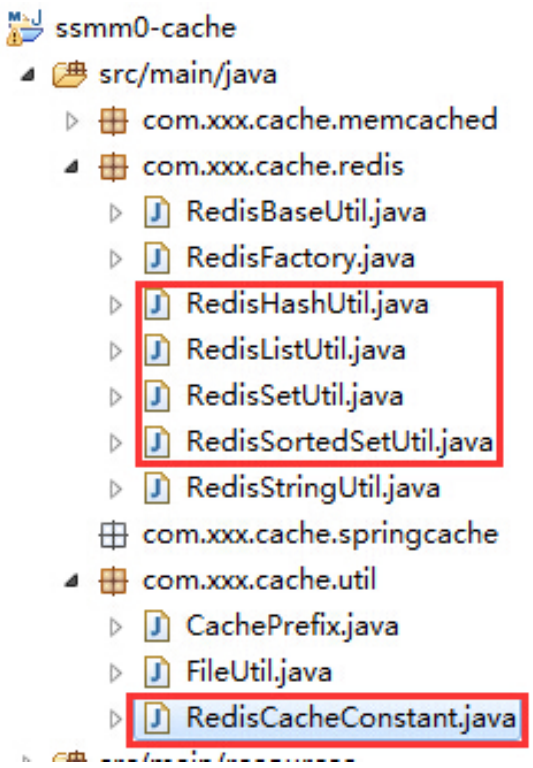
第十章 企业项目开发--分布式缓存Redis(2)

