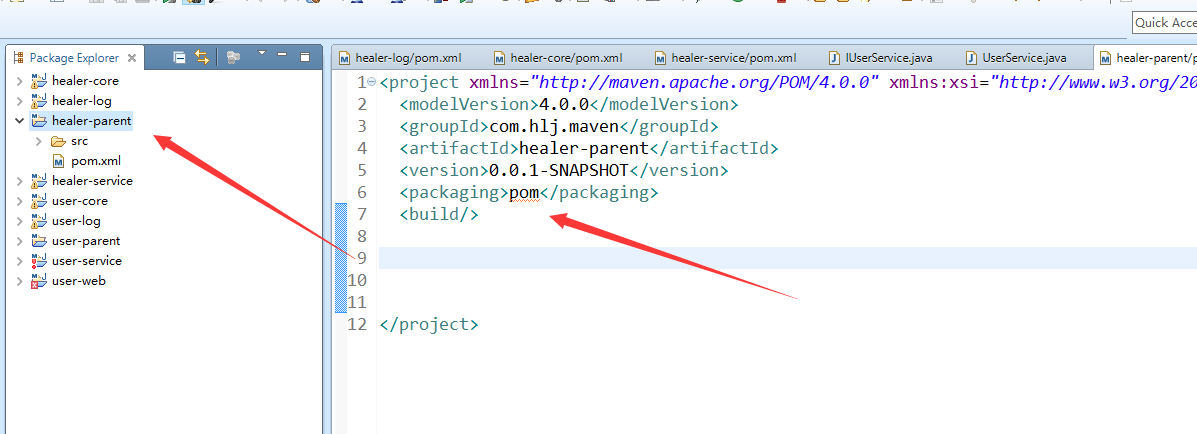
# 接着之前的3个项目healer-core，healer-log，healer-service来讲，不知道为什么没有jar，应该默认就是jar，所以不写jar

