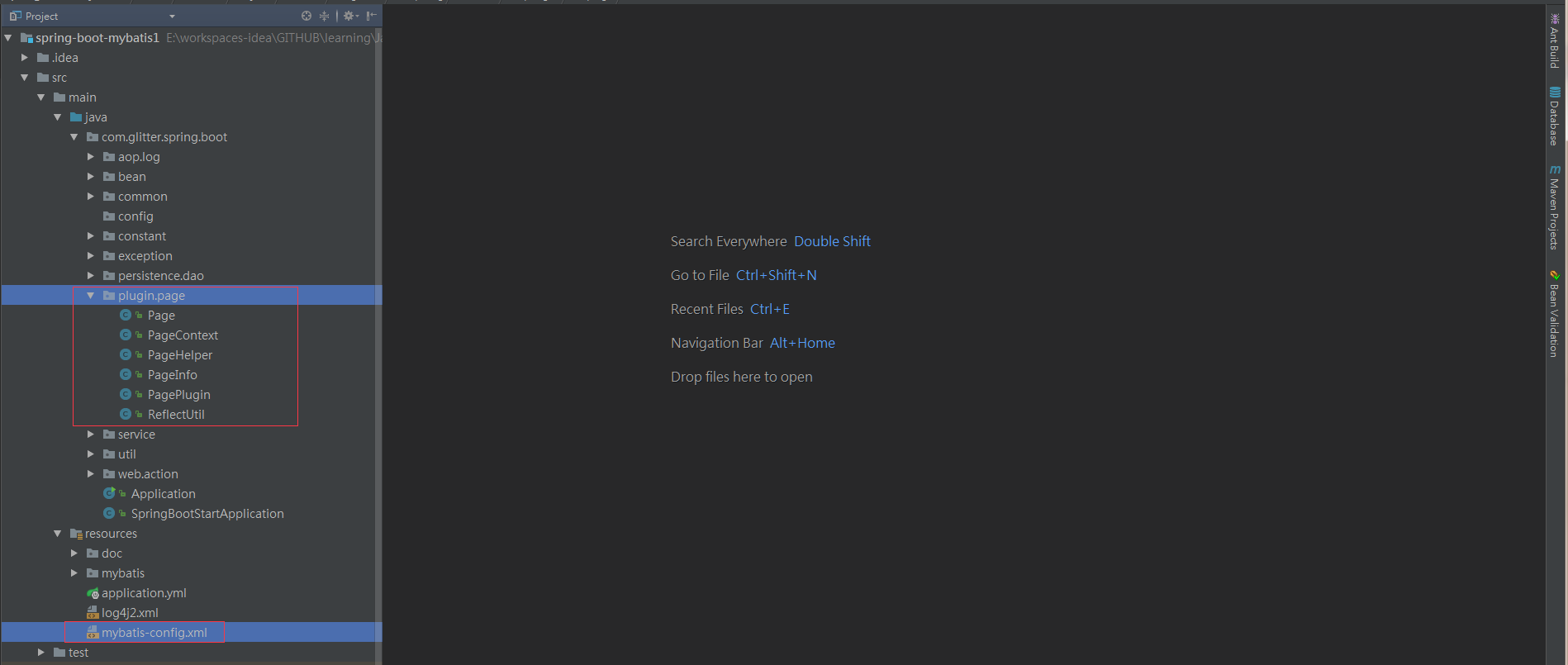
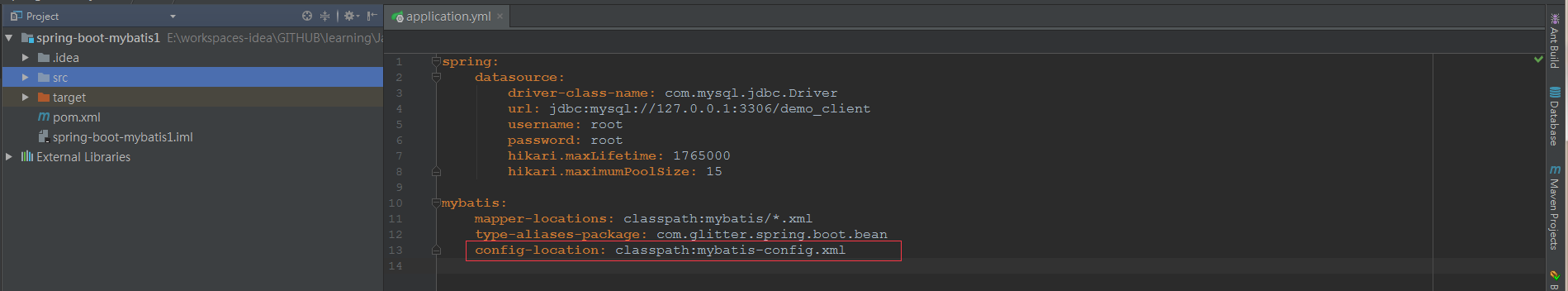
1.导入相关java文件，和mybatis-config.xml文件。