注意：本章代码是在上一章的基础上进行添加修改，上一章链接《第九章 企业项目开发--分布式缓存Redis(1)》

上一章说了ShardedJedisPool的创建过程，以及redis五种数据结构的第一种String类型的常用缓存操作方法。下面说余下的四种：

- list（有序列表）
- set（无序集合）
- sorted set（有序集合）
- hash

1、ssmm0-cache



1.1、RedisListUtil（有序列表工具类）

```
package com.xxx.cache.redis;

import java.util.List;

import redis.clients.jedis.ShardedJedis;

/**
 * list缓存操作类
 * 1、顺序为插入list的顺序
 * 2、允许存放重复元素
 * 3、可用作模拟队列（queue）、堆栈（stack），支持双向操作（L--首部或者R--尾部）
 * 4、index从0开始 -1表示结尾  -2表示倒数第二个
 * 5、API中的 start end参数 都是包前也包后的
 */
public class RedisListUtil extends RedisBaseUtil{

    /**************************添加缓存*****************************/
    /**
     * 从左边（首部）加入列表
     * 注意：
     * 1、可以一次性入队n个元素（这里使用了不定参数,当然可以换做数组）
     * 2、不定参数必须放在所有参数的最后边
     * 3、左边入队，相当于在队头插入元素，则之后的元素都要后移一位；而右边入队的话元素直接插在队尾，之前的元素的索引不变
     * 4、没有从list中获取指定value的运算
     */
    public static void lpush(String list, String... values){
        boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
        ShardedJedis jedis = null;
        try {
```

```
30              jedis = getJedis();//获取jedis实例
31              if(jedis==null){
32                  broken = true;
33                  return;
34              }
35              /*
36               * lpush(String key, String... strings);
37               * 返回push之后的list中包含的元素个数
38               */
39              jedis.lpush(list, values);
40          } catch (Exception e) {
41              broken = true;
42          }finally{
43              returnJedis(jedis, broken);
44          }
45      }
46
47      /**
48       * 从左边（首部）加入列表
49       * 并指定列表缓存过期时间
50       */
51      public static void lpush(String list, int expire, String... values){
52          boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
53          ShardedJedis jedis = null;
54          try {
55              jedis = getJedis();//获取jedis实例
56              if(jedis==null){
57                  broken = true;
58                  return;
59              }
60              /*
61               * lpush(String key, String... strings);
62               * 返回push之后的list中包含的元素个数
63               */
64              jedis.lpush(list, values);
65              jedis.expire(list, expire);//为该list设置缓存过期时间
66          } catch (Exception e) {
67              broken = true;
68          }finally{
69              returnJedis(jedis, broken);
70          }
71      }
72
73      /**
74       * 从右边（尾部）加入列表
75       */
76      public static void rpush(String list, String... values){
77          boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
78          ShardedJedis jedis = null;
79          try {
80              jedis = getJedis();//获取jedis实例
81              if(jedis==null){
82                  broken = true;
83                  return;
84              }
85              jedis.rpush(list, values);
86          } catch (Exception e) {
87              broken = true;
88          }finally{
89              returnJedis(jedis, broken);
90          }
91      }
92
93      /**
94       * 从右边（尾部）加入列表
```

```
95      * 并设置缓存过期时间
96      */
97     public static void rpush(String list, int expire, String... values){
98         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
99         ShardedJedis jedis = null;
100        try {
101            jedis = getJedis();//获取jedis实例
102            if(jedis==null){
103                broken = true;
104                return;
105            }
106            jedis.rpush(list, values);
107            jedis.expire(list, expire);//设置缓存过期时间
108        } catch (Exception e) {
109            broken = true;
110        }finally{
111            returnJedis(jedis, broken);
112        }
113    }
114
115    /**
116     * 设置list中index位置的元素
117     * index==-1表示最后一个元素
118     */
119    public static void lSetIndex(String list, long index, String value){
120        boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
121        ShardedJedis jedis = null;
122        try {
123            jedis = getJedis();//获取jedis实例
124            if(jedis==null){
125                broken = true;
126                return;
127            }
128            jedis.lset(list, index, value);
129        } catch (Exception e) {
130            broken = true;
131        }finally{
132            returnJedis(jedis, broken);
133        }
134    }
135
136    /**************************获取缓存*****************************/
137    /**
138     * 从左边（首部）出列表
139     */
140    public static String lpop(String list){
141        boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
142        ShardedJedis jedis = null;
143        try {
144            jedis = getJedis();//获取jedis实例
145            if(jedis==null){
146                broken = true;
147                return null;
148            }
149            return jedis.lpop(list);
150        } catch (Exception e) {
151            broken = true;
152        }finally{
153            returnJedis(jedis, broken);
154        }
155        return null;
156    }
157
158    /**
159     * 从右边出列表
```

```
160      */
161     public static String rpop(String list){
162         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
163         ShardedJedis jedis = null;
164         try {
165             jedis = getJedis();//获取jedis实例
166             if(jedis==null){
167                 broken = true;
168                 return null;
169             }
170             return jedis.rpop(list);
171         } catch (Exception e) {
172             broken = true;
173         }finally{
174             returnJedis(jedis, broken);
175         }
176         return null;
177     }
178 
179     /**
180      * 返回list中index位置的元素
181      */
182     public static String lGetIndex(String list, long index){
183         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
184         ShardedJedis jedis = null;
185         try {
186             jedis = getJedis();//获取jedis实例
187             if(jedis==null){
188                 broken = true;
189                 return null;
190             }
191             return jedis.lindex(list, index);
192         } catch (Exception e) {
193             broken = true;
194         }finally{
195             returnJedis(jedis, broken);
196         }
197         return null;
198     }
199 
200     /**
201      * 返回list指定区间[start,end]内的元素
202      */
203     public static List<String> lrange(String list, long start, long end){
204         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
205         ShardedJedis jedis = null;
206         try {
207             jedis = getJedis();//获取jedis实例
208             if(jedis==null){
209                 broken = true;
210                 return null;
211             }
212             return jedis.lrange(list, start, end);//
213         } catch (Exception e) {
214             broken = true;
215         }finally{
216             returnJedis(jedis, broken);
217         }
218         return null;
219     }
220 
221     /**
222      * 返回list内的全部元素
223      */
224     public static List<String> lrange(String list){
```

```
225         return lrange(list, 0, -1);
226     }
227 
228     /*** 删除缓存（删除整个list，直接用RedisStringUtil的delete就好）******/
229     /**
230      * 让list只保留指定区间[start,end]内的元素，不在指定区间内的元素都将被删除
231      */
232     public static void ltrim(String list, long start, long end){
233         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
234         ShardedJedis jedis = null;
235         try {
236             jedis = getJedis();//获取jedis实例
237             if(jedis==null){
238                 broken = true;
239                 return;
240             }
241             jedis.ltrim(list, start, end);
242         } catch (Exception e) {
243             broken = true;
244         }finally{
245             returnJedis(jedis, broken);
246         }
247     }
248 
249     /**
250      * 删除list中所有与value相等的元素
251      * 注意：
252      * count
253      * ==0 ：删除表中所有与value相等的元素
254      * >0：从表头开始向表尾搜索，移除count个与value相等的元素
255      * <0：从表尾开始向表头搜索，移除count个与value相等的元素
256      */
257     public static void lremove(String list, long count, String value){
258         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
259         ShardedJedis jedis = null;
260         try {
261             jedis = getJedis();//获取jedis实例
262             if(jedis==null){
263                 broken = true;
264                 return;
265             }
266             jedis.lrem(list, count, value);//返回删除了多少个元素
267         } catch (Exception e) {
268             broken = true;
269         }finally{
270             returnJedis(jedis, broken);
271         }
272     }
273 
274     /**
275      * 删除list中所有与value相等的元素
276      */
277     public static void lremove(String list, String value){
278         lremove(list, 0, value);
279     }
280 
281     /*************************其他*****************************/
282     /**
283      * 返回list中共有多少个元素
284      */
285     public static long llength(String list){
286         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
287         ShardedJedis jedis = null;
288         try {
289             jedis = getJedis();//获取jedis实例
```

```
290             if(jedis==null){
291                 broken = true;
292                 return 0;
293             }
294             return jedis.llen(list);
295         } catch (Exception e) {
296             broken = true;
297         }finally{
298             returnJedis(jedis, broken);
299         }
300         return 0;
301     }
302 
303     public static void main(String[] args) {
304         lpush("adminList", "jigang");
305         lpush("adminList", "nana");//头部
306         System.out.println(llength("adminList"));
307         System.out.println(lrange("adminList"));
308         //lpop("adminList");
309         //System.out.println(llength("adminList"));
310         //ltrim("adminList", 0, 1);
311         //System.out.println(lrange("adminList"));
312         //System.out.println(lpop("adminList"));//左边进左边出，栈（后进先出）
313         //System.out.println(rpop("adminList"));//左边进右边出，队列（先进先出）
314         System.out.println(lGetIndex("adminList",1));
315 
316     }
317 }
```

