尚马教育 JAVA 高级课程

Redis 缓存数据库

文档编号：C09

创建日期： 2017-07-07

最后修改日期：2021-02-22

版 本 号：V3.6

电子版文件名：尚马教育-第三阶段-9.redis 缓存数据库.docx

文档修改记录：

更新日期	更新作者	更新说明	版本号
2017-07-30	张元林	初始版本	V1.0
2018-08-01	王绍成	Mybatis 版本更新	V2.0
2019-10-25	冯勇涛	课件格式以及课程深度加深	V3.0
2021-02-22	冯勇涛	修改 spring 整合	V3.5
2021-03-15	冯勇涛	添加 windows 服务	V3.6

目录

1. Nosql 介绍

1.1. Nosql 起源

随着 web2.0 时代的快速发展，非关系型、分布式数据存储得到了快速的发展，它们不保证关系数据的 ACID 特性。NoSQL 概念在 2009 年被提了出来。NoSQL 最常见的解释是“non-relational”(非关系)，“Not Only SQL”(不仅仅是 SQL)也被很多人接受。

NoSQL，指的是非关系型的数据库。NoSQL 有时也称作 Not Only SQL 的缩写，是对不同于传统的关系型数据库的数据库管理系统的统称。

NoSQL 用于超大规模数据的存储。（例如谷歌或 Facebook 每天为他们的用户收集万亿比特的数据）。这些类型的数据存储不需要固定的模式，无需多余操作就可以横向扩展。

1.2. 常见 nosql 类型

键值数据库：可以理解为一个分布式的 Hashmap，支持 SET/GET 操作，值是 string，list，set，zset，hash 等。

列式数据库：可以理解为一个每行列数可变的数据表。

文档数据库：也是键值形式存储，键值数据库的一种衍生品。值是文档，文档格式包括 XML、YAML、JSON 和 BSON 等，也可以使用二进制格式。

图形数据库：以图为数据模型来存储数据，图来表示对象的集合以及关系。适用于相互关联的数据，可以高效地处理实体间的关系，尤其适合于社交网络、依赖分析、推荐系统、路径寻找、科学论文引用等场景。

2. 认识 redis

2.1. Redis 介绍

Redis 是一个 key-value 存储系统。

Redis 是一种面向“键/值”对类型数据的分布式 NoSQL 数据库系统。

它支持存储的类型包括 string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和 hash（哈希类型）。这些数据类型都支持 push/pop、add/remove 及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis 支持各种不同方式的排序。

为了保证效率，数据都是缓存在内存中。同时,redis 会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了 master-slave(主从)同步。

2.2. Redis 特点

高性能，持久存储，适应高并发的应用场景。

相比许多键值数据存储，Redis 拥有一套较为丰富的数据类型。

Redis 数据库完全在内存中，使用磁盘仅用于持久性。

Redis 可以将数据复制到任意数量的从服务器。

2.3. Redis 优势

异常快速：Redis 的速度非常快，支持丰富的数据类型。读的速度是 110000 次/s,写的速度是 81000 次/s 。

类型：Redis 支持多数开发人员已经知道的类型。像列表，集合，有序集合，散列数据类型。这使得它非常容易解决各种各样的问题,因为我们知道哪些问题是可以处理通过它的数据类型更好。

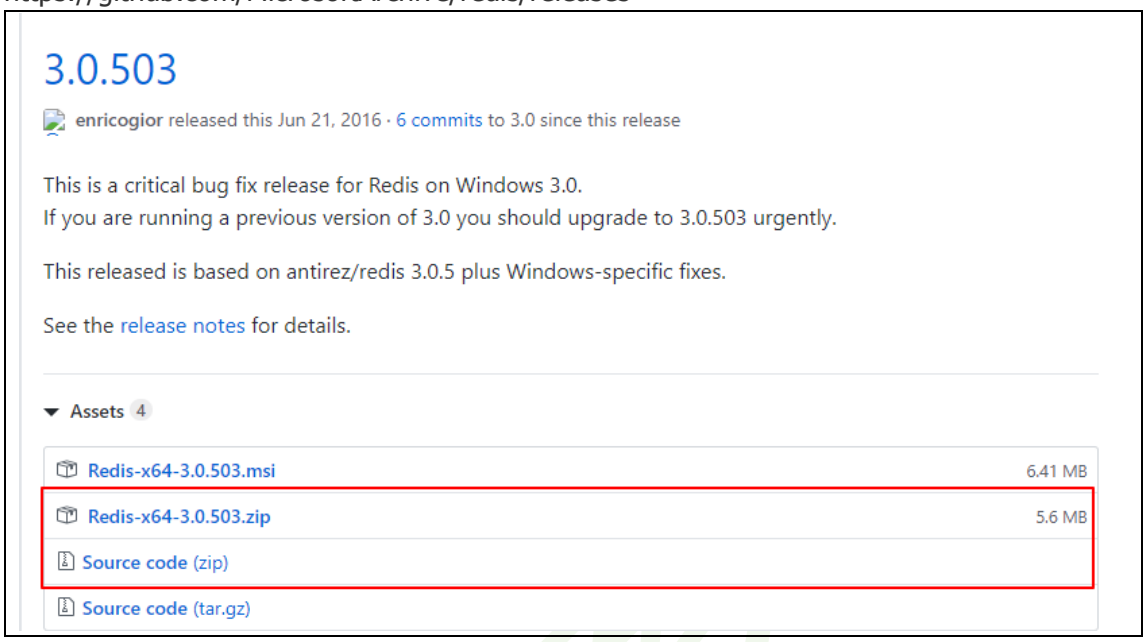
操作都是原子性：所有 Redis 操作是原子的，这保证了如果两个客户端同时访问的 Redis 服务器将获得更新后的值。

