Redis快速入门

Redis的常见命令和客户端使用

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

今日课程介绍

键值数据库

Key	Value
id	1001
name	张三
age	21
1001	NoSql

```
{
  "id": 1001,
  "name": "张三",
  "age": 21
}
```

目录

Contents

学习目标

Learning Objectives

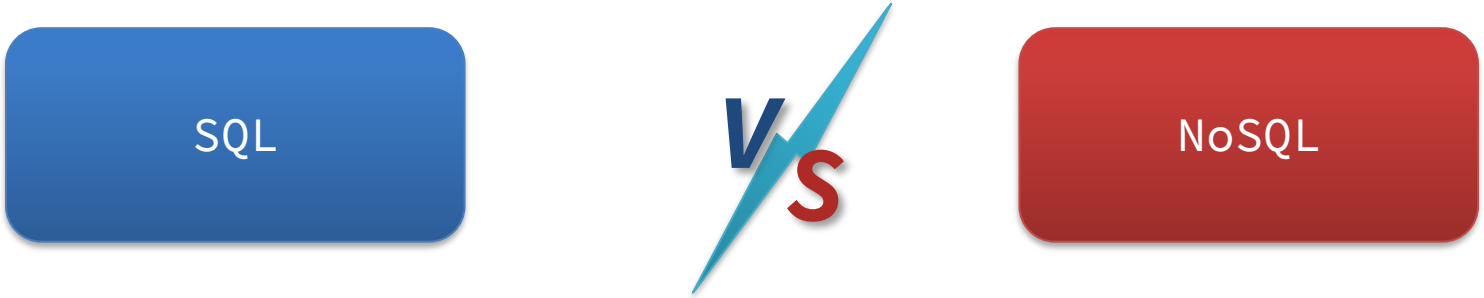
1. 知道NoSQL与SQL的差别
2. 熟悉Redis的常用5种数据结构
3. 熟悉Redis的常用命令
4. 熟练使用Jedis或SpringDataRedis

01 初识Redis

目录

Contents

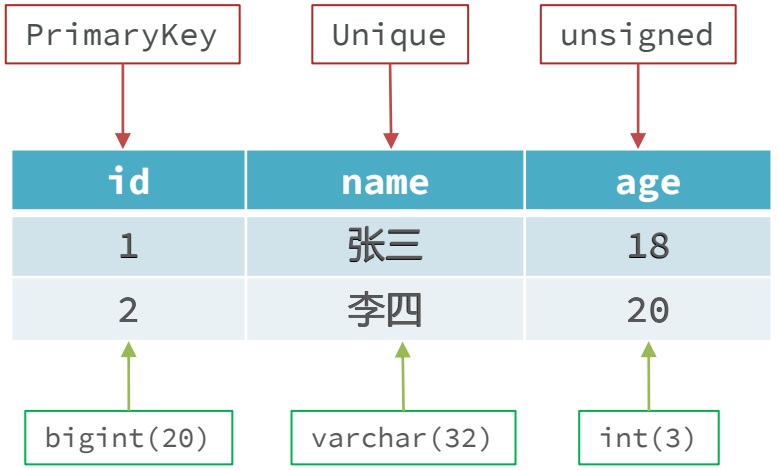
认识NoSQL



认识NoSQL

SQL VS NoSQL

#1 结构化(Structured)



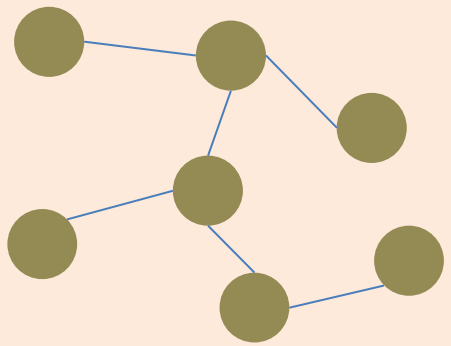
Key	Value
id	
name	
age	

Document

```
{
  id:
  name: "     ",
  age:
}
```

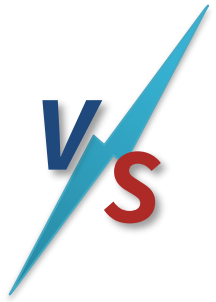
非结构化 #1

Graph



认识NoSQL

SQL



NoSQL

#1 结构化(Structured)

#2 关联的(Relational)

tb_user

id	name	age
1	张三	18
2	李四	20

tb_item

id	title	price
10	荣耀6	4999
20	小米11	3999

tb_order

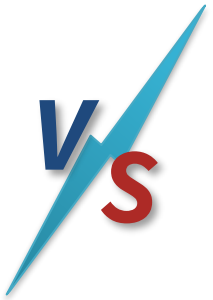
id	user_id	item_id
1	1	10
2	1	20

非结构化 #1

无关联的 #2

认识NoSQL

SQL



NoSQL

#1 结构化(Structured)

#2 关联的(Relational)

#3 SQL查询

```
SQL          SELECT id, name age FROM tb_user WHERE id = 1
Redis        get user:1
MongoDB      db.users.find({_id: 1})
elasticsearch GET http://localhost:9200/users/1
```

非结构化 #1

无关联的 #2

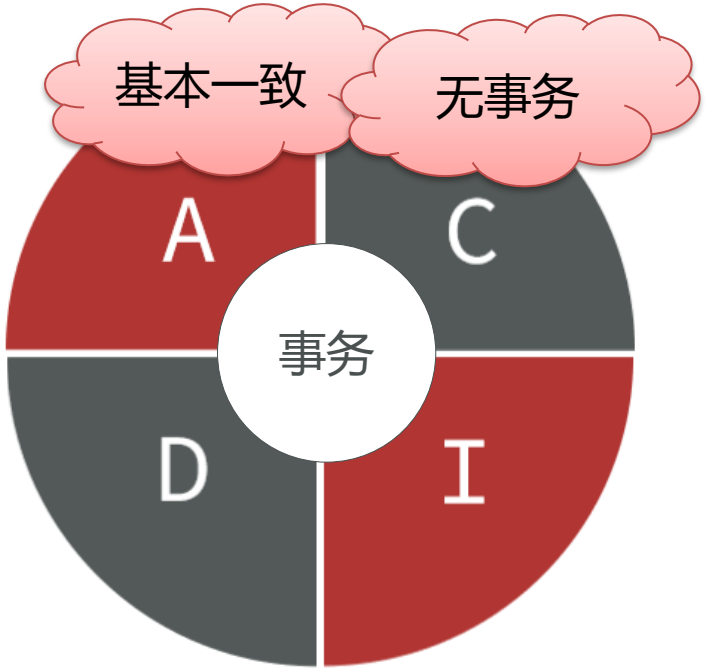
非SQL #3

认识NoSQL

SQL

VS

NoSQL



#1 结构化(Structured)