注意：

- 元素在list中的存放顺序为：插入list的顺序（从左边插入在头部，从右边插入在尾部）
- 允许存放重复元素
- 可用作模拟队列（queue）、堆栈（stack），支持双向操作（L--首部或者R--尾部）
- index从0开始 -1表示结尾 -2表示倒数第二个
- API中的 start end参数 都是包前也包后的
- 左边入队，相当于在队头插入元素，则之后的元素都要后移一位；而右边入队的话元素直接插在队尾，之前的元素的索引不变（推荐使用右边入队，即队尾入队）
- 没有从list中获取指定value的运算（这也是set/sorted set所没有的）
- 没有直接的指定缓存过期的API（这也是set/sorted set/hash所没有的），但是可以按例如如下的方式指定缓存过期时间

```
1     /**
2      * 从左边（首部）加入列表
3      * 并指定列表缓存过期时间
4      */
5     public static void lpush(String list, int expire, String... values){
6         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
7         ShardedJedis jedis = null;
8         try {
9             jedis = getJedis();//获取jedis实例
10            if(jedis==null){
11                broken = true;
12                return;
13            }
14            /*
15             * lpush(String key, String... strings);
16             * 返回push之后的list中包含的元素个数
17             */
18            jedis.lpush(list, values);
19            jedis.expire(list, expire);//为该list设置缓存过期时间
20        } catch (Exception e) {
21            broken = true;
```

```
        }finally{
            returnJedis(jedis, broken);
        }
    }
```

- 删除整个list可以直接使用jedis.del(list)（set/sorted set/hash同理）

1.2、RedisSetUtil（无序集合工具类）

```
package com.xxx.cache.redis;

import java.util.Set;

import redis.clients.jedis.ShardedJedis;

/**
 * set缓存操作类
 * 1、无序集合，最后的顺序不一定是插入顺序
 * 2、元素不能重复
 * 3、对于set而言，Jedis有交集、差集、并集运算，可是ShardJedis没有
 * 4、没有从set中获取指定value的运算
 */
public class RedisSetUtil extends RedisBaseUtil {
    /**************************添加缓存*****************************/
    /**
     * 添加缓存
     * @param set    缓存将要添加到的set集合
     * @param values 添加的缓存元素
     */
    public static void sadd(String set, String... values){
        boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
        ShardedJedis jedis = null;
        try {
            jedis = getJedis();//获取jedis实例
            if(jedis==null){
                broken = true;
                return;
            }
            /*
             * 对比：
             * lpush(String key, String... strings);
             * 返回push之后的list中包含的元素个数
             *
             * sadd(String key, String... members)
             * 1：添加元素成功
             * 0：set中已经有要添加的元素了
             */
            jedis.sadd(set, values);
        } catch (Exception e) {
            broken = true;
        }finally{
            returnJedis(jedis, broken);
        }
    }

    /**************************获取缓存*****************************/
    /**
     * 获取set集合中的所有缓存
     * @param set
     */
```

```
52      public static Set<String> smembers(String set){
53          boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
54          ShardedJedis jedis = null;
55          try {
56              jedis = getJedis();//获取jedis实例
57              if(jedis==null){
58                  broken = true;
59                  return null;
60              }
61              return jedis.smembers(set);
62          } catch (Exception e) {
63              broken = true;
64          }finally{
65              returnJedis(jedis, broken);
66          }
67          return null;
68      }
69
70      /**************************删除缓存*****************************/
71      /**
72       * 删除缓存
73       * @param set
74       * @param values
75       */
76      public static void sremove(String set, String... values){
77          boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
78          ShardedJedis jedis = null;
79          try {
80              jedis = getJedis();//获取jedis实例
81              if(jedis==null){
82                  broken = true;
83                  return;
84              }
85              jedis.srem(set, values);
86          } catch (Exception e) {
87              broken = true;
88          }finally{
89              returnJedis(jedis, broken);
90          }
91      }
92
93      /**************************其他*****************************/
94      /**
95       * set集合是否包含value
96       * @param set
97       */
98      public static boolean sismembers(String set, String value){
99          boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
100         ShardedJedis jedis = null;
101         try {
102             jedis = getJedis();//获取jedis实例
103             if(jedis==null){
104                 broken = true;
105                 return false;
106             }
107             return jedis.sismember(set, value);
108         } catch (Exception e) {
109             broken = true;
110         }finally{
111             returnJedis(jedis, broken);
112         }
113         return false;
114     }
115
116     /**
```

```
117      * 返回set集合的元素个数
118      * @param set
119      */
120     public static long ssize(String set){
121         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
122         ShardedJedis jedis = null;
123         try {
124             jedis = getJedis();//获取jedis实例
125             if(jedis==null){
126                 broken = true;
127                 return 0;
128             }
129             return jedis.scard(set);
130         } catch (Exception e) {
131             broken = true;
132         }finally{
133             returnJedis(jedis, broken);
134         }
135         return 0;
136     }
137 
138     /**
139      * 测试
140      */
141     public static void main(String[] args) {
142         sadd("adminset", "nana","jigang");
143         sadd("adminset", "nana2");
144         System.out.println(smembers("adminset"));
145         System.out.println(ssize("adminset"));
146         System.out.println(sismembers("adminset", "jigang"));
147         sremove("adminset", "jigang");
148         System.out.println(sismembers("adminset", "jigang"));
149     }
150 
151 }
```

