

苍穹外卖项目涉及的新技术

1.JWT

2.Nginx

3.Apifox (设计阶段)

4.Swagger+Knife4j (测试阶段)

5.ThreadLocal

6.扩展SpringMVC的消息转换器

7.动态SQL

8.逻辑外键

9.阿里云OSS

10.Redis

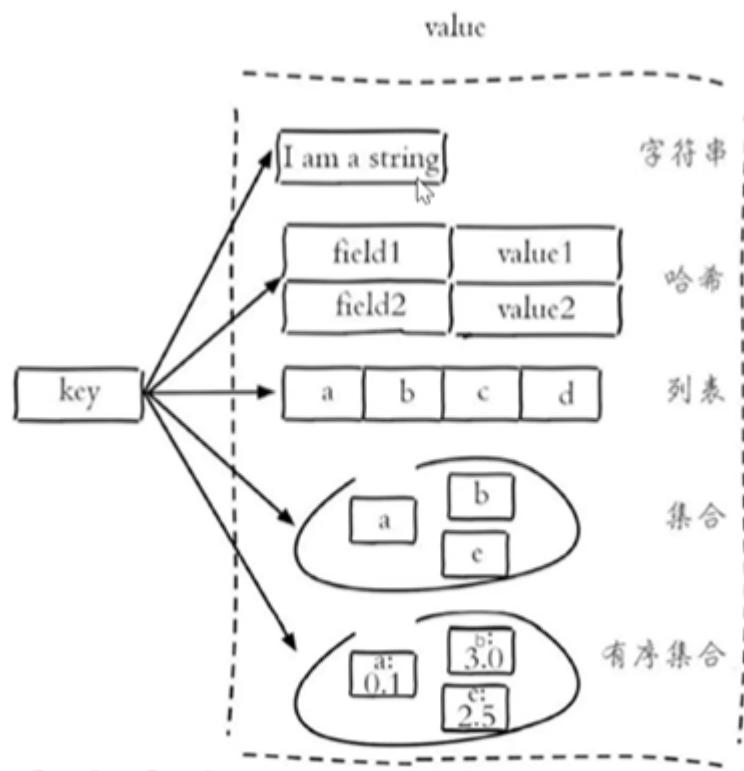
10.1 概述

- Redis是一个基于**内存**的key-value结构的数据库。
- **优点**: 基于内存存储，读写性能高。
- **适用场景**: 适合存储热点数据，比如热点商品、资讯、新闻。

10.2 Redis数据类型

Redis存储的是key-value结构的数据，其中key是字符串类型，value则有以下5种常用的数据类型：

- **字符串(string)**: 普通字符串，是Redis中最简单的数据类型。
- **哈希(hash)**: 也叫散列，类似于Java中的[HashMap](#)，适合存储Java对象。
- **列表(list)**: 按照插入顺序排序，可以有重复元素，类似于Java中的[LinkedList](#)。
- **集合(set)**: 无序集合，不能有重复元素，类似于Java中的[HashSet](#)。
- **有序集合(sorted set/zset)**: 集合中的每个元素关联着一个权重，按照权重升序排序，不能有重复元素，适合存储排行榜、投票……



10.3 Redis常用命令

10.3.1 字符串操作命令

命令	说明
<code>set key value</code>	设置指定key的value。
<code>get key</code>	获取指定key的value。
<code>setex key seconds value</code>	设置指定key的value，并将key的过期时间设置为seconds秒。
<code>setnx key value</code>	只有在key不存在时才设置key的value。

示例如下：

```
> connecting.....  
> myredis connected!  
> set name jack  
OK  
> get name  
jack  
> get abc  
null  
> setex code 30 123456  
OK  
> get code  
123456  
> get code  
null  
> setnx nickname jk  
1  
> setnx nickname jkk  
0  
> get nickname  
jk
```

