1. catch 里面打日志
2. public class SHAUtil {  
    public static String shaEncode(String str){  
    MessageDigest messageDigest;  
    String encdeStr = "";  
    try {  
    messageDigest = MessageDigest.*getInstance*("SHA-256");  
    byte[] hash = messageDigest.digest(str.getBytes("UTF-8"));  
    encdeStr = Hex.*encodeHexString*(hash);  
    } catch (NoSuchAlgorithmException e) {  
    e.printStackTrace();  
    } catch (UnsupportedEncodingException e) {  
    e.printStackTrace();  
    }  
    return encdeStr;  
    }  
     
    */\*\*  
    \* 测试主函数  
    \** ***@param*** *args  
    \** ***@throws*** *Exception  
    \*/* public static void main(String args[]) throws Exception {  
    String str = new String("123456");  
    System.*out*.println("原始：" + str);  
    System.*out*.println("SHA后：" + *shaEncode*(str));  
    }  
   }
3. @Component  
   public class RedisUtil<T> {  
    @Autowired  
    private RedisTemplate<String, String> redisTemplate;  
     
    @Autowired  
    private ValueOperations valueOperations;  
     
    public void add(String key, T entity) {  
    add(key, entity, 3600L);  
    }  
     
    public void addList(String key, List<T> list) {  
    addList(key, list, 3600L);  
    }  
     
    public void add(String key, T entity, Long time) {  
    try {  
    valueOperations.set(key, JSON.*toJSONString*(entity), time, TimeUnit.*MINUTES*);  
    } catch (Exception e) {  
    e.printStackTrace();  
    }  
    }  
     
    public void addList(String key, List<T> list, Long time) {  
    try {  
    valueOperations.set(key, JSON.*toJSONString*(list), time, TimeUnit.*MINUTES*);  
    } catch (Exception e) {  
    e.printStackTrace();  
    }  
    }  
     
    public T get(String key, Class<T> c) {  
    try {  
    T entity = null;  
    String json = (String) valueOperations.get(key);  
    if (StringUtils.*isNotEmpty*(json)) {  
    entity = JSON.*parseObject*(json, c);  
    }  
    return entity;  
    } catch (Exception e) {  
    return null;  
    }  
    }  
     
    public List<T> getList(String key, Class<T> c) {  
    try {  
    List<T> list = null;  
    String json = (String) valueOperations.get(key);  
    if (StringUtils.*isNotEmpty*(json)) {  
    list = JSONObject.*parseArray*(json, c);  
    }  
    return list;  
    } catch (Exception e) {  
    return null;  
    }  
    }  
     
    public void delete(String key) {  
    try {  
    if (key.contains("\*")) {  
    Set<String> set = redisTemplate.keys(key);  
    Iterator<String> it = set.iterator();  
    while (it.hasNext())  
    redisTemplate.delete(it.next());  
    } else  
    redisTemplate.delete(key);  
    } catch (Exception e) {  
    e.printStackTrace();  
    }  
    }  
   }
4. */\*\*  
    \* author: zhouhao8  
    \* Date: 2017/10/10  
    \* Time: 16:52  
    \*/*public class IPUtil {  
     
   // public static void main(String[] args) {  
   // System.out.println(isIPv4Address("127.0.0.1"));  
   // System.out.println(isIPv4Address("254.21.0.198"));  
   // System.out.println(isIPv4Address("255.777.0.198"));  
   // System.out.println(isIPv4Address("07.25.8.198"));  
   // System.out.println(isIPv4Address("1207.25.8.198"));  
   // System.out.println(isIPv4Address("qq107.25.8.198"));  
   // System.out.println(isIPv4Address("\\.\\.\\.107.25.8.198"));  
   // System.out.println(isIPv4Address("\\.\\.\\. 7.25.8.198"));  
   // System.out.println(isIPv4Address("107.25.r8.1982"));  
   // System.out.println(isIPv4Address("107.225.8.19"));  
   // System.out.println(isIPv4Address("0.0.0.0"));  
   // System.out.println(isIPv4Address("00.0.0.0"));  
   // System.out.println(isIPv4Address("0.202.1.0"));  
   // System.out.println(isIPv4Address("0.202.1.226"));  
   // System.out.println(isIPv4Address("249.202.1.0"));  
   // }  
     
