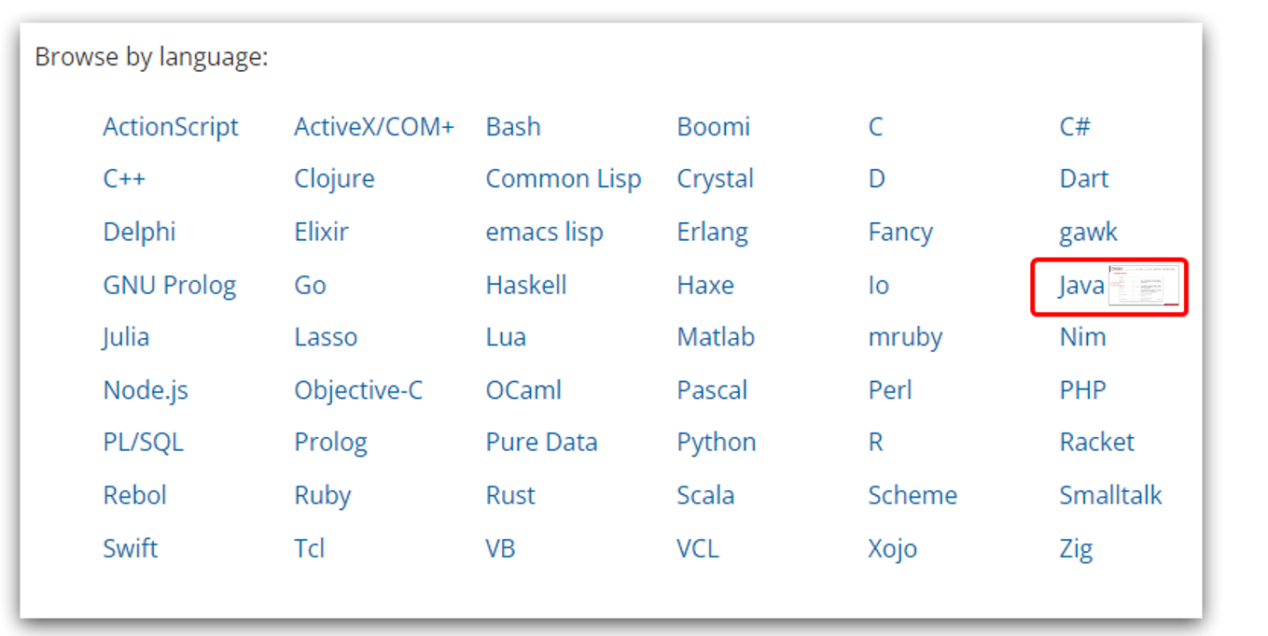
# Redis的Java客户端

经过前面的学习，我们已经基本了解了Redis五种常见的数据结构，以及每种结构的常见命令和命令基本用法。不过最终我们实现业务肯定是通过代码编码的方式来实现。

Redis官网给我们提供了各种编程语言的客户端供我们使用，当然我们作为Java程序员，关注的当然是Java语言的客户端。

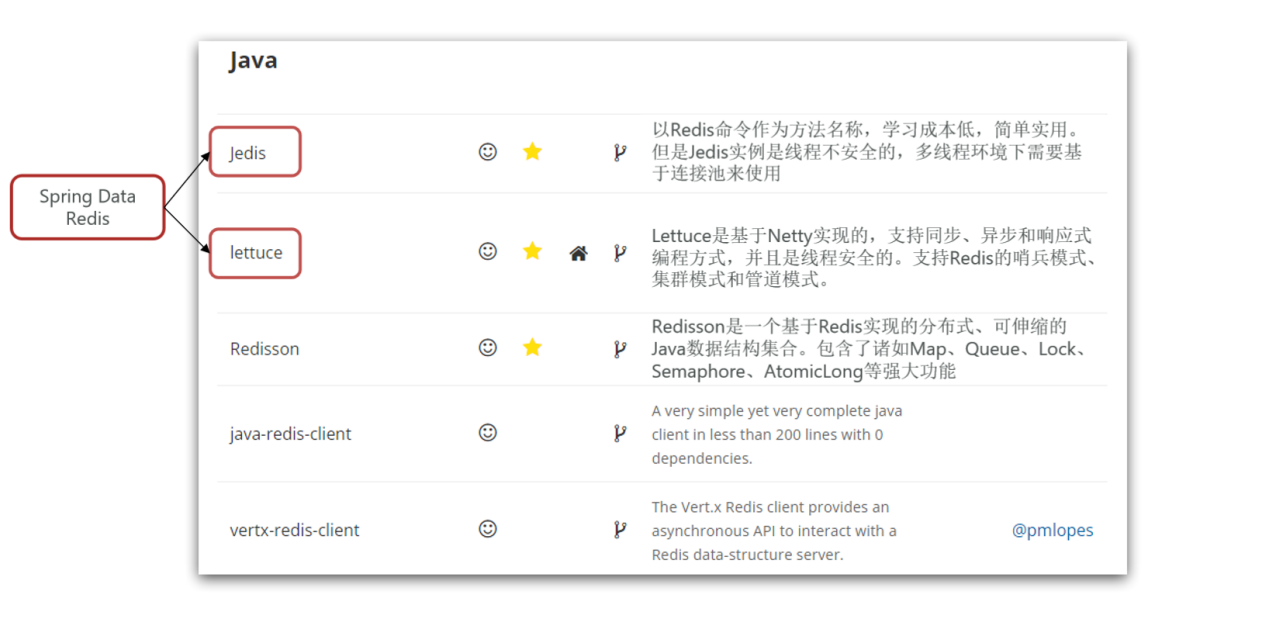


可以看到Java的Redis客户端有三个带有星，说明都是官方认可推荐使用的。那么我们到底应该使用哪一个客户端呢？

Jedis：其实就是Java Redis的组成单词，这个客户端最大的特点就是客户端方法都是以Redis命令作为方法命名的。学习成本低，基本上命令你会了，方法你以为就会了，简单实用，正因为此，Jedis刚推出就得到了广泛使用，可以认为在最开始是使用最多的客户端。包括现在都很多人在使用。但是Jedis有弊端，就是Jedis实例是线程不安全的，也就是说创建一个Jedis实例，多线程并发运行的时候是由线程安全问题的，所以在多线程使用时，必须为每一个线程创建独立的Jedis连接，那么就必须使用连接池的方式来使用！

Lettuce：地城是基于Netty实现的，Netty是一个高性能的网络编程框架，支持同步、异步和响应式编程，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。即和Spring结合得天衣无缝！因此Spring官方默认兼容的就是Lettuce客户端！

Redisson：特点不在于对Redis的基本操作，是一个基于Redis实现了一系列的工具：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。需要分布式场景下使用Redis就用Redisson。



我们通过学习Spring Data Redis来学习Jedis和Lettuce两个客户端，因为Spring Data Redis底层可以兼容Jedis和Lettuce。即Spring Data Redis定义了一套API，这套API既可以用Jedis实现，也可以用Lettuce实现！所以你学习了Spring Data Redis，等于就来学习了Jedis和Lettuce！

当然了，很多企业并没有尝试用Spriing这套API，还是喜欢用旧的Jedis这一套，因为命令名就是方法名，非常稳定，所以我们把Jedis原生API也会简单学习一下，然后再主要去详细学习Spring Data Redis！

