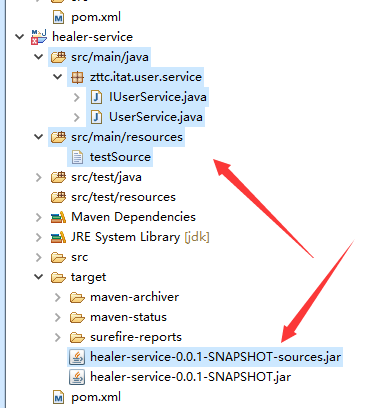
# 1、healer-service建立插件

## 1、使用maven-source-plugin插件进行打包（可以对main文件或者是test文件进行（这里的打包就是提取资源文件（main 、resource或者test 、resource）将原代码打到jar包（不包括编译后的class,以及其他依赖）里面

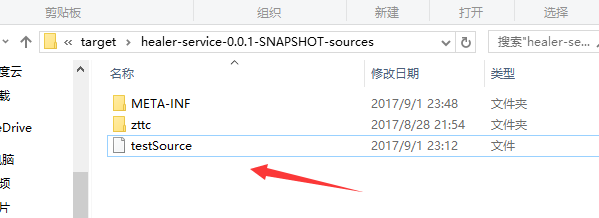
### 1、execution表示执行，phase 阶段的意思，这里写的是compile，表示执行完compile，就会对main下的文件进行打包

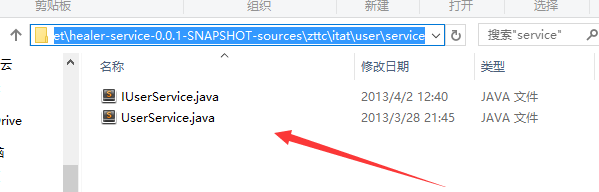
|  |
| --- |
| <build>  <!-- 22、插件的使用 -->  <plugins>  <!-- 23、将java代码引入jar包中（包括main、test下的java） -->    <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-source-plugin</artifactId>  <version>2.2.1</version>  <executions>  <execution>  <phase>compile</phase> <!-- 执行完compile再执行下面的命令 -->  <goals>  <goal>jar</goal>  <!-- <goal>test jar</goal>-->  </goals>  </execution>  </executions>  </plugin>  </plugins>  </build> |

### 2、测试<goal>jar</goal>完成，



### 3、解压后观察到main下的resource和java都有





# 2、healer-parent开始使用插件

### 1、build建立建立插件管理器pluginManagement ，子类不能够直接继承，需要像依赖管理器一样，添加groupId和artifactId

|  |
| --- |
| <!-- 23、从healer-service拷贝过来的插件，开始在父类中使用 -->  <build>  <pluginManagement>  <!-- 22、插件的使用 -->  <plugins>  <!-- 23、将java代码引入jar包中（包括main、test下的java） -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-source-plugin</artifactId>  <version>2.2.1</version>  <executions>  <execution>  <phase>compile</phase> <!-- 执行完compile再执行下面的命令 -->  <goals>  <goal>jar</goal>  <!-- <goal>test jar</goal>-->  </goals>  </execution>  </executions>  </plugin>  </plugins>  </pluginManagement>  </build> |

### 2、子类healer-servie中删除版本号和其他在父类中可以继承下来的

|  |
| --- |
| <build>  <!-- 22、插件的使用 -->  <plugins>  <!-- 23、将java代码引入jar包中（包括main、test下的java） -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-source-plugin</artifactId>  </plugin>  </plugins>  </build> |