#2 关联的(Relational)

#3 SQL查询

#4 ACID

非结构化 #1

无关联的 #2

非SQL #3

BASE #4

认识NoSQL

	SQL	NoSQL
数据结构	结构化(Structu	
数据关联	关联的(Relational)	
查询方式	SQL查询	
事务特性	ACID	BASE

#1 键值类型（Redis）

#2 文档类型（MongoDB）

#3 列类型（HBase）

#4 Graph类型（Neo4j）

目录

Contents

认识Redis

Redis诞生于2009年全称是**Remote Dictionary Server**，远程词典服务器，是一个基于内存的键值型NoSQL数据库。

特征：

- 键值（key-value）型，value支持多种不同数据结构，功能丰富
- 单线程，每个命令具备原子性
- 低延迟，速度快（基于内存、IO多路复用、良好的编码）。
- 支持数据持久化
- 支持主从集群、分片集群
- 支持多语言客户端

目录

Contents

安装Redis

参考课前资料《Redis安装说明》

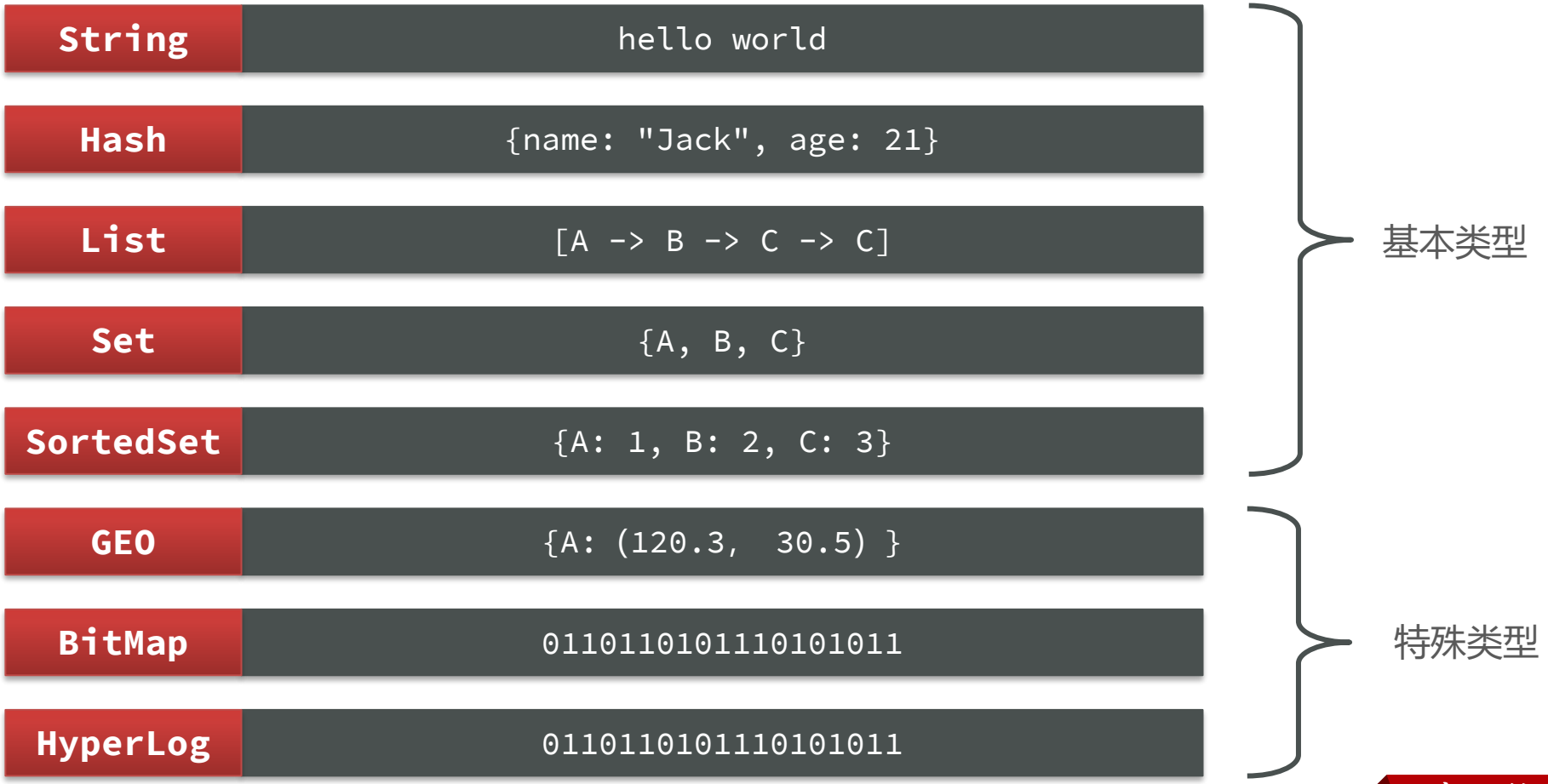
02 Redis常见命令

目录

Contents

Redis数据结构介绍

Redis是一个key-value的数据库，key一般是String类型，不过value的类型多种多样：



Redis数据结构介绍

Redis为了方便我们学习，将操作不同数据类型的命令也做了分组，在官网（ https://redis.io/commands ）可以查看到不同的命令：

```
127.0.0.1:6379> help
redis-cli 6.2.6
To get help about Redis commands type:
      "help @<group>" to get a list of commands in <group>
      "help <command>" for help on <command>
      "help <tab>" to get a list of possible help topics
      "quit" to exit

To set redis-cli preferences:
      ":set hints" enable online hints
      ":set nohints" disable online hints
Set your preferences in ~/.redisclirc
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> help @generic
```

查看通用的命令

目录

Contents

Redis通用命令

通用指令是部分数据类型的，都可以使用的指令，常见的有：

- KEYS：查看符合模板的所有key，不建议在生产环境设备上使用
- DEL：删除一个指定的key
- EXISTS：判断key是否存在
- EXPIRE：给一个key设置有效期，有效期到期时该key会被自动删除
- TTL：查看一个KEY的剩余有效期

通过help [command] 可以查看一个命令的具体用法，例如：

```
127.0.0.1:6379> help keys

  KEYS pattern
  summary: Find all keys matching the given pattern
  since: 1.0.0
  group: generic
```

目录

Contents

String类型

String类型，也就是字符串类型，是Redis中最简单的存储类型。

其value是字符串，不过根据字符串的格式不同，又可以分为3类：

- string：普通字符串
- int：整数类型，可以做自增、自减操作
- float：浮点类型，可以做自增、自减操作

不管是哪种格式，底层都是字节数组形式存储，只不过是编码方式不同。字符串类型的最大空间不能超过512m.