多功能实用工具：Redis 是一个多实用的工具，例如缓存，消息队列使用(Redis 原生支持发布/订阅)，任何短暂的数据，应用程序，如 Web 应用程序会话，网页命中计数等。

3. Redis 安装：

redis 的 windows 版本下载：

https://github.com/MicrosoftArchive/redis/releases



3.1. 安装文件目录说明

3.2. Redis 配置

```
41  port 6379                                  → 默认端口6379

113 databases 16                               → 一个redis实例下默认创建的数据库个数，数据库名:0-15
114

137 save 900 1                                 → redis的db持久化策略：save 秒 数据改变个数 → RDB
138 save 300 10
139 save 60 10000

171 # The filename where to dump the DB        → rdb持久化文件名
172 dbfilename dump.rdb
173

182 dir ./                                     → rdb存储路径
183

385 #
386 # requirepass foobared                     → 客户端连接密码设置
387

520
521 appendonly no                              → 是否开启aof持久化
522

523 # The name of the append only file (default: "appendonly.aof")
524 appendfilename "appendonly.aof"            → aof持久化文件名
525

589
590 auto-aof-rewrite-percentage 100            → 达到上个文件的百分百大小时候进行重写
591 auto-aof-rewrite-min-size 64mb             → 当第一个文件达到64M的时候重写
```

3.3. Redis 服务启动

(1) 启动服务

在 cmd 中执行：

redis-server.exe redis.windows.conf

或双击 redis-servier.exe

(2) Redis 安装为 window 服务

在 cmd 中执行:

redis-server.exe --service-install redis.windows.conf

(3) 卸载 windows 服务

```
redis-server --service-uninstall
```

3.4. Redis 客户端工具

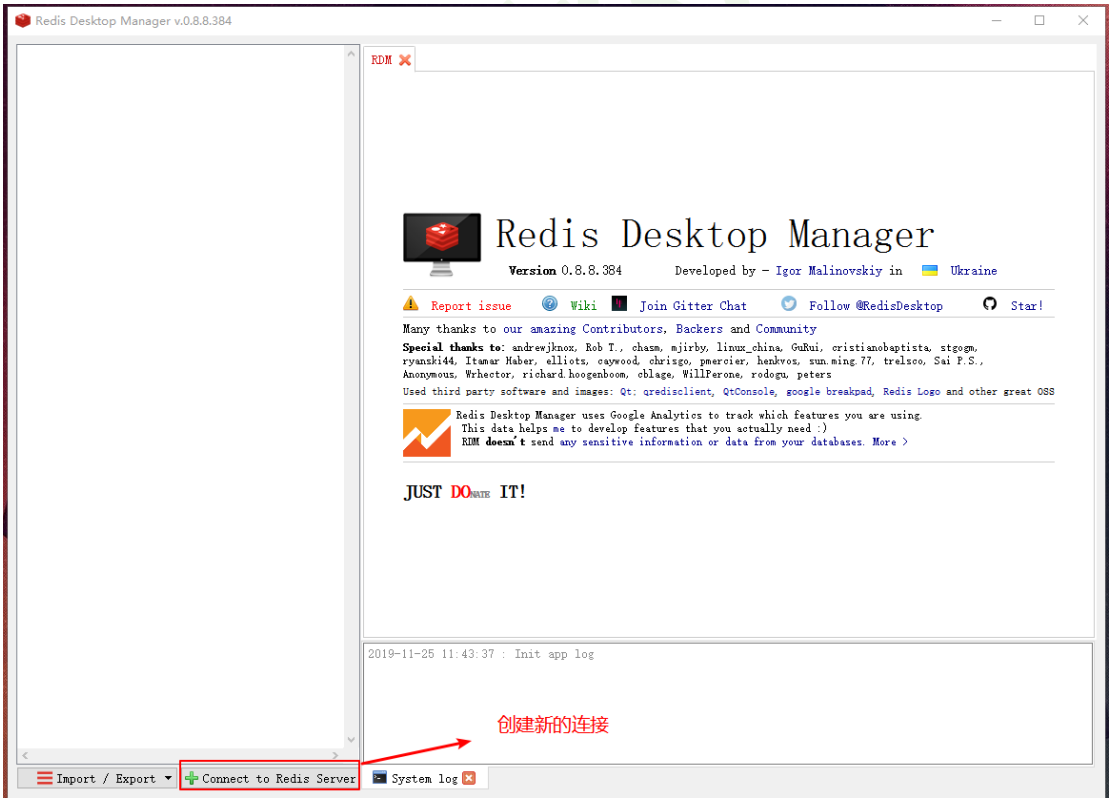
3.4.1. Redis-client 客户端

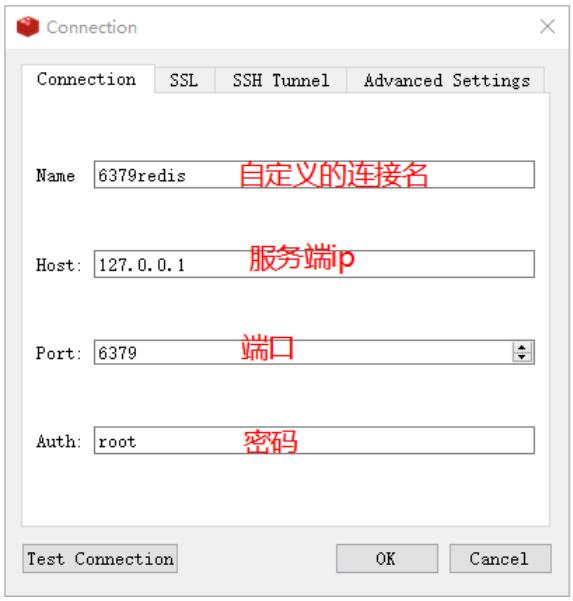
在 cmd 中执行：

redis-cli.exe -h ip 地址 -p 端口 -a 密码

3.4.2. RedisDesktopManager 可视化工具

安装并使用：




Connection
Connection
SSL
SSH Tunnel
Advanced Settings
Name
6379redis
自定义的连接名
Host:
127.0.0.1
服务端ip
Port:
6379
端口
Auth:
root
密码
Test Connection
OK
Cancel


Redis Desktop Manager v.0.8.8.384
6379redis
db0 (4/4)
db1 (0)
db2 (0)
db3 (0)
db4 (0)
db5 (0)
db6 (0)
db7 (0)
db8 (0)
db9 (0)
db10 (0)
db11 (0)
db12 (0)
db13 (0)
db14 (0)
db15 (0)
6379redis::db0::as
STRING:
as
TTL: -1
Rename
Delete
Reload Value
Set TTL
Value: size in bytes: 2
View as:
Plain Text
sd

4. Redis 数据类型

4.1. string 字符串

String 类型介绍：存储字符串类型的值,一般也可以做计数器使用，单个 String 最多能存储 512M。

String 语法介绍：

SET KEY VALUE [px 毫秒] [ex 秒] [nx] - 新增或修改 key 的值

GET [KEY] - 获取 key 的值

GETSET KEY VALUE -返回旧值，设置新值

INCR [KEY] - 将 key 中存储的 value 值做+1 操作

INCRBY [KEY] [INCREMENT] - 将 key 中存储的 value 加上指定的增量值(increment)

DECR [KEY] - 将 key 中存储的 value 值做-1 操作

DECRBY [KEY] [INCREMENT] - 将 key 中存储的 value 减去指定的增量值(increment)

4.2. List 列表

List 类型介绍：一个字符串列表,可从头/尾添加元素.一般用来存储经常访问的数据模型的 ID 列表/消息队列/红包奖池等等，每个列表最多可以存储 2^{32} - 1 个元素(40 多亿)。

List 语法介绍：

LPUSH [KEY] [VALUE] [VALUE1] - 将一个或多个值插入到列表头部

LPOP [KEY] - 移除并获取列表的第一个元素

LLEN KEY - 获取列表长度

LINDEX [KEY] [INDEX] - 通过索引获取列表中的元素

LSET [KEY] [INDEX] [VALUE] - 通过索引设置列表元素的值

LRANGE [KEY] [START] [STOP] - 获取列表指定范围内的元素

4.3. Hash 哈希

Hash 类型介绍:键值对映射表,一般用来存储对象，每个 hash 最多可以存储 2^{32} -1 键值对（40 多亿）。

Hash 语法介绍：

HSET [KEY] [FIELD] [VALUE] – 将哈希表 key 中的字段 field 的值设置为 value

HGET [KEY] [FIELD] – 获取 key 中 field 的值

HMSET [KEY] [FIELD] [VALUE] [FIELD1] [VALUE1] – 同时将多个 field-value 设置到哈希表 key 中

HMGET [KEY] [FIELD] [FIELD1] – 获取所有给定字段的值

HGETALL [KEY] – 获取哈希表中所有的字段和值

HKEYS [KEY] – 获取哈希表中所有的字段

HVALS KEY -获取哈希表中所有的值

HEXISTS [KEY] [FIELD] – 查看哈希表中指定 key 中指定的字段是否存在

HDEL [KEY] [FIELD] – 删除哈希表中指定 key 中指定的字段

4.4. Set 集合

Set 类型介绍：一个无序集合。集合成员是唯一的。可进行交集并集差集运算,一般用作关系处理,如:好友关系等。每个集合最多可以存储 232 - 1 个元素(40 多亿)。

Set 语法介绍：

SADD [KEY] [MEMBER] [MEMBER1] – 向集合添加一个或多个成员

SMEMBERS [KEY] - 获取集合中所有的成员

SPOP [KEY] – 删除并返回集合中的一个随机元素

SDIFF [KEY][KEY1] – 获取给定集合的差集

SDIFFSTORE [DESTINATION] [KEY] [KEY1] – 获取指定的集合的差集并存储至 destination 指定的 key 中

SINTER [KEY][KEY1] - 获取给定集合中的交集

SINTERSTORE [DESTINATION] [KEY] [KEY1] - 获取指定集合的交集并存储至 destination 指定的 key 中

SUNION [KEY][KEY1] – 获取给定集合中的并集

SUNIONSTORE [DESTINATION] [KEY] [KEY1] – 获取指定集合中的并集并存储至 destination 指定的 key 中

4.5. zSet 排序集合

SortedSet 类型介绍：有序集合和集合一样也是 string 类型元素的集合,且不允许重复的成员。不同的是每个元素都会关联一个 double 类型的分数。可以通过分数进行排序。一般在排行榜、热度排序等业务场景中使用。每个集合最多可以存储 232 - 1 个元素(40 多亿)

