RedisTemplate用法详解

原创： 麦小刚 麦小七的夏天 2019-03-29

Redis 可以存储键与5种不同数据结构类型之间的映射，这5种数据结构类型分别为String（字符串）、List（列表）、Set（集合）、Hash（散列）和 Zset（有序集合）。

RedisTemplate介绍

Spring封装了RedisTemplate对象来进行对Redis的各种操作，它支持所有的Redis原生的api。

Type Parameters:
K：模板中的Redis key的类型（通常为String）如：RedisTemplate<String, Object>

注意：**如果没特殊情况，切勿定义成RedisTemplate<Object, Object>**，否则根据里氏替换原则，使用的时候会造成类型错误 。

V：模板中的Redis value的类型

单机redis使用工具类

application.properties配置内容

-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

```
#redis配置#单机模式#redis数据库索引，默认为0spring.redis.database=0#redis服务器地址spring.redis.host=127.0.0.1#redis服务器连接端口号spring.redis.port=6379#redis服务器连接密码，默认为空spring.redis.password=#redis连接池最大活跃连接数spring.redis.pool.max-active=100#
```

redis连接池最大阻塞等待时间spring.redis.pool.max.wait=-1#redis连接池最大空闲连接数srping.redis.pool.max-idle=8#redis连接池最小空闲连接数spring.redis.pool.min-idle=0#redis连接池超时时间，单位为毫秒spring.redis.pool.timeout=60000

Redis的String数据结构

set void set(K key, V value);

redisTemplate.opsForValue().set("num","123");redisTemplate.opsForValue().get("num") 输出结果为123

set void set(K key, V value, long timeout, TimeUnit unit);

redisTemplate.opsForValue().set("num","123",10, TimeUnit.SECONDS);redisTemplate.opsForValue().get("num")设置的是10秒失效，十秒之内查询有结果，十秒之后返回为null

set void set(K key, V value, long offset);

覆写(overwrite)给定 key 所储存的字符串值，从偏移量 offset 开始

template.opsForValue().set("key","hello world");template.opsForValue().set("key","redis", 6);System.out.println("***************"+template.opsForValue().get("key"));结果：***************hello redis

get V get(Object key);

template.opsForValue().set("key","欢迎关注麦小七的夏天");System.out.println("**********"+template.opsForValue().get("key"));结果：***************欢迎关注麦小七的夏天

getAndSet V getAndSet(K key, V value);

设置键的字符串值并返回其旧值

template.opsForValue().set("getSetTest","test");System.out.println(template.opsForValue().getAndSet("getSetTest","test2"));结果：test

append Integer append(K key, String value);

如果key已经存在并且是一个字符串，则该命令将该值追加到字符串的末尾。如果键不存在，则它被创建并设置为空字符串，因此APPEND在这种特殊情况下将类似于SET。

template.opsForValue().append("test","欢迎关注");System.out.println(template.opsForValue().get("test"));template.opsForValue().append("test","麦小七的夏天");System.out.

```
println(template.opsForValue().get("test"));欢迎关注欢迎关注麦小七的夏天
```

size Long size(K key);
返回key所对应的value值得长度

```
template.opsForValue().set("key","麦小七的夏天");System.out.println("**************
"+template.opsForValue().size("key"));***************6
```

Redis的List数据结构

Long size(K key);
返回存储在键中的列表的长度。如果键不存在，则将其解释为空列表，并返回0。当key存储的值不是列表时返回错误。

```
System.out.println(template.opsForList().size("list"));6
```

Long leftPush(K key, V value);
将所有指定的值插入存储在键的列表的头部。如果键不存在，则在执行推送操作之前将其创建为空列表。（从左边插入）

```
template.opsForList().leftPush("list","java");template.opsForList().leftPush("list
","python");template.opsForList().leftPush("list","c++");返回的结果为推送操作后的列表的
长度123
```

Long leftPushAll(K key, V... values);
批量把一个数组插入到列表中

```
String[] strs = new String[]{"1","2","3"};template.opsForList().leftPushAll("list"
,strs);System.out.println(template.opsForList().range("list",0,-1));[3, 2, 1]
```

Long rightPush(K key, V value);
将所有指定的值插入存储在键的列表的头部。如果键不存在，则在执行推送操作之前将其创建为空列表。（从右边插入）

```
template.opsForList().rightPush("listRight","java");template.opsForList().rightPus
h("listRight","python");template.opsForList().rightPush("listRight","c++");123
```

Long rightPushAll(K key, V... values);

```
String[] strs = new String[]{"1","2","3"};template.opsForList().rightPushAll("list
",strs);System.out.println(template.opsForList().range("list",0,-1));[1, 2, 3]
```

void set(K key, long index, V value);
在列表中index的位置设置value值

```
System.out.println(template.opsForList().range("listRight",0,-1));template.opsForL
ist().set("listRight",1,"setValue");System.out.println(template.opsForList().range
("listRight",0,-1));[java, python, oc, c++][java, setValue, oc, c++]
```

