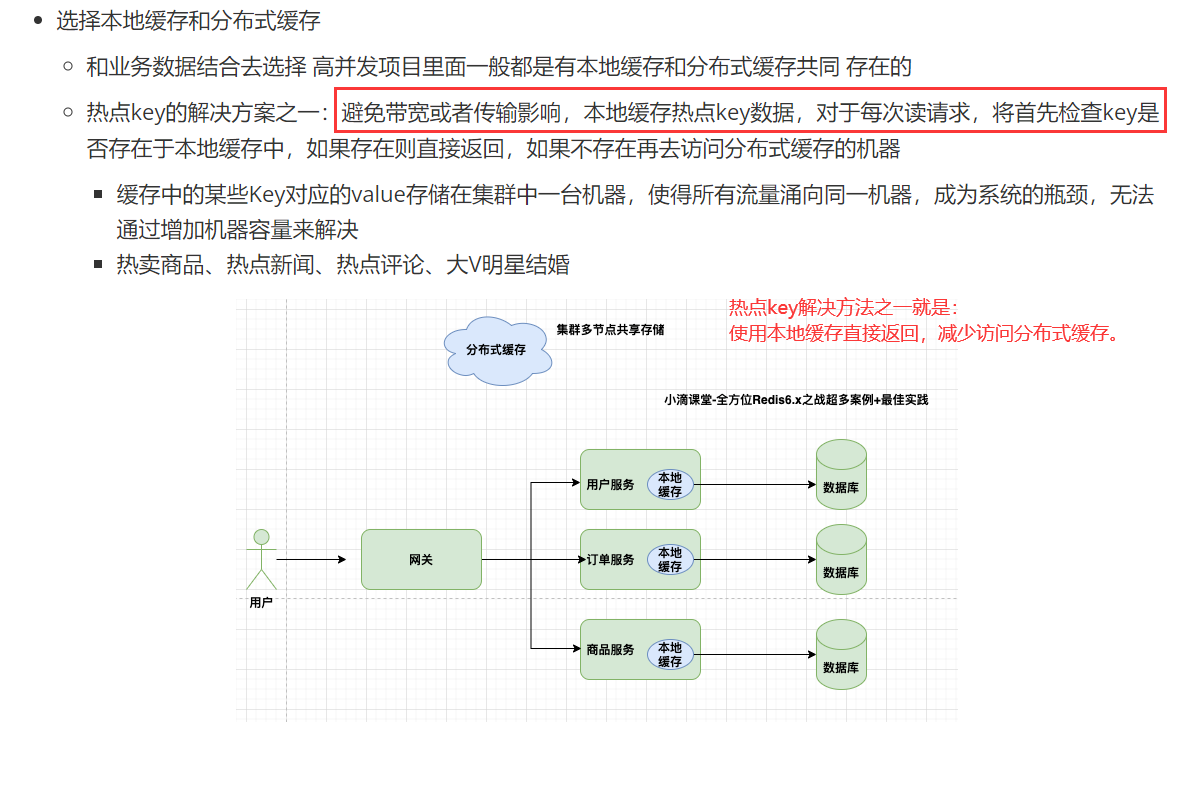
# 分布式缓存和本地缓存:



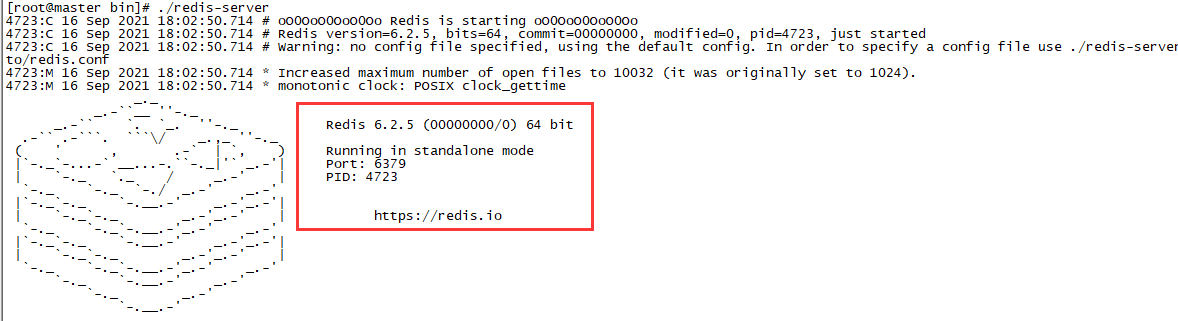
本地缓存肯定是最快的，但就是有单机这么个属性存在。

分布式中，分布式缓存和本地缓存是并存的，各有各的优势。



# 启动redis，规范:

./redis-server 即可启动:



启动信息。

这里需要自己配置redis的config和一些可连接配置:

1. 首先在redis目录下创建三个文件夹，不同的功能:

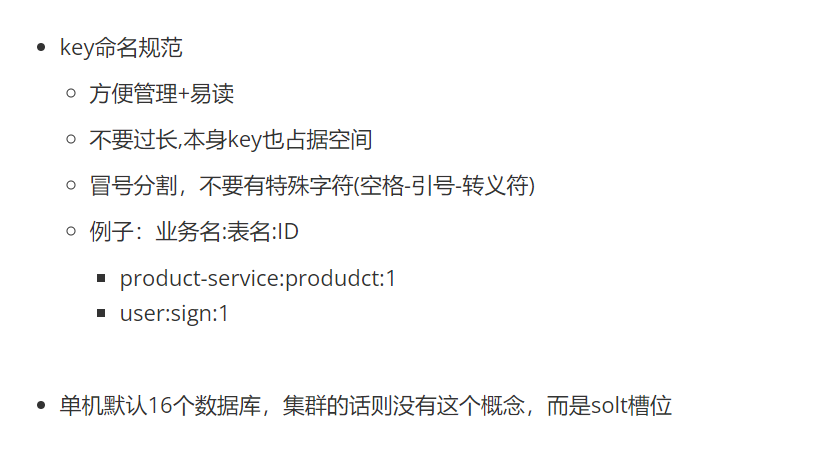


conf中创建redis.config配置文件（也可以使用它自带的，主要注意ip那块暂时所有可以访问，以后得指定ip访问）

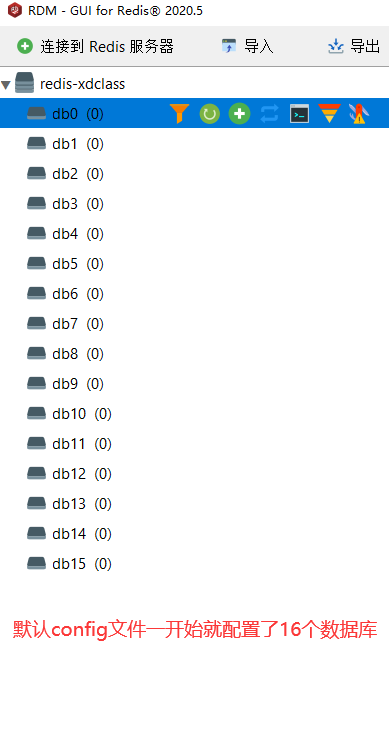
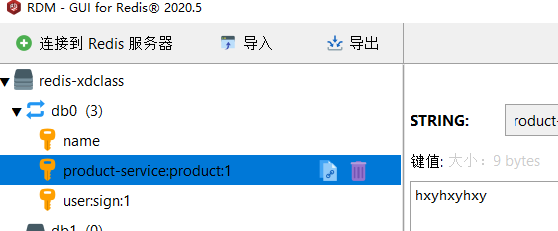


./redis-server ../conf/redis.conf 指定配置文件启动

2.key的规范如下，不同层级之间需要用“:”分割，在可视化工具中就会变成folds(本地这里没变，不知道为啥)



可视化工具:



# 3.各种数据结构和应用场景:

数据结构都差不多，就是应用场景得熟悉:

## 1.String:

String类型，value不能超过512MB，注意命名规范



mset/mget：一次性多设置和一次性多获取：

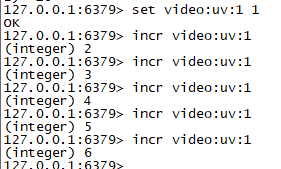




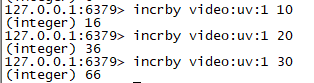
Mget：



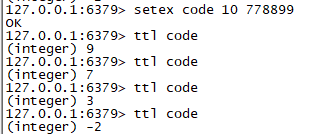
Incr 原子递增1:



Incrby 原子递增指定数值:



Setex 设置时指定过期时间:



Setnx:

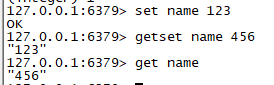
Set if not exist的意思，即不存在就set，存在就啥也不做；



原来name是已经存在了的。

Getset:

设置新的值返回旧的值:

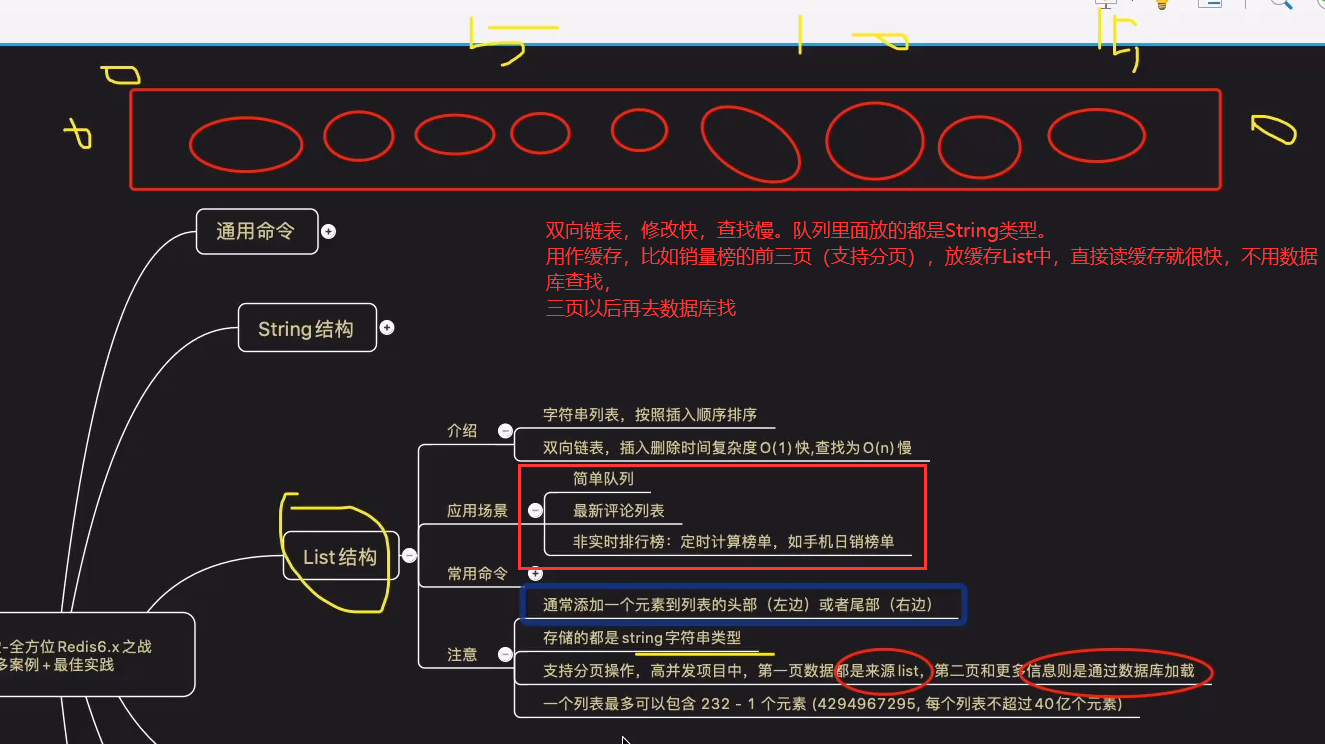


返回了旧的值。

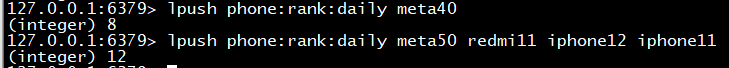
