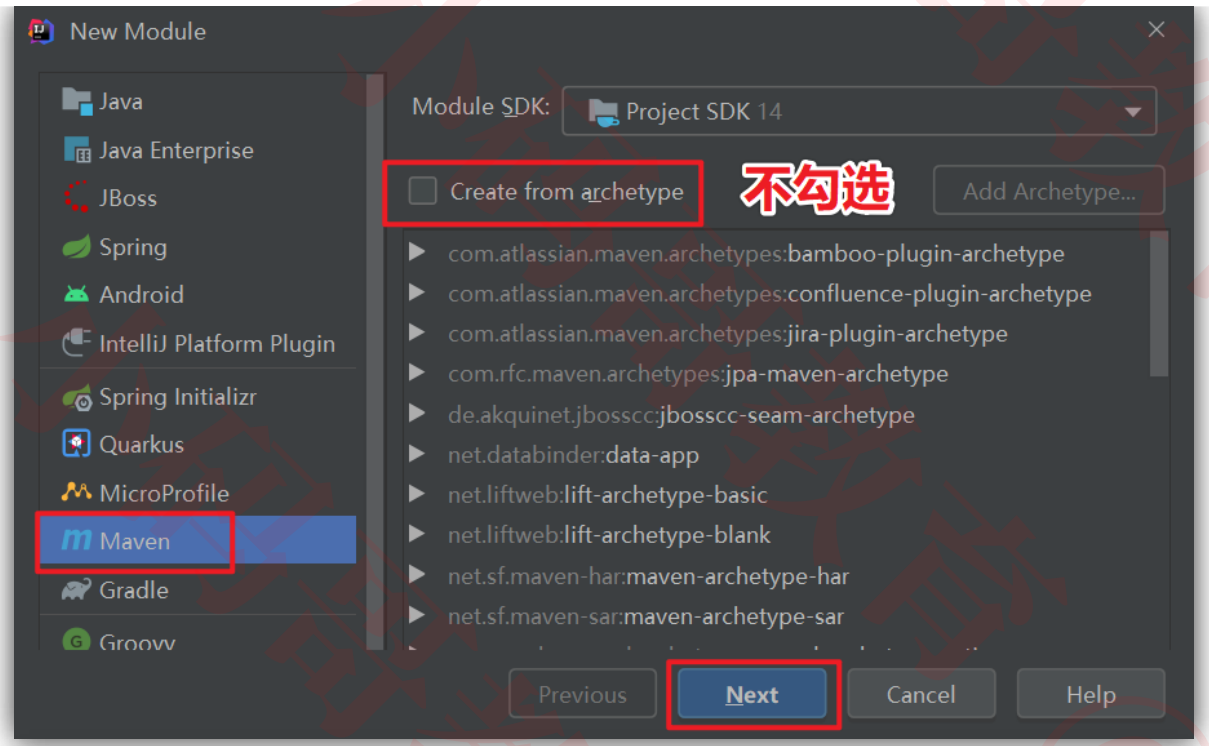
> 设置项目信息

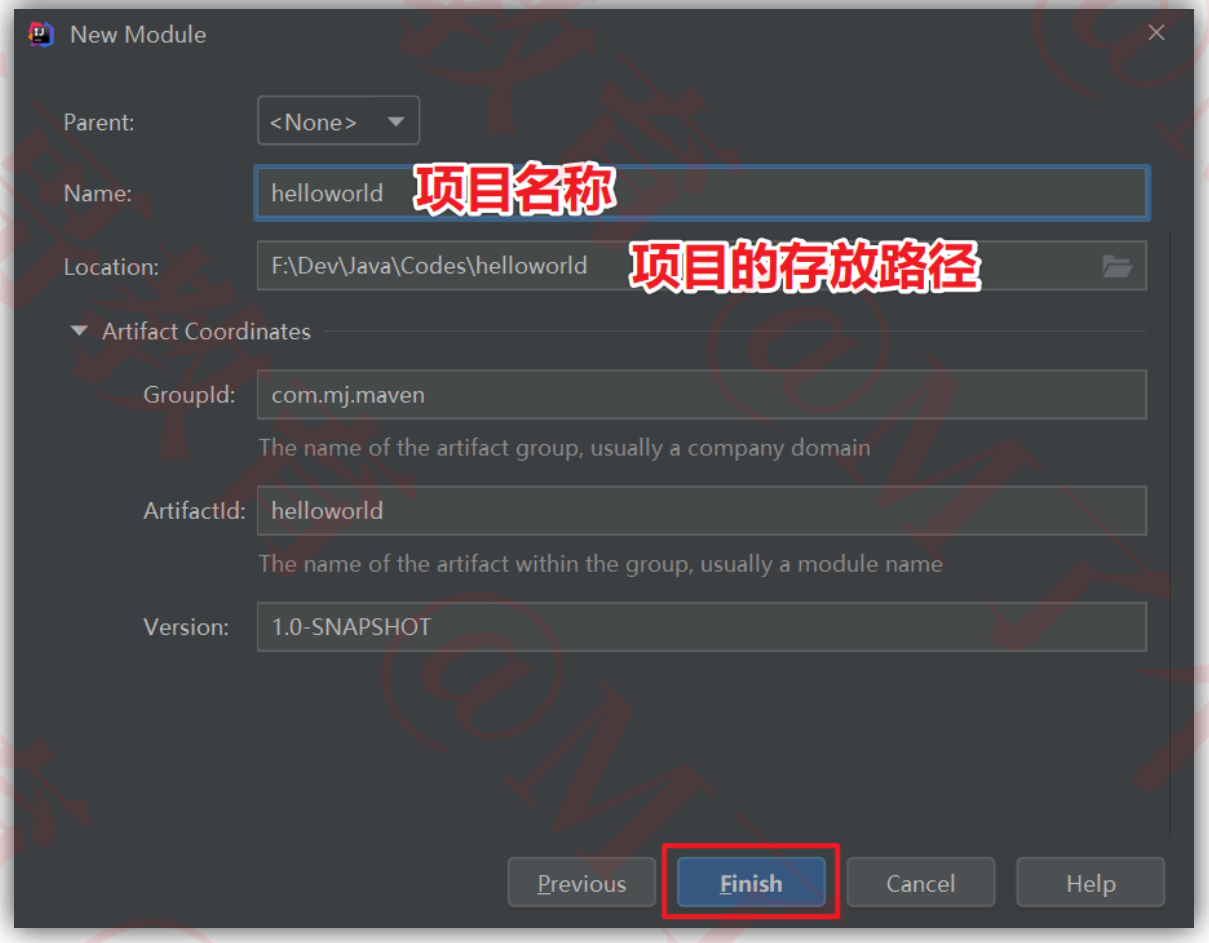




方法②