SortedSet 语法介绍：

ZADD [KEY] [SCORE] [MEMBER] [SCORE1] [MEMBER1] – 向集合添加一个或多个成员及分数,或者更新成员分数

ZINCRBY [KEY] [INCREMENT] [MEMBER] –有序集合中对指定成员增加 increment 值

ZRANGE [KEY] [START] [STOP] – 返回有序集合中指定索引区间的成员

ZRANGEBYSCORE [KEY] [MIN] [MAX] – 返回指定分数区间的成员升序排列

ZREVRANGE [KEY] [START] [STOP] – 返回指定索引区间的成员,降序排列

ZREVRANGEBYSCORE [KEY] [MAX] [MIN] – 返回指定分数区间的成员,降序排列

ZREVRANK [KEY] [MEMBER] – 返回指定成员的排名,从 0 开始

ZSCORE [KEY] [MEMBER] – 返回指定成员的分数

4.6. key 的操作命令

KEYS * - 得到当前实例下所有的 key

DEL KEY [KEY...] -删除一个或多个 key

EXISTS KEY - 判断指定 key 是否存在

EXPIRE KEY seconds - 设置 key 超时时间，过时删除。

TYPE KEY - 获取 key 的类型

SELECT dbIndex - 切换库（0-15）

5. Redis 持久化机制

5.1. RDB

按照持久化测量定期持久化数据到磁盘文件,存的是数据（key-value）

rdb 的持久化策略：

```
137 save 900 1
138 save 300 10
139 save 60 10000
```

→ redis的db持久化策略：save 秒 数据改变个数 → RDB

5.2. AOF

每秒持久化 redis 命令到磁盘文件,定期做文件压缩。

Aof 没有持久化策略，就是一秒持久化一次命令。

只有压缩策略：

```
589
590 auto-aof-rewrite-percentage 100
591 auto-aof-rewrite-min-size 64mb
```

→ 达到上个文件的百分百大小时候进行重写

→ 当第一个文件达到64M的时候重写

6. Redis-java

6.1. Jedis 单机使用

Jedis 是 Java 连接 Redis 的驱动包。

具备操作 Redis 的所有 API，而且使用简单。

6.1.1. 环境准备

jedis-2.7.2.jar

commons-pool2-2.3.jar

6.1.2. 运行代码

```
//单机 jedis，随用随创建连接,用完连接关闭，不建议使用
Jedis j  =new Jedis("127.0.0.1",6379);
j.auth("root");
j.set("bb","12");
j.close();
```

6.1.3. 使用连接池

```
//单机 jedis,使用连接池
JedisPoolConfig config = new JedisPoolConfig();
config.setMaxTotal(20);
JedisPool pool = new
JedisPool(config,"127.0.0.1",6379,1000*2,"root");//单利


Jedis jedis = pool.getResource();
String bb = jedis.get("bb");
```

```
System.out.println(bb);

jedis.close();
```

6.2. SharedJedis 客户端分片集群

Redis 分片集群，不能满足高可用性，同时集群不可扩展，因此仅供了解，项目中不使用。

6.2.1. 多 redis 实例启动

拷贝 redis.windows.conf，修改新的端口为 6380，启动第二个 redis 实例。

6.2.2. 运行代码

```
JedisPoolConfig config = new JedisPoolConfig();

    config.setMinIdle(5);

    //一个 JedisShardInfo 就是一台 redis 实例的连接信息对象

    List<JedisShardInfo> l = new ArrayList<>();

    l.add(new JedisShardInfo(new URI("redis://x:root@127.0.0.1:6379/0")));

    l.add(new JedisShardInfo(new URI("redis://x:root@127.0.0.1:6380/0")));

    ShardedJedisPool pool = new ShardedJedisPool(config,l);//全局唯一

    ShardedJedis resource = pool.getResource();//底层对数据做分片存储。

    for(int i=0;i<1000;i++) {

        resource.set("a"+i, i+"");

    }

    resource.close();
```

7. Redis 整合 spring

Zai

7.1. 添加 redis 与连接池依赖包

jedis-2.7.2.jar

commons-pool2-2.3.jar

7.2. 创建 redis.properties

```
redis.minIdle=10
redis.MaxTotal=100
redis.host=127.0.0.1
redis.port=6379
redis.timeout=3000
redis.auth=root
```

7.3. 创建 redis.xml

```
<context:property-placeholder
location="classpath:redis.properties" ignore-
unresolvable="true"></context:property-placeholder>
<!--Jedis 连接池配置信息-->
<bean id="poolConfig"
class="redis.clients.jedis.JedisPoolConfig">
    <property name="minIdle"
value="${redis.minIdle}"></property>
    <property name="maxTotal"
value="${redis.MaxTotal}"></property>
</bean>
<!--Jedis 连接池-->
<bean id="jedisPool" class="redis.clients.jedis.JedisPool">
```

```
    <constructor-arg index="0" ref="poolConfig"></constructor-
arg>
    <constructor-arg index="1"
value="${redis.host}"></constructor-arg>
    <constructor-arg index="2"
value="${redis.port}"></constructor-arg>
    <constructor-arg index="3"
value="${redis.timeout}"></constructor-arg>
    <constructor-arg index="4"
value="${redis.auth}"></constructor-arg>
</bean>
```