## Jedis客户端简单使用

### 1：创建maven项目redis-client-dmeo

### 2：引入Jedis依赖和Junit依赖

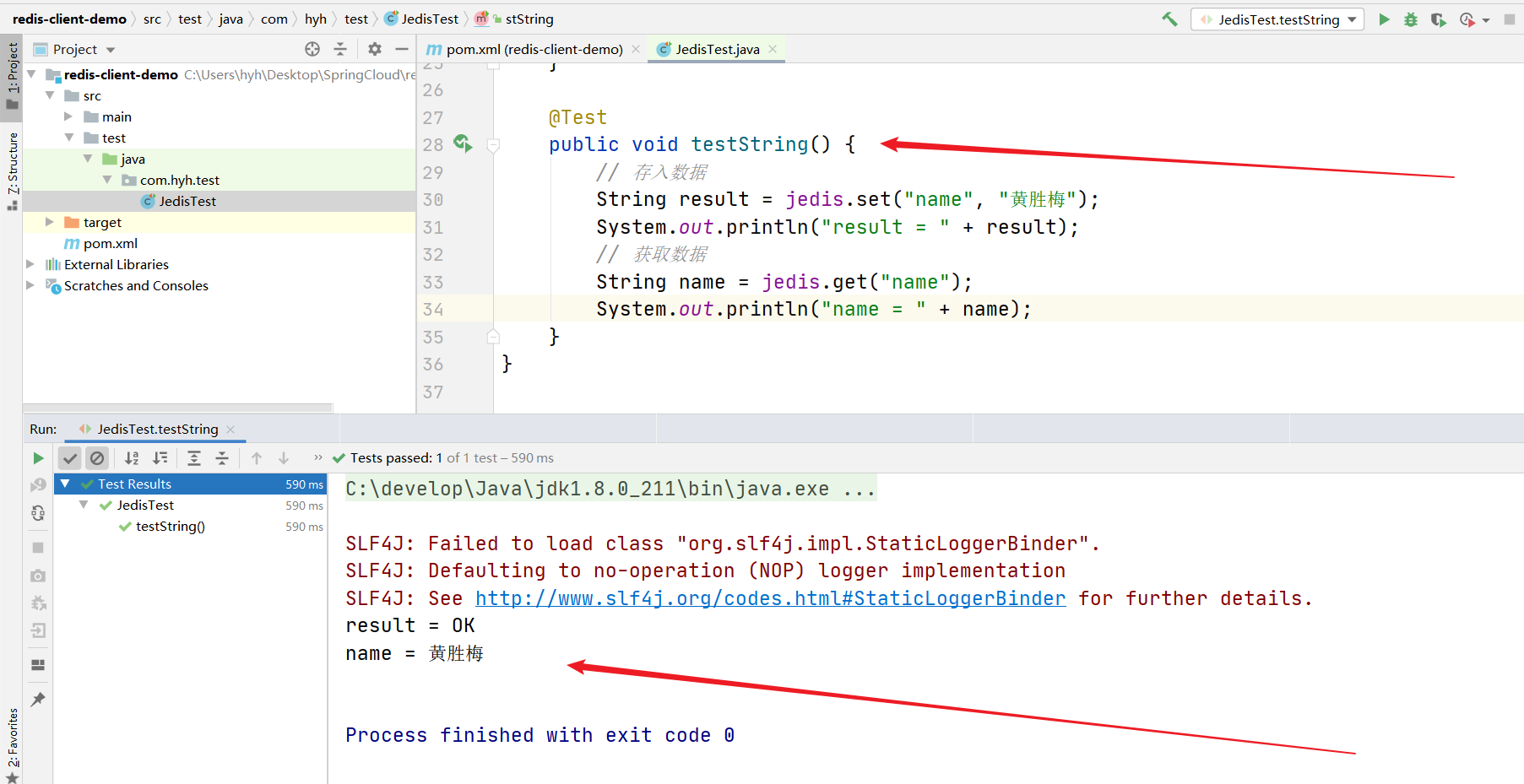


### 3：编写测试用例，使用Jedis客户端连接Redis

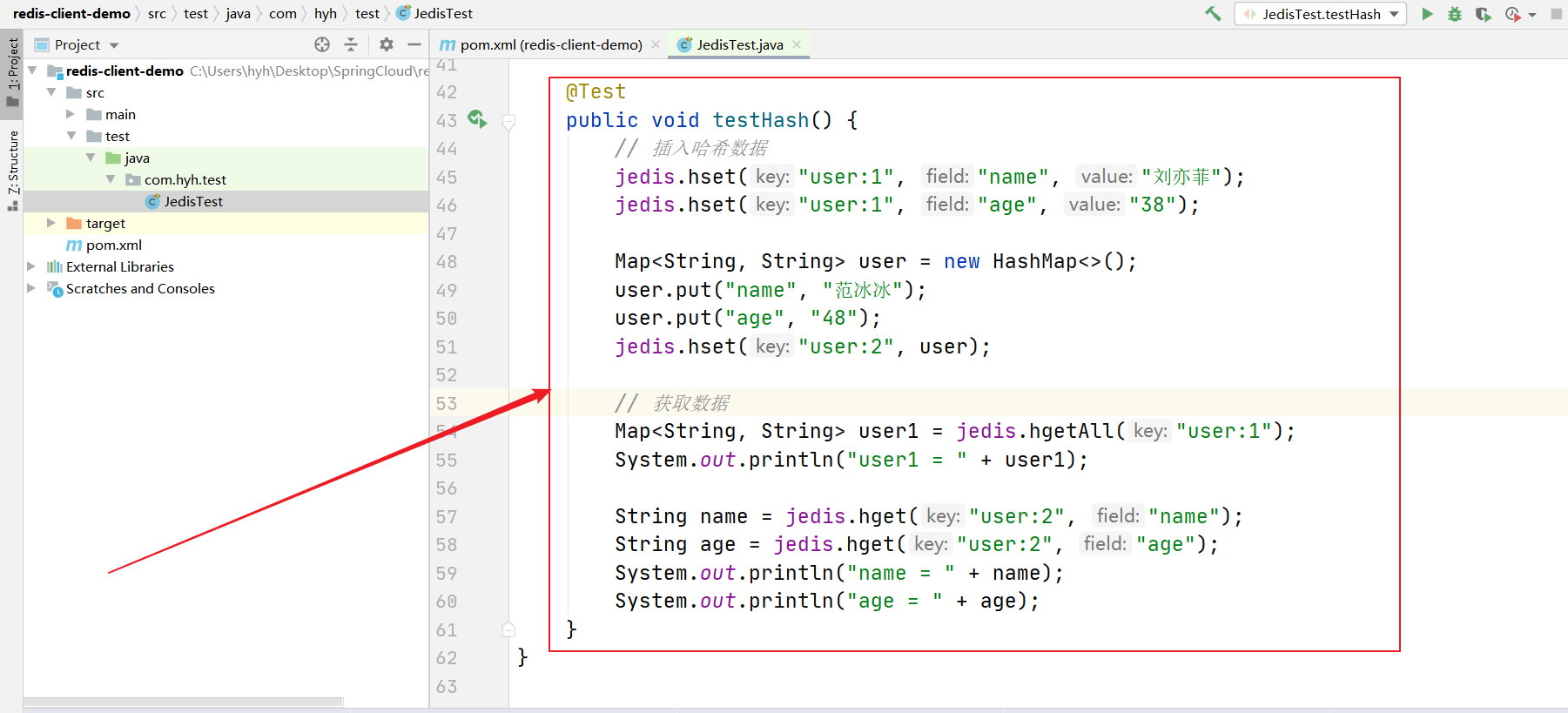


### 4：存储数据和获取数据

#### 4.1：String类型：向Redis存储数据和获取数据



#### 4.2：Hash类型：向Redis存储数据和获取数据



### Jedis使用基本步骤

1：引入依赖

2：创建jedis对象，建立连接

3：使用jedis，方法名和Redis命令一样

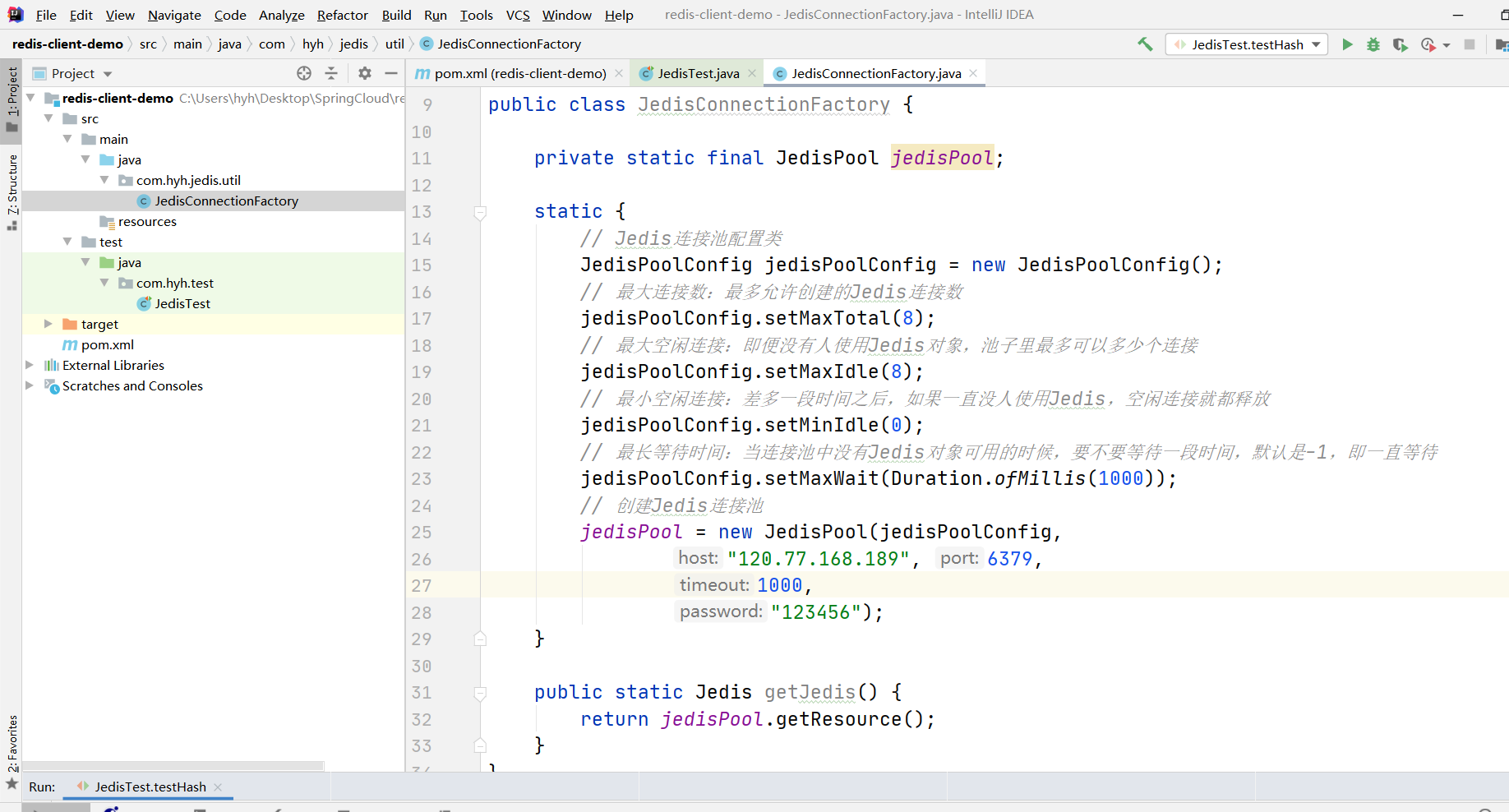
4：释放资源

## Jedis连接池：

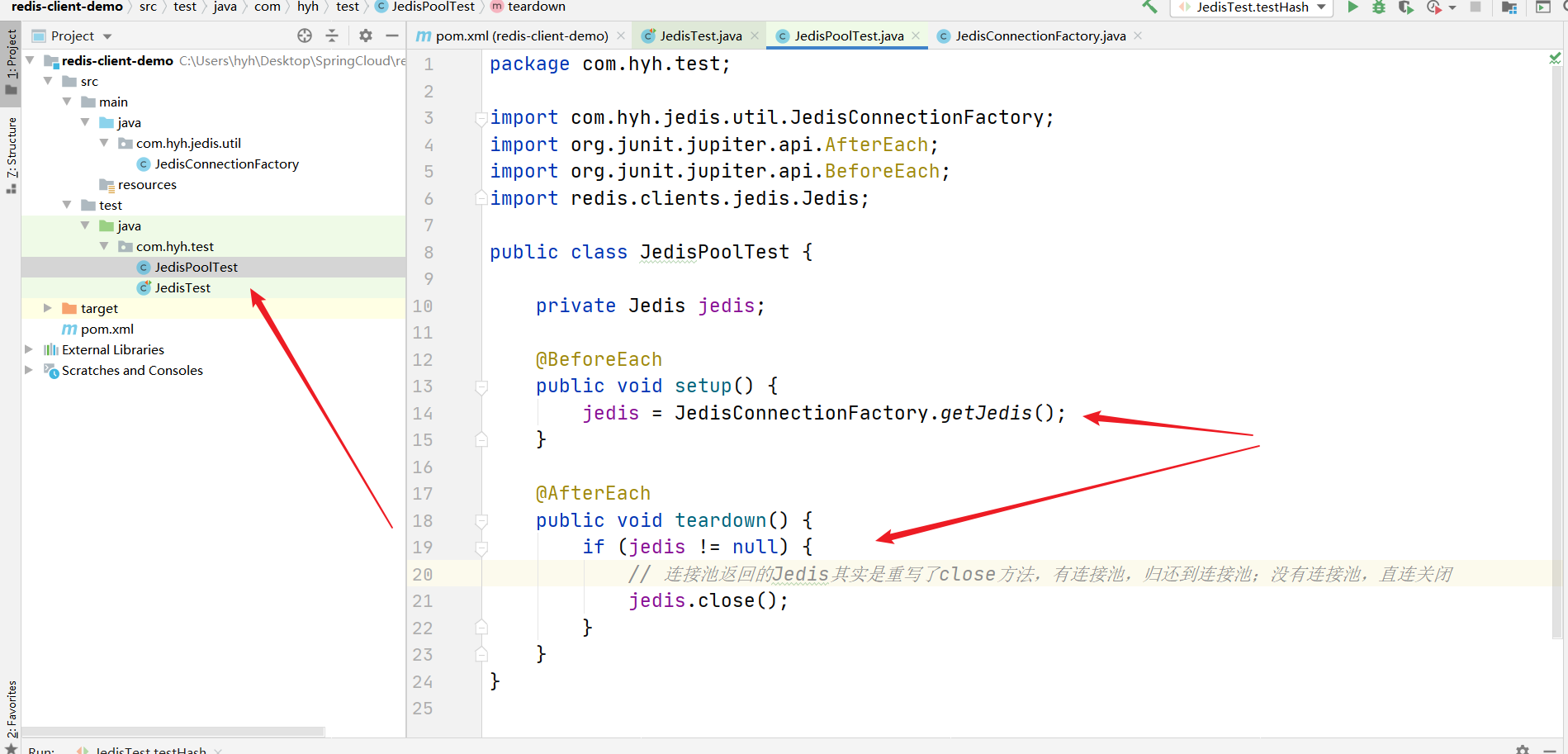
上面已经学习了Jedis的基本用法，但是我们也说过，Jedis本身是线程不安全的，如果在多线程的环境下并发地去访问，大家都来使用一个Jedis对象的话，是有可能出现线程安全问题的！因此在并发环境下，一定要给每一个线程创建独立的Jedis对象，此时最好的方案就是使用线程池！因为频繁地创建和销毁Jedis对象是由很大的性能损耗的！

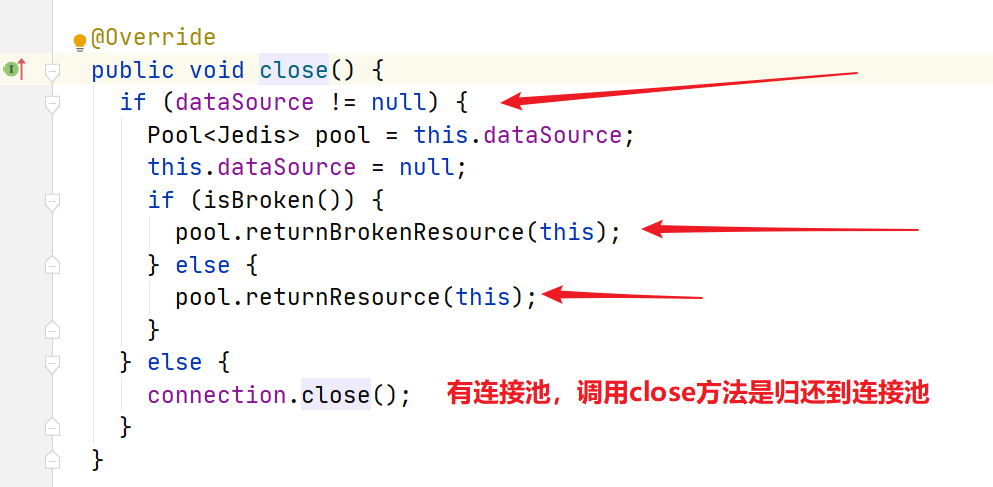