10.3.2 哈希操作命令

命令	说明
<code>hset key field value</code>	为哈希表key设置指定field的value。
<code>hget key field</code>	获取存储在哈希表中指定field的value。
<code>hdel key field</code>	删除存储在哈希表中指定field的value。
<code>hkeys key</code>	获取哈希表中的所有field。
<code>hvals key</code>	获取哈希表中的所有value。

示例如下：

```
> connecting.....  
> myredis connected!  
> hset student name zsh  
1  
> hset student age 18  
1  
> hget student name  
zsh  
> hget student age  
18  
> hset student sex male  
1  
> hdel student sex  
1  
> hkeys student  
name  
age  
> hvals students  
  
> hvals student  
zsh  
18
```

10.3.3 列表操作命令（可重）

命令	说明
<code>lpush key value1 [value2] ... [valueN]</code>	将一个或多个value插入到列表key的头部。
<code>lrange key start stop</code>	获取列表指定范围内的元素。（索引从0开始，-1表示末尾元素）
<code>rpop key</code>	移除并获取列表最后一个元素。
<code>llen key</code>	获取列表长度。

示例如下：

```
> connecting.....  
> myredis connected!  
> lpush hobbies sing dance rap  
3  
> lpush hobbies basketball  
4  
> lrange hobbies 0 -1  
basketball  
rap  
dance  
sing
```

ID (Total: 4)	Value	操作
1	basketball	编辑 删除 </>
2	rap	编辑 删除 </>
3	dance	编辑 删除 </>
4	sing	编辑 删除 </>

```
> connecting.....  
> myredis connected!  
> lrange hobbies 0 -1  
basketball  
rap  
dance  
sing  
> lrange hobbies 1 2  
rap  
dance  
> rpop hobbies  
sing  
> lrange hobbies 0 -1  
basketball  
rap  
dance  
> llen hobbies  
3
```

10.3.4 集合操作命令（不可重）

命令	说明
<code>sadd key member1 [member2] ... [memberN]</code>	向集合key中添加一个或多个member。
<code>smembers key</code>	返回集合中的所有member。
<code>scard key</code>	获取集合的member数量。
<code>sinter key1 [key2] ... [keyN]</code>	返回所有给定集合的交集。
<code>sunion key1 [key2] ... [keyN]</code>	返回所有给定集合的并集。
<code>srem key member1 [member2] ... [memberN]</code>	删除集合中的一个或多个member。

示例如下：

```
> connecting.....  
> myredis connected!  
> sadd set1 a b c d  
4  
> sadd set1 a  
0  
> smembers set1  
d  
b  
a  
c  
> scard set1  
4  
> sadd set2 c d e f  
4  
> sinner set1 set2  
ERR unknown command `sinner`, with args beginning with: `set1`, `set2`  
> sinter set1 set2  
d  
c  
> sunion set1 set2  
d  
b  
f  
e  
a  
c  
> srem set1 a  
1  
> smembers set1  
c  
b  
d
```

10.3.5 有序集合操作命令（不可重）

命令	说明
<code>zadd key score1 member1 [score2 member2] ... [scoreN memberN]</code>	向有序集合key中添加一个或多个member，同时指定它们的score (double类型)。
<code>zrange key start stop [withscores]</code>	返回有序集合中指定区间内的member，可以顺带返回score。(索引从0开始，-1表示末尾元素)
<code>zincrby key increment member</code>	对有序集合中的指定member的score加上增量increment。
<code>zrem key member1 [member2] ... [memberN]</code>	移除有序集合中的一个或多个member。

示例如下：

```
> connecting.....  
> myredis connected!  
> zadd zset1 10.0 a 10.5 b  
2  
> zadd zset1 10.2 c  
1  
> zrange zset1 0 -1  
a  
c  
b  
> zrange zset1 0 -1 withscores  
a  
10  
c  
10.199999999999999  
b  
10.5  
> ZINCRBY zset1 1.0 a  
11  
> zrange zset1 0 -1 withscores  
c  
10.199999999999999  
b  
10.5  
a  
11  
> zadd zset1 11.5 d 12.0 e  
2  
> zrange zset1 0 -1 withscores  
c  
10.199999999999999  
b  
10.5  
a  
11  
d  
11.5  
e  
12  
> zrem zset1 b  
1  
> zrange zset1 0 -1 withscores  
c  
10.199999999999999  
a  
11  
d
```