7.4. 创建 RedisService 工具类

```
package com.javasm.commons.service;


import java.util.Collection;
import java.util.List;
import java.util.Map;
import java.util.Set;
import javax.annotation.Resource;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Service;
import redis.clients.jedis.*;
import redis.clients.jedis.BinaryClient.LIST_POSITION;


@Service
public class RedisService {
```

```
private static final Logger log = LoggerFactory.getLogger(RedisService.class);


@Resource
private JedisPool jedisPool;


/**
 *
 * @Title: set @Description:  设置单个值  @param @param key @param @param
 *              value @param @return @return String @throws
 */
public String set(String key, String value) {
    Jedis Jedis = jedisPool.getResource();

    String result = null;
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.set(key, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}


/**
 *
```

```
  * @Title: get @Description: 获取单个值 @param @param key @param @return @return
  *              String @throws
  */
public String get(String key) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.get(key);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

/**
  *
  * @Title: exists @Description: 确认一个 key 是否存在 @param @param
  *              key @param @return @return Boolean @throws
  */
public Boolean exists(String key) {
    Boolean result = false;
    Jedis Jedis = jedisPool.getResource();
```

```
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.exists(key);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    /**
     *
     * @Title: type @Description: 返回值的类型 @param @param key @param @return
@return
     *              String @throws
     */
    public String type(String key) {
        String result = null;
        Jedis Jedis = jedisPool.getResource();

        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.type(key);
```

```
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    /**
     *
     * @Title: expire @Description: 设定一个 key 的活动时间（s） @param @param key
@param @param
     *              seconds @param @return @return Long @throws
     */
    public Long expire(String key, int seconds) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.expire(key, seconds);

        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
```

```
    }

    /**
     *
     * @Title: expireAt @Description: 在某个时间点失效 @param @param key @param
@param
     *              unixTime @param @return @return Long @throws
     */
    public Long expireAt(String key, long unixTime) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.expireAt(key, unixTime);

        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    /**
     *
     * @Title: ttl @Description: 获得一个 key 的活动时间 @param @param
     *              key @param @return @return Long @throws
```

```
 */
public Long ttl(String key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.ttl(key);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public boolean setbit(String key, long offset, boolean value) {

    boolean result = false;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.setbit(key, offset, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
```

```
    } finally {
        Jedis.close();
    }
    return result;
}

public boolean getbit(String key, long offset) {

    boolean result = false;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.getbit(key, offset);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public long setrange(String key, long offset, String value) {

    long result = 0;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
```

```
    }
    try {
        result = Jedis.setrange(key, offset, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String getrange(String key, long startOffset, long endOffset) {

    String result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.getrange(key, startOffset, endOffset);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String getSet(String key, String value) {
```

```
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.getSet(key, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long setnx(String key, String value) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.setnx(key, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
```

```
    return result;
}

public String setex(String key, int seconds, String value) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.setex(key, seconds, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long decrBy(String key, long integer) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.decrBy(key, integer);
```

```
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long decr(String key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.decr(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long incrBy(String key, long integer) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
```

```
    }
    try {
        result = Jedis.incrBy(key, integer);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long incr(String key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.incr(key);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}
```

```
public Long append(String key, String value) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.append(key, value);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String substr(String key, int start, int end) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.substr(key, start, end);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
```

```
    } finally {
        Jedis.close();
    }
    return result;
}

public Long hset(String key, String field, String value) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hset(key, field, value);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String hget(String key, String field) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
```

```
    }
    try {
        result = Jedis.hget(key, field);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long hsetnx(String key, String field, String value) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hsetnx(key, field, value);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}
```

```
public String hmset(String key, Map<String, String> hash) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hmset(key, hash);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public List<String> hmget(String key, String... fields) {
    List<String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hmget(key, fields);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
```

```
    } finally {
        Jedis.close();
    }
    return result;
}

public Long hincrBy(String key, String field, long value) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hincrBy(key, field, value);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Boolean hexists(String key, String field) {
    Boolean result = false;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
```

```
    try {
        result = Jedis.hexists(key, field);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long del(String key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.del(key);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long hdel(String key, String field) {
```

```
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hdel(key, field);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long hlen(String key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hlen(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
```

```
    return result;
}

public Set<String> hkeys(String key) {
    Set<String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hkeys(key);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public List<String> hvals(String key) {
    List<String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hvals(key);
```

```
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Map<String, String> hgetAll(String key) {
    Map<String, String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hgetAll(key);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

// ================list ====== l 表示  list 或  left, r 表示 right====================
/**
 *
 * @Title: rpush @Description:  在指定 Key 所关联的 List
```

```
    *            Value 的尾部插入参数中给出的所有 Values。如果该 Key 不存在，该命令将在插入之前创建一个与该 Key 关联的空链表，之后再将数据从链表的尾部插入。如果该键的 Value 不是链表类型，该命令将返回相关的错误信息。 @param @param
    *            key @param @param string @param @return @return Long @throws
    */
   public Long rpush(String key, String string) {
      Long result = null;
      Jedis Jedis = jedisPool.getResource();
      if (Jedis == null) {
            return result;
      }
      try {
            result = Jedis.rpush(key, string);

      } catch (Exception e) {
            log.error(e.getMessage(), e);
      } finally {
            Jedis.close();
      }
      return result;
   }

   /**
    *
    * @Title: lpush @Description:  在指定 Key 所关联的 List
    *            Value 的头部插入参数中给出的所有 Values。如果该 Key 不存在，该命令将在插入之前创建一个与该 Key 关联的空链表，之后再将数据从链表的头部插入。如果该键的 Value 不是链表类型，该命令将返回相关的错误信息。 @param @param
    *            key @param @param string @param @return @return Long @throws
```

```
 */
public Long lpush(String key, String string) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.lpush(key, string);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long lpushx(String key, String string) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.lpushx(key, string);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
```

```
    } finally {
        Jedis.close();
    }
    return result;
}

public Long llen(String key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.llen(key);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public List<String> lrange(String key, long start, long end) {
    List<String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
```

```
    try {
        result = Jedis.lrange(key, start, end);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String ltrim(String key, long start, long end) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.ltrim(key, start, end);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}
```

```
public String lindex(String key, long index) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.lindex(key, index);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String lset(String key, long index, String value) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.lset(key, index, value);

    } catch (Exception e) {
```

```
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long lrem(String key, long count, String value) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.lrem(key, count, value);

        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public String lpop(String key) {
        String result = null;
        Jedis Jedis = jedisPool.getResource();

        if (Jedis == null) {
```

```
            return result;
        }
        try {
            result = Jedis.lpop(key);

        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public String rpop(String key) {
        String result = null;
        Jedis Jedis = jedisPool.getResource();

        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.rpop(key);