> 不勾选**Create from archetype**



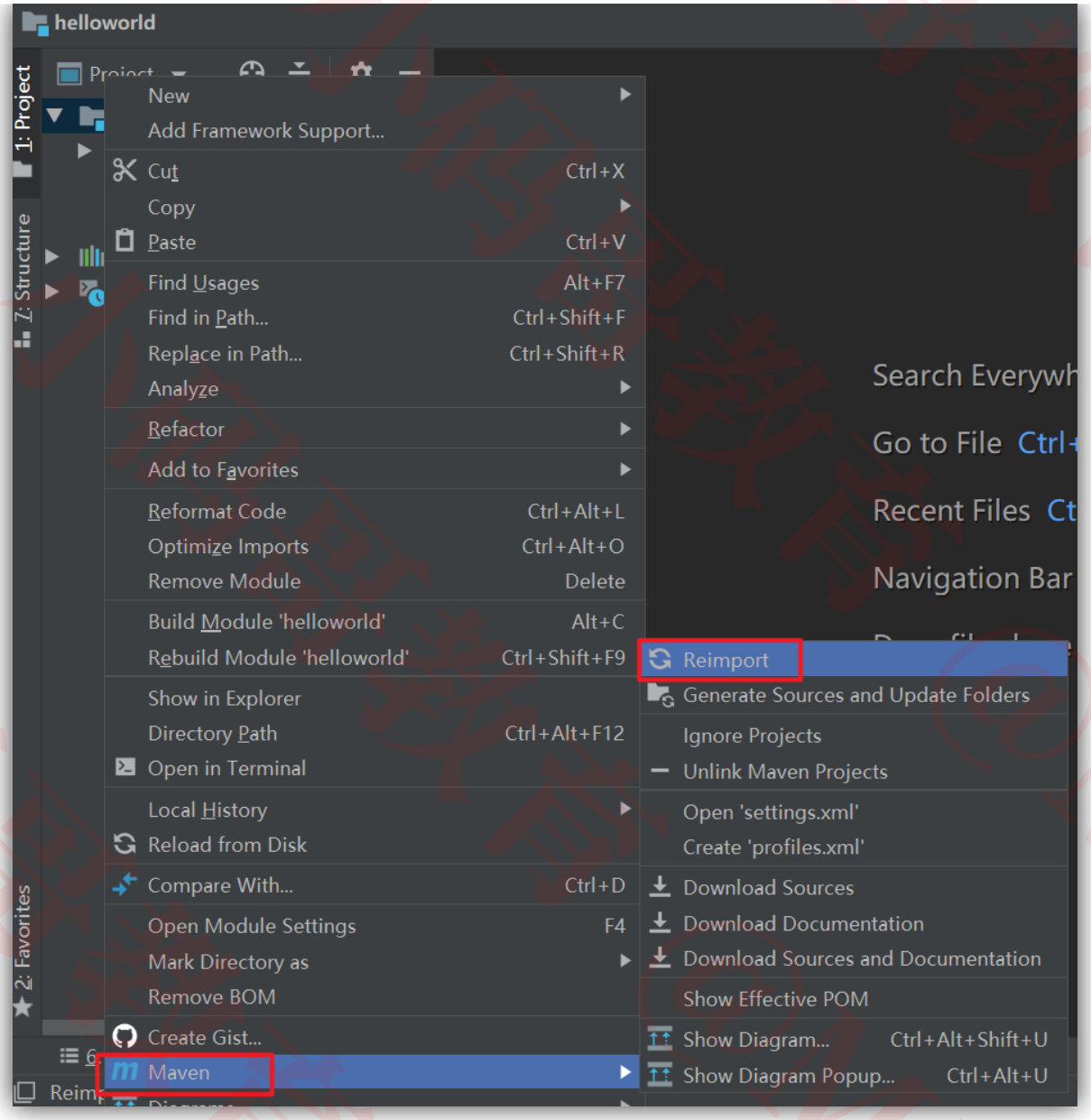


pom.xml配置

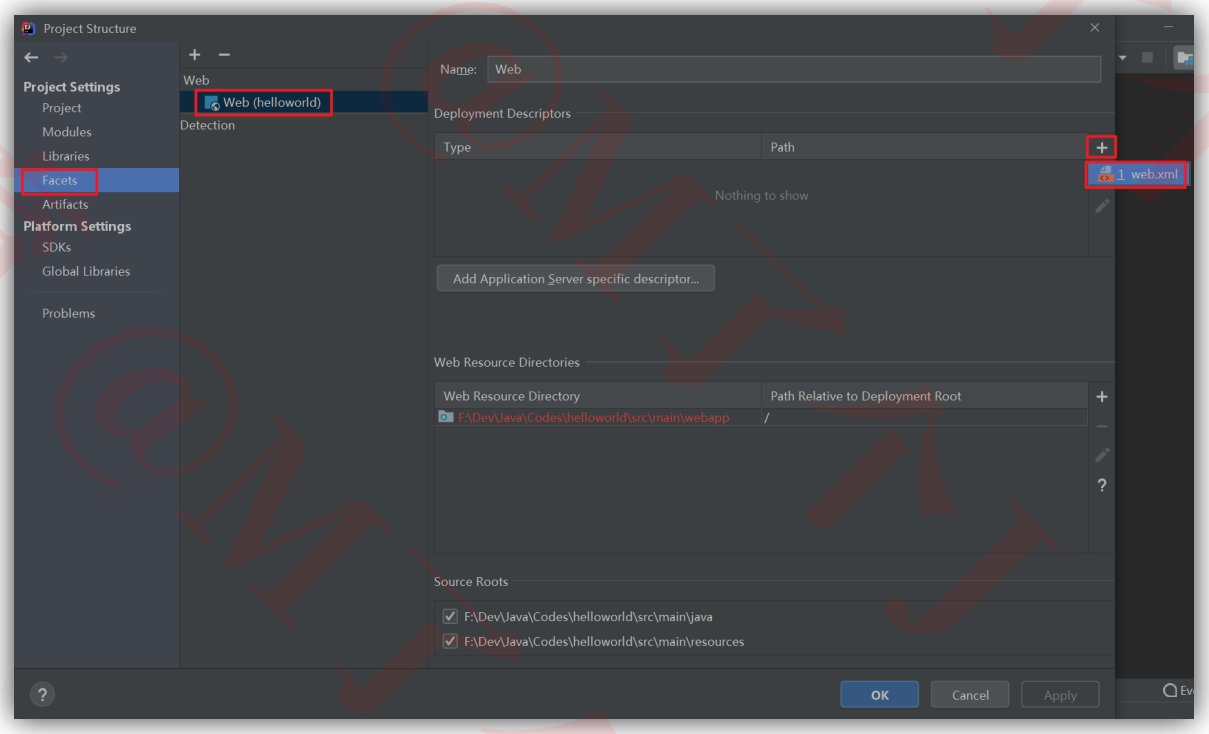
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <!-- 模型版本 -->
    <modelVersion>4.0.0</modelVersion>
```

