<https://blog.csdn.net/weixin_37490221/article/details/78135036>

# RedisTemplate访问Redis数据结构（三）——Hash

2017年09月29日 16:21:43 [枣面包](https://me.csdn.net/weixin_37490221) 阅读数：6214 标签： [spring-data-redis](http://so.csdn.net/so/search/s.do?q=spring-data-redis&t=blog)[RedisTemplate](http://so.csdn.net/so/search/s.do?q=RedisTemplate&t=blog)[散列](http://so.csdn.net/so/search/s.do?q=%E6%95%A3%E5%88%97&t=blog)[hash](http://so.csdn.net/so/search/s.do?q=hash&t=blog)[redis](http://so.csdn.net/so/search/s.do?q=redis&t=blog) 更多

个人分类： [后台开发](https://blog.csdn.net/weixin_37490221/article/category/7994204)

所属专栏： [和枣面包一起学习Redis](https://blog.csdn.net/column/details/24544.html)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/weixin\_37490221/article/details/78135036

[Redis ——Linux系统下的编译安装](https://blog.csdn.net/weixin_37490221/article/details/81513800)   
[RedisTemplate访问Redis数据结构（前言）](http://blog.csdn.net/weixin_37490221/article/details/78134105)   
[RedisTemplate访问Redis数据结构（一）——String](http://blog.csdn.net/weixin_37490221/article/details/78134521)   
[RedisTemplate访问Redis数据结构（二）——List](http://blog.csdn.net/weixin_37490221/article/details/78134748)   
[RedisTemplate访问Redis数据结构（三）——Hash](http://blog.csdn.net/weixin_37490221/article/details/78135036)   
[RedisTemplate访问Redis数据结构（四）——Set](http://blog.csdn.net/weixin_37490221/article/details/78135202)   
[RedisTemplate访问Redis数据结构（五）——ZSet](http://blog.csdn.net/weixin_37490221/article/details/78135815)   
[Github源码地址](https://github.com/zaomianbao/RedisDemo)   
[Redis ——事务详解](https://blog.csdn.net/weixin_37490221/article/details/78197613)

Redis的散列可以让用户将多个键值对存储到一个Redis键里面。此处我们使用redisTemplate，配置如下（详情请见链接：[RedisTemplate访问Redis数据结构（前言）](http://blog.csdn.net/weixin_37490221/article/details/78134105)）

<bean id="redisTemplate" class="org.springframework.data.redis.core.RedisTemplate">

<property name="connectionFactory" ref="jedisConnFactory"/>

<property name="keySerializer">

<bean class="org.springframework.data.redis.serializer.StringRedisSerializer"/>

</property>

<property name="hashKeySerializer">

<bean class="org.springframework.data.redis.serializer.StringRedisSerializer"/>

</property>

<property name="valueSerializer">

<bean class="org.springframework.data.redis.serializer.GenericJackson2JsonRedisSerializer"/>

</property>

<property name="hashValueSerializer">

<bean class="org.springframework.data.redis.serializer.GenericJackson2JsonRedisSerializer"/>

</property>

</bean>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15

HashOperations提供一系列方法操作hash。首先初始化spring工厂获得redisTemplate和opsForHash

private RedisTemplate<String,Object> redisTemplate;

private HashOperations<String,String,Object> opsForHash;

@SuppressWarnings("unchecked")

@Before

public void before(){

@SuppressWarnings("resource")

ApplicationContext context = new ClassPathXmlApplicationContext("/applicationContext.xml");

redisTemplate = (RedisTemplate<String,Object>)context.getBean("redisTemplate");

opsForHash = redisTemplate.opsForHash();

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

##### void put(H key, HK hashKey, HV value);

##### Map< HK, HV > entries(H key);

@Test

public void testPut(){

opsForHash.put("he1", "key1", "a");

opsForHash.put("he1", "key2", "b");

opsForHash.put("he1", "key3", "c");

Map<String, Object> entries = opsForHash.entries("he1");

System.out.println(entries);*//{key3=c, key1=a, key2=b}(无序)*

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

##### void putAll(H key, Map< ? extends HK, ? extends HV > m);

@Test

public void testPutAll(){

Map<String,Object> param = new HashMap<String,Object>();

param.put("key1", "a");

param.put("key2", "b");

param.put("key3", "c");

opsForHash.putAll("he2", param);

System.out.println(opsForHash.entries("he2"));*//{key2=b, key1=a, key3=c}*

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

##### Long delete(H key, Object… hashKeys);

@Test

public void testDelete(){

Map<String,Object> param = new HashMap<String,Object>();

param.put("key1", "a");

param.put("key2", "b");

param.put("key3", "c");

opsForHash.putAll("he3", param);