2.mybatis-config.xml内容如下，其中的plugins插件引入也可以通过在application.yml中配置的方式来引入，比较推荐，因为spring boot的核心思想就是去掉xml等配置文件，多采用约定，自动和注解的方式来进行配置。内容中的settings配置如果不使用xml而是直接在application.yml配置，这个事情有待研究。

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
 <settings>  
 <setting name="cacheEnabled" value="false" />  
 <setting name="lazyLoadingEnabled" value="false" />  
 <setting name="aggressiveLazyLoading" value="false" />  
 <setting name="multipleResultSetsEnabled" value="true" />  
 <setting name="useColumnLabel" value="true" />  
 <setting name="useGeneratedKeys" value="true" />  
 <setting name="autoMappingBehavior" value="FULL" />  
 <setting name="defaultStatementTimeout" value="25000" />  
 </settings>  
 <plugins>  
 <plugin interceptor="com.glitter.spring.boot.plugin.page.PagePlugin">  
 <property name="dialect" value="mysql" />  
 </plugin>  
 </plugins>  
</configuration>

3.在application.yml中增加对mybatis-config.xml的指定。



4.项目中分页代码。有各种情况吧，简单说就是需要分页就在dao方法查询前使用PageHelper.startPage(pageNum, pageSize);，这会将本次请求的page参数放入本次请求的线程中，在整个线程中都存在page参数。

后续的其他任何查询方法如果没有重新进行PageHelper.startPage(pageNum, pageSize);的声明或者清理PageHelper.clearPage();，则会一直沿用最第一个，后面有声明或清理都会以最接近调用方法的那次PageHelper调用为准，每调用一次都会影响这个线程中的page参数值而已。

最佳实践:所以一般我们一个分页查询完毕后，就手动调用一下PageHelper.clearPage();即可，避免对后续的其他查询造成干扰，后面的查询如果又涉及到分页查询，那么就再重新声明一下PageHelper.startPage(pageNum, pageSize);，这样就比较好了。

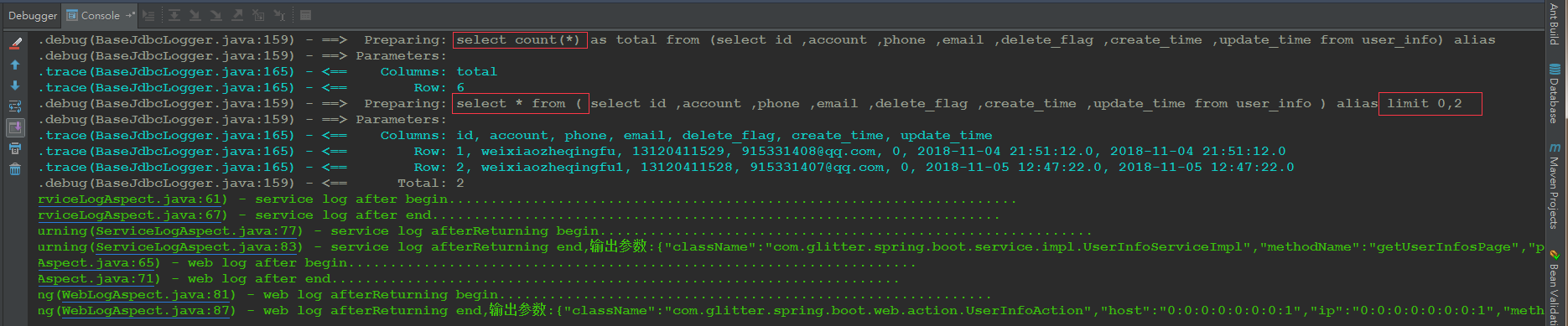
当然这个声明有时候声明一次就可以一直管用，比如先声明PageHelper.startPage(pageNum, pageSize);来一次分页查询，然后调用PageHelper.clearPage();进行上下文清理，那么如果后续的几个其他查询方法都不需要分页的话，其实都不需要再做任何声明了，PageHelper.clearPage();声明作为最后一次

