Redis 快速入门

Redis 基础

• 什么是Redis?

Redis 是一个基于 内存 的 key-value 结构数据库。

- 基于内存存储，读写性能高
- 适合存储热点数据（热点商品、资讯、新闻）
- 企业应用广泛

Redis 入门

• Redis 简介

Redis 是一个开源的内存中的数据结构存储系统，它可以用作：数据库、缓存和消息中间件。

官网：https://redis.io

Redis 是用C语言开发的一个开源的高性能键值对(key-value)数据库，官方提供的数据是可以达到100000+的QPS（每秒内查询次数）。它存储的value类型比较丰富，也被称为结构化的NoSql数据库。

NoSql（Not Only SQL），不仅仅是SQL，泛指 非关系型数据库 。NoSql数据库并不是要取代关系型数据库，而是关系型数据库的补充。

- 关系型数据库（RDBMS）
    - Mysql
    - Oracle
    - DB2
    - SQLServer

- 非关系型数据库（NoSql）
    - Redis
    - Mongo db
    - MemCached

Redis 应用场景：

- 缓存
- 任务队列
- 消息队列
- 分布式锁

Redis 下载与安装

Redis 安装包分为Windows版和Linux版：

- Windows版下载地址：https://github.com/microsoftarchive/redis/releases
- Windows版下载地址：https://github.com/tporadowski/redis/releases
- Linux版下载地址：https://download.redis.io/releases/

Redis 服务启动与停止

Linux 中Redis服务启动，可以使用 `redis-server`，默认端口号是6379

`Ctrl+C`停止Redis服务

Windows系统中启动Redis，直接双击redis-server.exe即可启动Redis服务，Redis默认服务端口号为6379

`Ctrl+C`停止Redis服务

数据类型

• 介绍

Redis存储的是key-value结构的数据，其中key是字符串类型。

• Redis 5种常用类型

value有5种常用的数据类型：

- 字符串 string——普通字符串，常用
- 哈希 hash——适合存储对象
- 列表 list——按照插入顺序排序，可以有重复元素
- 集合 set——无序集合，没有重复元素
- 有序集合 sorted set——有序集合，没有重复元素

常用命令

• 字符串 string 操作命令

```
SET key value           #设置指定key的值
Get key                 #获取指定key的值
SETEX key seconds value #设置指定key的值，并将key的过期时间设为seconds秒
SETNX key value         #只有在key不存在时设置key的值
```

更多命令可以参考Redis中文网：https://www.redis.net.cn

• 哈希 hash 操作命令

```
HSET key field value #将哈希表key中的字段field的值设置为value
HGET key field       #获取存储在哈希表中指定字段的值
HDEL key field       #删除存储在哈希表中的指定字段
HKEYS key            #获取哈希表中所有字段
HVALS key            #获取哈希表中所有值
HGETALL key          #获取在哈希表中指定key的所有字段和值
```

• 列表 list 操作命令

```
LPUSH key value1 [value2] #将一个或多个值插入到列表头部
LRANGE key start stop     #获取列表指定范围内的元素
RPOP key                  #移除并获取列表最后一个元素
LLEN key                  #获取列表长度
BRPOP key1 [key2] timeout #移除并获取列表的最后一个元素，如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止
```

- **集合 set 操作命令**

```
SADD key member1 [member2] #向集合添加一个或多个成员
SMEMBERS key               #返回集合中的所有成员
SCARD key                  #获取集合的成员数
SINTER key1 [key2]         #返回给定所有集合的交集
SUNION key1 [key2]         #返回给定所有集合的并集
SDIFF key1 [key2]          #返回给定所有集合的差集
SREM key member1 [member2] #移除集合中一个或多个成员
```

- **有序集合 sorted set 操作命令**

Redis sorted set 有序集合是 string 类型元素的集合，且不允许重复的成员。每个元素都会关联一个 double 类型的分数（score）。Redis 正是通过分数来为集合中的成员进行从小到大排序。有序集合的成员是唯一的，但分数却可以重复。

```
ZADD key score1 member1 [score2 member2] #向有序集合添加一个或多个成员，或者更新已存在成员的分数
ZRANGE key start stop [WITHSCORES]       #通过索引区间返回有序集合中指定区间内的成员
ZINCRBY key increment member             #有序集合中对指定成员的分数加上增量 increment
ZREM key member [member ... ]            #移除有序集合中的一个或多个成员
```