```
    <!-- 项目信息 -->
    <groupId>com.mj.maven</groupId>
    <artifactId>helloworld</artifactId>
    <version>1.0-SNAPSHOT</version>

    <!-- 打包方式 -->
    <packaging>war</packaging>

    <!-- 文件编码 -->
    <properties>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
    </properties>

    <!-- 依赖 -->
    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
    </dependencies>

    <!-- 构建信息 -->
    <build>
        <!-- 打包后的文件名 -->
        <finalName>helloworld</finalName>
    </build>
</project>
```

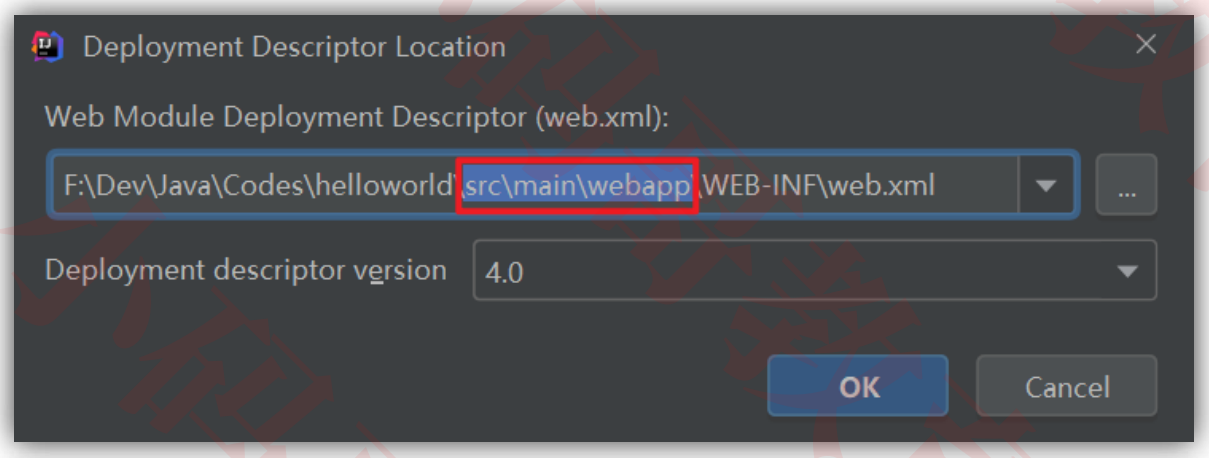
> 对项目进行**Reimport**

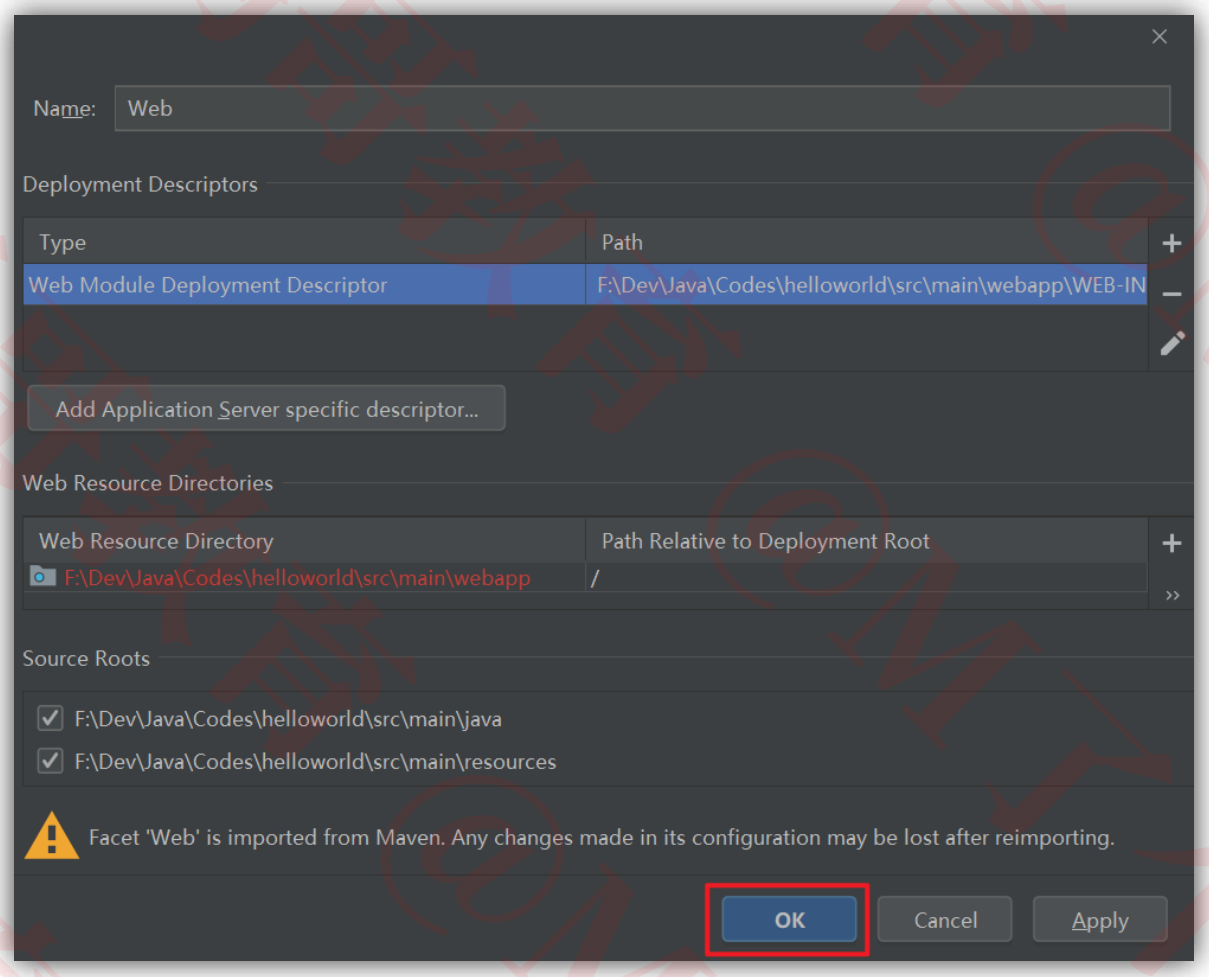


打开**Project Structure**，添加**web.xml**



> 将**web.xml**放到**src/main/webapp/WEB-INF**目录





生成Runnable Jar

方法①：maven-jar-plugin

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <version>3.0.2</version>
    <configuration>
        <archive>
```

```
            <manifest>
                <addClasspath>true</addClasspath>
                <classpathPrefix>lib</classpathPrefix>
                <mainClass>主类</mainClass>
            </manifest>
        </archive>
        <finalName>jar的文件名</finalName>
    </configuration>
</plugin>
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-dependency-plugin</artifactId>
    <version>3.1.2</version>
    <executions>
        <execution>
            <!-- <phase>package</phase> -->
            <goals>
                <goal>copy-dependencies</goal>
            </goals>
            <configuration>

 <outputDirectory>${project.build.directory}/lib</outputDirectory>
            </configuration>
        </execution>
    </executions>
</plugin>
```

