





俊俊的小熊饼干  
码龄4年 暂无认证

19  
原创

18万+  
周排名

14万+  
总排名

5万+  
访问

等级

676  
积分

26  
粉丝

45  
获赞

24  
评论

138  
收藏



私信

关注

搜博主文章

Q

热门文章

- Redis实现排行榜功能(实战)  23583
- springboot aop 自定义注解方式实现一套完善的日志记录（完整源码）  7282
- 正则表达式-linux路径匹配  5969
- 解决springcloud Eureka启动报错:Unable to start embedded Tomcat  5405
- npm设置和查看仓库源  3845

分类专栏

-  java 11篇
-  spring 1篇
-  js 4篇
-  vue 3篇
-  redis 2篇
-  linux 1篇

最新评论

- Redis实现排行榜功能(实战)  
俊俊的小熊饼干: redis是缓存，存的是热点数据哈，具体看自己的情况
- Redis实现排行榜功能(实战)  
Luhuaiwei: 那岂不是相当于就是替代了一个OrderBy的作用而已
- Redis实现排行榜功能(实战)  
Hai - W: 如果排行榜需要显示各学员的头像，头像地址存数据库吗 每更新一次排...
- Redis实现排行榜功能(实战)  
俊俊的小熊饼干: 历史数据可以批量加到redis,和新数据一样就行
- Redis实现排行榜功能(实战)  
俊俊的小熊饼干: 维护还是用数据库吧，用这个功能本身就是为了排行，并不是为...

最新文章

- 让线程按顺序执行8种方式
- Elasticsearch与Solr 选型
- BlockingQueue 阻塞队列实现异步事件

|       |     |       |    |
|-------|-----|-------|----|
| 2020年 | 1篇  | 2019年 | 3篇 |
| 2018年 | 16篇 |       |    |

目录

需求


分析

实现

- 一.redis sorts sets简介
- 二.springboot 中使用RedisTemplate
- 三.代码实现
- 四.归纳

结语

## Redis实现排行榜功能(实战)

原创 俊俊的小熊饼干 2018-10-08 18:04:01  23586  收藏 90 版权

分类专栏：[java](#) [redis](#) 文章标签：[redis](#) [RedisTemplate](#) [排行榜](#)

转载请注明出处：https://blog.csdn.net/m0\_37459380/article/details/82971525

### 需求

前段时间，做了一个世界杯竞猜积分排行榜。对世界杯64场球赛胜负平进行猜测，猜对+1分，错误+0分，一人一场只能猜一次。

- 展示前一百名列表。
- 展示个人排名(如：张三，您当前的排名106579)。

### 分析

一开始打算直接使用mysql数据库来做，遇到一个问题，每个人的分数都会变化,如何能够获取到个人的排名呢？数据库可以通过分数进行row\_num排序，但是这个方法需要进行全表扫描，当参与的人数达到10000的时候查询就非常慢了。

redis的排行榜功能就完美契合了这个需求。来看看我是怎么实现的吧。

### 实现

#### 一.redis sorts sets简介

Sorted Sets数据类型就像是set和hash的混合。与sets一样，Sorted Sets是唯一的，不重复的字符串组成。可以说Sorted Sets也是Sets的一种。

Sorted Sets是通过Skip List(跳跃表)和hash Table(哈希表)的双端口数据结构实现的，因此每次添加元素时，Redis都会执行O(log(N))操作。所以当我们要求排序的时候，Redis根本不需要做任何工作了，早已经全部排好序了。元素的分数可以随时更新。

#### 二.springboot 中使用RedisTemplate

本文主要通过redisTemplate来操作redis,当然也可以使用redis-client,看个人喜好。

我在本机开启了一个单点的redis，配置文件如下

```
1 server:
2   port: 9001
3 spring:
4   redis:
5     database: 0
6     url: redis://user:123@127.0.0.1:6379
7     host: 127.0.0.1
8     password: 123
9     port: 6379
10    ssl: false
11    timeout: 5000
```

Maven依赖引入如下

```
1 <parent>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-parent</artifactId>
4   <version>2.0.4.RELEASE</version>
5 </parent>
6
7 <dependencies>
8   <dependency>
9     <groupId>org.springframework.boot</groupId>
10    <artifactId>spring-boot-starter-web</artifactId>
11  </dependency>
12  <dependency>
13    <groupId>org.springframework.boot</groupId>
14    <artifactId>spring-boot-starter-data-redis</artifactId>
15  </dependency>
16  <dependency>
17    <groupId>org.springframework.boot</groupId>
18    <artifactId>spring-boot-starter-test</artifactId>
19  </dependency>
20 </dependencies>
```

#### 三.代码实现

1.注入redis，将key声明为常量SCORE\_RANK

```
1 @Autowired
2 private StringRedisTemplate redisTemplate;
3
4 public static final String SCORE_RANK = "score_rank";
```