Jedis本身是线程不安全的，并且频繁创建和销毁连接会有性能损耗，，因此推荐使用Jedis链接池代替Jedis的直连方式，Jedis官方是基于Apache的Commons Pool实现了连接池。

### 1：创建Jedis连接池



### 2：：使用Jedis连接池





## SpringDataRedis

SpringData是Spring中关于数据操作的模块，包含Spring对各种数据库的集成，其中对Redis的集成模块就叫做SpringDataRedis。

Spring从来不会去重复造轮子，都是集成，SpringDataRedis最重要的就是对其他Redis客户端的整合！

**1:提供了对不同Redis客户端的整合（Lettuce和Jedis）**

**2:提供了RedisTemplate统一API来操作Redis**

**3支持Redis的发布订阅模型**

**4：支持Redis哨兵和Redis集群**

**5：支持基于Lettuce的响应式编程**

**6：支持基于JDK、JSON、字符串、Spring对象的数据序列化和反序列化。**

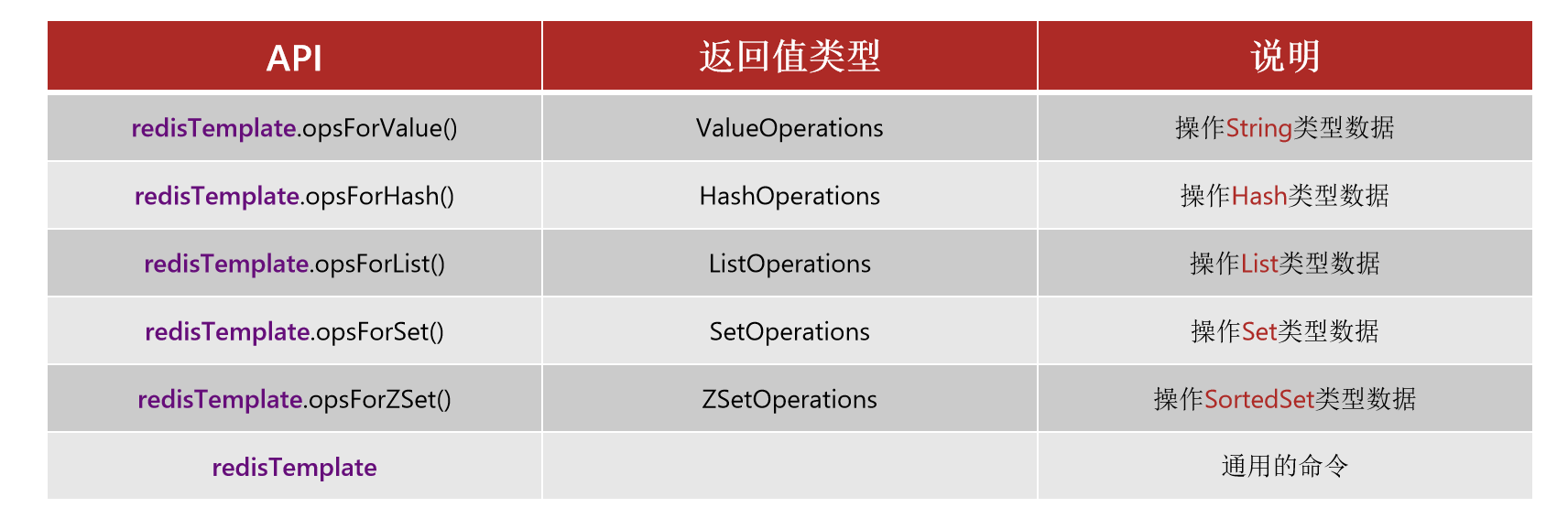
就是原生的Jedis客户端只支持存储字符串或者字节数组的数据，而SpringDataRedis可以直接存储复杂的Java对象，而无需我们手动去把Java对象转成字符串或者字节！