        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
```

```
}

// return 1 add a not exist value ,
// return 0 add a exist value
public Long sadd(String key, String member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.sadd(key, member);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Set<String> smembers(String key) {
    Set<String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.smembers(key);
```

```
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long srem(String key, String member) {

    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.srem(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String spop(String key) {

    String result = null;
```

```
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.spop(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long scard(String key) {

    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.scard(key);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
```

```
    return result;
}

public Boolean sismember(String key, String member) {

    Boolean result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.sismember(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String srandmember(String key) {

    String result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.srandmember(key);
```

```
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long zadd(String key, double score, String member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zadd(key, score, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Set<String> zrange(String key, int start, int end) {
    Set<String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
```

```
    }
    try {
        result = Jedis.zrange(key, start, end);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long zrem(String key, String member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrem(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Double zincrby(String key, double score, String member) {
    Double result = null;
```

```
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zincrby(key, score, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long zrank(String key, String member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrank(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}
```

```
public Long zrevrank(String key, String member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrevrank(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Set<String> zrevrange(String key, int start, int end) {
    Set<String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {

        result = Jedis.zrevrange(key, start, end);

    } catch (Exception e) {
        log.error(e.getMessage(), e);
```

```
    } finally {
        Jedis.close();
    }
    return result;
}

public Set<Tuple> zrangeWithScores(String key, int start, int end) {
    Set<Tuple> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrangeWithScores(key, start, end);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Set<Tuple> zrevrangeWithScores(String key, int start, int end) {
    Set<Tuple> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
```

```
            result = Jedis.zrevrangeWithScores(key, start, end);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long zcard(String key) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zcard(key);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Double zscore(String key, String member) {
        Double result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
```

```
            return result;
        }
        try {
            result = Jedis.zscore(key, member);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public List<String> sort(String key) {
        List<String> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.sort(key);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public List<String> sort(String key, SortingParams sortingParameters) {
```

```
    List<String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.sort(key, sortingParameters);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long zcount(String key, double min, double max) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zcount(key, min, max);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
```

```
}

public Set<String> zrangeByScore(String key, double min, double max) {
    Set<String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrangeByScore(key, min, max);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Set<String> zrevrangeByScore(String key, double max, double min) {
    Set<String> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrevrangeByScore(key, max, min);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
```

```
            Jedis.close();
        }
        return result;
    }

    public Set<String> zrangeByScore(String key, double min, double max, int offset, int count) {
        Set<String> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zrangeByScore(key, min, max, offset, count);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<String> zrevrangeByScore(String key, double max, double min, int offset, int count)
{
        Set<String> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
```

```
            result = Jedis.zrevrangeByScore(key, max, min, offset, count);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<Tuple> zrangeByScoreWithScores(String key, double min, double max) {
        Set<Tuple> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zrangeByScoreWithScores(key, min, max);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<Tuple> zrevrangeByScoreWithScores(String key, double max, double min) {
        Set<Tuple> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
```

```
            return result;
        }
        try {
            result = Jedis.zrevrangeByScoreWithScores(key, max, min);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<Tuple> zrangeByScoreWithScores(String key, double min, double max, int offset,
int count) {
        Set<Tuple> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zrangeByScoreWithScores(key, min, max, offset, count);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }
```

```
    public Set<Tuple> zrevrangeByScoreWithScores(String key, double max, double min, int offset, int count) {
        Set<Tuple> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zrevrangeByScoreWithScores(key, max, min, offset, count);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long zremrangeByRank(String key, int start, int end) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zremrangeByRank(key, start, end);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
```

```
    }
    return result;
}

public Long zremrangeByScore(String key, double start, double end) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zremrangeByScore(key, start, end);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long linsert(String key, LIST_POSITION where, String pivot, String value) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.linsert(key, where, pivot, value);
    } catch (Exception e) {
```

```
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public String set(byte[] key, byte[] value) {
        String result = null;
        Jedis Jedis = jedisPool.getResource();

        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.set(key, value);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public byte[] get(byte[] key) {
        byte[] result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
```

```
    }
    try {
        result = Jedis.get(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Boolean exists(byte[] key) {
    Boolean result = false;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.exists(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String type(byte[] key) {
    String result = null;
```

```
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.type(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long expire(byte[] key, int seconds) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.expire(key, seconds);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
```

```
}

public Long expireAt(byte[] key, long unixTime) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.expireAt(key, unixTime);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long ttl(byte[] key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.ttl(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
```

```
            Jedis.close();
        }
        return result;
    }

    public byte[] getSet(byte[] key, byte[] value) {
        byte[] result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.getSet(key, value);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long setnx(byte[] key, byte[] value) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.setnx(key, value);
```

```
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String setex(byte[] key, int seconds, byte[] value) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.setex(key, seconds, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long decrBy(byte[] key, long integer) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
```

```
            return result;
        }
        try {
            result = Jedis.decrBy(key, integer);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long decr(byte[] key) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.decr(key);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long incrBy(byte[] key, long integer) {
```

```
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.incrBy(key, integer);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long incr(byte[] key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.incr(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
```

```
}

public Long append(byte[] key, byte[] value) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.append(key, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public byte[] substr(byte[] key, int start, int end) {
    byte[] result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {

        result = Jedis.substr(key, start, end);

    } catch (Exception e) {
```

```
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long hset(byte[] key, byte[] field, byte[] value) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.hset(key, field, value);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public byte[] hget(byte[] key, byte[] field) {
        byte[] result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
```

```
    try {
        result = Jedis.hget(key, field);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long hsetnx(byte[] key, byte[] field, byte[] value) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hsetnx(key, field, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String hmset(byte[] key, Map<byte[], byte[]> hash) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();
```

```
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hmset(key, hash);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public List<byte[]> hmget(byte[] key, byte[]... fields) {
    List<byte[]> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hmget(key, fields);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}