2.新增默认排行数据

```
1 /**
2  * 批量新增
3  */
4 @Test
5 public void batchAdd() {
6     Set<ZSetOperations.TypedTuple<String>> tuples = new HashSet<>();
7     long start = System.currentTimeMillis();
8     for (int i = 0; i < 100000; i++) {
9         DefaultTypedTuple<String> tuple = new DefaultTypedTuple<>("张三" + i, 10);
10        tuples.add(tuple);
11    }
12    System.out.println("循环时间:" + (System.currentTimeMillis() - start));
13    Long num = redisTemplate.opsForZSet().add(SCORE_RANK, tuples);
14    System.out.println("批量新增时间:" + (System.currentTimeMillis() - start));
15    System.out.println("受影响行数: " + num);
16 }
```

```
1 //输出
2 循环时间:56
3 批量新增时间:1015
4 受影响行数: 100000
```

3.获取前10名(根据分数倒序)

```
1 /**
2  * 获取排行列表
3  */
4 @Test
5 public void list() {
6
7
8     Set<String> range = redisTemplate.opsForZSet().reverseRange(SCORE_RANK, 0,
9     System.out.println("获取到的排行列表:" + JSON.toJSONString(range));
10    Set<ZSetOperations.TypedTuple<String>> rangeWithScores = redisTemplate.opsF
11    System.out.println("获取到的排行和分数列表:" + JSON.toJSONString(rangeWithScore
12 }
```

```
1 //输出
2 获取到的排行列表:["张三99999", "张三99998", "张三99997", "张三99996", "张三99995", "张三99994",
3 获取到的排行和分数列表:[{"score":100000.0, "value": "张三99999"}, {"score":99999.0, "value":
```

4.新增李四的分数



举报

```
1  /**
2   * 单个新增
3   */
4  @Test
5  public void add() {
6      redisTemplate.opsForZSet().add(SCORE_RANK, "李四", 8899);
7  }
```

5.获取李四单人的排行

```
1  /**
2   * 获取单个的排行
3   */
4  @Test
5  public void find(){
6      Long rankNum = redisTemplate.opsForZSet().reverseRank(SCORE_RANK, "李四");
7      System.out.println("李四的个人排名: " + rankNum);
8
9      Double score = redisTemplate.opsForZSet().score(SCORE_RANK, "李四");
10     System.out.println("李四的分数: " + score);
11 }
```

```
1 //输出
2 李四的个人排名: 91101
3 李四的分数:8899.0
```

6.统计分数之间有多少人

```
1  /**
2   * 统计两个分数之间的人数
3   */
4  @Test
5  public void count(){
6      Long count = redisTemplate.opsForZSet().count(SCORE_RANK, 8001, 9000);
7      System.out.println("统计8001-9000之间的人数:" + count);
8  }
```

```
1 //输出
2 统计8001-9000之间的人数:1001
```

7.获取集合的基数(数量大小)

```
1  /**
2   * 获取整个集合的基数(数量大小)
3   */
4  @Test
5  public void zCard(){
6      Long aLong = redisTemplate.opsForZSet().zCard(SCORE_RANK);
7      System.out.println("集合的基数为: " + aLong);
8  }
```

```
1 //输出
2 集合的基数为: 100001
```

8.使用加法操作分数

```
1  /**
2   * 使用加法操作分数
3   */
4  @Test
5  public void incrementScore(){
6      Double score = redisTemplate.opsForZSet().incrementScore(SCORE_RANK, "李四",
7      System.out.println("李四分数+1000后: " + score);
8  }
```

```
1 //输出
2 李四分数+1000后: 9899.0
```

四.归纳

在以上测试类中我们使用了redis的那些功能呢？在以上的例子中我们使用了单个新增，批量新增，获取前十，获取单人排名这些操作，但是redisTemplate还提供了更多的方法。

新增or更新

有三种方式，一种是单个，一种是批量，对分数使用加法(如果不存在，则从0开始加)。

```
1 //单个新增or更新
2 Boolean add(K key, V value, double score);
3 //批量新增or更新
4 Long add(K key, Set<TypedTuple<V>> tuples);
5 //使用加法操作分数
6 Double incrementScore(K key, V value, double delta);
```

删除

删除提供了三种方式：通过key/values删除，通过排名区间删除，通过分数区间删除。

```
1 //通过key/value删除
2 Long remove(K key, Object... values);
3
4 //通过排名区间删除
5 Long removeRange(K key, long start, long end);
6
7 //通过分数区间删除
8 Long removeRangeByScore(K key, double min, double max);
```

查

1.列表查询:分为两大类，正序和逆序。以下只列表正序的，逆序的只需在方法前加上reverse即可：

//通过排名区间获取列表值集合

```
1 Set<V> range(K key, long start, long end);
2
3 //通过排名区间获取列表值和分数集合
4 Set<TypedTuple<V>> rangeWithScores(K key, long start, long end);
5
6 //通过分数区间获取列表值集合
7 Set<V> rangeByScore(K key, double min, double max);
8
9 //通过分数区间获取列表值和分数集合
10 Set<TypedTup<V>> rangeByScoreWithScores(K key, double min, double max);
11
12 //通过Range对象筛选再获取集合排行
13 Set<V> rangeByLex(K key, Range range);
14
15 //通过Range对象筛选再获取Limit数量的集合排行
16 Set<V> rangeByLex(K key, Range range, Limit limit);
```

2.单人查询

可获取单人排行，和通过key/value获取分数。以下只列表正序的，逆序的只需在方法前加上reverse即可：

```
1 //获取个人排行
2 Long rank(K key, Object o);
3
4 //获取个人分数
5 Double score(K key, Object o);
```