**7：支持基于Redis的JDKCollection实现。**

JDKCollection就是JDK中的各种集合，为什么要基于Redis重新实现集合，比如队列，链表等等，因为基于Redis的实现是分布式的、跨系统的。

### API介绍：

SpringDataRedis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封装到不同的类型中：



操作String类型数据：RedisTemplate.opsForValue().xxx

操作Hash类型数据：RedisTemplate.opsForHash().xxx

操作List类型数据：RedisTemplate.opsForList().xxx

操作Set类型数据：RedisTemplate.opsForSet().xxx

操作SortedSet类型数据：RedisTemplate.opsForZSet().xxx

### 快速入门

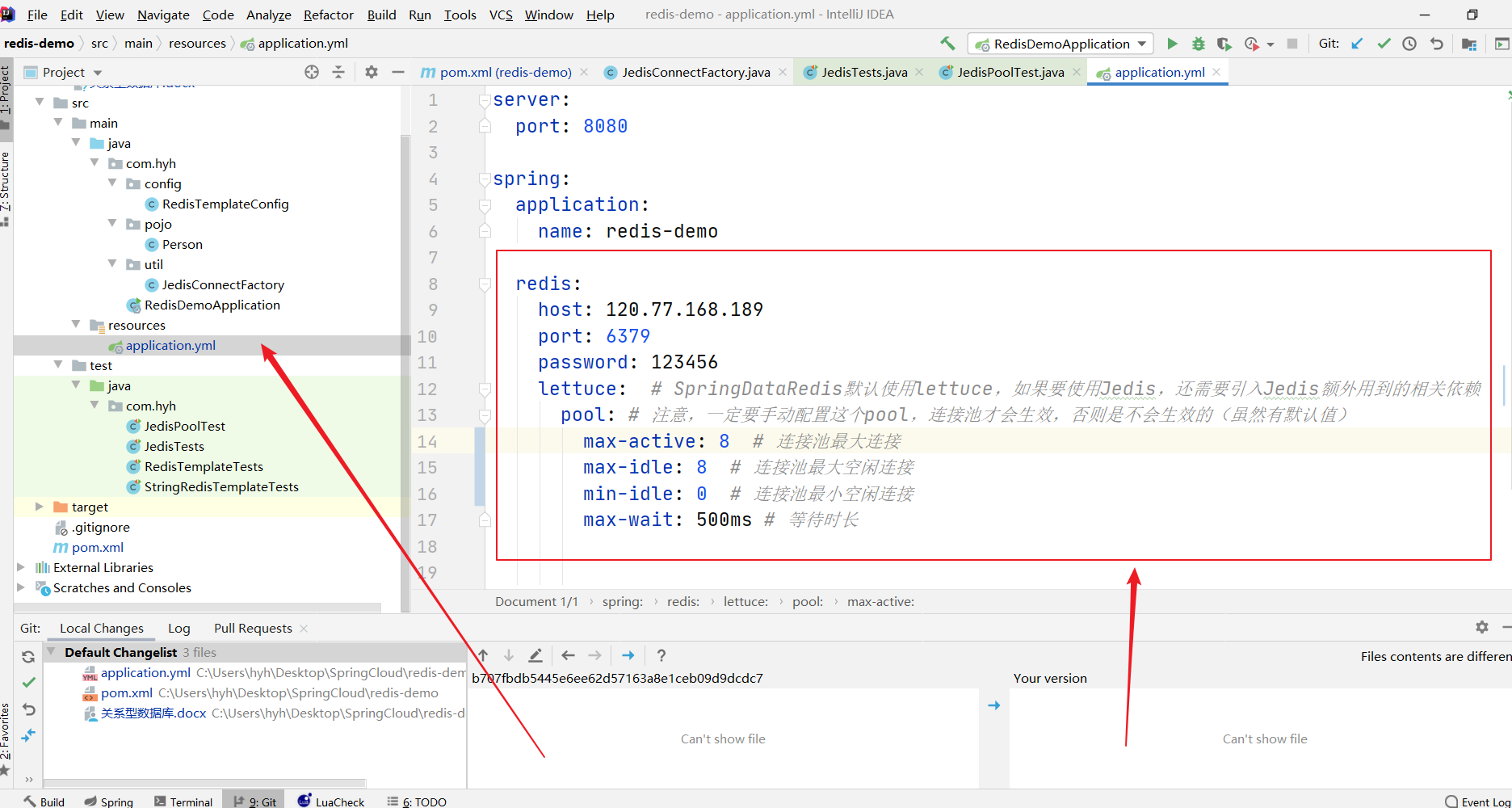
#### 0：创建SpringBoot项目

#### 1：添加spring-boot-starter-data-redis依赖

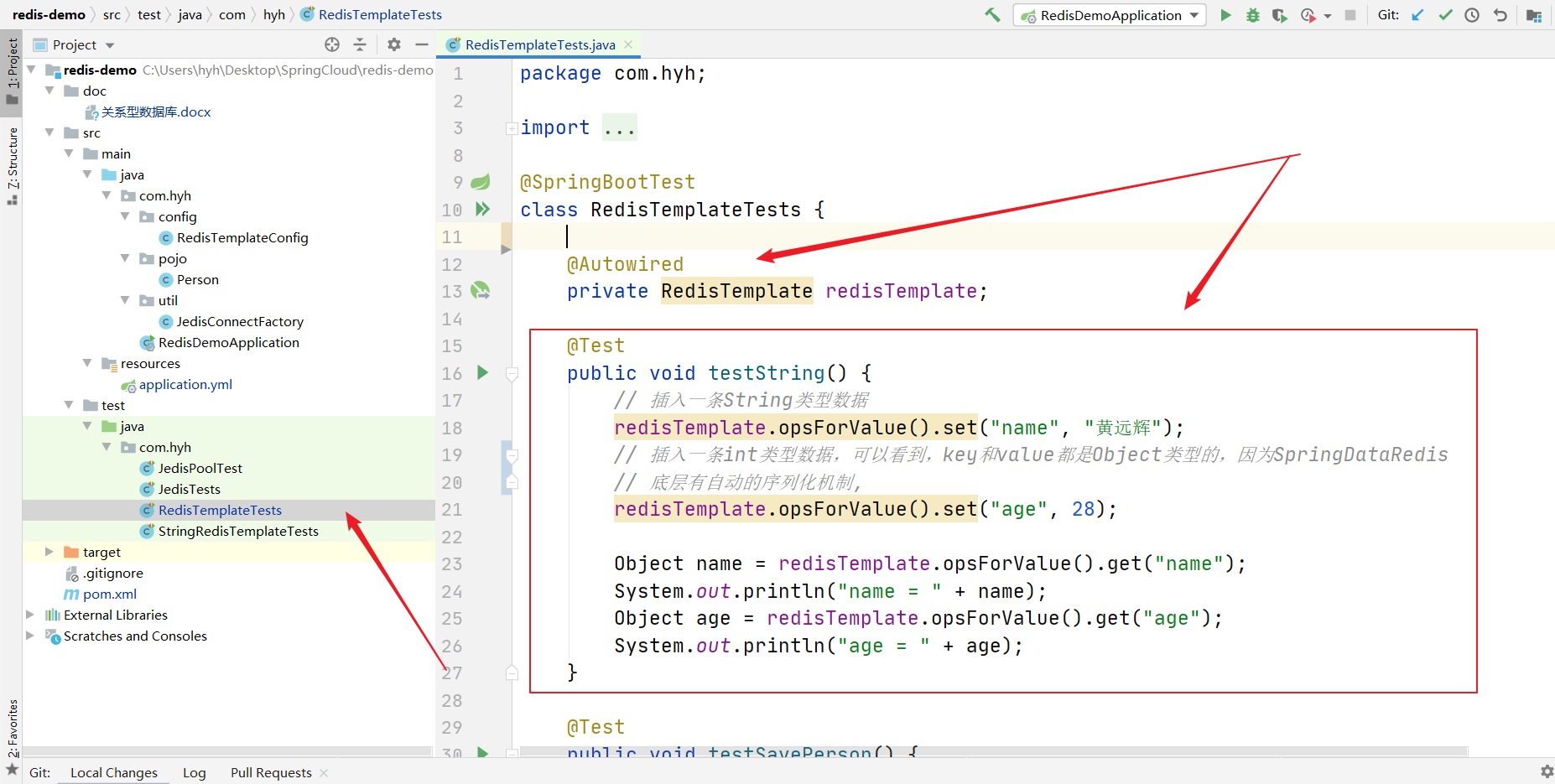


#### 2：在application.yml中配置redis信息

SpringBoot自动装配的好处就是我们不用去写编码，Spring会自动帮我们根据配置信息完成Bean创建，注入！