Long remove(K key, long count, Object value);
从存储在键中的列表中删除等于值的元素的第一个计数事件。
计数参数以下列方式影响操作：
count> 0：删除等于从头到尾移动的值的元素。
count <0：删除等于从尾到头移动的值的元素。
count = 0：删除等于value的所有元素。

```
System.out.println(template.opsForList().range("listRight",0,-1));template.opsForL
ist().remove("listRight",1,"setValue");//将删除列表中存储的列表中第一次次出现的"setValue"
。System.out.println(template.opsForList().range("listRight",0,-1));[java, setValue
, oc, c++][java, oc, c++]
```

V index(K key, long index);
根据下表获取列表中的值，下标是从0开始的

```
System.out.println(template.opsForList().range("listRight",0,-1));System.out.print
ln(template.opsForList().index("listRight",2));[java, oc, c++]c++
```

V leftPop(K key);
弹出最左边的元素，弹出之后该值在列表中将不复存在

```
System.out.println(template.opsForList().range("list",0,-1));System.out.println(te
mplate.opsForList().leftPop("list"));System.out.println(template.opsForList().rang
e("list",0,-1));[c++, python, oc, java, c#, c#]c++[python, oc, java, c#, c#]
```

V rightPop(K key);
弹出最右边的元素，弹出之后该值在列表中将不复存在

```
System.out.println(template.opsForList().range("list",0,-1));System.out.println(te
mplate.opsForList().rightPop("list"));System.out.println(template.opsForList().ran
ge("list",0,-1));[python, oc, java, c#, c#]c#[python, oc, java, c#]
```

Redis的Hash数据机构

Long delete(H key, Object... hashKeys);
删除给定的哈希hashKeys

```
System.out.println(template.opsForHash().delete("redisHash","name"));System.out.pr
intln(template.opsForHash().entries("redisHash"));1{class=6, age=28.1}
```

Boolean hasKey(H key, Object hashKey);
确定哈希hashKey是否存在

```
System.out.println(template.opsForHash().hasKey("redisHash","666"));System.out.pri
ntln(template.opsForHash().hasKey("redisHash","777"));truefalse
```

HV get(H key, Object hashKey);
从键中的哈希获取给定hashKey的值

```
System.out.println(template.opsForHash().get("redisHash","age"));26
```

Set<HK> keys(H key);
获取key所对应的散列表的key

```
System.out.println(template.opsForHash().keys("redisHash"));//redisHash所对应的散列表为{class=1, name=666, age=27}[name, class, age]
```

Long size(H key);
获取key所对应的散列表的大小个数

```
System.out.println(template.opsForHash().size("redisHash"));//redisHash所对应的散列表为{class=1, name=666, age=27}3
```

void putAll(H key, Map<? extends HK, ? extends HV> m);
使用m中提供的多个散列字段设置到key对应的散列表中

```
Map<String,Object> testMap = new HashMap();testMap.put("name","666");testMap.put("age",27);testMap.put("class","1");template.opsForHash().putAll("redisHash1",testMap);System.out.println(template.opsForHash().entries("redisHash1"));{class=1, name=jack, age=27}
```

void put(H key, HK hashKey, HV value);
设置散列hashKey的值

```
template.opsForHash().put("redisHash","name","666");template.opsForHash().put("redisHash","age",26);template.opsForHash().put("redisHash","class","6");System.out.println(template.opsForHash().entries("redisHash"));{age=26, class=6, name=666}
```

List<HV> values(H key);
获取整个哈希存储的值根据密钥

```
System.out.println(template.opsForHash().values("redisHash"));[tom, 26, 6]
```

Map<HK, HV> entries(H key);
获取整个哈希存储根据密钥

```
System.out.println(template.opsForHash().entries("redisHash"));{age=26, class=6, name=tom}
```

Cursor<Map.Entry<HK, HV>> scan(H key, ScanOptions options);
使用Cursor在key的hash中迭代，相当于迭代器。

```
Cursor<Map.Entry<Object, Object>> curosr = template.opsForHash().scan("redisHash",
   ScanOptions.ScanOptions.NONE);    while(curosr.hasNext()){        Map.Entry<Obj
ect, Object> entry = curosr.next();        System.out.println(entry.getKey()+":"+e
ntry.getValue());    }age:27class:6name:666
```

Redis的Set数据结构

Long add(K key, V... values);
无序集合中添加元素，返回添加个数
也可以直接在add里面添加多个值 如：template.opsForSet().add("setTest","aaa","bbb")

```
String[] strs= new String[]{"str1","str2"};System.out.println(template.opsForSet()
.add("setTest", strs));2
```

Long remove(K key, Object... values);
移除集合中一个或多个成员

```
String[] strs = new String[]{"str1","str2"};System.out.println(template.opsForSet(
).remove("setTest",strs));2
```