方法②：maven-assembly-plugin

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-assembly-plugin</artifactId>
    <version>3.3.0</version>
    <configuration>
        <archive>
            <manifest>
                <mainClass>主类</mainClass>
            </manifest>
        </archive>
        <descriptorRefs>
            <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
        <finalName>jar的文件名</finalName>
        <appendAssemblyId>false</appendAssemblyId>
    </configuration>
    <executions>
        <execution>
            <phase>package</phase>
            <goals>
                <goal>single</goal>
            </goals>
        </execution>
```

```
    </executions>
</plugin>
```

方法③：maven-shade-plugin

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-shade-plugin</artifactId>
    <version>3.2.3</version>
    <executions>
        <execution>
            <!-- <phase>package</phase> -->
            <goals>
                <goal>shade</goal>
            </goals>
            <configuration>

 <shadedArtifactAttached>true</shadedArtifactAttached>
                <transformers>
                    <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestReso
urceTransformer">
                        <mainClass>主类</mainClass>
                    </transformer>
                </transformers>
                <finalName>jar的文件名</finalName>
            </configuration>
        </execution>
    </executions>
</plugin>
```

方法④：spring-boot-maven-plugin

```
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <version>2.3.0.RELEASE</version>
    <executions>
        <execution>
            <!-- <phase>package</phase> -->
            <goals>
                <goal>repackage</goal>
            </goals>
            <configuration>
                <mainClass>主类</mainClass>
                <finalName>jar的文件名</finalName>
            </configuration>
        </execution>
    </executions>
</plugin>
```

安装本地jar到Maven的LocalRepository

```
mvn install:install-file -Dfile=jar的路径 -DgroupId=组织 -DartifactId=
库名 -Dversion=版本 -Dpackaging=jar
```

Maven配置Tomcat插件

使用Maven内置的Tomcat

> 集成**tomcat7-maven-plugin**

```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <path>/context_path</path>
        <port>8080</port>
        <uriEncoding>UTF-8</uriEncoding>
    </configuration>
</plugin>
```

> 为了避免jar包冲突，修改servlet的scope为**provided**

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
</dependency>
```

使用独立安装的Tomcat9

> 在**%TOMCAT_HOME%/conf/tomcat-users.xml**中添加用户

```
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml
tomcat-users.xsd"
              version="1.0">
    <role rolename="manager-gui"/>
    <role rolename="manager-script"/>
    <user username="root" password="root" roles="manager-
gui,manager-script" />
</tomcat-users>
```

> 在%MAVEN_HOME%/conf/settings.xml的<servers>中添加<server>

```
<server>
    <id>tomcat9</id>
    <username>root</username>
    <password>root</password>
</server>
```

> 集成tomcat7-maven-plugin

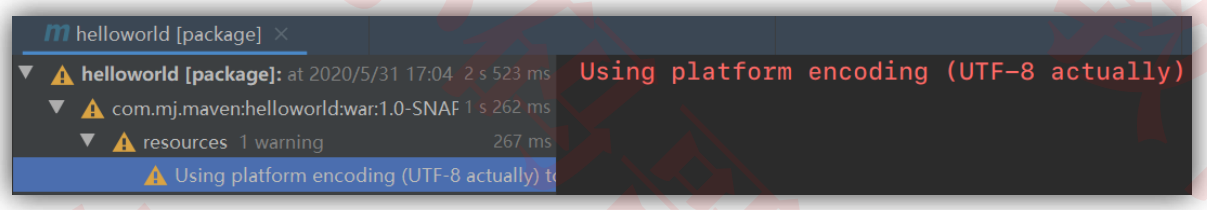
```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <path>/context_path</path>
        <update>true</update>
        <!-- <url>http://localhost:8080/manager/text</url> -->
        <!-- <uriEncoding>UTF-8</uriEncoding> -->
        <server>tomcat9</server>
        <!--
            <username>root</username>
            <password>root</password>
        -->
    </configuration>
</plugin>
```

> 启动本地的Tomcat9，通过tomcat7-maven-plugin的命令部署项目到Tomcat9