#### 3：注入RedisTemplate即可使用



#### 总结：SpringDataRedis的使用步骤

1：引入spring-boot-starter-data-redis依赖

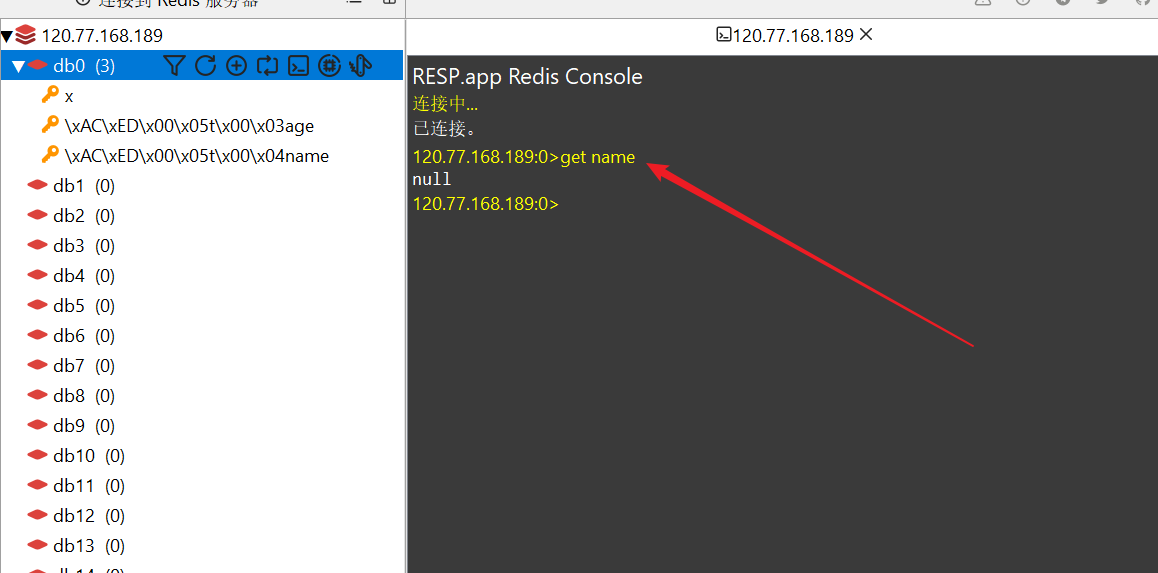
2：在application.yml配置Redis信息

3：注入RedisTemplate

### RedisTemplate的RedisSerializer

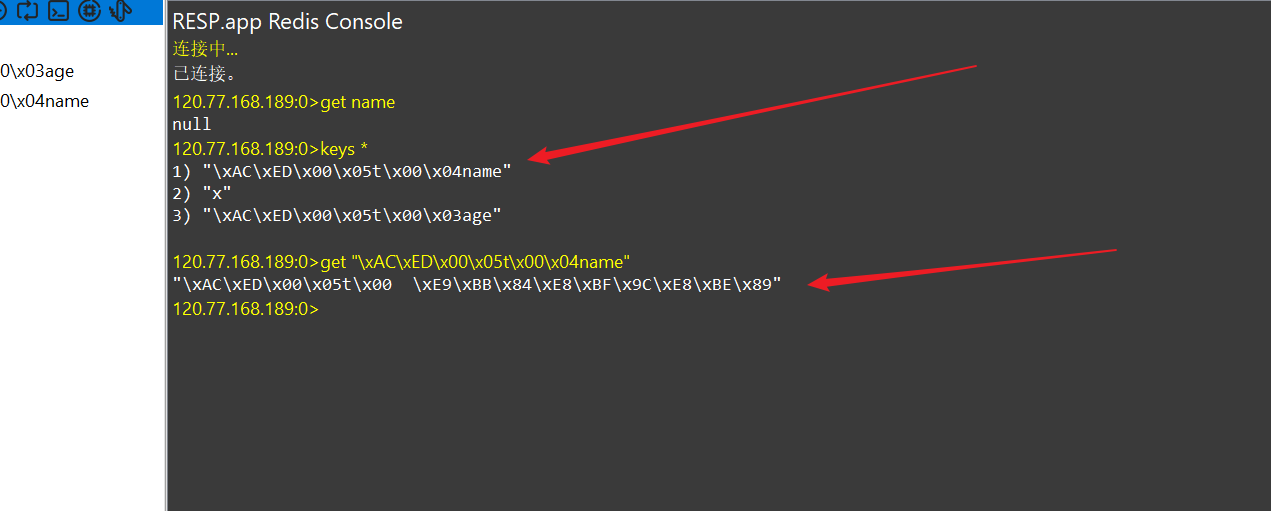
#### 默认序列化器

上面已经完成RedisTemplate的简单使用，我们向Redis中写入了一个key为“name”,value为“黄远辉”的数据！我们在控制台看一看！居然发现通过：get name获取不到存储进去的值“黄远辉”

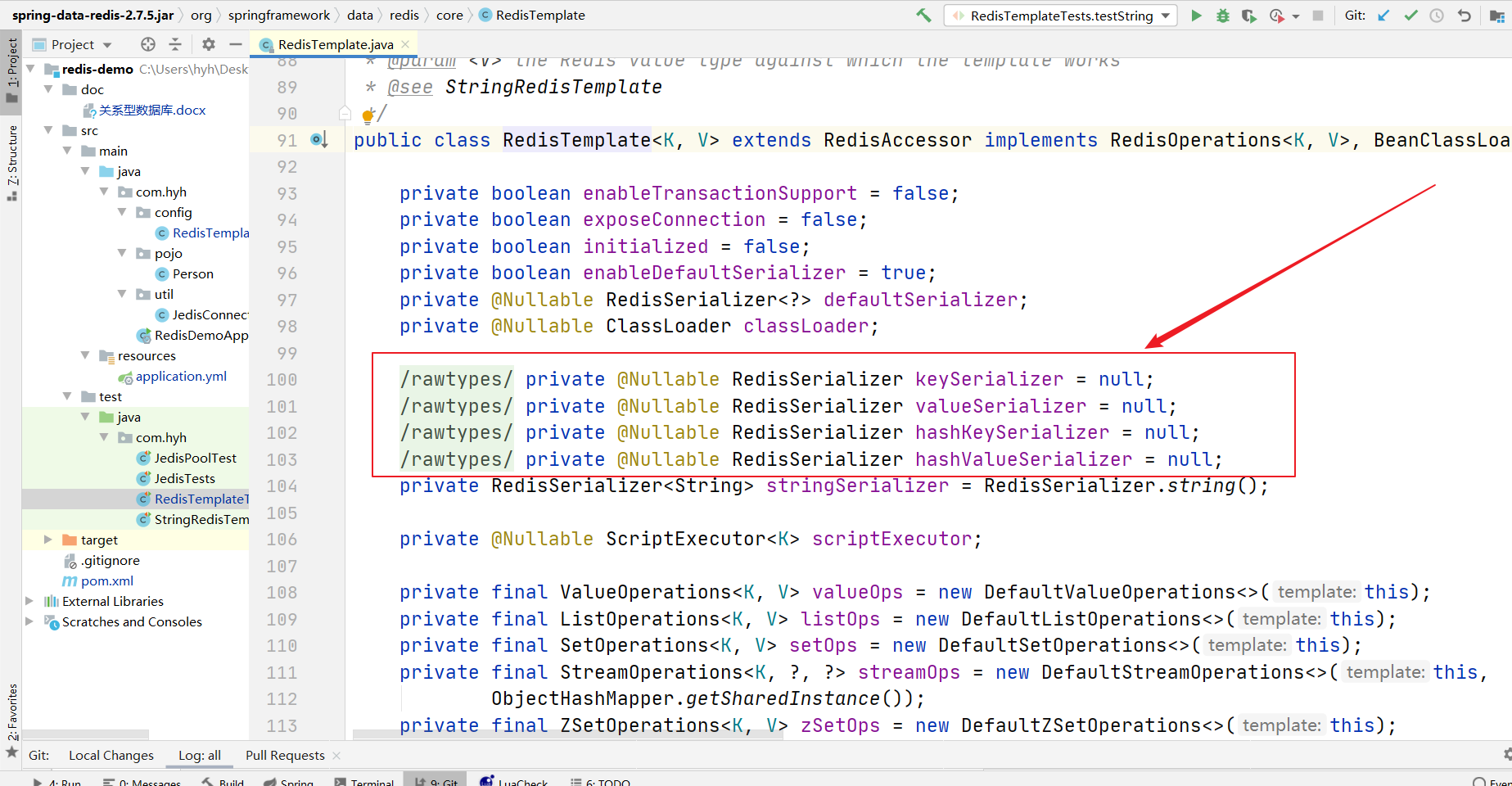


为什么我命名写入了key为“name”,value为“黄远辉”的数据，但是却通过管理工具获取不到呢？发生了什么事？

其实数据确实存进去了，只是变了样！看下图：



为什么会这样，这就要说到序列化了。RedisTemplate中的set方法接收的参数是Object类型的，并不是字符串，这就是我们之前说过的SpringDataRedis的特殊功能：它可以接收任何类型的对象，然后帮我们转成Redis可以处理的字节！所以我们存进去的key（name）和value（黄远辉）都被当成了Java对象了。而RedisTemplate底层默认对这些对象的处理方式就是利用JDK的序列化工具帮我们：