KEY	VALUE

String类型的常见命令

String的常见命令有：

- SET：添加或者修改已经存在的一个String类型的键值对
- GET：根据key获取String类型的value
- MSET：批量添加多个String类型的键值对
- MGET：根据多个key获取多个String类型的value
- INCR：让一个整型的key自增1
- INCRBY:让一个整型的key自增并指定步长，例如：incrby num 2 让num值自增2
- INCRBYFLOAT：让一个浮点类型的数字自增并指定步长
- SETNX：添加一个String类型的键值对，前提是这个key不存在，否则不执行
- SETEX：添加一个String类型的键值对，并且指定有效期

Redis没有类似MySQL中的Table的概念，我们该如何区分不同类型的key呢?

- 例如，需要存储用户、商品信息到redis，有一个用户id是1，有一个商品id恰好也是1

key的结构

Redis的key允许有多个单词形成层级结构，多个单词之间用':'隔开，格式如下：

```
项目名:业务名:类型:id
```

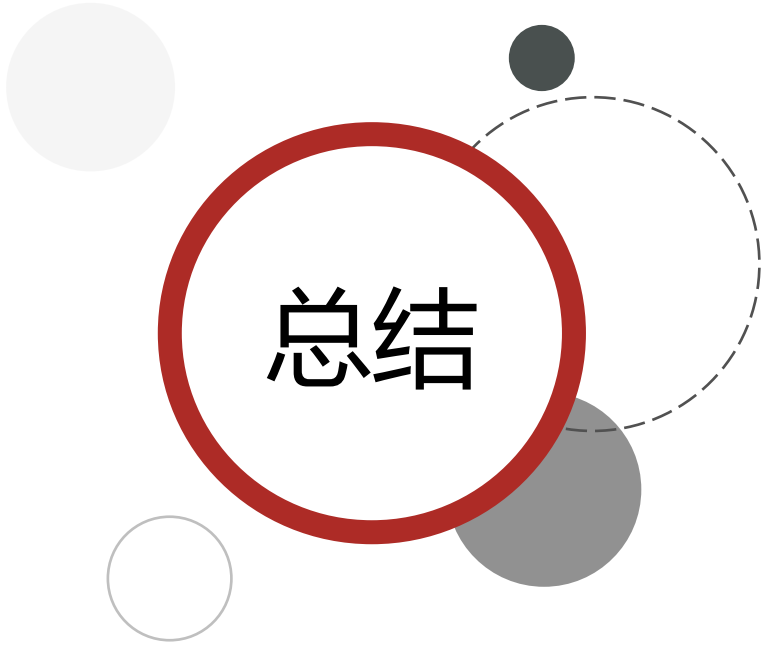
这个格式并非固定，也可以根据自己的需求来删除或添加词条。

例如我们的项目名称叫 heima，有user和product两种不同类型的数据，我们可以这样定义key：

- user相关的key：heima:user:1
- product相关的key：heima:product:1

如果Value是一个Java对象，例如一个User对象，则可以将对象序列化为JSON字符串后存储：

KEY	VALUE
heima:user:1	{"id":1, "name": "Jack", "age": 21}
heima:product:1	{"id":1, "name": "小米11", "price": 4999}



String类型的三种格式：

- 字符串
- int
- float

Redis的key的格式：

- [项目名]:[业务名]:[类型]:[id]

目录

Contents

Hash类型

Hash类型，也叫散列，其value是一个无序字典，类似于Java中的HashMap结构。

String结构是将对象序列化为JSON字符串后存储，当需要修改对象某个字段时很不方便：

KEY	VALUE
heima:user:1	{name:"Jack", age:21}
heima:user:2	{name:"Rose", age:18}

Hash结构可以将对象中的每个字段独立存储，可以针对单个字段做CRUD：

KEY	VALUE	
	field	value
heima:user:1	name	Jack
	age	21
heima:user:2	name	Rose
	age	18

Hash类型的常见命令

Hash的常见命令有：

- HSET key field value：添加或者修改hash类型key的field的值
- HGET key field：获取一个hash类型key的field的值
- HMSET：批量添加多个hash类型key的field的值
- HMGET：批量获取多个hash类型key的field的值
- HGETALL：获取一个hash类型的key中的所有的field和value
- HKEYS：获取一个hash类型的key中的所有的field
- HVALS：获取一个hash类型的key中的所有的value
- HINCRBY:让一个hash类型key的字段值自增并指定步长
- HSETNX：添加一个hash类型的key的field值，前提是这个field不存在，否则不执行