    private static final Pattern *IPV4\_PATTERN* =  
    Pattern.*compile*(  
    "^(25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)(\\.(25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)){3}$");  
    private static final Pattern *PORT\_PATTERN* =  
    Pattern.*compile*(  
    "/^[1-9]$|(^[1-9][0-9]$)|(^[1-9][0-9][0-9]$)|(^[1-9][0-9][0-9][0-9]$)|(^[1-6][0-5][0-5][0-3][0-5]$)/");  
    private static final Pattern *IPV6\_STD\_PATTERN* =  
    Pattern.*compile*(  
    "^(?:[0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}$");  
     
    private static final Pattern *IPV6\_HEX\_COMPRESSED\_PATTERN* =  
    Pattern.*compile*(  
    "^((?:[0-9A-Fa-f]{1,4}(?::[0-9A-Fa-f]{1,4})\*)?)::((?:[0-9A-Fa-f]{1,4}(?::[0-9A-Fa-f]{1,4})\*)?)$");  
     
    public static boolean isIPv4Address(final String input) {  
    return *IPV4\_PATTERN*.matcher(input).matches();  
    }  
     
    public static boolean isIPv6StdAddress(final String input) {  
    return *IPV6\_STD\_PATTERN*.matcher(input).matches();  
    }  
     
    public static boolean isIPv6HexCompressedAddress(final String input) {  
    return *IPV6\_HEX\_COMPRESSED\_PATTERN*.matcher(input).matches();  
    }  
     
    public static boolean isIPv6Address(final String input) {  
    return *isIPv6StdAddress*(input) || *isIPv6HexCompressedAddress*(input);  
    }  
     
    public static boolean isPort(final String input) {  
    return *PORT\_PATTERN*.matcher(input).matches();  
    }  
   }
5. */\*\*  
    \* 工具类  
    \*  
    \** ***@Author*** *wenjiahui  
    \** ***@Create*** *2017/10/11 14:07  
    \** ***@Modify*** *By:  
    \*\*/*public class BaseUtils {  
     
    */\*\*  
    \** ***@param*** *str  
    \** ***@return*** *boolean  
    \** ***@Description:*** *判断字符串对象是否为null或者字符串为空窜  
    \** ***@author*** *wenjiahui 2017/10/11 14:07  
    \*/* public static boolean isNotNullOrEmpty(String str) {  
    return !*isNullOrEmpty*(str);  
    }  
     
    */\*\*  
    \** ***@param*** *str  
    \** ***@return*** *boolean  
    \** ***@Description:*** *是否为null或空字符串  
    \** ***@author*** *wenjiahui 2017/10/11 14:07  
    \*/* public static boolean isNullOrEmpty(String str) {  
    return str == null || str.isEmpty();  
    }  
     
    */\*\*  
    \** ***@param*** *list  
    \** ***@return*** *boolean  
    \** ***@Description:*** *判断list是否有合法数据  
    \** ***@author*** *wenjiahui 2017/10/11 14:07  
    \*/* public static boolean listNotEmpty(Collection<?> list) {  
    return null != list && list.size() > 0;  
    }  
     
    */\*\*  
    \** ***@param*** *list  
    \** ***@return*** *boolean  
    \** ***@Description:*** *判断array是否有合法数据  
    \** ***@author*** *wenjiahui 2017/10/11 14:07  
    \*/* public static boolean listNotEmpty(Object[] list) {  
    return null != list && list.length > 0;  
    }  
     
    */\*\*  
    \** ***@param*** *num  
    \** ***@return*** *boolean  
    \** ***@Description:*** *判断数字是否合法  
    \** ***@author*** *wenjiahui 2017/10/11 14:07  
    \*/* public static boolean numValidate(Integer num) {  
    return num != null && num.intValue() > 0;  
    }  
     