注意：

- 元素在set中的存放顺序为：与插入set的先后书顺序无关（即无序）
- 不允许存放重复元素
- 对于set而言，Jedis有交集、差集、并集运算，可是ShardJedis没有

1.3、RedisSortedSet（有序集合工具类）

```
 1 package com.xxx.cache.redis;
 2 
 3 import java.util.Map;
 4 import java.util.Set;
 5 
 6 import redis.clients.jedis.ShardedJedis;
 7 import redis.clients.jedis.Tuple;
 8 
 9 /**
10  * sorted set缓存操作类
11  * 1、有序集合，最后的顺序是按照score从小到大的顺序排列
12  * 2、元素不能重复
13  * 3、没有从set中获取指定value的运算
14  */
15 public class RedisSortedSetUtil extends RedisBaseUtil {
16     /*************************添加缓存*****************************/
```

```
17      /**
18       * 添加缓存(一个)
19       * @param sortedSet 添加入的集合
20       * @param score        权重
21       * @param value        值
22       */
23      public static void zadd(String sortedSet,double score, String value){
24          boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
25          ShardedJedis jedis = null;
26          try {
27              jedis = getJedis();//获取jedis实例
28              if(jedis==null){
29                  broken = true;
30                  return;
31              }
32              jedis.zadd(sortedSet, score, value);
33          } catch (Exception e) {
34              broken = true;
35          }finally{
36              returnJedis(jedis, broken);
37          }
38      }
39
40      /**
41       * 添加缓存(一次可添加多个)
42       * @param sortedSet        添加入的集合
43       * @param value2score    加入集合的元素集
44       */
45      public static void zadd(String sortedSet,Map<String, Double> value2score){
46          boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
47          ShardedJedis jedis = null;
48          try {
49              jedis = getJedis();//获取jedis实例
50              if(jedis==null){
51                  broken = true;
52                  return;
53              }
54              jedis.zadd(sortedSet, value2score);
55          } catch (Exception e) {
56              broken = true;
57          }finally{
58              returnJedis(jedis, broken);
59          }
60      }
61
62      /**************************获取缓存*****************************/
63      /**
64       * 返回sortedSet内[start,end]索引的元素set
65       * 1、在sortedSet中，元素是按照score从小到大排列的，
66       *          此方法从前向后获取元素（即按元素的score从小到大排列）
67       * @param sortedSet
68       * @param start
69       * @param end
70       */
71      public static Set<String> zrange(String sortedSet, long start, long end){
72          boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
73          ShardedJedis jedis = null;
74          try {
75              jedis = getJedis();//获取jedis实例
76              if(jedis==null){
77                  broken = true;
78                  return null;
79              }
80              return jedis.zrange(sortedSet, start, end);
81          } catch (Exception e) {
```

```
82              broken = true;
83          }finally{
84              returnJedis(jedis, broken);
85          }
86          return null;
87      }
88
89      /**
90       * 返回sortedSet内所有元素，元素按照score从小到大排列
91       */
92      public static Set<String> zrange(String sortedSet){
93          return zrange(sortedSet, 0, -1);
94      }
95
96      /**
97       * 返回sortedSet集合[start, end]中的元素
98       * 1、此方法相当于从后向前取元素，即元素从大到小排列
99       * 或者相当于将sortedSet从大到小排列，然后从前向后去元素
100      * @param sortedSet
101      * @param start
102      * @param end
103      * @return
104      */
105     public static Set<String> zrevrange(String sortedSet, long start, long end){
106         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
107         ShardedJedis jedis = null;
108         try {
109             jedis = getJedis();//获取jedis实例
110             if(jedis==null){
111                 broken = true;
112                 return null;
113             }
114             return jedis.zrevrange(sortedSet, start, end);
115         } catch (Exception e) {
116             broken = true;
117         }finally{
118             returnJedis(jedis, broken);
119         }
120         return null;
121     }
122
123     /**
124      * 返回sortedSet内所有元素，元素按照score从大到小排列
125      */
126     public static Set<String> zrevrange(String sortedSet){
127         return zrevrange(sortedSet, 0, -1);
128     }
129
130     /**
131      * 获取sortedSet内[minScore, maxScore]的元素
132      * @param sortedSet
133      * @param minScore
134      * @param maxScore
135      * @return
136      */
137     public static Set<String> zrangeByScore(String sortedSet, double minScore, double maxScore){
138         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
139         ShardedJedis jedis = null;
140         try {
141             jedis = getJedis();//获取jedis实例
142             if(jedis==null){
143                 broken = true;
144                 return null;
145             }
146             return jedis.zrangeByScore(sortedSet, minScore, maxScore);
```

```
147         } catch (Exception e) {
148             broken = true;
149         }finally{
150             returnJedis(jedis, broken);
151         }
152         return null;
153     }
154 
155     /**
156      * 获取Set<Tuple>集合，其中Tuple是value与score的结构体
157      * @param sortedSet
158      * @param minScore
159      * @param maxScore
160      * @return
161      */
162     public static Set<Tuple> zrevrangeByScoreWithScores(String sortedSet, double minScore, double
maxScore){
163         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
164         ShardedJedis jedis = null;
165         try {
166             jedis = getJedis();//获取jedis实例
167             if(jedis==null){
168                 broken = true;
169                 return null;
170             }
171             return jedis.zrevrangeByScoreWithScores(sortedSet, maxScore, minScore);
172         } catch (Exception e) {
173             broken = true;