10.3.6 通用命令

命令	说明
keys pattern	查找所有符合指定模式pattern的key。
exists key	检查指定key是否存在。
type key	返回key存储的value的类型。
del key	若key存在则删除该key。

示例如下：

```
> connecting.....  
> myredis connected!  
> keys *  
set2  
name  
zset1  
student  
hobbies  
nickname  
set1  
> keys set*  
set2  
set1  
> keys *1  
zset1  
set1  
> exists name  
1  
> exits age  
ERR unknown command `exits`, with args beginning with: `age`,  
> exists age  
0  
> type student  
hash  
> del nickname  
1  
> keys *  
set2  
name  
zset1  
student  
hobbies  
set1
```

10.4 在Java中操作Redis

10.4.1 Redis的Java客户端

Redis的主流Java客户端有以下几个：

- Jedis
- Lettuce
- Spring Data Redis

其中，Spring Data Redis是Spring的一部分，对Redis底层开发包进行了高度封装。

10.4.2 Spring Data Redis

1. 导入Spring Data Redis的maven坐标：

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-data-redis</artifactId>
4 </dependency>
```

2. 配置Redis数据源：

```
1 spring:
2   redis:
3     # 无密码
4     host:
5     port:
6     database:
```

3. 编写配置类，创建RedisTemplate对象：

```
1 /**
2  * Redis配置类
3 */
```

```
4 @Configuration
5 @Slf4j
6 public class RedisConfig {
7
8     @Bean
9     public RedisTemplate redisTemplate(RedisConnectionFactory
factory){
10         log.info("开始创建Redis模板对象...");  

11         RedisTemplate redisTemplate=new RedisTemplate();  

12
13         // 设置Redis的连接工厂对象
14         redisTemplate.setConnectionFactory(factory);
15         // 设置Redis key的序列化器
16         redisTemplate.setKeySerializer(new
StringRedisSerializer());
17
18         return redisTemplate;
19     }
20 }
```

4. 通过RedisTemplate对象操作Redis:

```
1 @SpringBootTest
2 public class SpringDataRedisTest {
3
4     @Autowired
5     private RedisTemplate redisTemplate;
6
7     @Test
8     public void testRedisTemplate(){
9         System.out.println(redisTemplate);
10    }
11
12
13     // 操作字符串类型的数据
14     @Test
15     public void testString(){
16         ValueOperations ops = redisTemplate.opsForValue();
17
18         // set
19         ops.set("city","Swatow");
20
21         // get
22         String city = (String) ops.get("city");
```

```
23     System.out.println("city: "+city);
24
25     // setex
26     ops.set("code","123456",30, TimeUnit.SECONDS);
27
28     // setnx
29     ops.setIfAbsent("lock","1");
30     ops.setIfAbsent("lock","2");// 设置失败
31
32 }
33
34
35 // 操作哈希类型的数据
36 @Test
37 public void testHash(){
38     HashOperations ops = redisTemplate.opsForHash();
39
40     // hset
41     Object key="student";
42     ops.put(key,"name","zjl");
43     ops.put(key,"age","99");
44     ops.put(key,"sex","male");
45
46     // hget
47     String name = (String) ops.get(key, "name");
48     System.out.println("name: "+name);
49
50     // hdel
51     ops.delete(key,"age");
52
53     // hkeys
54     System.out.println(ops.keys(key));
55
56     // hvals
57     System.out.println(ops.values(key));
58 }
59 }
```

11.HttpClient

12.SpringCache

13.冗余字段

14.SpringTask

15. WebSocket