目录

Contents

List类型

Redis中的List类型与Java中的LinkedList类似，可以看做是一个双向链表结构。既可以支持正向检索和也可以支持反向检索。

特征也与LinkedList类似：

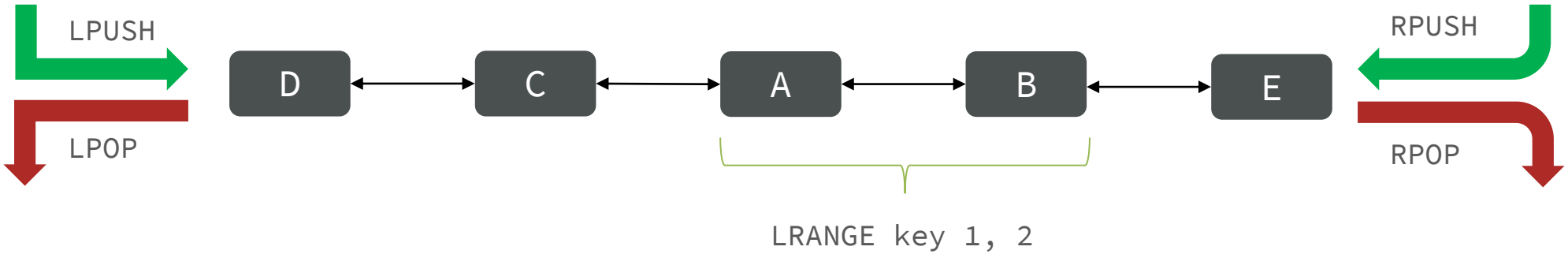
- 有序
- 元素可以重复
- 插入和删除快
- 查询速度一般

常用来存储一个有序数据，例如：朋友圈点赞列表，评论列表等。

List类型的常见命令

List的常见命令有：

- LPUSH key element ... ：向列表左侧插入一个或多个元素
- LPOP key：移除并返回列表左侧的第一个元素，没有则返回nil
- RPUSH key element ... ：向列表右侧插入一个或多个元素
- RPOP key：移除并返回列表右侧的第一个元素
- LRANGE key star end：返回一段角标范围内的所有元素
- BLPOP和BRPOP：与LPOP和RPOP类似，只不过在没有元素时等待指定时间，而不是直接返回nil



如何利用List结构模拟一个栈?

- 入口和出口在同一边

如何利用List结构模拟一个队列?

- 入口和出口在不同边

如何利用List结构模拟一个阻塞队列?

- 入口和出口在不同边
- 出队时采用BLPOP或BRPOP

目录

Contents

Set类型

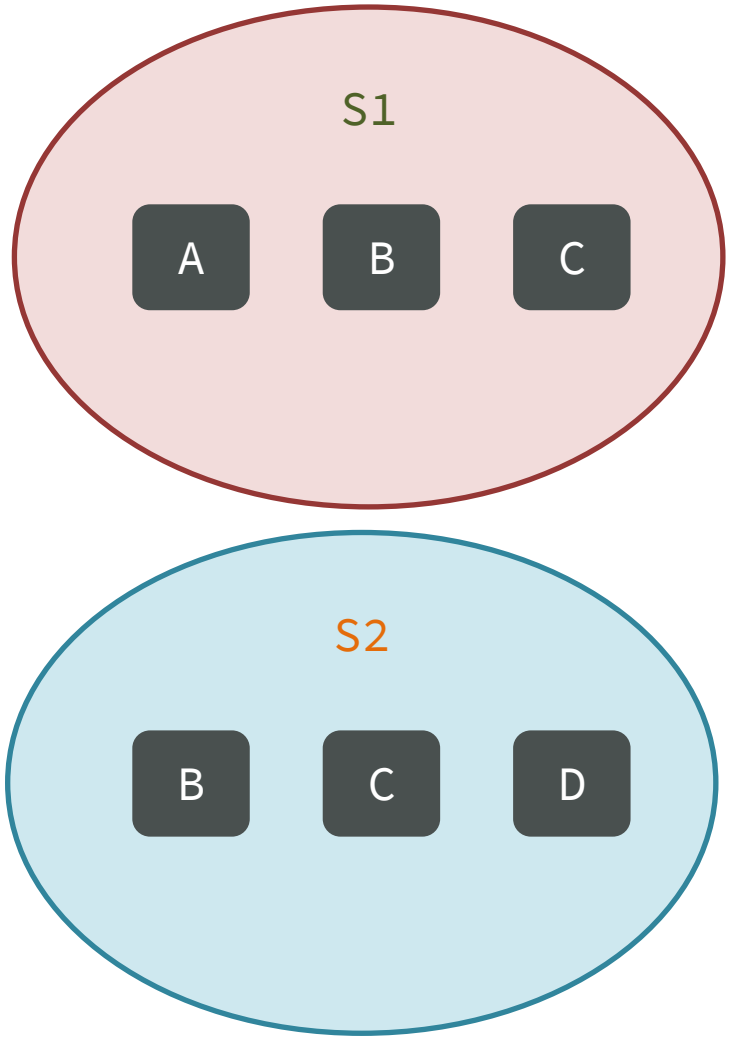
Redis的Set结构与Java中的HashSet类似，可以看做是一个value为null的HashMap。因为也是一个hash表，因此具备与HashSet类似的特征：

- 无序
- 元素不可重复
- 查找快
- 支持交集、并集、差集等功能

Set类型的常见命令

String的常见命令有：

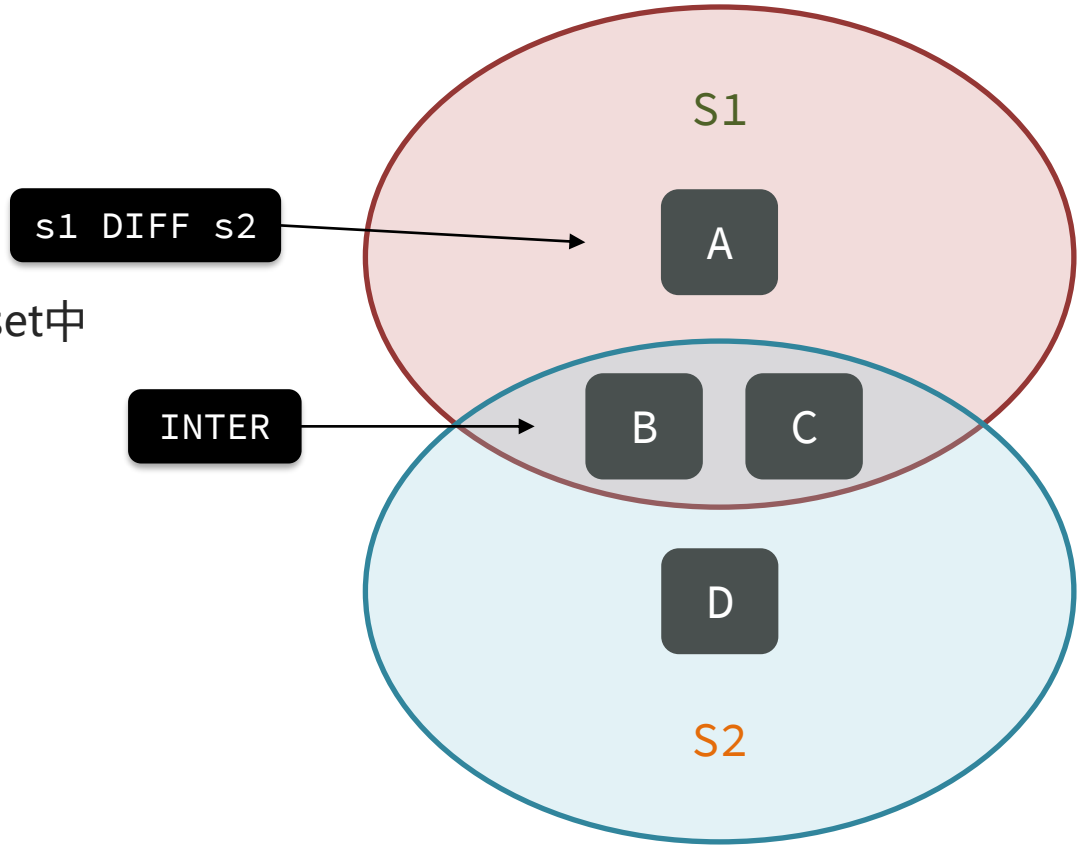
- SADD key member ... ： 向set中添加一个或多个元素
- SREM key member ... : 移除set中的指定元素
- SCARD key： 返回set中元素的个数
- SISMEMBER key member：判断一个元素是否存在于set中
- SMEMBERS：获取set中的所有元素
- SINTER key1 key2 ... ： 求key1与key2的交集