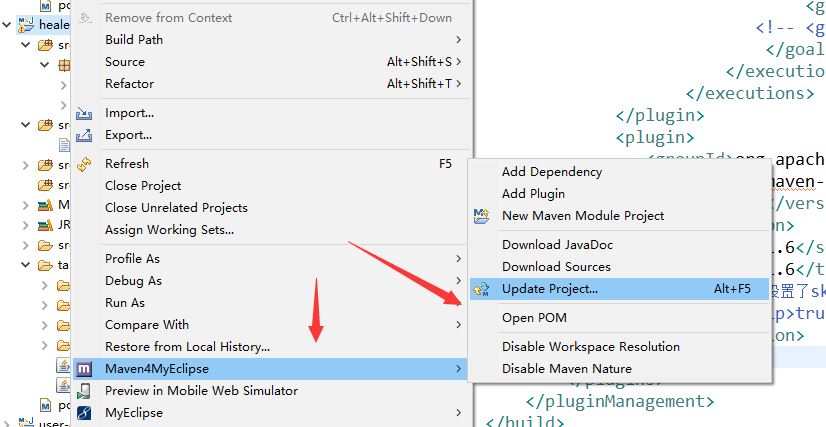
## 1、插件将项目使用的jdk版本变成其他的版本

### 1、source，target 写为1.6作为版本

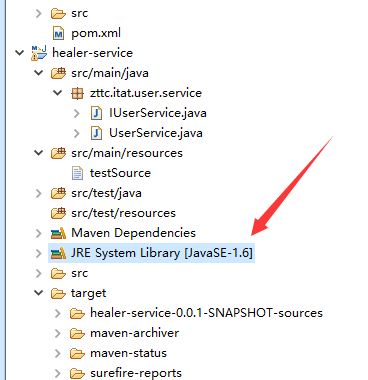
### 2、<skip>true</skip> 表示如果test测试类有了错在编译时候可以跳过，不会报错

|  |
| --- |
| <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.0</version>  <configuration>  <source>1.6</source>  <target>1.6</target>  <!-- 此处设置了skip等于设置了把所有的测试编译都跳过，如果测试类写得有问题，也不会报错，所以一般不使用 -->  <!-- <skip>true</skip> -->  </configuration>  </plugin> |

### 3、观察版本号，如果没有及时更新，反而报错，选择更新项目



#### 1、现在版本变为1.6，我的计算机的版本是1.8来着

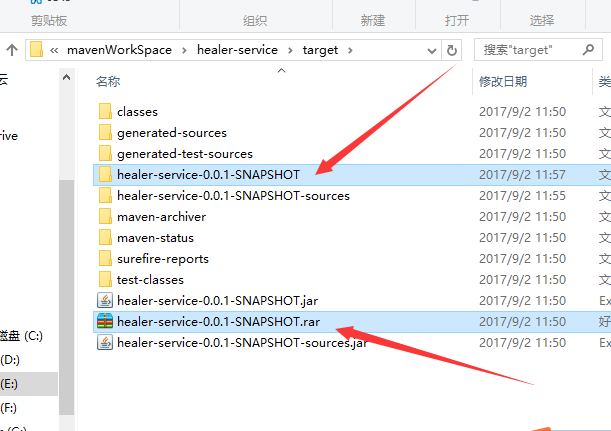


# 3、rar ，将自己项目所有的依赖导入到rar包中，可以包括 自己，没有

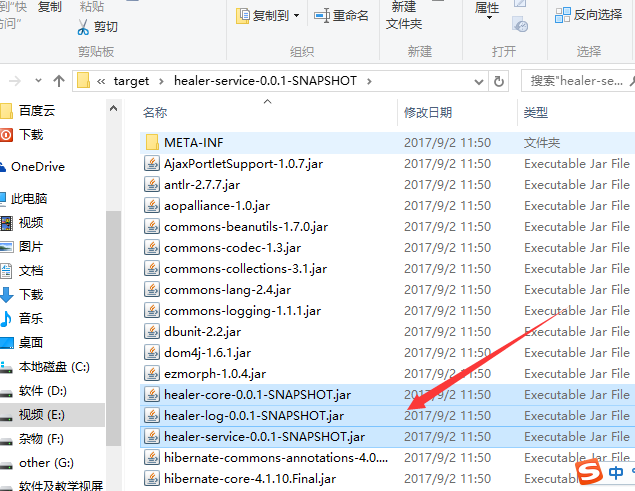
## 1、<includeJar>true</includeJar> 表示是否包含自己

|  |
| --- |
| <!-- 25、rar 将自己项目所有的依赖导入到rar包中，包括 自己依赖的其他项目  <includeJar>true</includeJar> 表示是否包含自己 -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-rar-plugin</artifactId>  <version>2.3</version>  <executions>  <execution>  <phase>package</phase>  <goals>  <goal>rar</goal>  </goals>  </execution>  </executions>  <configuration>  <!-- 是否包含有我们自己的jar包 true 包含 -->  <includeJar>true</includeJar>  </configuration>  </plugin>  </plugins> |