    */\*\*  
    \** ***@param*** *num  
    \** ***@return*** *boolean  
    \** ***@Description:*** *判断数字是否合法  
    \** ***@author*** *wenjiahui 2017/10/11 14:07  
    \*/* public static boolean numValidate(Long num) {  
    return num != null && num.longValue() > 0;  
    }  
     
    */\*\*  
    \* id转换成List  
    \*  
    \** ***@param*** *id  
    \** ***@return*** *List*<*Long*>  
    *\** ***@author*** *wenjiahui 2017/10/11 14:07  
    \*/* public static List<Long> convertLongToList(Long id) {  
    List<Long> list = new ArrayList<Long>();  
    list.add(id);  
    return list;  
    }  
     
   }
6. */\*\*  
    \* 权限授予工具类  
    \*  
    \** ***@Author*** *haoyanxuan  
    \** ***@Create*** *2017/9/26 14:36  
    \** ***@Modify*** *By:  
    \*\*/*public class AuthorizeUtil {  
    */\*\*  
    \* 找出target数组中与source数组不同的元素  
    \*  
    \** ***@param*** *source  
    \** ***@param*** *target  
    \** ***@return*** *\*/* public static Set<Long> checkDif(String[] source, String[] target) {  
    Set<Long> difToSource = new HashSet<>();  
    Set<String> tmp = new HashSet<>();  
    for (String s : source) {  
    tmp.add(s);  
    }  
    for (String s : target) {  
    if (tmp.add(s)) {  
    difToSource.add(Long.*parseLong*(s));  
    }  
    }  
    return difToSource;  
    }  
   }
7. */\*\*  
    \* redis缓存配置类  
    \*  
    \** ***@Author*** *haoyanxuan  
    \** ***@Create*** *2017-09-21 11:33  
    \*\*/*@Configuration  
   @EnableCaching  
   public class RediesCacheConfig extends CachingConfigurerSupport {  
    //为什么加了以后会连接池无法取出连接(是因为没有关闭?)  
   // @Bean  
   // public RedisConnectionFactory redisConnectionFactory(){  
   // return new JedisConnectionFactory();  
   // }  
     
   // @Bean  
   // public RedisCacheManager cacheManager() {  
   // RedisCacheManager redisCacheManager = new RedisCacheManager(redisTemplate());  
   // return redisCacheManager;  
   // }  
     
    @Bean  
    public ValueOperations valueOperations(RedisTemplate<Object, Object> redisTemplate) {  
    return redisTemplate.opsForValue();  
    }  
     
    @Bean  
    public ListOperations listOperations(RedisTemplate<Object, Object> redisTemplate) {  
    return redisTemplate.opsForList();  
    }  
     
    @Bean  
    public HashOperations hashOperations(RedisTemplate<Object, Object> redisTemplate) {  
    return redisTemplate.opsForHash();  
    }  
     