174         }finally{
175             returnJedis(jedis, broken);
176         }
177         return null;
178     }
179 
180     /*************************删除缓存*****************************/
181     /**
182      * 删除多个缓存
183      * @param sortedSet
184      * @param values
185      */
186     public static void zremove(String sortedSet, String... values){
187         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
188         ShardedJedis jedis = null;
189         try {
190             jedis = getJedis();//获取jedis实例
191             if(jedis==null){
192                 broken = true;
193                 return;
194             }
195             jedis.zrem(sortedSet, values);
196         } catch (Exception e) {
197             broken = true;
198         }finally{
199             returnJedis(jedis, broken);
200         }
201     }
202 
203     /**
204      * 删除指定范围（按照索引，包前包后）的缓存
205      * @param sortedSet
206      * @param start
207      * @param end
208      */
209     public static void zremrangeByRank(String sortedSet, long start, long end){
210         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
```

```
211         ShardedJedis jedis = null;
212         try {
213             jedis = getJedis();//获取jedis实例
214             if(jedis==null){
215                 broken = true;
216                 return;
217             }
218             jedis.zremrangeByRank(sortedSet, start, end);
219         } catch (Exception e) {
220             broken = true;
221         }finally{
222             returnJedis(jedis, broken);
223         }
224     }
225 
226     /**
227      * 删除指定范围（按照分数，包前包后）的缓存
228      * @param sortedSet
229      * @param minScore
230      * @param maxScore
231      */
232     public static void zremrangeByScore(String sortedSet, double minScore, double maxScore){
233         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
234         ShardedJedis jedis = null;
235         try {
236             jedis = getJedis();//获取jedis实例
237             if(jedis==null){
238                 broken = true;
239                 return;
240             }
241             jedis.zremrangeByScore(sortedSet, minScore, maxScore);
242         } catch (Exception e) {
243             broken = true;
244         }finally{
245             returnJedis(jedis, broken);
246         }
247     }
248 
249     /**************************其他*****************************/
250     /**
251      * 获取集合sortedSet的长度
252      * @param sortedSet
253      * @return
254      */
255     public static long zlength(String sortedSet){
256         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
257         ShardedJedis jedis = null;
258         try {
259             jedis = getJedis();//获取jedis实例
260             if(jedis==null){
261                 broken = true;
262                 return 0;
263             }
264             return jedis.zcard(sortedSet);
265         } catch (Exception e) {
266             broken = true;
267         }finally{
268             returnJedis(jedis, broken);
269         }
270         return 0;
271     }
272 
273     /**
274      * 获取sortedSet中的value的权重score
275      * @param sortedSet
```

```
276      * @param value
277      * @return
278      */
279     public static double zscore(String sortedSet, String value){
280         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
281         ShardedJedis jedis = null;
282         try {
283             jedis = getJedis();//获取jedis实例
284             if(jedis==null){
285                 broken = true;
286                 return 0;
287             }
288             return jedis.zscore(sortedSet, value);
289         } catch (Exception e) {
290             broken = true;
291         }finally{
292             returnJedis(jedis, broken);
293         }
294         return 0;
295     }
296 
297     /**
298      * 为sortedSet中的value的权重加上增量score
299      * @param sortedSet
300      * @param score
301      * @param value
302      */
303     public static void zincrby(String sortedSet,double score, String value){
304         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
305         ShardedJedis jedis = null;
306         try {
307             jedis = getJedis();//获取jedis实例
308             if(jedis==null){
309                 broken = true;
310                 return;
311             }
312             jedis.zincrby(sortedSet, score, value);
313         } catch (Exception e) {
314             broken = true;
315         }finally{
316             returnJedis(jedis, broken);
317         }
318     }
319 
320     /**
321      * 测试
322      */
323     public static void main(String[] args) {
324         zadd("aaset", 1, "aa");
325         zadd("aaset", 4, "bb");
326         zadd("aaset", 0, "cc");
327 
328         System.out.println(zrange("aaset", 0, -1));//[cc, aa, bb]
329         System.out.println(zrange("aaset", 0, 1));//[cc, aa]
330         System.out.println(zrevrange("aaset", 0, -1));//[bb, aa, cc]
331         System.out.println(zrevrange("aaset", 0, 1));//[bb, aa]
332         System.out.println(zrangeByScore("aaset", 0, 2));//[cc, aa]
333         Set<Tuple> s = zrevrangeByScoreWithScores("aaset", 0, 2);
334         for(Tuple t : s){
335             System.out.println(t.getElement()+"-->"+t.getScore());//aa-->1.0  cc-->0.0
336         }
337 
338         System.out.println(zlength("aaset"));//3
339         System.out.println(zscore("aaset","bb"));//4.0
340         zincrby("aaset",10,"bb");
```

```
341         System.out.println(zscore("aaset","bb"));//14.0
342         zremove("aaset", "cc");
343         System.out.println(zrange("aaset"));//[aa, bb]
344         zremrangeByScore("aaset", 10, 20);
345         System.out.println(zrange("aaset"));//[aa]
346     }
347
348 }
```