- **通用命令**

```
KEYS pattern #查找所有符合给定模式 (pattern) 的key
EXISTS key   #检查给定key是否存在
TYPE key     #返回key所存储的值的类型
TTL key      #返回给定key的剩余生存时间 (TTL, time to live) , 以秒为单位
DEL key      #该命令用于在key存在时删除key
```

在 Java 中操作 Redis

介绍

Redis 的 Java 客户端很多，官方推荐的有三种：

- **jedis**
- Lettuce
- Redission

Spring 对 Redis 客户端进行了整合，提供了 **Spring Data Redis**，在 Spring Boot 项目中还提供了对应的 Starter，即 `spring-boot-starter-data-redis`

jedis

jedis 的 maven 坐标：

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>2.8.0</version>
</dependency>
```

使用 jedis 操作 Redis 的步骤：

1. 获取连接
2. 执行操作
3. 关闭连接

```
import org.junit.Test;
import redis.clients.jedis.Jedis;

import java.util.Set;

/**
 * 使用Jedis操作Redis
 */
public class JedisTest {

    @Test
    public void testRedis(){
        //1 获取连接
        Jedis jedis = new Jedis("localhost", 6379);

        //2 执行具体的操作
        jedis.set("username", "xiaoming");

        String value = jedis.get("username");
        System.out.println(value);
```

```
21
22          //jedis.del("username");
23
24          jedis.hset("myhash", "addr","bj");
25          String hValue = jedis.hget("myhash", "addr");
26          System.out.println(hValue);
27
28          Set<String> keys = jedis.keys("*");
29          for (String key : keys) {
30              System.out.println(key);
31          }
32
33          //3 关闭连接
34          jedis.close();
35      }
36  }
```

• Spring Data Redis

在 Spring Boot 项目中，可以使用 Spring Data Redis 来简化 Redis 操作，maven 坐标：

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

Spring Boot 中提供了一个高度封装的类：`RedisTemplate`，针对 jedis 客户端中大量 api 进行了归类封装，将同一类型操作封装为 `operation` 接口，具体分类如下：

- `ValueOperations` ：简单K-V操作
- `SetOperations` ：set类型数据操作
- `ZSetOperations` ：zset类型数据操作
- `HashOperations` ：针对map类型的数据操作
- `ListOperations` ：针对list类型的数据操作

App.java

```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class App {

    public static void main(String[] args) {
        SpringApplication.run(App.class,args);
    }
}
```

---

application.yml

```yaml
spring:
  application:
    name: springdataredis_demo
  #Redis相关配置
  redis:
    host: localhost
    port: 6379
    #password: 123456
    database: 0 #操作的是0号数据库
    jedis:
      #Redis连接池配置
      pool:
        max-active: 8 #最大连接数
        max-wait: 1ms #连接池最大阻塞等待时间
        max-idle: 4 #连接池中的最大空闲连接
        min-idle: 0 #连接池中的最小空闲连接
```

RedisConfig.java

```java
import org.springframework.cache.annotation.CachingConfigurerSupport;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.serializer.StringRedisSerializer;

```

```java
/**
 * Redis配置类
 */

@Configuration
public class RedisConfig extends CachingConfigurerSupport {

    @Bean
    public RedisTemplate<Object, Object> redisTemplate(RedisConnectionFactory connectionFactory) {

        RedisTemplate<Object, Object> redisTemplate = new RedisTemplate<>();

        //默认的Key序列化器为: JdkSerializationRedisSerializer
        redisTemplate.setKeySerializer(new StringRedisSerializer());
        redisTemplate.setHashKeySerializer(new StringRedisSerializer());

        redisTemplate.setConnectionFactory(connectionFactory);

        return redisTemplate;
    }
}
```

SpringDataRedisTest.java

```java
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.data.redis.connection.DataType;
import org.springframework.data.redis.core.*;
import org.springframework.test.context.junit4.SpringRunner;

import java.util.List;
import java.util.Set;
import java.util.concurrent.TimeUnit;

@SpringBootTest
@RunWith(SpringRunner.class)
public class SpringDataRedisTest {

    @Autowired
    private RedisTemplate redisTemplate;