    @Bean  
    public RedisTemplate<Object, Object> redisTemplate(RedisConnectionFactory redisConnectionFactory) {  
    RedisTemplate<Object, Object> redisTemplate = new RedisTemplate<Object, Object>();  
    redisTemplate.setConnectionFactory(redisConnectionFactory);  
    // 开启事务支持  
    redisTemplate.setEnableTransactionSupport(true);  
    // 使用String格式序列化缓存键  
    StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();  
    redisTemplate.setKeySerializer(stringRedisSerializer);  
    redisTemplate.setHashKeySerializer(stringRedisSerializer);  
    return redisTemplate;  
    }  
   }
8. #spring.profiles.active=dev  
   spring.profiles.active=test,debugS  
   #spring.profiles.active=pro  
   logging.level.root=info  
   logging.level.org.springframework=info
9. server.port=5555  
   spring.application.name=cmss  
   spring.cloud.zookeeper.connect-string=10.33.49.81:2181  
   spring.cloud.zookeeper.discovery.preferIpAddress=true  
   spring.redis.host=127.0.0.1  
   spring.redis.port=6379  
   spring.redis.pool.max-idle=5000  
   spring.redis.pool.min-idle=1000  
   spring.redis.pool.max-active=5000  
   spring.redis.pool.max-wait=10000  
   spring.jpa.show-sql=true  
   spring.jpa.hibernate.ddl-auto=update  
   spring.jpa.properties.hibernate.current\_session\_context\_class=org.springframework.orm.hibernate4.SpringSessionContext  
   spring.datasource.driver-class-name=oracle.jdbc.OracleDriver  
   spring.datasource.url=jdbc:oracle:thin:@10.33.25.24:1521:orcl  
   spring.datasource.username=newcomer  
   spring.datasource.password=123456  
   spring.http.encoding.charset=utf-8
10. server.port=5555  
    spring.application.name=cmss  
    spring.cloud.zookeeper.connect-string=10.33.49.81:2181  
    spring.cloud.zookeeper.discovery.preferIpAddress=true  
    spring.redis.host=10.33.49.81  
    spring.redis.port=6379  
    spring.redis.pool.max-idle=5000  
    spring.redis.pool.min-idle=1000  
    spring.redis.pool.max-active=5000  
    spring.redis.pool.max-wait=10000  
    spring.jpa.show-sql=true  
    spring.jpa.hibernate.ddl-auto=update  
    spring.jpa.properties.hibernate.current\_session\_context\_class=org.springframework.orm.hibernate4.SpringSessionContext  
    spring.datasource.driver-class-name=oracle.jdbc.OracleDriver  
    spring.datasource.url=jdbc:oracle:thin:@10.33.25.24:1521:orcl  
    spring.datasource.username=formal  
    spring.datasource.password=123456  
    spring.http.encoding.charset=utf-8
11. server.port=5555  
    spring.application.name=cmss  
    spring.cloud.zookeeper.connect-string=10.33.49.81:2181  
    spring.cloud.zookeeper.discovery.preferIpAddress=true  
    spring.redis.host=10.33.49.81  
    spring.redis.port=6379  
    spring.redis.pool.max-idle=5000  
    spring.redis.pool.min-idle=1000  
    spring.redis.pool.max-active=5000  
    spring.redis.pool.max-wait=10000  
    spring.jpa.show-sql=true  
    spring.jpa.hibernate.ddl-auto=update  
    spring.jpa.properties.hibernate.current\_session\_context\_class=org.springframework.orm.hibernate4.SpringSessionContext  
    spring.datasource.driver-class-name=oracle.jdbc.OracleDriver  
    spring.datasource.url=jdbc:oracle:thin:@10.33.25.24:1521:orcl  
    spring.datasource.username=newcomer  
    #spring.datasource.username=newcommer1  
    spring.datasource.password=123456  
    spring.http.encoding.charset=utf-8
12. <?xml version="1.0" encoding="UTF-8"?>  
      
    <configuration scan="true" scanPeriod="1 seconds" debug="false">  
     <include resource="org/springframework/boot/logging/logback/defaults.xml" />  
     <contextName>cmss</contextName>  
     <property name="FILE\_OUT" value="%d{yyyy-MM-dd'T'HH:mm:ss:SSS'Z'} [%thread] %logger{36} - %msg%n" />  
     <appender name="console" class="ch.qos.logback.core.ConsoleAppender">  
     <encoder>  
     <pattern>${CONSOLE\_LOG\_PATTERN}</pattern>  
     </encoder>  
     </appender>  
      
     <appender name="debugAppender" class="ch.qos.logback.core.rolling.RollingFileAppender">  
      
     <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">  
     <!-- 文件名称 -->  
     <fileNamePattern>${catalina.base}/logs/cmssDebug.%d{yyyy-MM-dd}-%i.log</fileNamePattern>  
     <!-- 文件最大保存历史数量 -->  
     <MaxHistory>50</MaxHistory>  
     <maxFileSize>10MB</maxFileSize>  
     </rollingPolicy>  
     <encoder>  
     <pattern>  
     ${FILE\_OUT}  
     </pattern>  
     <charset>UTF-8</charset>  
     </encoder>  
     <filter class="ch.qos.logback.classic.filter.ThresholdFilter">  
     <level>DEBUG</level>  
     </filter>  
     </appender>  
      
     <!-- INFO -->  
     <appender name="infoAppender" class="ch.qos.logback.core.rolling.RollingFileAppender">  
      