声明没有再被其他声明覆盖就会影响到整个线程的方法，会一直有效。

文字描述太复杂，其实超级简单了。

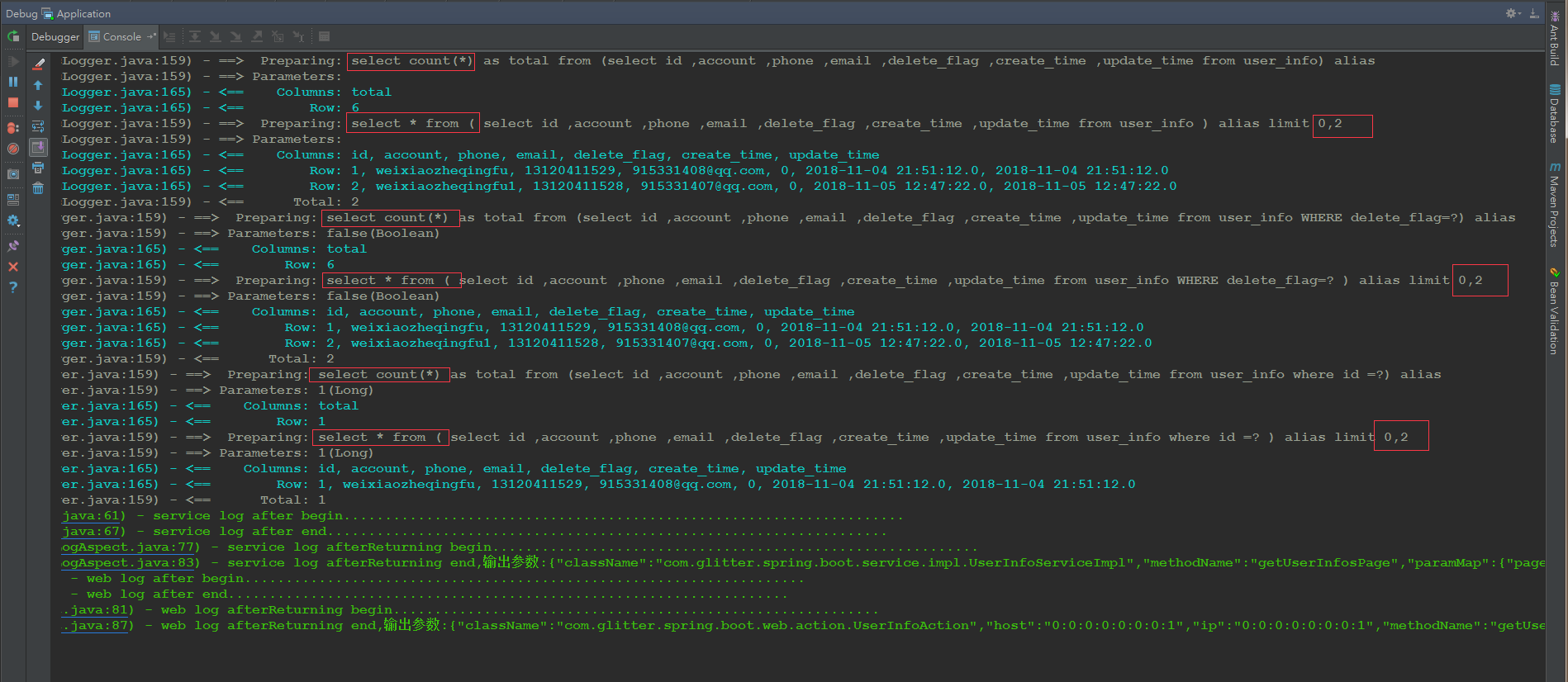
(1).常规情况，只有一个分页查询。

@Override  
public PageInfo<UserInfo> getUserInfosPage(Integer pageNum, Integer pageSize) {  
 PageHelper.*startPage*(pageNum, pageSize);  
 List<UserInfo> userInfos = userInfoDao.findAllList();  
 PageInfo<UserInfo> pageInfo = new PageInfo(userInfos);  
 return pageInfo;  
}