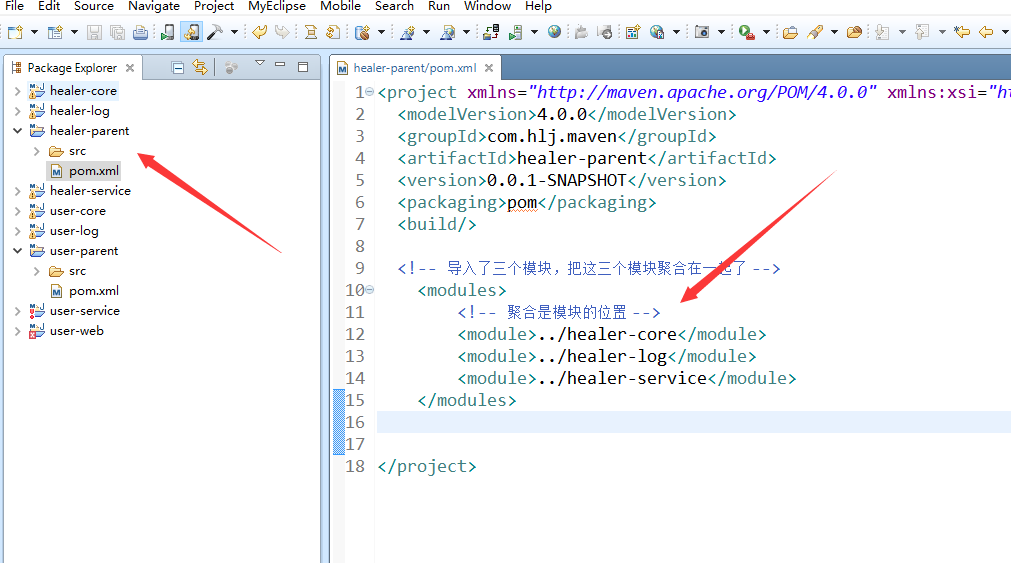
# 1、新建项目parent搭建聚合

## 1、新建一个简单的maven项目pom

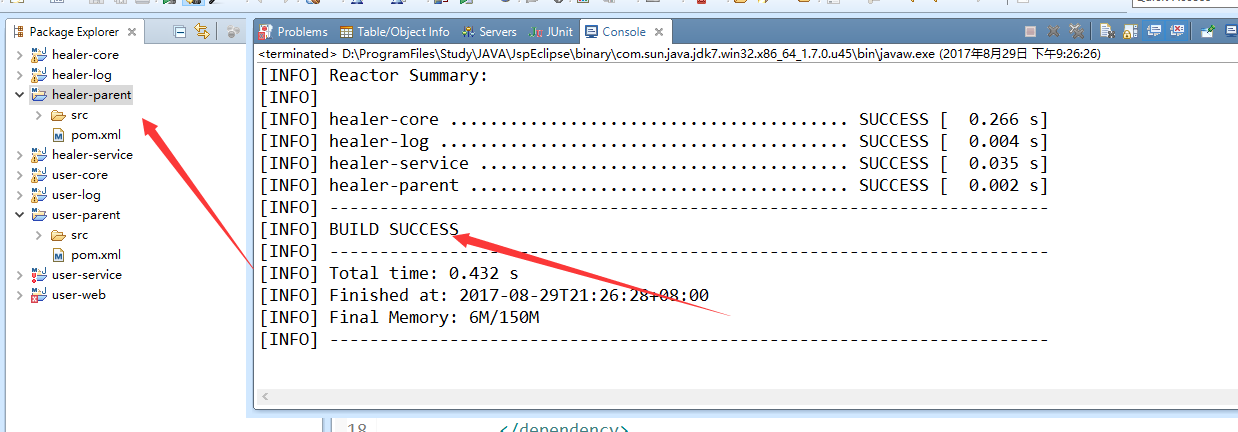


## 2、导入这三个module聚合工程

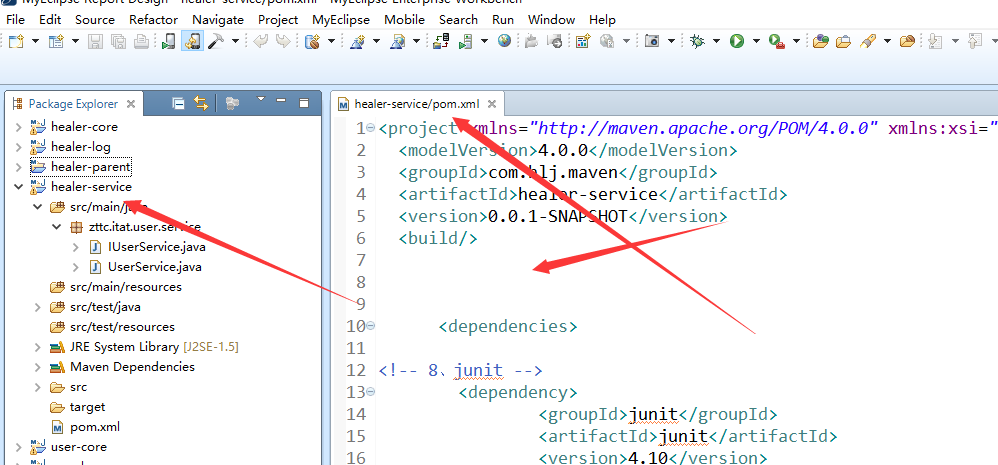
|  |
| --- |
| <!-- 导入了三个模块，把这三个模块聚合在一起了 -->  <modules>  <!-- 聚合是模块的位置 -->  <module>../healer-core</module>  <module>../healer-log</module>  <module>../healer-service</module>  </modules> |



