https://blog.csdn.net/linzhiqiang0316/article/details/52711908

**SpringBoot中Redis的set、map、list、value、实体类等基本操作介绍**

2016年09月30日 16:34:02 [林老师带你学编程](https://me.csdn.net/linzhiqiang0316) 阅读数：26235更多

个人分类： [SpringBoot](https://blog.csdn.net/linzhiqiang0316/article/category/6424831)

所属专栏： [SpringBoot](https://blog.csdn.net/column/details/19194.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/linzhiqiang0316/article/details/52711908

## 今天给大家介绍一下SpringBoot中Redis的set、map、list、value等基本操作的具体使用方法

上一节中给大家介绍了如何在SpringBoot中搭建Redis缓存数据库，这一节就针对于Redis的具体操作进行介绍。

关于Redis的历史，以及目前的普及程度大家应该都比我还清楚吧。作为key-value的nosql数据库，Redis的功能还是特别强大的。

下面就来介绍一下Redis中一些常用的操作：

**1.存储set值的案例介绍：**

1. Set<String>set1=new HashSet<String>();
2. set1.add("set1");
3. set1.add("set2");
4. set1.add("set3");
5. redisTemplate.opsForSet().add("set1",set1);
6. Set<String> resultSet =redisTemplate.opsForSet().members("set1");
7. System.out.println("resultSet:"+resultSet);

**运行结果为：**

resultSet:[[set3, set2, set1]]

**2.存储map值的案例介绍：**

1. Map<String,String> map=new HashMap<String,String>();
2. map.put("key1","value1");
3. map.put("key2","value2");
4. map.put("key3","value3");
5. map.put("key4","value4");
6. map.put("key5","value5");
7. redisTemplate.opsForHash().putAll("map1",map);
8. Map<String,String> resultMap= redisTemplate.opsForHash().entries("map1");
9. List<String>reslutMapList=redisTemplate.opsForHash().values("map1");
10. Set<String>resultMapSet=redisTemplate.opsForHash().keys("map1");
11. String value=(String)redisTemplate.opsForHash().get("map1","key1");
12. System.out.println("value:"+value);
13. System.out.println("resultMapSet:"+resultMapSet);
14. System.out.println("resultMap:"+resultMap);
15. System.out.println("resulreslutMapListtMap:"+reslutMapList);

**运行结果为：**

1. value:value1
2. resultMapSet:[key1, key2, key5, key3, key4]
3. resultMap:{key3=value3, key2=value2, key1=value1, key5=value5, key4=value4}
4. resulreslutMapListtMap:[value1, value2, value5, value3, value4]

**3.存储list值的案例介绍：**

1. List<String> list1=new ArrayList<String>();
2. list1.add("a1");
3. list1.add("a2");
4. list1.add("a3");
6. List<String> list2=new ArrayList<String>();
7. list2.add("b1");
8. list2.add("b2");
9. list2.add("b3");
10. redisTemplate.opsForList().leftPush("listkey1",list1);
11. redisTemplate.opsForList().rightPush("listkey2",list2);
12. List<String> resultList1=(List<String>)redisTemplate.opsForList().leftPop("listkey1");
13. List<String> resultList2=(List<String>)redisTemplate.opsForList().rightPop("listkey2");
14. System.out.println("resultList1:"+resultList1);
15. System.out.println("resultList2:"+resultList2);

**运行结果为：**

1. resultList1:[a1, a2, a3]
2. resultList2:[b1, b2, b3]

这里需要解释一下：不管是leftPush还是rightPush都可以用leftPop或者rightPoP任意一种获取到其中的值，不过就是获取的遍历方向不一样。有学过数据结构的人都知道里面循环链表是可以前后遍历的，就和这里的场景是一样的。如果还有不懂的话可以去看看这部分的源代码，其实就是遍历方向不同，所以效率也不同。所以最好leftPush用leftPoP遍历，rightPush用rightPoP遍历。

**4.存储key-value值的案例介绍：**

1. System.out.println("缓存正在设置。。。。。。。。。");
2. redisTemplate.opsForValue().set("key1","value1");
3. redisTemplate.opsForValue().set("key2","value2");
4. redisTemplate.opsForValue().set("key3","value3");
5. redisTemplate.opsForValue().set("key4","value4");
6. System.out.println("缓存已经设置完毕。。。。。。。");
7. String result1=redisTemplate.opsForValue().get("key1").toString();
8. String result2=redisTemplate.opsForValue().get("key2").toString();
9. String result3=redisTemplate.opsForValue().get("key3").toString();
10. System.out.println("缓存结果为：result："+result1+" "+result2+" "+result3);

**运行结果为：**

1. 缓存正在设置。。。。。。。。。
2. 缓存已经设置完毕。。。。。。。
3. 缓存结果为：result：value1 value2 value3

**4.存储实体类的案例介绍：**

1. List<Blacklist> blackList=blacklistDao.findAll();
2. redisTemplate.opsForValue().set("blacklist",blackList);
3. List<Blacklist> resultBlackList= redisTemplate.opsForValue().get("blacklist");
4. for(Blacklist blacklist:resultBlackList){
5. System.out.println("ip:"+blacklist.getIp());
6. }

**实体类：**

1. package example.entity;
2. import javax.persistence.\*;
3. import java.io.Serializable;
4. import java.util.Date;
5. @Entity
6. @Table(name = "blacklist")
7. public class Blacklist implements Serializable
8. {
9. private static final long serialVersionUID = -1L;
10. @Id
11. @GeneratedValue(strategy = GenerationType.AUTO)
12. private int id;
14. @Column(name = "ip", nullable = true, length = 30)
15. private String ip;
17. @Temporal(TemporalType.TIMESTAMP)
18. private Date iptime; *// 日期类型，格式：yyyy-MM-dd HH:mm:ss*
20. public Blacklist() {
21. }
23. public Blacklist(String ip, Date iptime) {
24. this.ip = ip;
25. this.iptime = iptime;
26. }
28. public static long getSerialVersionUID() {
29. return serialVersionUID;
30. }
32. public int getId() {
33. return id;
34. }
36. public void setId(int id) {
37. this.id = id;
38. }
40. public String getIp() {
41. return ip;
42. }
44. public void setIp(String ip) {
45. this.ip = ip;
46. }
48. public Date getIptime() {
49. return iptime;
50. }
52. public void setIptime(Date iptime) {
53. this.iptime = iptime;
54. }
55. }

**运行结果为：**

1. ip:127.0.0.2
2. ip:127.0.0.3
3. ip:127.0.0.4
4. ip:127.0.0.5
5. ip:127.0.0.10
6. ip:127.0.0.1

这里需要注意的是：实体类一定要实现序列话，否者不管set的值为多少，最后存储的结果都是null的。

这样关于Redis一些常用的存储方法就介绍完了，下一节将会给大家介绍一下Redis如何实现数据库的同步。

其实也就是实现查询和更新分离，查询的时查询Redis缓存中的数据，更新操作的时候执行数据库操作，然后同步更新Redis中的相关键值。  
明天就国庆了，我在这里提前祝大家国庆快乐呀！我会在国庆后更新博客内容，谢谢大家的观看阅读！

**如果大家对文章有什么问题或者疑意之类的，可以加我订阅号在上面留言，订阅号上面我会定期更新最新博客。如果嫌麻烦可以直接加我wechat：lzqcode**