Set类型的常见命令

String的常见命令有：

- SADD key member ... ： 向set中添加一个或多个元素
- SREM key member ... : 移除set中的指定元素
- SCARD key： 返回set中元素的个数
- SISMEMBER key member：判断一个元素是否存在于set中
- SMEMBERS：获取set中的所有元素
- SINTER key1 key2 ... ： 求key1与key2的交集
- SDIFF key1 key2 ... ： 求key1与key2的差集
- SUNION key1 key2 ..： 求key1和key2的并集



案例 Set命令的练习

将下列数据用Redis的Set集合来存储：

- 张三的好友有：李四、王五、赵六
- 李四的好友有：王五、麻子、二狗

利用Set的命令实现下列功能：

- 计算张三的好友有几人
- 计算张三和李四有哪些共同好友
- 查询哪些人是张三的好友却不是李四的好友
- 查询张三和李四的好友总共有哪些人
- 判断李四是否是张三的好友
- 判断张三是否是李四的好友
- 将李四从张三的好友列表中移除

目录

Contents

SortedSet类型

Redis的SortedSet是一个可排序的set集合，与Java中的TreeSet有些类似，但底层数据结构却差别很大。SortedSet中的每一个元素都带有一个score属性，可以基于score属性对元素排序，底层的实现是一个跳表（SkipList）加 hash表。

SortedSet具备下列特性：

- 可排序
- 元素不重复
- 查询速度快

因为SortedSet的可排序特性，经常被用来实现排行榜这样的功能。

SortedSet类型的常见命令

SortedSet的常见命令有：

- ZADD key score member：添加一个或多个元素到sorted set ，如果已经存在则更新其score值
- ZREM key member：删除sorted set中的一个指定元素
- ZSCORE key member : 获取sorted set中的指定元素的score值
- ZRANK key member：获取sorted set 中的指定元素的排名
- ZCARD key：获取sorted set中的元素个数
- ZCOUNT key min max：统计score值在给定范围内的所有元素的个数
- ZINCRBY key increment member：让sorted set中的指定元素自增，步长为指定的increment值
- ZRANGE key min max：按照score排序后，获取指定排名范围内的元素
- ZRANGEBYSCORE key min max：按照score排序后，获取指定score范围内的元素
- ZDIFF、ZINTER、ZUNION：求差集、交集、并集

注意：所有的排名默认都是升序，如果要降序则在命令的Z后面添加REV即可

案例

SortedSet命令练习

将班级的下列学生得分存入Redis的SortedSet中：

Jack 85, Lucy 89, Rose 82, Tom 95, Jerry 78, Amy 92, Miles 76

- 并实现下列功能：
- 删除Tom同学
- 获取Amy同学的分数
- 获取Rose同学的排名
- 查询80分以下有几个学生
- 给Amy同学加2分
- 查出成绩前3名的同学
- 查出成绩80分以下的所有同学

03 Redis的Java客户端

Redis的Java客户端

在Redis官网中提供了各种语言的客户端，地址：https://redis.io/clients

Browse by language:

ActionScript	ActiveX/COM+	Bash	Boomi	C	C#
C++	Clojure	Common Lisp	Crystal	D	Dart
Delphi	Elixir	emacs lisp	Erlang	Fancy	gawk
GNU Prolog	Go	Haskell	Haxe	Io	Java
Julia	Lasso	Lua	Matlab	mruby	Nim
Node.js	Objective-C	OCaml	Pascal	Perl	PHP
PL/SQL	Prolog	Pure Data	Python	R	Racket
Rebol	Ruby	Rust	Scala	Scheme	Smalltalk
Swift	Tcl	VB	VCL	Xojo	Zig

Redis的Java客户端

Spring Data Redis

Java

Jedis	☺	★			以Redis命令作为方法名称，学习成本低，简单实用。但是Jedis实例是线程不安全的，多线程环境下需要基于连接池来使用
lettuce	☺	★	⌂		Lettuce是基于Netty实现的，支持同步、异步和响应式编程方式，并且是线程安全的。支持Redis的哨兵模式、集群模式和管道模式。
Redisson	☺	★			Redisson是一个基于Redis实现的分布式、可伸缩的Java数据结构集合。包含了诸如Map、Queue、Lock、 Semaphore、AtomicLong等强大功能
java-redis-client	☺				A very simple yet very complete java client in less than 200 lines with 0 dependencies.
vertx-redis-client	☺				The Vert.x Redis client provides an asynchronous API to interact with a Redis data-structure server. @pmlopes

目录

Contents

Jedis

Jedis的官网地址： https://github.com/redis/jedis，我们先来个快速入门：

1. 引入依赖：

```xml
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>3.7.0</version>
</dependency>
```