## 3、直接build clean，发现控制台出现了我们所引入的项目

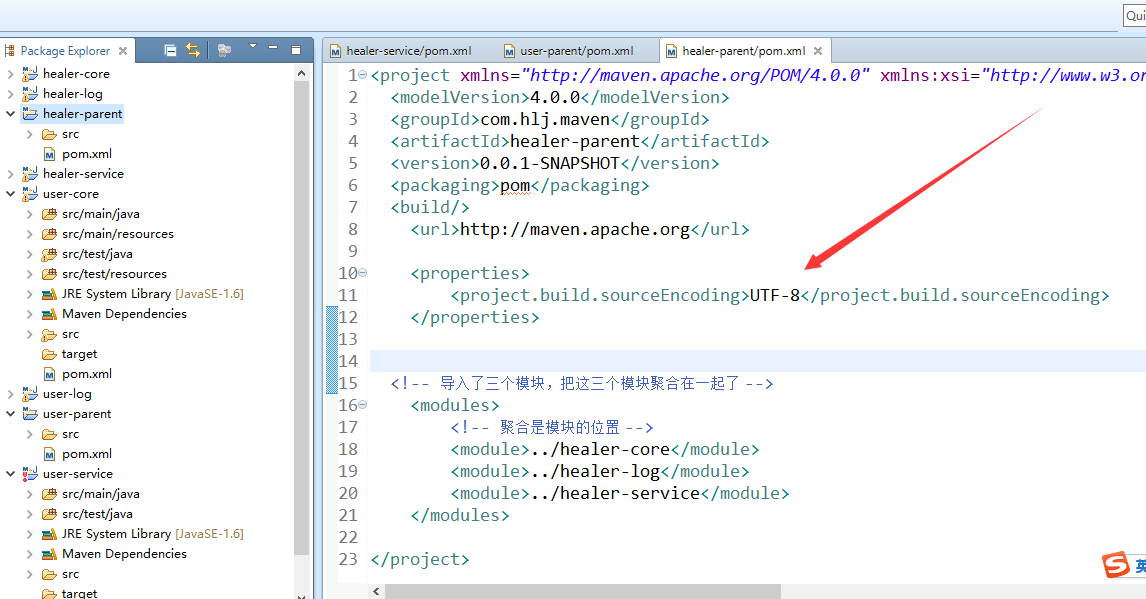


## 4、在其他三个的项目中这时候还没有出现相关的parent父类



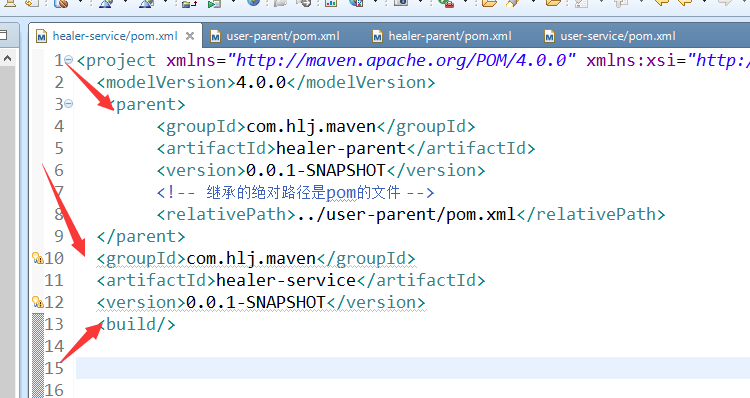
# 2、parent制作为父类（解决以上jar包的大量重复）

## 1、父类中实现编码UTF-8 则子类中就不需要写



|  |
| --- |
| <!-- 14 、实现父类UTF-8 -->  <url>http://maven.apache.org</url>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  </properties> |