     <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">  
     <!-- 文件名称 -->  
     <fileNamePattern>${catalina.base}/logs/cmssInfo.%d{yyyy-MM-dd}-%i.log</fileNamePattern>  
     <!-- 文件最大保存历史数量 -->  
     <MaxHistory>30</MaxHistory>  
     <maxFileSize>10MB</maxFileSize>  
     </rollingPolicy>  
     <encoder>  
     <pattern>  
     ${FILE\_OUT}  
     </pattern>  
     <charset>UTF-8</charset>  
     </encoder>  
     <filter class="ch.qos.logback.classic.filter.ThresholdFilter">  
     <level>INFO</level>  
     </filter>  
     </appender>  
      
     <!-- ERROR -->  
     <appender name="errorAppender" class="ch.qos.logback.core.rolling.RollingFileAppender">  
      
     <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">  
     <!-- 文件名称 -->  
     <fileNamePattern>${catalina.base}/logs/cmssError.%d{yyyy-MM-dd}-%i.log</fileNamePattern>  
     <!-- 文件最大保存历史数量 -->  
     <MaxHistory>10</MaxHistory>  
     <maxFileSize>5MB</maxFileSize>  
     </rollingPolicy>  
     <encoder>  
     <pattern>  
     ${FILE\_OUT}  
     </pattern>  
     <charset>UTF-8</charset>  
     </encoder>  
     <filter class="ch.qos.logback.classic.filter.ThresholdFilter">  
     <level>ERROR</level>  
     </filter>  
     </appender>  
      
     <!-- info环境. -->  
     <springProfile name="info">  
     <root level="info">  
     <appender-ref ref="console" />  
     <appender-ref ref="infoAppender" />  
     </root>  
     </springProfile>  
      
     <!-- debug环境. -->  
     <springProfile name="debugS">  
     <root level="debug">  
     <appender-ref ref="console" />  
     <appender-ref ref="debugAppender" />  
     </root>  
     </springProfile>  
      
     <!-- error环境. -->  
     <springProfile name="error">  
     <root level="error">  
     <appender-ref ref="console" />  
     <appender-ref ref="errorAppender" />  
     </root>  
     </springProfile>  
      
    </configuration>
13. */\*\*  
     \* 定时迁移数据表任务  
     \*  
     \** ***@Author*** *haoyanxuan  
     \** ***@Create*** *2017/9/27 16:05  
     \** ***@Modify*** *By:  
     \*\*/*@Component  
    public class MigrateTask {  
      
     @Autowired  
     MigrateService migrateService;  
      
     @Scheduled ( cron = "10 0 19 ? \* SAT" )  
     public void executeMigrateDeleted() {  
     Date currentTime = new Date();  
     //每周六晚七点对数据库进行迁移  
     try {  
     migrateModules(currentTime);  
     } catch (NoSuchMethodException e) {  
     e.printStackTrace();  
     } catch (IllegalAccessException e) {  
     e.printStackTrace();  
     } catch (InvocationTargetException e) {  
     e.printStackTrace();  
     } catch (InstantiationException e) {  
     e.printStackTrace();  
     }  
      
     }  
      
     public void migrateModules(Date currentTime) throws NoSuchMethodException, IllegalAccessException, InvocationTargetException, InstantiationException {  
     //部门管理模块  
     migrateDeletedData(Department.class, CommonConstant.*CVSDEPARTMENTHEADERS*, currentTime);  
     migrateDeletedData(DepartmentRoleMap.class, CommonConstant.*CVSDEPARTMENTAUTHROLEHEADERS*, currentTime);  
     migrateDeletedData(DepartmentAreaMap.class, CommonConstant.*CVSDEPARTMENTAUTHAREAHEADERS*, currentTime);  
      
     //用户角色模块  
     migrateDeletedData(RoleDO.class, CommonConstant.*CVSROLEHEADERS*, currentTime);  
     migrateDeletedData(FunctionDO.class, CommonConstant.*CVSFUCTIONHEADERS*, currentTime);  
     migrateDeletedData(RoleFunctionMap.class, CommonConstant.*CVSROLEFUNCTIONMAPHEADERS*, currentTime);  
     migrateDeletedData(UserDO.class, CommonConstant.*CVSUSERHEADERS*, currentTime);  
     migrateDeletedData(UserRoleMap.class, CommonConstant.*CVSUSERROLEMAPHEADERS*, currentTime);  
      