V pop(K key);
移除并返回集合中的一个随机元素

```
System.out.println(template.opsForSet().pop("setTest"));System.out.println(templat
e.opsForSet().members("setTest"));bbb[aaa, ccc]
```

Boolean move(K key, V value, K destKey);
将 member 元素从 source 集合移动到 destination 集合

```
template.opsForSet().move("setTest","aaa","setTest2");System.out.println(template.
opsForSet().members("setTest"));System.out.println(template.opsForSet().members("s
etTest2"));[ccc][aaa]
```

Long size(K key);
无序集合的大小长度

```
System.out.println(template.opsForSet().size("setTest"));1
```

Set<V> members(K key);
返回集合中的所有成员

```
System.out.println(template.opsForSet().members("setTest"));[ddd, bbb, aaa, ccc]
```

Cursor<V> scan(K key, ScanOptions options);
遍历set

```
Cursor<Object> curosr = template.opsForSet().scan("setTest", ScanOptions.NONE);  w
hile(curosr.hasNext()){     System.out.println(curosr.next());  }dddbbbaaaccc
```

Redis的ZSet数据结构

Boolean add(K key, V value, double score);
新增一个有序集合，存在的话为false，不存在的话为true

```
System.out.println(template.opsForZSet().add("zset1","zset-1",1.0));true
```

Long add(K key, Set<TypedTuple<V>> tuples);
新增一个有序集合

```
ZSetOperations.TypedTuple<Object> objectTypedTuple1 = new DefaultTypedTuple<>("zse
t-5",9.6);ZSetOperations.TypedTuple<Object> objectTypedTuple2 = new DefaultTypedTu
ple<>("zset-6",9.9);Set<ZSetOperations.TypedTuple<Object>> tuples = new HashSet<ZS
etOperations.TypedTuple<Object>>();tuples.add(objectTypedTuple1);tuples.add(object
TypedTuple2);System.out.println(template.opsForZSet().add("zset1",tuples));System.
```

```
out.println(template.opsForZSet().range("zset1",0,-1));[zset-1, zset-2, zset-3, zs
et-4, zset-5, zset-6]
```

Long remove(K key, Object... values);
从有序集合中移除一个或者多个元素

```
System.out.println(template.opsForZSet().range("zset1",0,-1));System.out.println(t
emplate.opsForZSet().remove("zset1","zset-6"));System.out.println(template.opsForZ
Set().range("zset1",0,-1));[zset-1, zset-2, zset-3, zset-4, zset-5, zset-6]1[zset-
1, zset-2, zset-3, zset-4, zset-5]
```

Long rank(K key, Object o);
返回有序集中指定成员的排名，其中有序集成员按分数值递增(从小到大)顺序排列

```
System.out.println(template.opsForZSet().range("zset1",0,-1));System.out.println(t
emplate.opsForZSet().rank("zset1","zset-2"));[zset-2, zset-1, zset-3, zset-4, zset
-5]0   //表明排名第一
```

Set<V> range(K key, long start, long end);
通过索引区间返回有序集合成指定区间内的成员，其中有序集成员按分数值递增(从小到大)顺序排列

```
System.out.println(template.opsForZSet().range("zset1",0,-1));[zset-2, zset-1, zse
t-3, zset-4, zset-5]
```

Long count(K key, double min, double max);
通过分数返回有序集合指定区间内的成员个数

```
System.out.println(template.opsForZSet().rangeByScore("zset1",0,5));System.out.pri
ntln(template.opsForZSet().count("zset1",0,5));[zset-2, zset-1, zset-3]3
```

Long size(K key);
获取有序集合的成员数，内部调用的就是zCard方法

```
System.out.println(template.opsForZSet().size("zset1"));6
```

Double score(K key, Object o);
获取指定成员的score值

```
System.out.println(template.opsForZSet().score("zset1","zset-1"));2.2
```

Long removeRange(K key, long start, long end);
移除指定索引位置的成员，其中有序集成员按分数值递增(从小到大)顺序排列

```
System.out.println(template.opsForZSet().range("zset2",0,-1));System.out.println(t
emplate.opsForZSet().removeRange("zset2",1,2));System.out.println(template.opsForZ
Set().range("zset2",0,-1));[zset-1, zset-2, zset-3, zset-4]2[zset-1, zset-4]
```

Cursor<TypedTuple<V>> scan(K key, ScanOptions options);
遍历zset

```
Cursor<ZSetOperations.TypedTuple<Object>> cursor = template.opsForZSet().scan("zzs
et1", ScanOptions.NONE);    while (cursor.hasNext()){       ZSetOperations.TypedTu
ple<Object> item = cursor.next();       System.out.println(item.getValue() + ":" +
 item.getScore());    }zset-1:1.0zset-2:2.0zset-3:3.0zset-4:6.0
```