## 2、heaer-core，log，service。module中开始写入parent实现继承 则会自动提示我们的groupId和version以及存在了，则主动删去



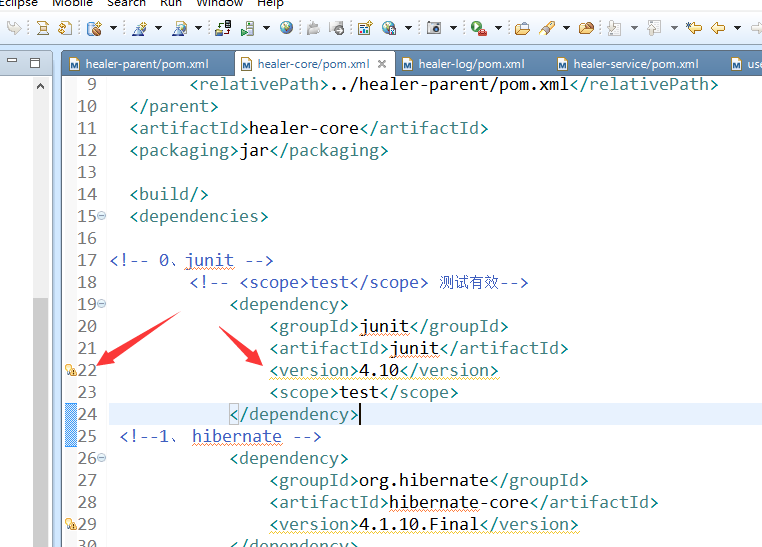
### 1、删除之后的xml如下

|  |
| --- |
| <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>com.hlj.maven</groupId>  <artifactId>healer-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  <!-- 继承的绝对路径是pom的文件 -->  <relativePath>../user-parent/pom.xml</relativePath>  </parent>  <artifactId>healer-log</artifactId>  <packaging>jar</packaging>    <build/> |

### 3、父类建立依赖管理，将子类中所有的依赖全部放进去（包括healer-core和healer-logy，因为healer-servie要用），这样即使父类有了，子类如果不写依赖配置，则不能继承。父类这个时候就相当于是一个大的项目工厂，子类要什么写什么即可

|  |
| --- |
| <dependencyManagement>  <dependencies>  各种依赖  </dependencies>  </dependencyManagement> |