2. 建立连接

```java
private Jedis jedis;

@BeforeEach
void setUp() {
    // 建立连接
    jedis = new Jedis("192.168.150.101", 6379);
    // 设置密码
    jedis.auth("123321");
    // 选择库
    jedis.select(0);
}
```

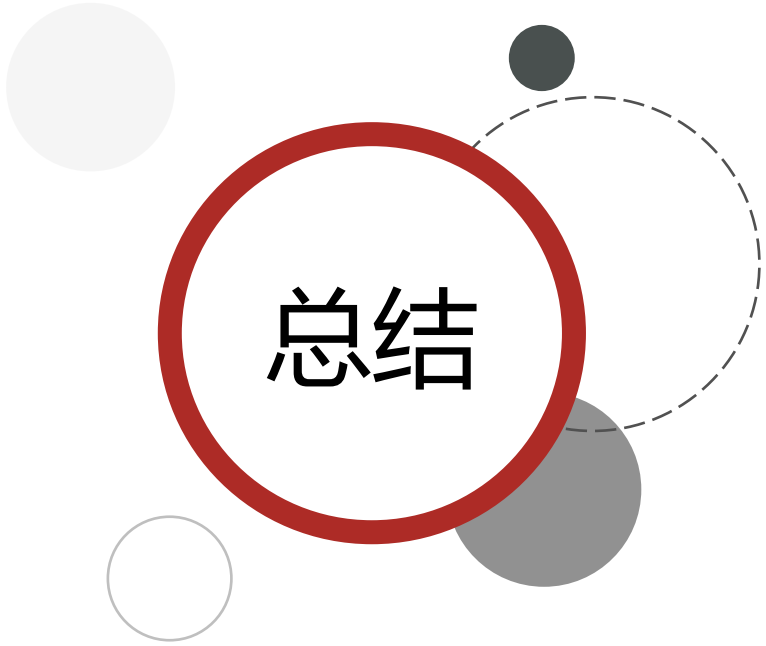
Jedis

3. 测试string

```
@Test
void testString() {
    // 插入数据，方法名称就是redis命令名称，非常简单
    String result = jedis.set("name", "张三");
    System.out.println("result = " + result);
    // 获取数据
    String name = jedis.get("name");
    System.out.println("name = " + name);
}
```

4. 释放资源

```
@AfterEach
void tearDown() {
    // 释放资源
    if (jedis != null) {
        jedis.close();
    }
}
```



Jedis使用的基本步骤：

1. 引入依赖
2. 创建Jedis对象，建立连接
3. 使用Jedis，方法名与Redis命令一致
4. 释放资源

Jedis连接池

Jedis本身是线程不安全的，并且频繁的创建和销毁连接会有性能损耗，因此我们推荐大家使用Jedis连接池代替Jedis的直连方式。

```
public class JedisConnectionFactory {
    private static final JedisPool jedisPool;

    static {
        JedisPoolConfig jedisPoolConfig = new JedisPoolConfig();
        // 最大连接
        jedisPoolConfig.setMaxTotal(8);
        // 最大空闲连接
        jedisPoolConfig.setMaxIdle(8);
        // 最小空闲连接
        jedisPoolConfig.setMinIdle(0);
        // 设置最长等待时间， ms
        jedisPoolConfig.setMaxWaitMillis(200);
        jedisPool = new JedisPool(jedisPoolConfig, "192.168.150.101", 6379,
                1000, "123321");
    }
    // 获取Jedis对象
    public static Jedis getJedis(){
        return jedisPool.getResource();
    }
}
```

练习

基于SpringBoot整合Jedis

SpringBoot已经成为企业开发的标配，你能不能基于SpringBoot来整合下Jedis的连接池呢?

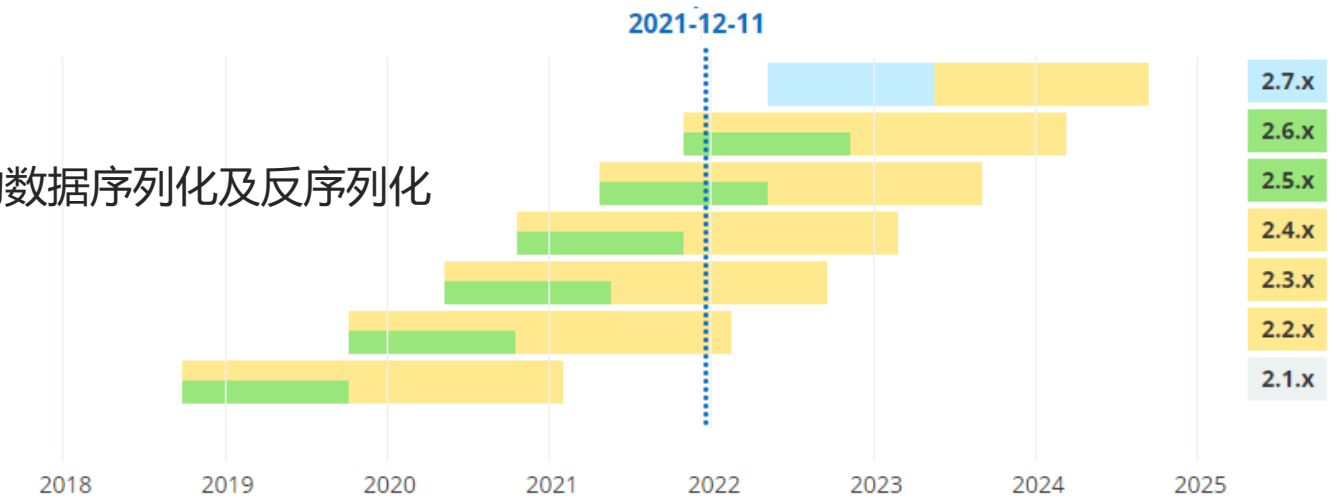
目录

Contents

SpringDataRedis

SpringData是Spring中数据操作的模块，包含对各种数据库的集成，其中对Redis的集成模块就叫做SpringDataRedis，官网地址：https://spring.io/projects/spring-data-redis

- 提供了对不同Redis客户端的整合（Lettuce和Jedis）
- 提供了RedisTemplate统一API来操作Redis
- 支持Redis的发布订阅模型
- 支持Redis哨兵和Redis集群
- 支持基于Lettuce的响应式编程
- 支持基于JDK、JSON、字符串、Spring对象的数据序列化及反序列化
- 支持基于Redis的JDKCollection实现