    /**
```

```
21          * 操作String类型数据
22          */
23         @Test
24         public void testString(){
25             redisTemplate.opsForValue().set("city123", "beijing");
26 
27             String value = (String) redisTemplate.opsForValue().get("city123");
28             System.out.println(value);
29 
30             redisTemplate.opsForValue().set("key1", "value1", 10l,
    TimeUnit.SECONDS);
31 
32             Boolean aBoolean =
    redisTemplate.opsForValue().setIfAbsent("city1234", "nanjing");
33             System.out.println(aBoolean);
34         }
35 
36         /**
37          * 操作Hash类型数据
38          */
39         @Test
40         public void testHash(){
41             HashOperations hashOperations = redisTemplate.opsForHash();
42 
43             //存值
44             hashOperations.put("002", "name", "xiaoming");
45             hashOperations.put("002", "age", "20");
46             hashOperations.put("002", "address", "bj");
47 
48             //取值
49             String age = (String) hashOperations.get("002", "age");
50             System.out.println(age);
51 
52             //获得hash结构中的所有字段
53             Set keys = hashOperations.keys("002");
54             for (Object key : keys) {
55                 System.out.println(key);
56             }
57 
58             //获得hash结构中的所有值
59             List values = hashOperations.values("002");
60             for (Object value : values) {
61                 System.out.println(value);
62             }
63         }
64 
65         /**
66          * 操作List类型的数据
67          */
68         @Test
69         public void testList(){
```

```
70            ListOperations listOperations = redisTemplate.opsForList();
71
72            //存值
73            listOperations.leftPush("mylist", "a");
74            listOperations.leftPushAll("mylist", "b", "c", "d");
75
76            //取值
77            List<String> mylist = listOperations.range("mylist", 0, -1);
78            for (String value : mylist) {
79                System.out.println(value);
80            }
81
82            //获得列表长度 llen
83            Long size = listOperations.size("mylist");
84            int lSize = size.intValue();
85            for (int i = 0; i < lSize; i++) {
86                //出队列
87                String element = (String) listOperations.rightPop("mylist");
88                System.out.println(element);
89            }
90        }
91
92        /**
93         * 操作Set类型的数据
94         */
95        @Test
96        public void testSet(){
97            SetOperations setOperations = redisTemplate.opsForSet();
98
99            //存值
100           setOperations.add("myset", "a", "b", "c", "a");
101
102           //取值
103           Set<String> myset = setOperations.members("myset");
104           for (String o : myset) {
105               System.out.println(o);
106           }
107
108           //删除成员
109           setOperations.remove("myset", "a", "b");
110
111           //取值
112           myset = setOperations.members("myset");
113           for (String o : myset) {
114               System.out.println(o);
115           }
116
117       }
118
119       /**
120        * 操作ZSet类型的数据
```

```
     */
    @Test
    public void testZset(){
        ZSetOperations zSetOperations = redisTemplate.opsForZSet();

        //存值
        zSetOperations.add("myZset", "a", 10.0);
        zSetOperations.add("myZset", "b", 11.0);
        zSetOperations.add("myZset", "c", 12.0);
        zSetOperations.add("myZset", "a", 13.0);

        //取值
        Set<String> myZset = zSetOperations.range("myZset", 0, -1);
        for (String s : myZset) {
            System.out.println(s);
        }

        //修改分数
        zSetOperations.incrementScore("myZset", "b", 20.0);

        //取值
        myZset = zSetOperations.range("myZset", 0, -1);
        for (String s : myZset) {
            System.out.println(s);
        }

        //删除成员
        zSetOperations.remove("myZset", "a", "b");

        //取值
        myZset = zSetOperations.range("myZset", 0, -1);
        for (String s : myZset) {
            System.out.println(s);
        }
    }

    /**
     * 通用操作，针对不同的数据类型都可以操作
     */
    @Test
    public void testCommon(){
        //获取Redis中所有的key
        Set<String> keys = redisTemplate.keys("*");
        for (String key : keys) {
            System.out.println(key);
        }

        //判断某个key是否存在
        Boolean itcast = redisTemplate.hasKey("itcast");
        System.out.println(itcast);

```

```
172             //删除指定key
173             redisTemplate.delete("myZset");
174 
175             //获取指定key对应的value的数据类型
176             DataType dataType = redisTemplate.type("myset");
177             System.out.println(dataType.name());
178 
179         }
180     }
```