可以在RedisTemplate源码中看到，里面有四个序列化器

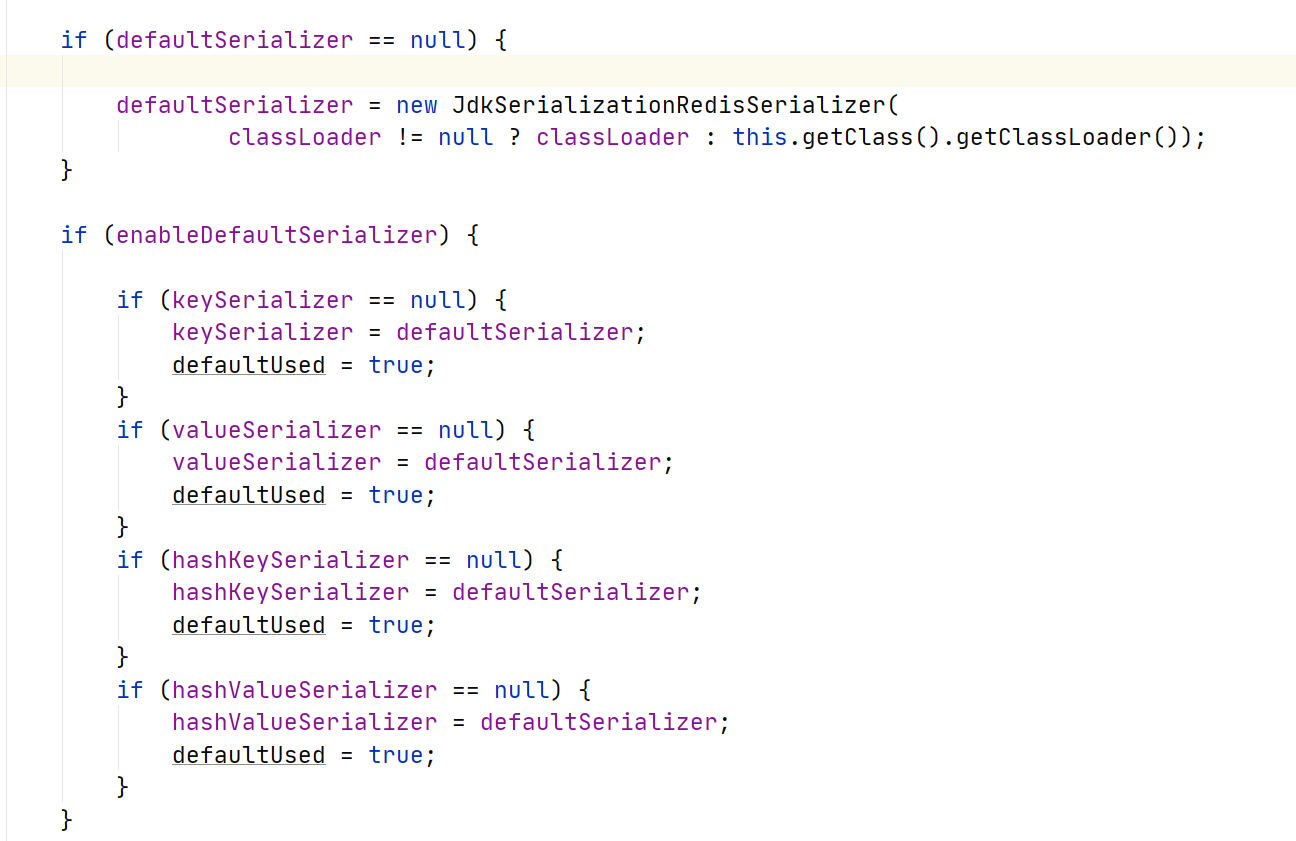
Key的序列化器：RedisSerializer keySerializer

Value的序列化器：RedisSerializer valueSerializer

Hash结构中字段的序列化器：RedisSerializer hashKeySerializer

Hash结构中字段值的序列化器：RedisSerializer hashValueSerializer

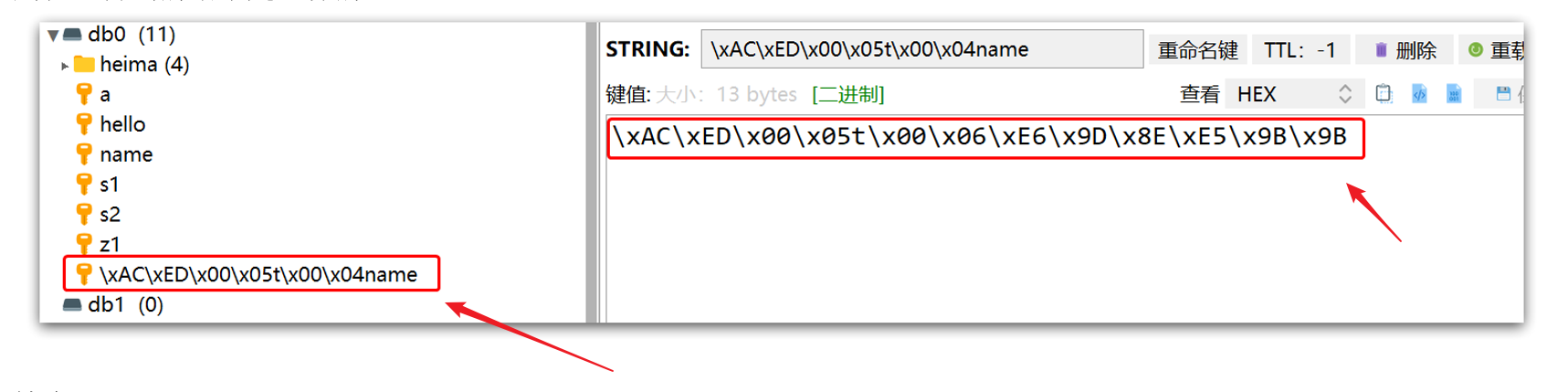
所以你看，我们利用RedisTemplate存入的一切数据，最终都会利用以上四个序列化器来去做序列化和反序列化！如果你没有给这些序列化器赋值的时候，就会创建的序列化器，默认的序列化器就是JDK的序列化器：**JdkSerializationRedisSerializer，这个序列化器就会把java对象转成字节！转成字节再写到Redis之后就是我们上面看到的哪个看不懂的样子了！**



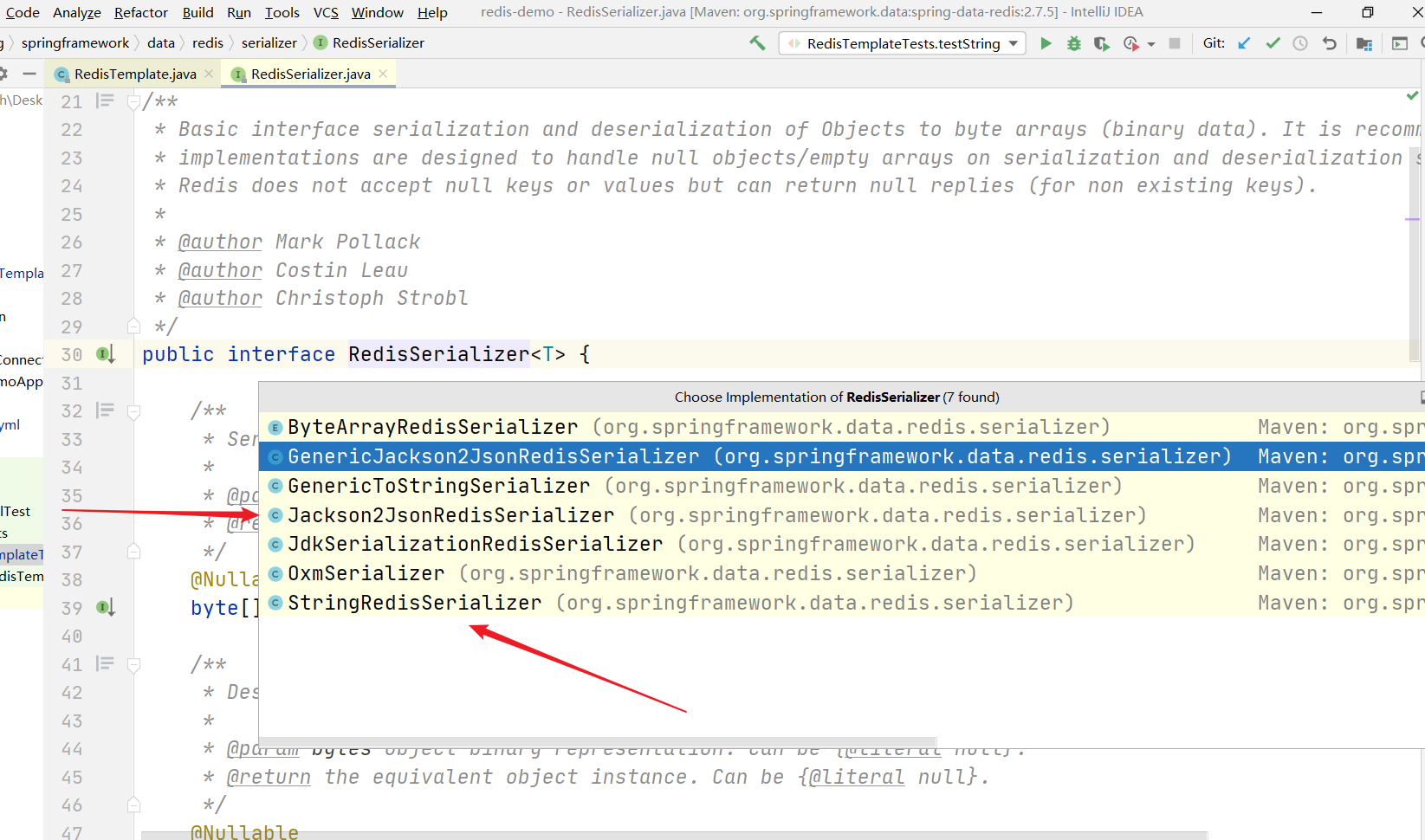
#### 默认的序列化方式存在的问题

1：可读性差，朋友们，如果我不告诉你，你知道这是“黄远辉”吗！而且还会存在bug，我明明通过代码设置的是name的值，但是可能会出现没有修改到我们想要的name！因为你的key也被序列化了！