注意：

- 元素在set中的存放顺序为：根据score（权重）从小到大排列
- 不允许存放重复元素

1.4、RedisHashUtil（hash工具类）

```
 1 package com.xxx.cache.redis;
 2
 3 import java.util.HashMap;
 4 import java.util.List;
 5 import java.util.Map;
 6 import java.util.Set;
 7
 8 import redis.clients.jedis.ShardedJedis;
 9
10 /**
11  * hash缓存操作类
12  */
13 public class RedisHashUtil extends RedisBaseUtil {
14     /**************************添加缓存*****************************/
15     /**
16      * 添加单个缓存key-value到map中
17      */
18     public static void hset(String map, String key, String value){
19         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
20         ShardedJedis jedis = null;
21         try {
22             jedis = getJedis();//获取jedis实例
23             if(jedis==null){
24                 broken = true;
25                 return;
26             }
27             jedis.hset(map, key, value);
28         } catch (Exception e) {
29             broken = true;
30         }finally{
31             returnJedis(jedis, broken);
32         }
33     }
34
35     /**
36      * 添加单个缓存key-value到map中
37      * 若已经存在于指定key相同的key，那么就不操作
38      */
39     public static void hsetnx(String map, String key, String value){
40         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
41         ShardedJedis jedis = null;
42         try {
43             jedis = getJedis();//获取jedis实例
44             if(jedis==null){
```

```
45                broken = true;
46                return;
47            }
48            jedis.hsetnx(map, key, value);
49        } catch (Exception e) {
50            broken = true;
51        }finally{
52            returnJedis(jedis, broken);
53        }
54    }
55
56    /**
57     * 在map中添加key2value的map，即一次性添加多条缓存
58     * @param map
59     * @param key2value
60     */
61    public static void hmset(String map, Map<String, String> key2value){
62        boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
63        ShardedJedis jedis = null;
64        try {
65            jedis = getJedis();//获取jedis实例
66            if(jedis==null){
67                broken = true;
68                return;
69            }
70            jedis.hmset(map, key2value);
71        } catch (Exception e) {
72            broken = true;
73        }finally{
74            returnJedis(jedis, broken);
75        }
76    }
77
78    /**************************获取缓存*****************************/
79    /**
80     * 获取map中key的集合
81     * @param set
82     */
83    public static Set<String> hkeys(String map){
84        boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
85        ShardedJedis jedis = null;
86        try {
87            jedis = getJedis();//获取jedis实例
88            if(jedis==null){
89                broken = true;
90                return null;
91            }
92            return jedis.hkeys(map);
93        } catch (Exception e) {
94            broken = true;
95        }finally{
96            returnJedis(jedis, broken);
97        }
98        return null;
99    }
100
101   /**
102    * 获取map中的所有key的value
103    */
104   public static List<String> hvals(String map){
105       boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
106       ShardedJedis jedis = null;
107       try {
108           jedis = getJedis();//获取jedis实例
109           if(jedis==null){
```

```
110                 broken = true;
111                 return null;
112             }
113             return jedis.hvals(map);
114         } catch (Exception e) {
115             broken = true;
116         }finally{
117             returnJedis(jedis, broken);
118         }
119         return null;
120     }
121 
122     /**
123      * 从map中获取多个key的value，并放在List集合中
124      */
125     public static List<String> hmget(String map, String... keys){
126         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
127         ShardedJedis jedis = null;
128         try {
129             jedis = getJedis();//获取jedis实例
130             if(jedis==null){
131                 broken = true;
132                 return null;
133             }
134             return jedis.hmget(map, keys);
135         } catch (Exception e) {
136             broken = true;
137         }finally{
138             returnJedis(jedis, broken);
139         }
140         return null;
141     }
142 
143     /**
144      * 从map中获取全部的缓存key-value对
145      */
146     public static Map<String, String> hgetAll(String map){
147         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
148         ShardedJedis jedis = null;
149         try {
150             jedis = getJedis();//获取jedis实例
151             if(jedis==null){
152                 broken = true;
153                 return null;
154             }
155             return jedis.hgetAll(map);
156         } catch (Exception e) {
157             broken = true;
158         }finally{
159             returnJedis(jedis, broken);
160         }
161         return null;
162     }
163 
164     /**
165      * 从map中获取相应key的缓存value
166      */
167     public static String hget(String map, String key){
168         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
169         ShardedJedis jedis = null;
170         try {
171             jedis = getJedis();//获取jedis实例
172             if(jedis==null){
173                 broken = true;
174                 return null;
```

```
175             }
176             return jedis.hget(map, key);
177         } catch (Exception e) {
178             broken = true;
179         }finally{
180             returnJedis(jedis, broken);
181         }
182         return null;
183     }
184 
185     /**************************删除缓存*****************************/
186     /**
187      * 从map中删除多个缓存
188      */
189     public static void hdel(String map, String... keys){
190         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
191         ShardedJedis jedis = null;
192         try {
193             jedis = getJedis();//获取jedis实例
194             if(jedis==null){
195                 broken = true;
196                 return;
197             }
198             jedis.hdel(map, keys);
199         } catch (Exception e) {
200             broken = true;
201         }finally{
202             returnJedis(jedis, broken);
203         }
204     }
205 
206     /**************************其他*****************************/
207     /**
208      * 获取map中的key-value数
209      */
210     public static long hlen(String map){
211         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
212         ShardedJedis jedis = null;
213         try {
214             jedis = getJedis();//获取jedis实例
215             if(jedis==null){
216                 broken = true;
217                 return 0;
218             }
219             return jedis.hlen(map);
220         } catch (Exception e) {
221             broken = true;
222         }finally{
223             returnJedis(jedis, broken);
224         }
225         return 0;
226     }
227 
228     /**
229      * map中是否存在键为key的缓存
230      */
231     public static boolean hexists(String map, String key){
232         boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
233         ShardedJedis jedis = null;
234         try {
235             jedis = getJedis();//获取jedis实例
236             if(jedis==null){
237                 broken = true;
238                 return false;
239             }
```

```
            return jedis.hexists(map, key);
        } catch (Exception e) {
            broken = true;
        }finally{
            returnJedis(jedis, broken);
        }
        return false;
    }

    /**
     * 测试
     */
    public static void main(String[] args) {
        hset("aamap", "aa1", "aa11");
        Map<String,String> maps = new HashMap<String, String>();
        maps.put("aa2", "aa22");
        maps.put("aa3", "aa33");
        hmset("aamap", maps);

        System.out.println(hkeys("aamap"));//[aa3, aa2, aa1]
        System.out.println(hvals("aamap"));//[aa33, aa22, aa11]
        System.out.println(hgetAll("aamap"));//{aa3=aa33, aa2=aa22, aa1=aa11}
        System.out.println(hget("aamap","aa2"));//aa22
        System.out.println(hmget("aamap","aa2","aa1"));//[aa22, aa11]

        System.out.println(hlen("aamap"));//3
        System.out.println(hexists("aamap","aa3"));//true
        System.out.println(hexists("aamap","aa0"));//false

        hdel("aamap","aa0");
        hdel("aamap","aa1");
        System.out.println(hgetAll("aamap"));//{aa3=aa33, aa2=aa22}

    }
}
```