     //资源管理模块  
     migrateDeletedData(DeviceDO.class, CommonConstant.*CVSDEVICEHEADERS*, currentTime);  
     migrateDeletedData(PointDO.class, CommonConstant.*CVSPOINTHEADERS*, currentTime);  
      
     //日志模块  
     migrateDeletedData(OperationLog.class, CommonConstant.*CVSOPERATIONLOGHEADERS*, currentTime);  
      
     //组织模块  
     migrateDeletedData(AreaDO.class, CommonConstant.*CVSAREAHEADERS*, currentTime);  
      
     //登录模块  
      
     }  
      
     public <T> void migrateDeletedData(Class<T> c, String[] csvHeaders, Date currentTime) throws NoSuchMethodException, InvocationTargetException, IllegalAccessException, InstantiationException {  
     List<T> list = migrateService.findByUpdateTime(currentTime, c);  
     System.*out*.println("查询完毕");  
     writeCSV(c.getName().substring(c.getName().lastIndexOf('.') + 1), csvHeaders, list, currentTime);  
     migrateService.hardDeleteByUpdateTime(currentTime, c);  
     }  
      
      
     public <T> void writeCSV(String moduleName, String[] csvHeaders, List<T> data, Date currentTime) {  
     // 定义一个CSV路径  
     SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd-HHmmss");  
     String dateString = formatter.format(currentTime);  
     String csvFilePath = "E://cmss\_doc//Deleted\_" + moduleName + "\_" + dateString + ".csv";  
     try {  
     // 创建CSV写对象 例如:CsvWriter(文件路径，分隔符，编码格式);  
     CsvWriter csvWriter = new CsvWriter(csvFilePath, ',', Charset.*forName*("GB2312"));  
     // 写表头  
     csvWriter.writeRecord(csvHeaders);  
     // 写内容  
     for (int i = 0; i < data.size(); i++) {  
     T t = data.get(i);  
     String[] csvContent = t.toString().split(",");  
     csvWriter.writeRecord(csvContent);  
     }  
     csvWriter.close();  
     System.*out*.println("--------CSV文件已经写入--------");  
     } catch (IOException e) {  
     e.printStackTrace();  
     }  
     }  
    }
14. */\*\*  
     \* 迁移删除数据的接口实现  
     \*  
     \** ***@Author*** *haoyanxuan  
     \** ***@Create*** *2017/10/10 19:28  
     \** ***@Modify*** *By:  
     \*\*/*@Service  
    public class MigrateServiceImpl<T> implements MigrateService<T> {  
     @Autowired  
     private SessionFactory sessionFactory;  
      
     //获取session  
     private Session getSession() {  
     return sessionFactory.openSession();  
     }  
      
     @Override  
     public void hardDeleteByUpdateTime(Date date, Class clazz) {  
     String hql = "delete from " + clazz.getName() + " t where t.deleted = 1 and t.updateTime < ?";  
     Query query = getSession().createQuery(hql);  
     query.setDate(0, date);  
     query.executeUpdate();  
     }  
      
     public List<T> findByUpdateTime(Date date, Class clazz) {  
     String hql = "from " + clazz.getName() + " t where t.deleted = 1 and t.updateTime < ?";  
     Query query = getSession().createQuery(hql);  
     query.setDate(0, date);  
     List list = query.list();  
     return list;  
     }  
      
    }
15. */\*\*  
     \* 迁移删除数据的服务接口  
     \*  
     \** ***@Author*** *haoyanxuan  
     \** ***@Create*** *2017/10/10 19:25  
     \** ***@Modify*** *By:  
     \*\*/*public interface MigrateService<T> {  
     void hardDeleteByUpdateTime(Date date, Class clazz);  
      
     List<T> findByUpdateTime(Date date, Class clazz);  
    }