2：内存占用大，看下图很明显啦，明明短小精悍的，序列化后却变得这么大！

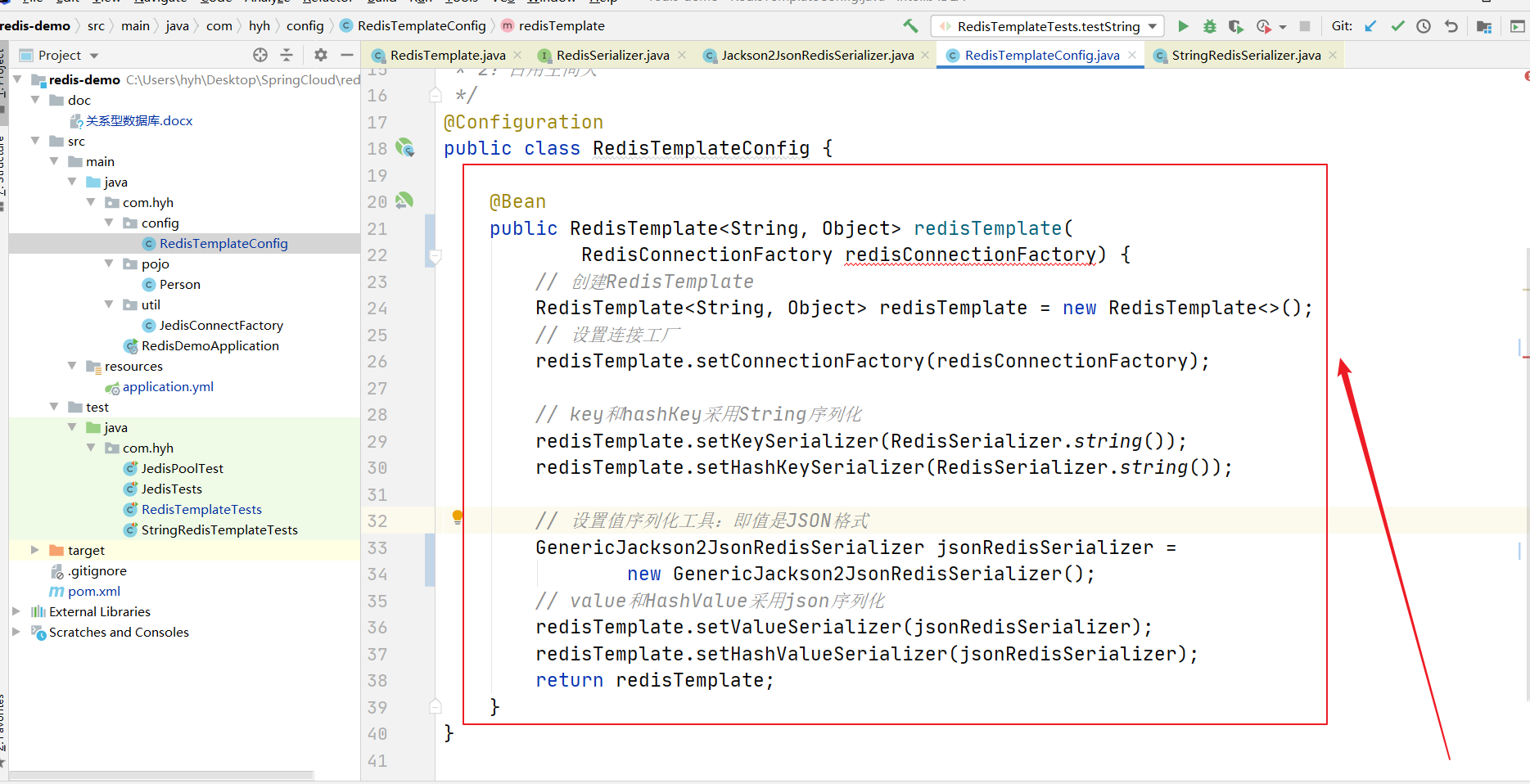


#### 改变RedisTemplate默认的序列化方式



可以看到RedisSerializer实现有上图中的几个，一般key的序列化器实现使用StringRedisSerializer，value的序列化器实现使用Jackson2JsonRedisSerializer。

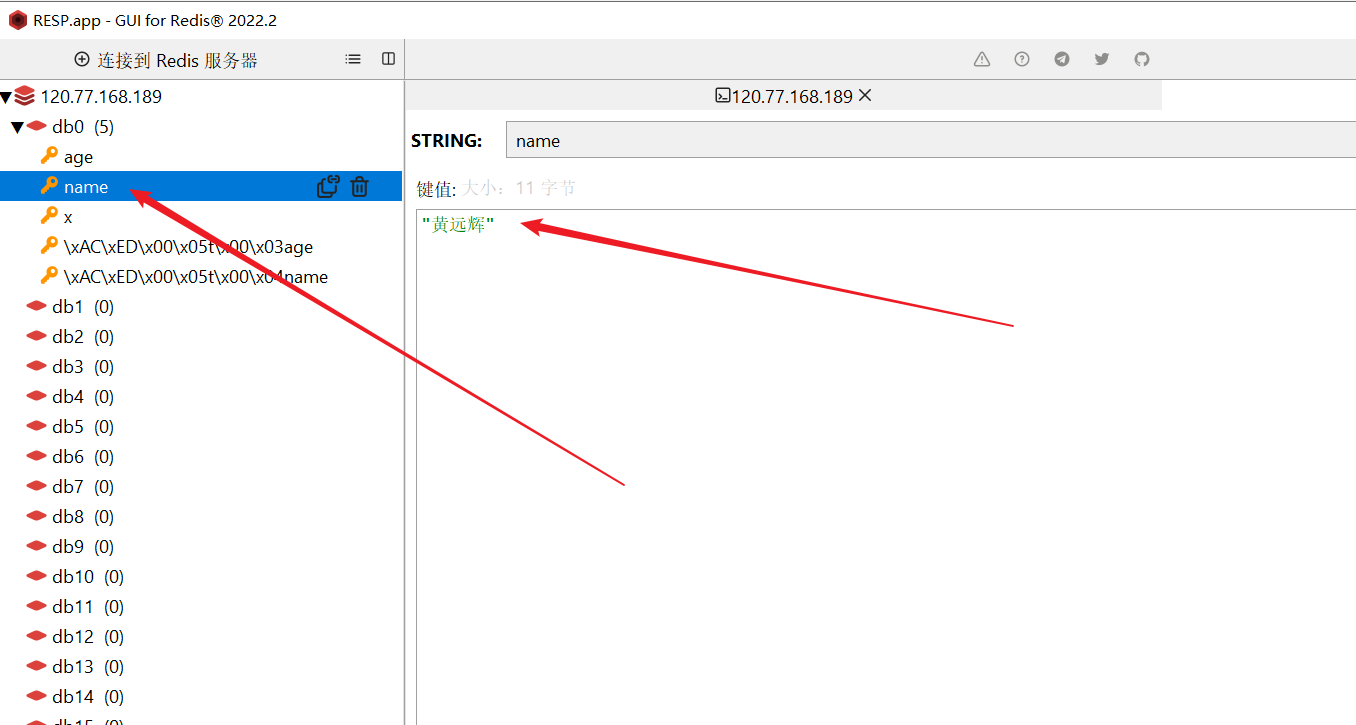
##### 设置key、hashKey和value、hashValue的序列化器实现：