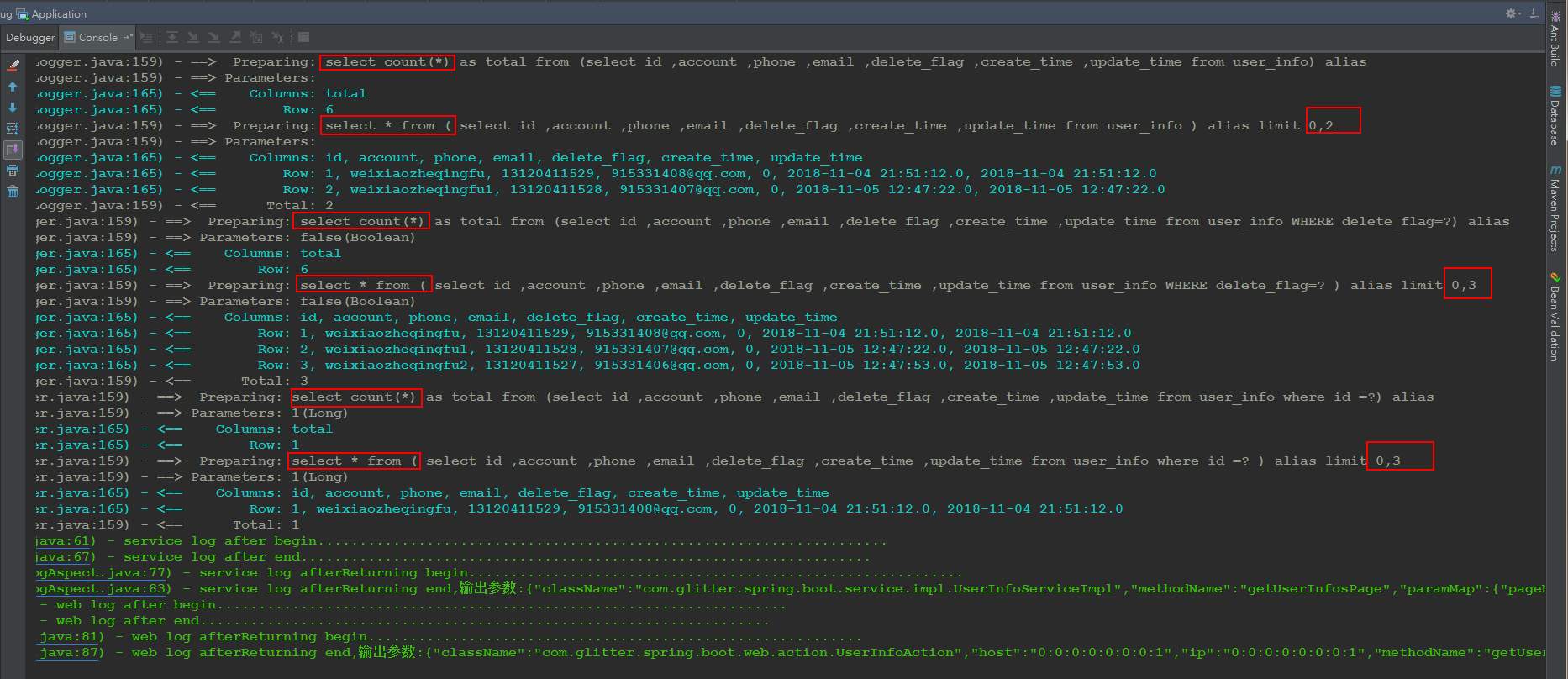
(2).如果后续还有其他查询sql语句需要分页，如果pageNum和pageSize保持和上面的pageNum和pageSize一样，那么可以直接顺着写。

@Override  
public PageInfo<UserInfo> getUserInfosPage(Integer pageNum, Integer pageSize) {  
 PageHelper.*startPage*(pageNum, pageSize);  
 List<UserInfo> userInfos = userInfoDao.findAllList();  
 PageInfo<UserInfo> pageInfo = new PageInfo(userInfos);  
  
 UserInfo record = new UserInfo();  
 record.setDeleteFlag(false);  
 List<UserInfo> list = userInfoDao.findList(record);  
  