SpringDataRedis快速入门

SpringDataRedis中提供了RedisTemplate工具类，其中封装了各种对Redis的操作。并且将不同数据类型的操作API封装到了不同的类型中：

API	返回值类型	说明
redisTemplate.opsForValue()	ValueOperations	操作String类型数据
redisTemplate.opsForHash()	HashOperations	操作Hash类型数据
redisTemplate.opsForList()	ListOperations	操作List类型数据
redisTemplate.opsForSet()	SetOperations	操作Set类型数据
redisTemplate.opsForZSet()	ZSetOperations	操作SortedSet类型数据
redisTemplate		通用的命令

SpringDataRedis快速入门

SpringBoot已经提供了对SpringDataRedis的支持，使用非常简单：

1. 引入依赖

```
<!--Redis依赖-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
<!--连接池依赖-->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-pool2</artifactId>
</dependency>
```

SpringDataRedis快速入门

2. 配置文件

```yaml
spring:
  redis:
    host: 192.168.150.101
    port: 6379
    password: 123321
    lettuce:
      pool:
        max-active: 8 # 最大连接
        max-idle: 8 # 最大空闲连接
        min-idle: 0 # 最小空闲连接
        max-wait: 100 # 连接等待时间
```

SpringDataRedis快速入门

3. 注入RedisTemplate

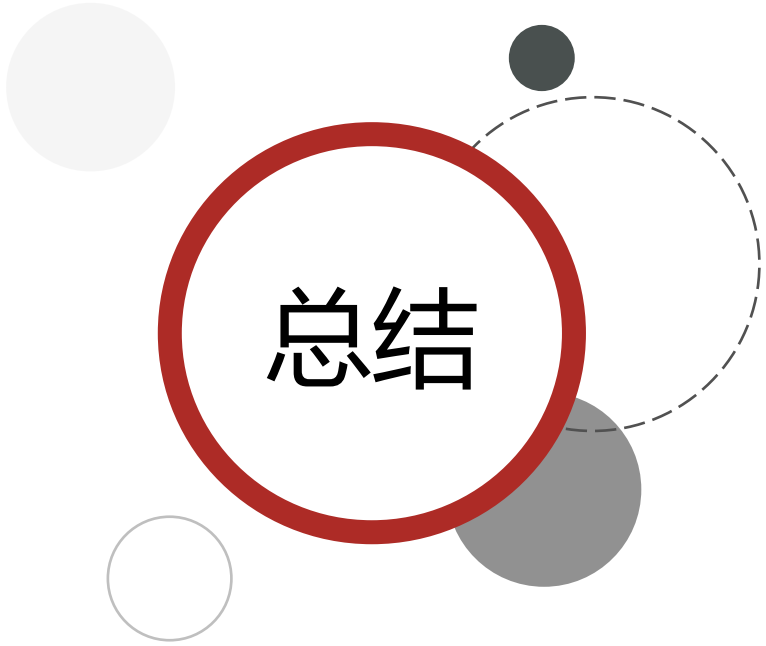
```
@Autowired
private RedisTemplate redisTemplate;
```

4. 编写测试

```
@SpringBootTest
public class RedisTest {

    @Autowired
    private RedisTemplate redisTemplate;

    @Test
    void testString() {
        // 插入一条string类型数据
        redisTemplate.opsForValue().set("name", "李四");
        // 读取一条string类型数据
        Object name = redisTemplate.opsForValue().get("name");
        System.out.println("name = " + name);
    }
}
```

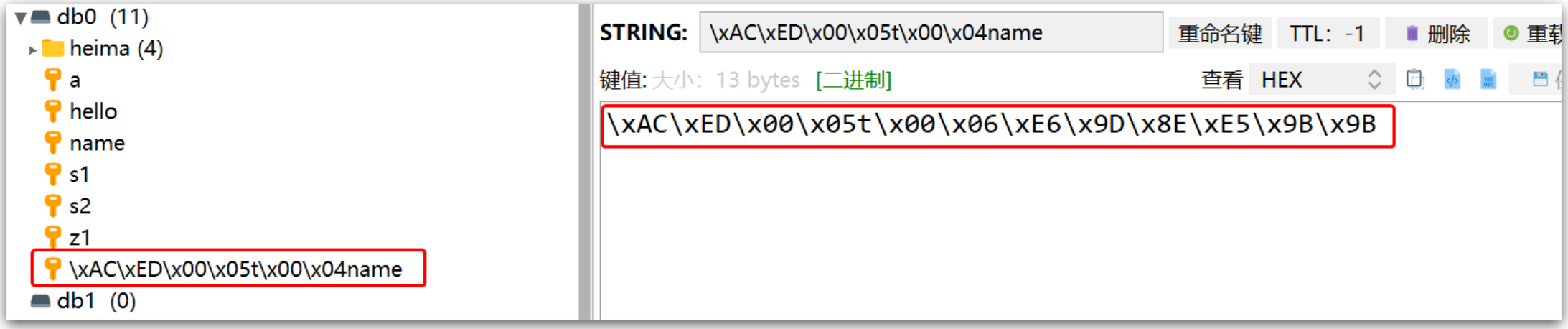


SpringDataRedis的使用步骤：

1. 引入spring-boot-starter-data-redis依赖
2. 在application.yml配置Redis信息
3. 注入RedisTemplate

SpringDataRedis的序列化方式

RedisTemplate可以接收任意Object作为值写入Redis，只不过写入前会把Object序列化为字节形式，默认是采用JDK序列化，得到的结果是这样的：



缺点：

- 可读性差
- 内存占用较大

SpringDataRedis的序列化方式

我们可以自定义RedisTemplate的序列化方式，代码如下：

```
@Bean
public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory redisConnectionFactory)
        throws UnknownHostException {
    // 创建Template
    RedisTemplate<String, Object> redisTemplate = new RedisTemplate<>();
    // 设置连接工厂
    redisTemplate.setConnectionFactory(redisConnectionFactory);
    // 设置序列化工具
    GenericJackson2JsonRedisSerializer jsonRedisSerializer =
                                            new GenericJackson2JsonRedisSerializer();
    // key和 hashKey采用 string序列化
    redisTemplate.setKeySerializer(RedisSerializer.string());
    redisTemplate.setHashKeySerializer(RedisSerializer.string());
    // value和 hashValue采用 JSON序列化
    redisTemplate.setValueSerializer(jsonRedisSerializer);
    redisTemplate.setHashValueSerializer(jsonRedisSerializer);
    return redisTemplate;
}
```