## 2、执行clean package，出现了rar 包，没有class文件



### 1、观察，所有healer-sevice的依赖全部放到一起，因为上面出现了include，所以自己的healer-sevice也被打包成jar封装了起来



# 4、sql-maven-plugin ，关于sql的一个插件，可以在编译命令之后创建数据库，和创建表，以及执行一些sql命令（根据指定目录下的sql文件）

## 1、插件代码如下，仔细观察里面的注释，

### 1、里面有连个excution，表示会在两个编译命令（compile和test）之后执行两次sql语句，第一次是语句形式，第二次会找目录下sql文件中的sql语句

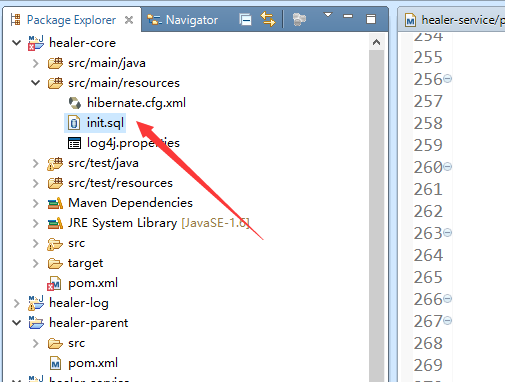
### 2、以下srcFiles代码中给的src下面的路径，只要子类项目（本例healer-core）的pom.xml中，添加了如下插件plugin片段，并且healer-core所在的src目录中具备sql文件，就会执行该目录下的sql文件中的命令

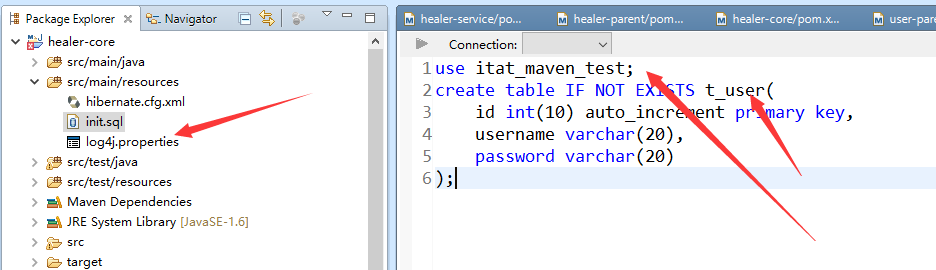
|  |
| --- |
| <!-- 26、sql-maven-plugin 编译时候 创建数据库 test时候创建表 -->  <plugin>  <groupId>org.codehaus.mojo</groupId>  <artifactId>sql-maven-plugin</artifactId>  <version>1.5</version>  <!-- 使用插件依然可以指定相应的依赖 ,但是仅仅在插件编译的时候有用-->  <dependencies>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.24</version>  </dependency>  </dependencies>  <!-- 因为我们上面设置了property标签，里面设置了值，所以能够使用 ${jdbc.driver}-->  <configuration>  <driver>${jdbc.driver}</driver>  <url>${jdbc.url}</url>  <username>${jdbc.username}</username>  <password>${jdbc.password}</password>  </configuration>  <executions>  <execution>  <id>create-db</id>  <phase>compile</phase>  <goals>  <goal>execute</goal><!-- 编译之后执行下面的sql语句 -->  </goals>  <configuration>  <sqlCommand>  create database IF NOT EXISTS itat\_maven\_test  </sqlCommand>  </configuration>  </execution>  <execution>  <id>init-table</id>  <phase>test-compile</phase>  <goals>  <goal>execute</goal>  </goals>  <configuration>  <srcFiles>  <srcFile>src/main/resources/init.sql</srcFile>  </srcFiles>  </configuration>  </execution>  </executions>  </plugin> |