注意：

- 有根据key获取缓存value的方法

1.5、RedisCacheConstant（创建一些redis使用的自定义的map/set/list/soretd set名）

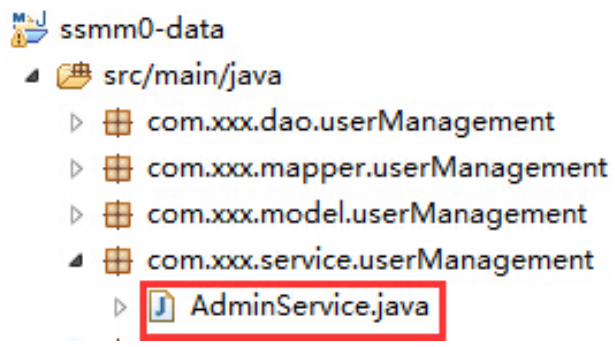
```
package com.xxx.cache.util;

/**
 * 创建一些redis使用的自定义的map/set/list/soretd set名
 */
public enum RedisCacheConstant {
    USER_MANAGEMENT_MAP,    //人员管理业务类缓存map
    HOTEL_MANAGEMENT_MAP;    //酒店管理业务类缓存map
}
```

注意：

- 这一块儿与业务有关
- 在我们的API的设计过程中，可以直接将自己封装的方法中的list/set/sorted set/map参数的类型有String改为RedisCacheConstant，而在方法内部你调用ShardJedis的API的时候，使用String.valueOf(RedisCacheConstant)这样的方式，这样就会减少在实际开发中的代码量。