 UserInfo userInfo = userInfoDao.getById(1L);  
 return pageInfo;  
}



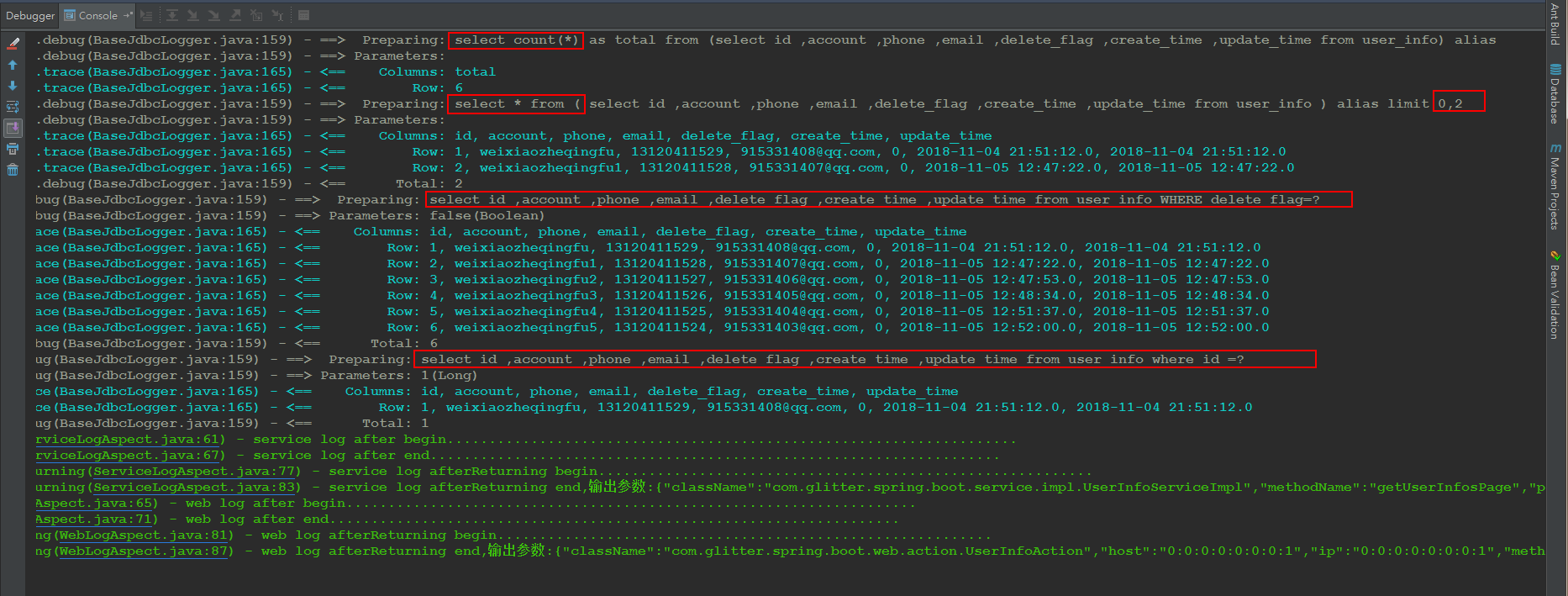
(3).如果后续还有其他查询sql语句需要分页，如果pageNum和pageSize想使用新的值。

@Override  
public PageInfo<UserInfo> getUserInfosPage(Integer pageNum, Integer pageSize) {  
 PageHelper.*startPage*(pageNum, pageSize);  
 List<UserInfo> userInfos = userInfoDao.findAllList();  
 PageInfo<UserInfo> pageInfo = new PageInfo(userInfos);  
  
 PageHelper.*startPage*(1, 3);  
 UserInfo record = new UserInfo();  
 record.setDeleteFlag(false);  
 List<UserInfo> list = userInfoDao.findList(record);  
  
 UserInfo userInfo = userInfoDao.getById(1L);  
 return pageInfo;  
}



(4).如果后续还有其他查询sql语句不需要分页。

@Override  
public PageInfo<UserInfo> getUserInfosPage(Integer pageNum, Integer pageSize) {  
 PageHelper.*startPage*(pageNum, pageSize);  
 List<UserInfo> userInfos = userInfoDao.findAllList();  
 PageInfo<UserInfo> pageInfo = new PageInfo(userInfos);  
  
 PageHelper.*clearPage*();  
  
 UserInfo record = new UserInfo();  
 record.setDeleteFlag(false);  
 List<UserInfo> list = userInfoDao.findList(record);  
  
 UserInfo userInfo = userInfoDao.getById(1L);  
 return pageInfo;  
}



(2).如果开始的sql不需要分页，后续还有其他查询sql语句需要分页。

@Override  
public PageInfo<UserInfo> getUserInfosPage(Integer pageNum, Integer pageSize) {  
 UserInfo record = new UserInfo();  
 record.setDeleteFlag(false);  
 List<UserInfo> list = userInfoDao.findList(record);  
  
 UserInfo userInfo = userInfoDao.getById(1L);  
  
 PageHelper.*startPage*(pageNum, pageSize);  
 List<UserInfo> userInfos = userInfoDao.findAllList();  
 PageInfo<UserInfo> pageInfo = new PageInfo(userInfos);  
  
 return pageInfo;  
}