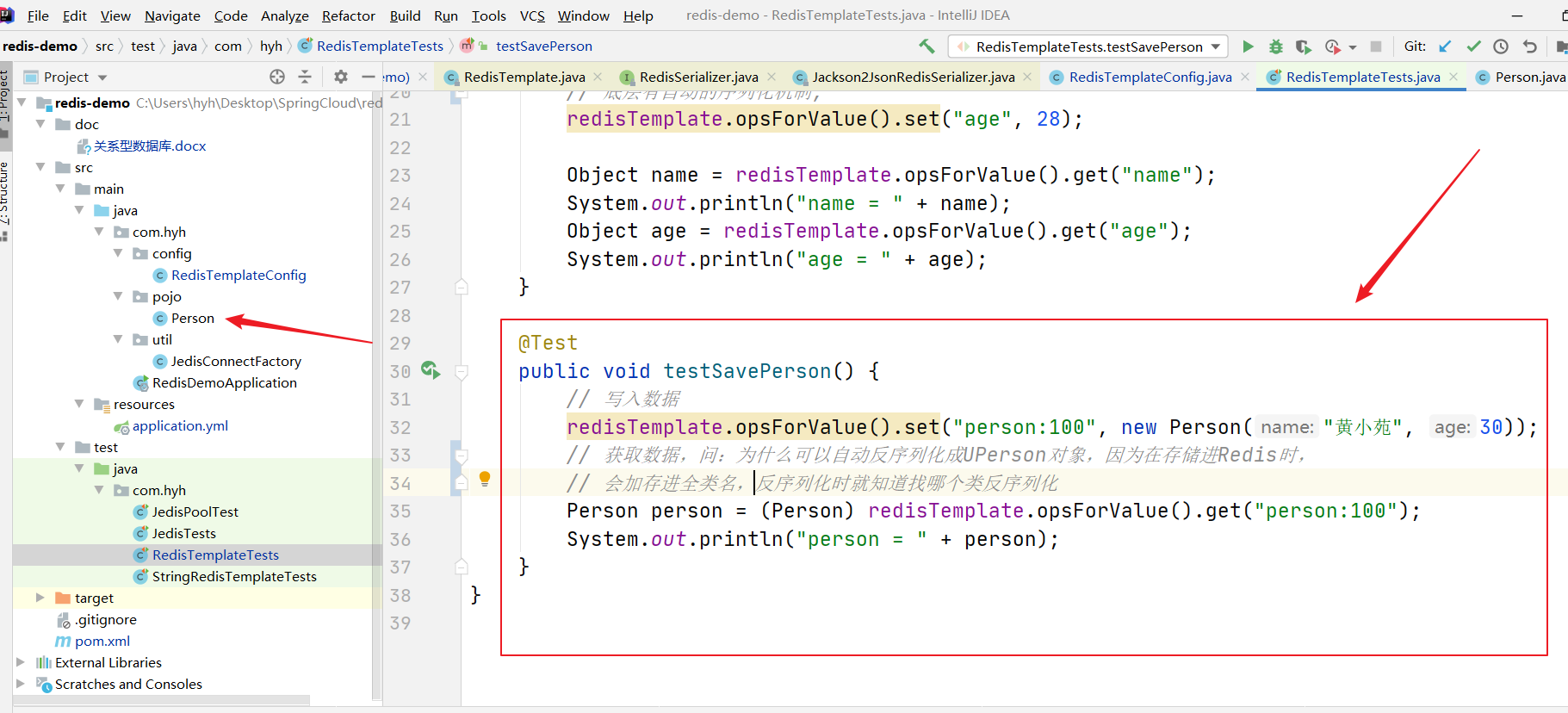
##### 引入jackson-databind依赖



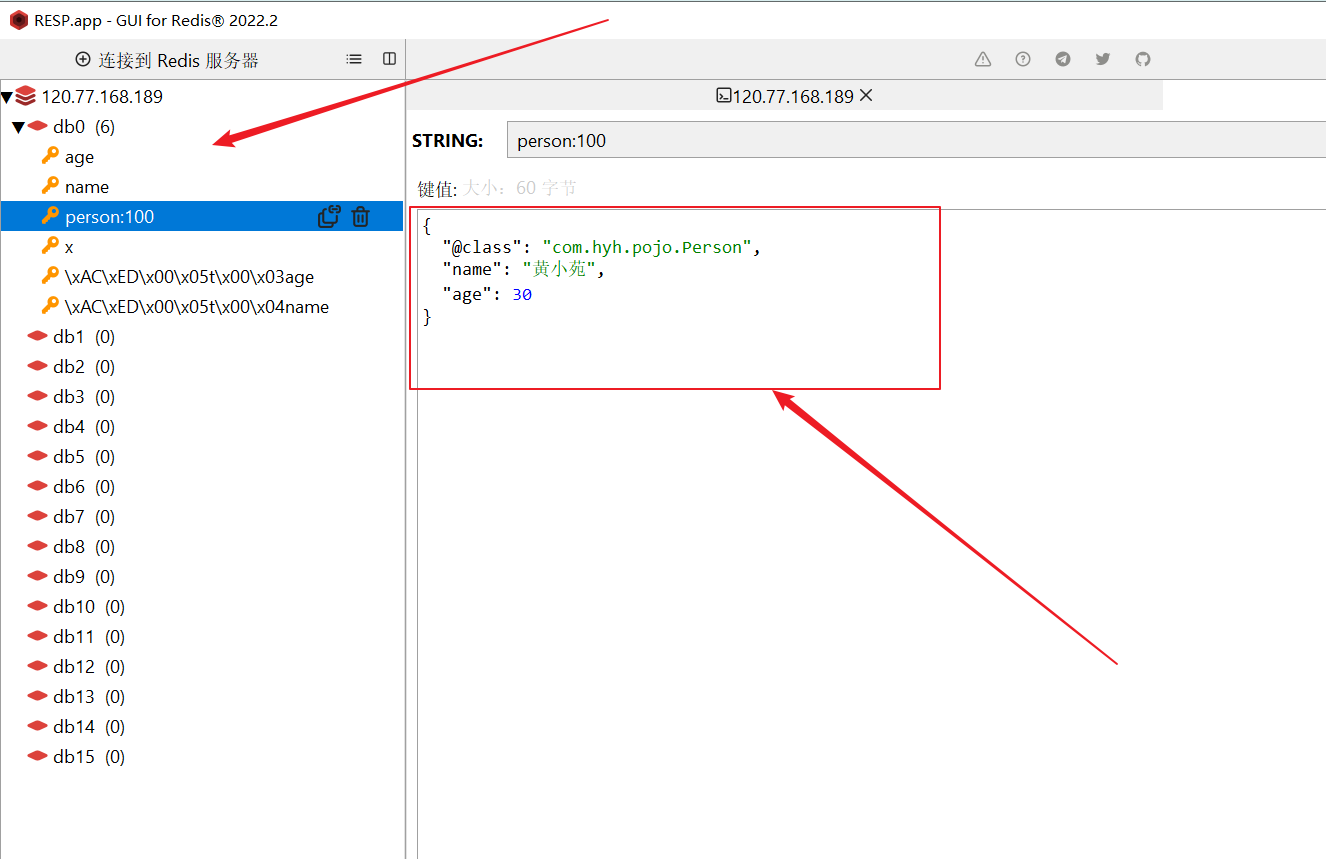
##### 运行测试方法，可以看到序列化方式正常了：



##### 验证向Redis存储对象的时候，看能不能序列化和反序列化



你看你看，序列化后变成我们想要的JSON格式的，也就是说，将来我们再去存储任何数据，不管是Java对象还是普通字符串，你随便存，全部都交给自动化的序列化工具去处理，而我们要做的就是定义好序列化器：



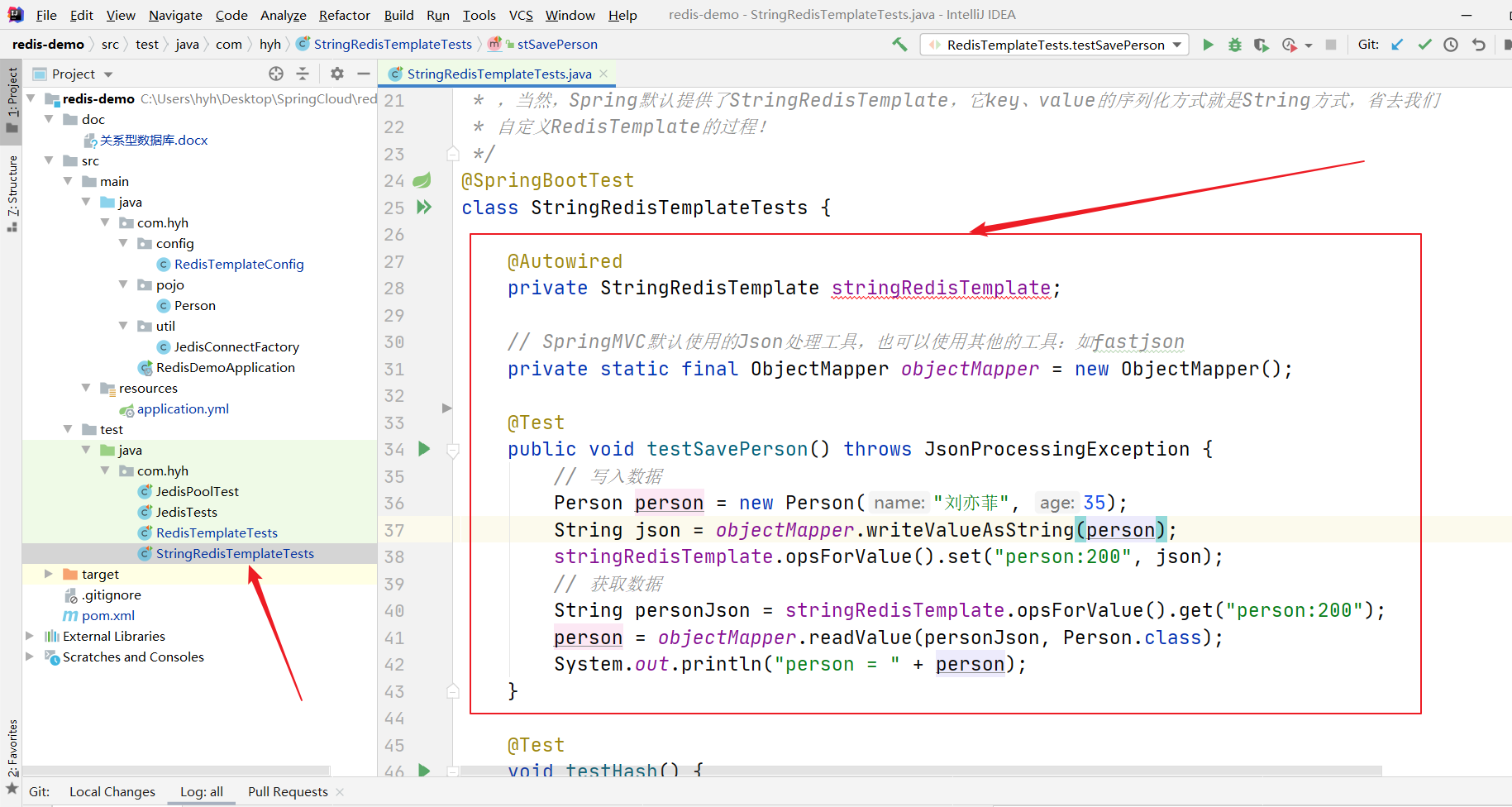
### StringRedisTemplate

前面例子我们已经学习了自定义序列化器来实现全自动的数据序列化和反序列化，这样一来，当我们向Redis写入一个Java对象的时候，我们根本不用操心，RedisTemplate可以帮助我们自动地把对象序列化成JSON，当我们读的时候还能自动帮把JSON我们反序列化Java对象，使用起来很爽。

但是却存在小小问题，上面例子的RedisTemplate的value序列化工具在保存到redis时，会有一个class属性（全类名），消耗内存！实际工作中，Redis会存储成千上万的对象，这样占用的内就相当大了！所以，为了节省内存空间，我们实际上并不会使用JSON序列化器来处理value，而是统一使用Spring序列化器，即要求只能存储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。

因此，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。

即最佳的方式就是key、value的序列化方式都是String，由程序员做一下额外的操作，当然，Spring默认提供了StringRedisTemplate，它key、value的序列化方式就是String方式，省去我们自定义RedisTemplate的过程！如下：



#### 总结：RedisTemplate的两种序列化实践方案：

##### 方案一：

自定义RedisTemplate

修改RedisTemplate的序列化器为GenericJackson2JsonRedisSerializer

##### 方案二：

使用StringRedisTemplate

写入Redis时，手动把对象序列化为JSON

读取Redis时，手动把读取到的JSON反序列化为对象

### StringRedisTemplate操作Hash类型