```

```
public Long hincrBy(byte[] key, byte[] field, long value) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hincrBy(key, field, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Boolean hexists(byte[] key, byte[] field) {
    Boolean result = false;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hexists(key, field);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
```

```
    }
    return result;
}

public Long hdel(byte[] key, byte[] field) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hdel(key, field);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long hlen(byte[] key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hlen(key);
    } catch (Exception e) {
```

```
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<byte[]> hkeys(byte[] key) {
        Set<byte[]> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.hkeys(key);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Collection<byte[]> hvals(byte[] key) {
        Collection<byte[]> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
```

```
    try {
        result = Jedis.hvals(key);
    } catch (Exception e) {

        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Map<byte[], byte[]> hgetAll(byte[] key) {
    Map<byte[], byte[]> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.hgetAll(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long rpush(byte[] key, byte[] string) {
    Long result = null;
```

```
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.rpush(key, string);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long lpush(byte[] key, byte[] string) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.lpush(key, string);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}
```

```
public Long llen(byte[] key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.llen(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public List<byte[]> lrange(byte[] key, int start, int end) {
    List<byte[]> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.lrange(key, start, end);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
```

```
    }
    return result;
}

public String ltrim(byte[] key, int start, int end) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.ltrim(key, start, end);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public byte[] lindex(byte[] key, int index) {
    byte[] result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.lindex(key, index);
```

```
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public String lset(byte[] key, int index, byte[] value) {
    String result = null;
    Jedis Jedis = jedisPool.getResource();

    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.lset(key, index, value);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long lrem(byte[] key, int count, byte[] value) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
```

```
            return result;
        }
        try {
            result = Jedis.lrem(key, count, value);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public byte[] lpop(byte[] key) {
        byte[] result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.lpop(key);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public byte[] rpop(byte[] key) {
```

```
    byte[] result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.rpop(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long sadd(byte[] key, byte[] member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.sadd(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
```

```
}

public Set<byte[]> smembers(byte[] key) {
    Set<byte[]> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.smembers(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long srem(byte[] key, byte[] member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.srem(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
```

```
            Jedis.close();
        }
        return result;
    }

    public byte[] spop(byte[] key) {
        byte[] result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.spop(key);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long scard(byte[] key) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.scard(key);
```

```
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Boolean sismember(byte[] key, byte[] member) {
    Boolean result = false;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.sismember(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public byte[] srandmember(byte[] key) {
    byte[] result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
```

```
    }
    try {
        result = Jedis.srandmember(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long zadd(byte[] key, double score, byte[] member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zadd(key, score, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Set<byte[]> zrange(byte[] key, int start, int end) {
    Set<byte[]> result = null;
```

```
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrange(key, start, end);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long zrem(byte[] key, byte[] member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrem(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}
```

```
public Double zincrby(byte[] key, double score, byte[] member) {
    Double result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zincrby(key, score, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long zrank(byte[] key, byte[] member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrank(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
```

```
    }
    return result;
}

public Long zrevrank(byte[] key, byte[] member) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrevrank(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Set<byte[]> zrevrange(byte[] key, int start, int end) {
    Set<byte[]> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrevrange(key, start, end);
    } catch (Exception e) {
```

```
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<Tuple> zrangeWithScores(byte[] key, int start, int end) {
        Set<Tuple> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zrangeWithScores(key, start, end);
        } catch (Exception e) {

            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<Tuple> zrevrangeWithScores(byte[] key, int start, int end) {
        Set<Tuple> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
```

```
    }
    try {
        result = Jedis.zrevrangeWithScores(key, start, end);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long zcard(byte[] key) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zcard(key);
    } catch (Exception e) {

        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Double zscore(byte[] key, byte[] member) {
```

```
    Double result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zscore(key, member);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public List<byte[]> sort(byte[] key) {
    List<byte[]> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.sort(key);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
```

```
}

public List<byte[]> sort(byte[] key, SortingParams sortingParameters) {
    List<byte[]> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.sort(key, sortingParameters);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Long zcount(byte[] key, double min, double max) {
    Long result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zcount(key, min, max);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
```

```
            Jedis.close();
        }
        return result;
    }

    public Set<byte[]> zrangeByScore(byte[] key, double min, double max) {
        Set<byte[]> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zrangeByScore(key, min, max);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<byte[]> zrangeByScore(byte[] key, double min, double max, int offset, int count) {
        Set<byte[]> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zrangeByScore(key, min, max, offset, count);
```

```
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<Tuple> zrangeByScoreWithScores(byte[] key, double min, double max) {
        Set<Tuple> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zrangeByScoreWithScores(key, min, max);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<Tuple> zrangeByScoreWithScores(byte[] key, double min, double max, int offset,
int count) {
        Set<Tuple> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
```

```
            return result;
        }
        try {
            result = Jedis.zrangeByScoreWithScores(key, min, max, offset, count);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<byte[]> zrevrangeByScore(byte[] key, double max, double min) {
        Set<byte[]> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zrevrangeByScore(key, max, min);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Set<byte[]> zrevrangeByScore(byte[] key, double max, double min, int offset, int count)
```

```
{
    Set<byte[]> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrevrangeByScore(key, max, min, offset, count);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
    return result;
}

public Set<Tuple> zrevrangeByScoreWithScores(byte[] key, double max, double min) {
    Set<Tuple> result = null;
    Jedis Jedis = jedisPool.getResource();
    if (Jedis == null) {
        return result;
    }
    try {
        result = Jedis.zrevrangeByScoreWithScores(key, max, min);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
    } finally {
        Jedis.close();
    }
```

```
        return result;
    }

    public Set<Tuple> zrevrangeByScoreWithScores(byte[] key, double max, double min, int offset, int count) {
        Set<Tuple> result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zrevrangeByScoreWithScores(key, max, min, offset, count);
        } catch (Exception e) {

            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long zremrangeByRank(byte[] key, int start, int end) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
```

```
            result = Jedis.zremrangeByRank(key, start, end);

        } catch (Exception e) {

            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long zremrangeByScore(byte[] key, double start, double end) {
        Long result = null;
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
            return result;
        }
        try {
            result = Jedis.zremrangeByScore(key, start, end);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        } finally {
            Jedis.close();
        }
        return result;
    }

    public Long linsert(byte[] key, LIST_POSITION where, byte[] pivot, byte[] value) {
        Long result = null;
```

```
        Jedis Jedis = jedisPool.getResource();
        if (Jedis == null) {
                return result;
        }
        try {
                result = Jedis.linsert(key, where, pivot, value);
        } catch (Exception e) {
                log.error(e.getMessage(), e);
        } finally {
                Jedis.close();
        }
        return result;
    }
}
```