2、ssmm0-data



AdminService：

```
    /**********************redis hash********************/
    /*
     * 此处用set、list、sorted set都不太好，因为三者都不具备根据key查找值的能力，
     * 以set为例，你缓存的时候，只能缓存一个id进去，最后查询缓存，查到缓存中有ID之后，还需要再根据此ID查询数据库，才能返回具体的admin，还不如直接根据ID去数据库查询
     *
     * set的一个典型应用场景：
     * 当有用户注册或者用户信息修改或用户被删除之后，我们将其ID放入缓存，
     * 之后可能会启动一个定时任务，定时扫描该set中是否有ID存在，如果有，说明有用户信息发生变化，
     * 然后再进行一些操作，操作之后将set清空。之后继续循环上述的方式
     *
     * 这里有个问题？set的操作是线程安全的吗？
     */
    public Admin findAdminByIdFromRedisHash(int id) {
        //从缓存中获取数据：注意这里可以直接将RedisHashUtil中的hget方法的map改为RedisCacheConstant类型，下边同理，其他set、list、sorted set也同理
        String adminStr = RedisHashUtil.hget(String.valueOf(RedisCacheConstant.USER_MANAGEMENT_MAP), String.valueOf(id));
        //若缓存中有，直接返回
        if(StringUtils.isNoneBlank(adminStr)){
            return Admin.parseJsonToAdmin(adminStr);
        }
        //若缓存中没有，从数据库查询
        Admin admin = adminDao.getUserById(id);
        //若查询出的数据不为null
        if(admin!=null){
            //将数据存入缓存
            RedisHashUtil.hset(String.valueOf(RedisCacheConstant.USER_MANAGEMENT_MAP), String.valueOf(id), admin.toJson());
        }
        //返回从数据库查询的admin（当然也可能数据库中也没有，就是null）
        return admin;
    }
```

说明：在AdminService类中只添加了如上方法。

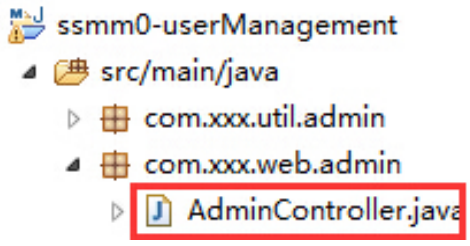
注意：

- String、hash具有按key查找value的功能
- list、set、sorted set没有按key查找的功能

适用场景：

- 需要按key查找value的，用hash和String
- set类的两种典型应用场景：（归结为一句话：存储特殊的key，之后可能还需要根据这些key进行数据库的查询）
  - 当有用户注册或者用户信息修改或用户被删除之后，我们将其ID放入缓存，之后可能会启动一个定时任务，定时扫描该set中是否有ID存在，如果有，说明有用户信息发生变化，然后再进行一些操作，操作之后将set清空。之后继续循环上述的方式
  - 存储一个网站的活跃的用户ID，之后我们可以确定set中存在的所有ID都是活跃用户，之后可以按照他们的ID进行数据库的查询；如果用MySQL去做这个事儿，可能需要扫描全表，然后再进行一些操作

3、ssmm0-userManagement



AdminController：

```
1     /**
2      * 根据id查找Admin
3      */
4     @ResponseBody
5     @RequestMapping("/findAdminByIdFromRedisHash")
6     public Admin findAdminByIdFromRedisHash(@RequestParam(value="id") int id){
7
8         return adminService.findAdminByIdFromRedisHash(id);
9     }
```

说明：AdminController中只添加了如上方法。

4、测试

测试方法同上一章。

总结：

- list
  - 元素在list中的存放顺序为：插入list的顺序（从左边插入在头部，从右边插入在尾部）
  - 允许存放重复元素
  - 可用作模拟队列（queue）、堆栈（stack），支持双向操作（L--首部或者R--尾部）
  - 左边入队，相当于在队头插入元素，则之后的元素都要后移一位；而右边入队的话元素直接插在队尾，之前的元素的索引不变（推荐使用右边入队，即队尾入队）
- set
  - 元素在set中的存放顺序为：与插入set的先后书顺序无关（即无序）
  - 不允许存放重复元素
  - 对于set而言，Jedis有交集、差集、并集运算，可是ShardJedis没有
- soretd set
  - 元素在set中的存放顺序为：根据score（权重）从小到大排列
  - 不允许存放重复元素

相同点：

- index
  - 从0开始 -1表示结尾 -2表示倒数第二个
- API中的 start end参数
  - 都是包前也包后的
- 按key查找功能
  - list、set、sorted set没有按key查找的功能
  - String、hash具有按key查找value的功能
- 直接的指定缓存过期的API
  - String有
  - list、set、sorted set、hash没有，但是可以按例如如下的方式指定缓存过期时间

```
1     /**
2      * 从左边（首部）加入列表
3      * 并指定列表缓存过期时间
4      */
```

```
    public static void lpush(String list, int expire, String... values){
        boolean broken = false;//标记：该操作是否被异常打断而没有正常结束
        ShardedJedis jedis = null;
        try {
            jedis = getJedis();//获取jedis实例
            if(jedis==null){
                broken = true;
                return;
            }
            /*
             * lpush(String key, String... strings);
             * 返回push之后的list中包含的元素个数
             */
            jedis.lpush(list, values);
            jedis.expire(list, expire);//为该list设置缓存过期时间
        } catch (Exception e) {
            broken = true;
        }finally{
            returnJedis(jedis, broken);
        }
    }
```

- 删除整个元素
  - jedis.del(list)：可用于五种结构

疑问：关于jedis、xmemcached的线程安全问题，是如何解决的？