```
mvn tomcat7:deploy
# 或者
mvn tomcat7:redeploy
```

常见问题解决

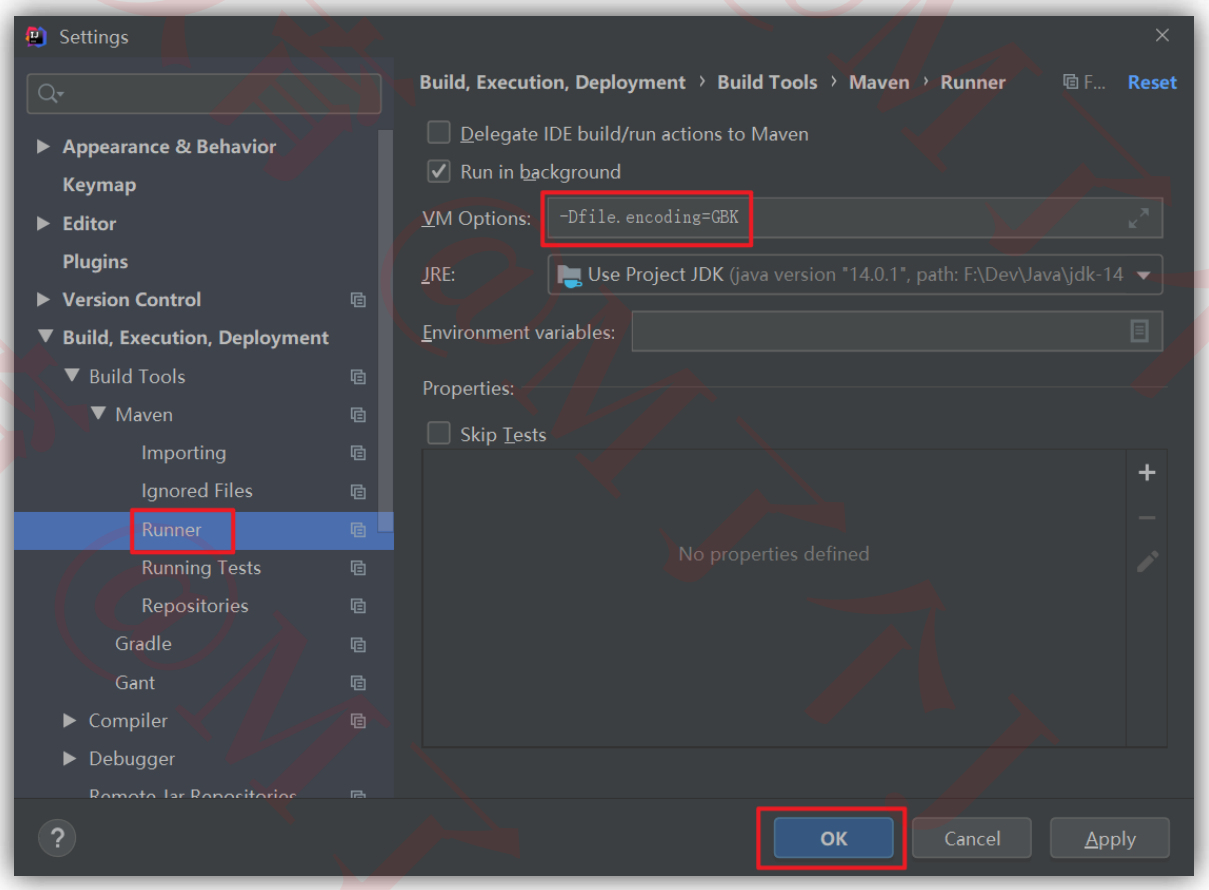
> 解决文件编码的警告：在**pom.xml**中添加




```
<properties>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
</properties>
```

> 解决IDEA控制台输出乱码：
>
> Settings -> Build, Execution, Deployment -> Build Tools -> Maven -> Runner
>
> 在VM Options中添加**-Dfile.encoding=GBK**

```
Customer{name='����', age=20, height=1.67}
Customer{name='����', age=4324, height=4354.0}
Customer{name='����44', age=56757, height=657657.0}
Customer{name='888', age=999, height=666.0}
Customer{name='123343', age=111545, height=123.0}
Customer{name='����', age=20, height=1.89}
Customer{name='12', age=13, height=14.0}
Customer{name='', age=0, height=0.0}
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.795 sec
```



> 对Maven项目使用IDEA内置的Build，可能会出现以下问题（找不到依赖包）
> 目前发现升级到**IDEA 2020**开始就会出现这个问题，可以考虑退回到**IDEA 2019**