7.5. 使用 RedisService

以发送验证码以及登录为例，通过 redis 临时保存服务端生成的手机验证码，通过 redis 缓存登录用户信息。以下为 service 层方法：

```
@Resource
        private RedisService rs;

        @Override
        public boolean sendValiCode(String uphone) {
                String valicode = StringUtils.getSixNumCode();
                System.out.println("验证按："+valicode);
                //TODO 给 uphone 手机号异步发送验证码
```

```
        //暂存 valicode 到 redis
        rs.setex(RedisKey.phonecode+uphone,5*60,valicode);
        return true;
    }

    @Override
    public RB phoneLogin(String uphone, String inputCode) {
        String s = rs.get(RedisKey.phonecode + uphone);
        if(s==null)
            return RB.status(SE.VALICODE_ERROR);
        if(!s.equals(inputCode))
            return RB.status(SE.VALICODE_ERROR);

        //redis 中根据手机号查询登录用户数据，hash 类型的用户数据
        String user = rs.hget(RedisKey.sysusers, uphone);
        if(user!=null){
            Sysuser sysuser = JSON.parseObject(user, Sysuser.class);
            sysuser.setUpwd("");
            return RB.result(SE.SUC,sysuser);
        }

        //缓存中没有，去查询数据库
        Sysuser t = new Sysuser();
        t.setUphone(uphone);
        List<Sysuser> users = sm.selectUsers(t);
        if(users!=null && users.size()==1){
            Sysuser u    =users.get(0);
            rs.hset(RedisKey.sysusers,uphone,JSON.toJSONString(u));
            u.setUpwd("");
```

```
            return RB.result(SE.SUC,u);
        }
        //数据库中也没有,自动注册
        Sysuser u = new Sysuser();
        u.setUphone(uphone);
        u.setUname("尚马老 K");
        u.setUpwd(uphone.substring(5));
        u.setNewuser("yes");//为了前端能够弹出 dialog，提示完善信息。
        addUser(u);//保存用户对象到 mybatis
       //redis 中缓存登录用户数据
        rs.hset(RedisKey.sysusers,uphone,JSON.toJSONString(u));

        return RB.result(SE.SUC,u);
    }
```