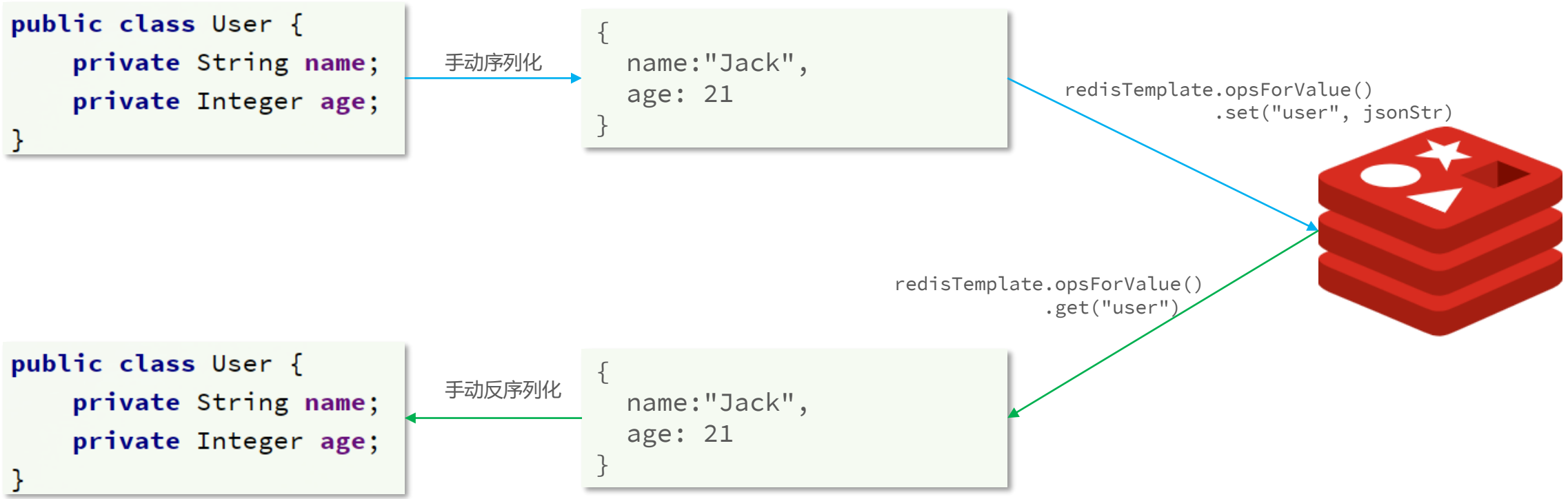
StringRedisTemplate

尽管JSON的序列化方式可以满足我们的需求，但依然存在一些问题，如图：

为了在反序列化时知道对象的类型，JSON序列化器会将类的class类型写入json结果中，存入Redis，会带来额外的内存开销。

StringRedisTemplate

为了节省内存空间，我们并不会使用JSON序列化器来处理value，而是统一使用String序列化器，要求只能存储String类型的key和value。当需要存储Java对象时，手动完成对象的序列化和反序列化。

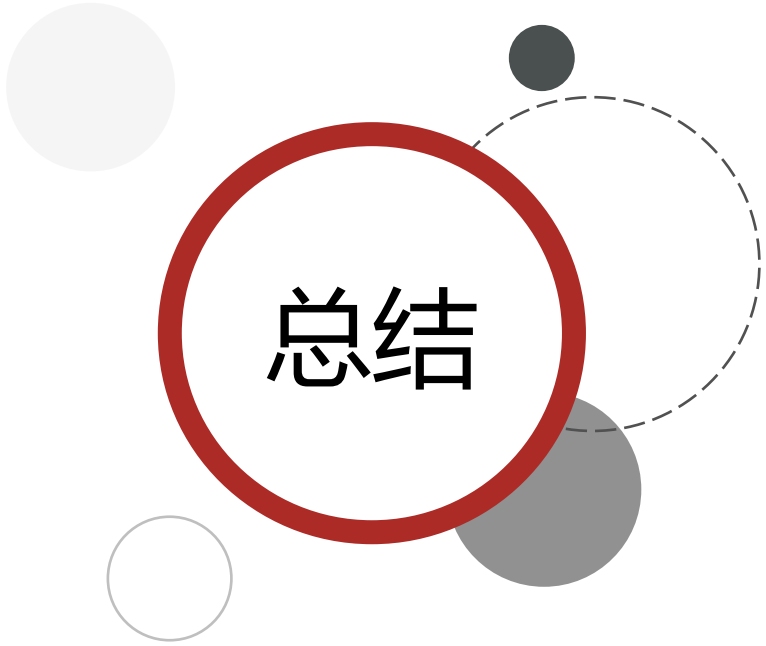


StringRedisTemplate

Spring默认提供了一个StringRedisTemplate类，它的key和value的序列化方式默认就是String方式。省去了我们自定义RedisTemplate的过程：

```
@Autowired
private StringRedisTemplate stringRedisTemplate;
// JSON工具
private static final ObjectMapper mapper = new ObjectMapper();
@Test
void testStringTemplate() throws JsonProcessingException {
    // 准备对象
    User user = new User("虎哥", 18);
    // 手动序列化
    String json = mapper.writeValueAsString(user);
    // 写入一条数据到redis
    stringRedisTemplate.opsForValue().set("user:200", json);

    // 读取数据
    String val = stringRedisTemplate.opsForValue().get("user:200");
    // 反序列化
    User user1 = mapper.readValue(val, User.class);
    System.out.println("user1 = " + user1);
}
```



RedisTemplate的两种序列化实践方案：

方案一：

1. 自定义RedisTemplate
2. 修改RedisTemplate的序列化器为GenericJackson2JsonRedisSerializer

方案二：

1. 使用StringRedisTemplate
2. 写入Redis时，手动把对象序列化为JSON
3. 读取Redis时，手动把读取到的JSON反序列化为对象

传智教育旗下高端IT教育品牌