System.out.println(opsForHash.entries("he3"));*//{key3=c, key2=b, key1=a}*

opsForHash.delete("he3", "key1");

System.out.println(opsForHash.entries("he3"));*//{key2=b, key3=c}*

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

##### Boolean hasKey(H key, Object hashKey);

@Test

public void testHashKey(){

Map<String,Object> param = new HashMap<String,Object>();

param.put("key1", "a");

param.put("key2", "b");

param.put("key3", "c");

opsForHash.putAll("he4", param);

System.out.println(opsForHash.hasKey("he", "key2"));*//false*

System.out.println(opsForHash.hasKey("he4", "key4"));*//false*

System.out.println(opsForHash.hasKey("he4", "key2"));*//true*

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

##### HV get(H key, Object hashKey);

@Test

public void testGet(){

Map<String,Object> param = new HashMap<String,Object>();

param.put("key1", "a");

param.put("key2", "b");

param.put("key3", "c");

opsForHash.putAll("he5", param);

System.out.println(opsForHash.get("he5", "key1"));*//a*

System.out.println(opsForHash.get("he5", "key"));*//null*

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

##### List< HV > multiGet(H key, Collection< HK > hashKeys);

@Test

public void testMultiGet(){

Map<String,Object> param = new HashMap<String,Object>();

param.put("key1", "a");

param.put("key2", "b");

param.put("key3", "c");

opsForHash.putAll("he6", param);

List<String> keys = new ArrayList<String>();

keys.add("key1");

keys.add("key");

keys.add("key2");

System.out.println(opsForHash.multiGet("he6", keys));*//[a, null, b]*

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13

##### Long increment(H key, HK hashKey, long delta);

##### Double increment(H key, HK hashKey, double delta);

@Test

public void testIncrement(){

Map<String,Object> param = new HashMap<String,Object>();

param.put("key1", "a");

param.put("key2", "b");

param.put("key3", "c");

param.put("key4", 4);

opsForHash.putAll("he7", param);

System.out.println(opsForHash.increment("he7", "key4", 1));*//5*

System.out.println(opsForHash.increment("he7", "key4", 1.1));*//6.1*

try {

opsForHash.increment("he7", "key1", 1);*//ERR hash value is not an integer*

} catch (Exception e) {

e.printStackTrace();

}

try {

opsForHash.increment("he7", "key1", 1.1);*//ERR hash value is not a float*

} catch (Exception e) {

e.printStackTrace();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21

##### Set< HK > keys(H key);

@Test

public void testKeys(){

redisTemplate.delete("he8");

Map<String,Object> param = new HashMap<String,Object>();

param.put("key4", "d");

param.put("key1", "a");

param.put("key3", "c");

param.put("key5", "e");

param.put("key2", "b");

opsForHash.putAll("he8", param);

Set<String> keys = opsForHash.keys("he8");

System.out.println(keys);*//[key4, key3, key5, key2, key1]*

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13

##### Long size(H key);

@Test

public void testSize(){

Map<String,Object> param = new HashMap<String,Object>();

param.put("key4", "d");

param.put("key1", "a");

param.put("key3", "c");

param.put("key5", "e");

param.put("key2", "b");

opsForHash.putAll("he9", param);

System.out.println(opsForHash.size("he9"));*//5*

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

##### Boolean putIfAbsent(H key, HK hashKey, HV value);

@Test

public void testPutIfAbsent(){

*//仅当hashKey不存在时才设置散列hashKey的值。*

System.out.println(opsForHash.putIfAbsent("he10", "key1", "a"));*//true*

System.out.println(opsForHash.putIfAbsent("he10", "key1", "a"));*//false*

}

* 1
* 2
* 3
* 4
* 5
* 6

##### List< HV > values(H key);

@Test

public void testValues(){

Map<String,Object> param = new HashMap<String,Object>();

param.put("key4", "d");

param.put("key1", "a");

param.put("key3", "c");

param.put("key5", "e");

param.put("key2", "b");

opsForHash.putAll("he11", param);

List<Object> values = opsForHash.values("he11");

System.out.println(values);*//[d, c, e, b, a]*

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12

##### Cursor< Map.Entry< HK, HV >> scan(H key, ScanOptions options);

@Test

public void testScan(){

Map<String,Object> param = new HashMap<String,Object>();

param.put("key4", "d");

param.put("key1", "a");

param.put("key3", "c");

param.put("key5", "e");

param.put("key2", "b");

opsForHash.putAll("he13", param);

Cursor<Map.Entry<String, Object>> curosr = opsForHash.scan("he13", ScanOptions.NONE);

while(curosr.hasNext()){

Map.Entry<String, Object> entry = curosr.next();

System.out.println(entry.getKey()+":"+entry.getValue());

}

/\*\*

key4:d

key3:c

key5:e

key2:b

key1:a

\*/

}