### 4、将healer-core healer-log healer-service中的依赖全部方进入父类parent之后，会发现这三个钟依赖的版本号已经写过了，这个时候不用写了，删掉子类相关的版本version，scope，



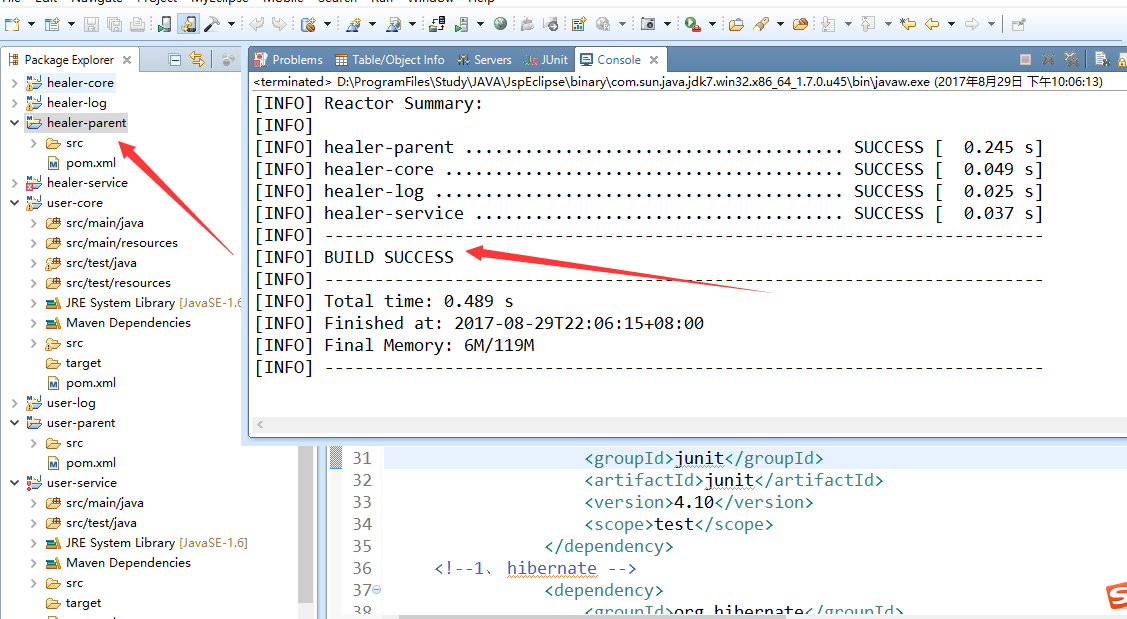
#### 1、以service举例

|  |
| --- |
| <dependencies>  <!-- 8、junit -->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  </dependency>  <!-- 9、healer-core -->    <dependency>  <groupId>com.hlj.maven</groupId>  <artifactId>healer-core</artifactId>  </dependency>    <!-- 10、healer-log 使用下面的变量不能够跳过去-->  <dependency>  <groupId>${project.groupId}</groupId>  <artifactId>healer-log</artifactId>  <!-- 12 、排除依赖 exclusions-->  <exclusions>  <!-- 可以排除依赖 -->  <exclusion>  <groupId>commons-logging</groupId>  <artifactId>commons-logging</artifactId>  </exclusion>  </exclusions>  </dependency>  <!-- 11、easymock -->  <dependency>  <groupId>org.easymock</groupId>  <artifactId>easymock</artifactId>  </dependency>    </dependencies> |

#### 2、父类parent中如下

|  |
| --- |
| <dependencyManagement>  <dependencies>  <!-- 0、junit -->  <!-- <scope>test</scope> 测试有效-->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.10</version>  <scope>test</scope>  </dependency>  <!--1、 hibernate -->  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>4.1.10.Final</version>  </dependency>    <!-- 2、 mysql 效-->  <!-- <scope>runtime</scope> 运行有效-->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.24</version>  <scope>runtime</scope>  </dependency>  <!-- 3、log4j -->  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.17</version>  </dependency>    <!-- 4、dbunit -->  <dependency>  <groupId>dbunit</groupId>  <artifactId>dbunit</artifactId>  <version>2.2</version>  </dependency>  <!--  5、junit 已存在  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.10</version>  <scope>test</scope>  </dependency> -->  <!-- 6、commons-logging -->    <dependency>  <groupId>commons-logging</groupId>  <artifactId>commons-logging</artifactId>  <version>1.1.1</version>  </dependency>  <!--  7、log4j 已经存在    <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.17</version>  </dependency> -->    <!-- 9、healer-core -->    <dependency>  <groupId>com.hlj.maven</groupId>  <artifactId>healer-core</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>    <!-- 10、healer-log 使用下面的变量不能够跳过去-->  <dependency>  <groupId>${project.groupId}</groupId>  <artifactId>healer-log</artifactId>  <version>${project.version}</version>  <!-- 12 、排除依赖 exclusions-->  <exclusions>  <!-- 可以排除依赖 -->  <exclusion>  <groupId>commons-logging</groupId>  <artifactId>commons-logging</artifactId>  </exclusion>  </exclusions>  </dependency>  <!-- 11、easymock -->  <dependency>  <groupId>org.easymock</groupId>  <artifactId>easymock</artifactId>  <version>3.1</version>  <scope>test</scope>  </dependency>  </dependencies>  </dependencyManagement> |

## 5、这个时候基本上是搭建完成了，父类 build clean package试试



# 3、聚合和继承的路径

## 1、聚合模块的位置

|  |
| --- |
| <modules>  <!-- 聚合是模块的位置 -->  <module>../healer-core</module>  <module>../healer-log</module>  <module>../healer-service</module>  </modules> |

## 2、继承

|  |
| --- |
| <parent>  <groupId>com.hlj.maven</groupId>  <artifactId>healer-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  <!-- 继承的绝对路径是pom的文件 -->  <relativePath>../healer-parent/pom.xml</relativePath>  </parent> |