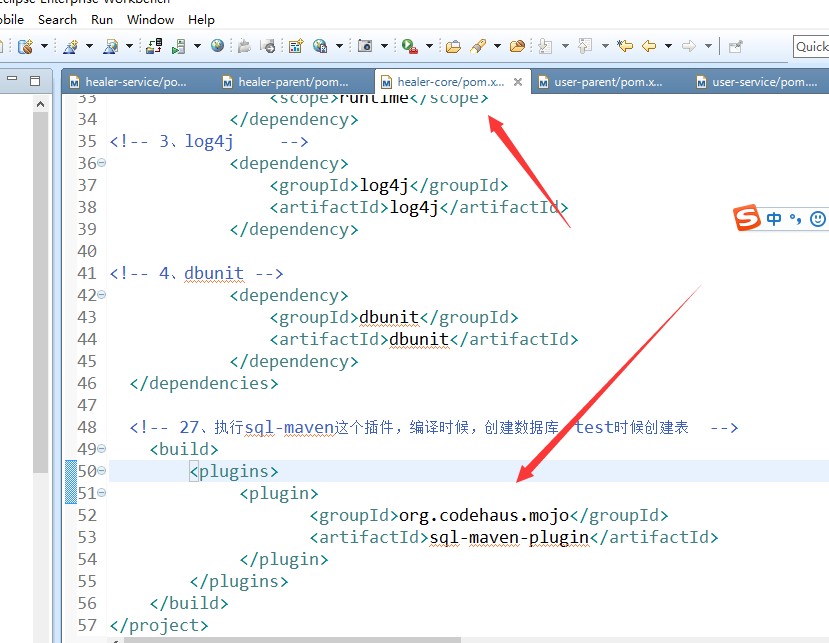
### 3、上面的propertits有了才能使用里面的${jdbc.driver}

|  |
| --- |
| <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <!-- 27、配置插件的sql-maven-plugin 里面需要的参数-->  <jdbc.driver>com.mysql.jdbc.Driver</jdbc.driver>  <jdbc.url>jdbc:mysql://localhost:3306</jdbc.url>  <jdbc.username>root</jdbc.username>  <jdbc.password>147094</jdbc.password>  </properties> |

### 4、healer-core，中src/main/resource目录下存在init.sql并且，pom.xml也引入了插件

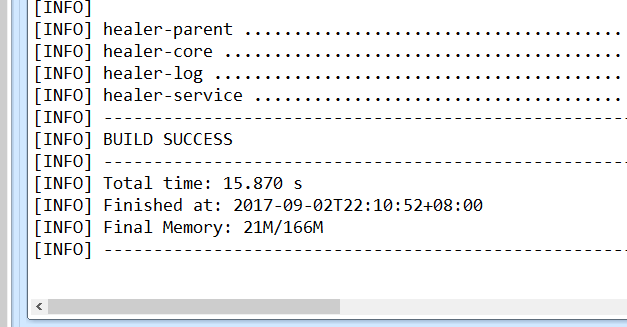


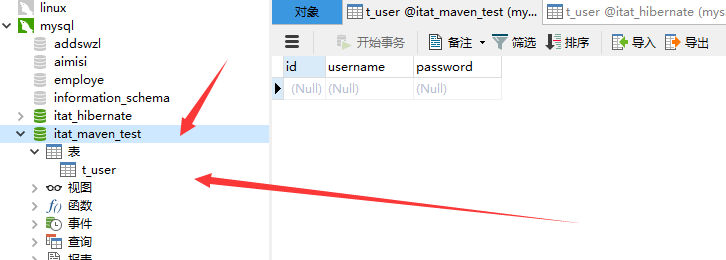




### 5、开始测试，虽然项目子项目报了点错，但是不影响

#### clean test，结果发现多了一个数据库和，一张表





### 6、解决5中的报错，父类中添加，lifecycle-mapping ，不需要添加到healer-core中，这个描述的插件解决healer-core使用sql-maven-plugin 出现的报错信息(虽然不影响编译，但是看着不舒服)

Plugin execution not covered by lifecycle configuration:

|  |
| --- |
| <!-- 28、lifecycle-mapping  解决healer-core使用sql-maven-plugin 出现的报错信息(虽然不影响编译，但是看着不舒服) -->  <plugin>  <groupId>org.eclipse.m2e</groupId>  <artifactId>lifecycle-mapping</artifactId>  <version>1.0.0</version>  <configuration>  <lifecycleMappingMetadata>  <pluginExecutions>  <pluginExecution>  <pluginExecutionFilter>  <groupId>org.codehaus.mojo</groupId>  <artifactId>sql-maven-plugin</artifactId>  <versionRange>[1.5,)</versionRange>  <goals>  <goal>execute</goal>  </goals>  </pluginExecutionFilter>  <action>  <ignore />  </action>  </pluginExecution>  </pluginExecutions>  </lifecycleMappingMetadata>  </configuration>  </plugin> |