## List:

应用场景即 列表:

主要可以用来做队列，一个list大小最好不要超过1k。



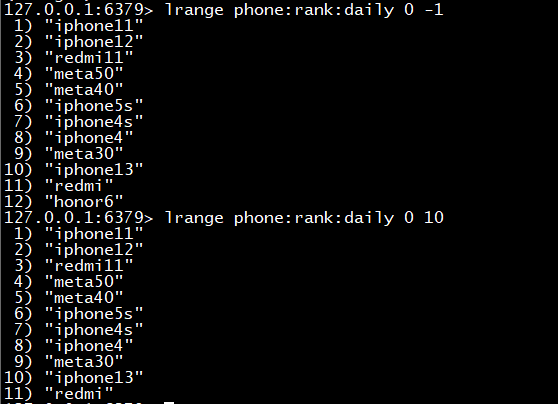
Lpush / rpush : 插入命令，可以一次插一个，也可以一次插多个。



Llen:查看列表长度。

Lrange: 查看列表元素，可指定范围，-1表示全部，输入两个同样的数字代表某个位置的值。

Lindex: 查看某个index的值



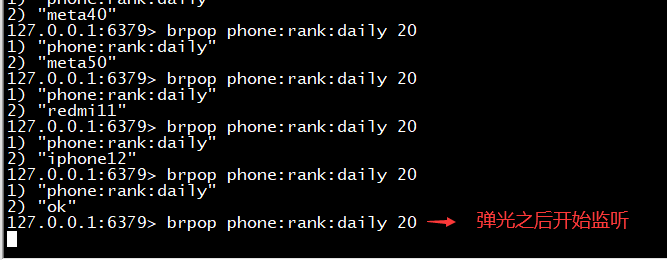


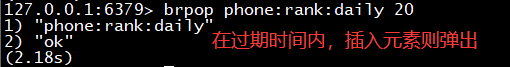
Lpop / Rpop：从队头或者队尾移出指定个数的元素。



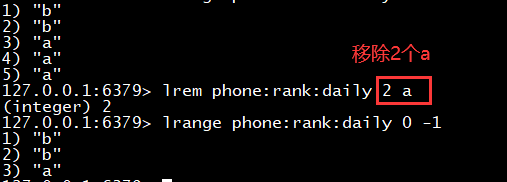
Brpop:阻塞式监听弹出元素:

从right（队尾）开始向上弹出，如果没有元素，则在设置时间内监听，如果有元素进队则弹出，没有的话超时就失效。

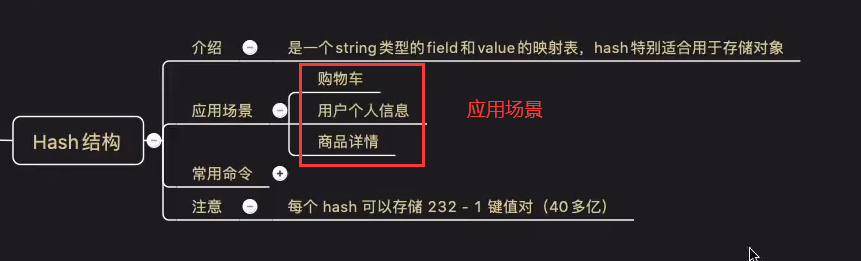




Lrem: 可以指定移除个数:



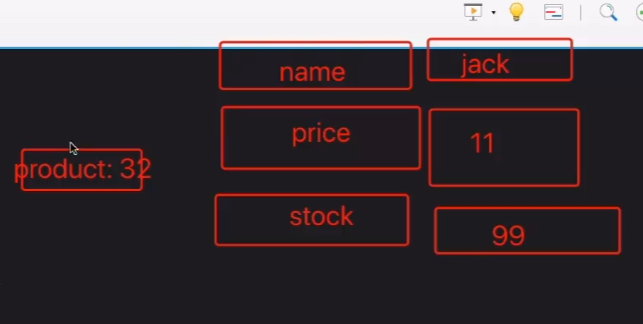
## HashMap:



也是为了提高性能，比如下面的购物车，uid为88的用户的购物车中的商品，就可以用HashMap<ID,JSON>的格式来存储，这样网页就更快加载。



还比如物品信息，也可以用Hashmap<attr,info>的格式:

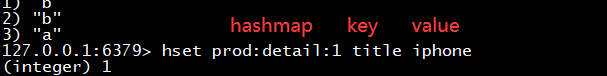


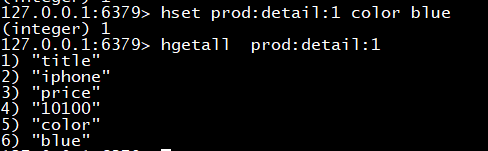
命令如下：

Hset: 设置某个hashmap的key-value, 注意，这里的key在redis中叫field

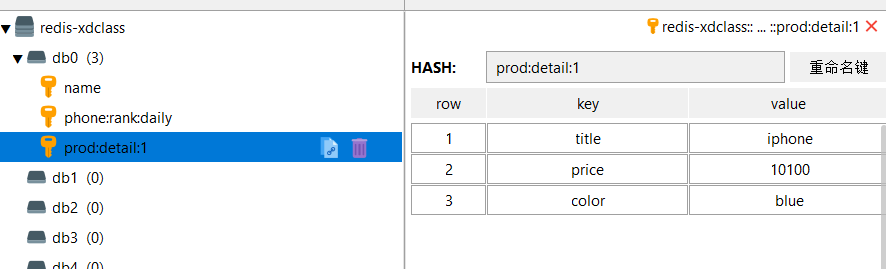
Hget:更据hashmap 的key值找value

HgetAll ：获取hashmap所有的key-value

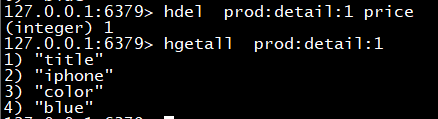




在数据表中是下面这样:

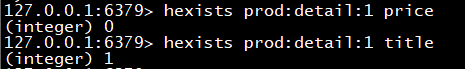


Hdel删除:



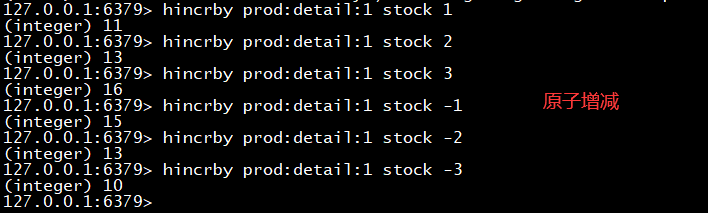
Hexist：

判断是否存在某个key(field)：



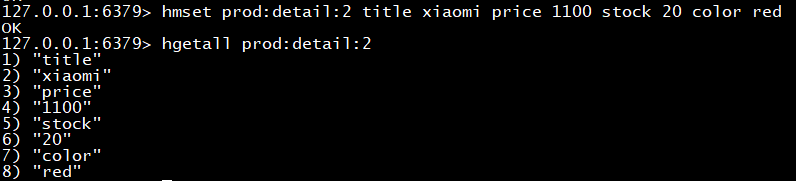
Hincrby:

对某个key(field)进行原子递增或者递减操作:



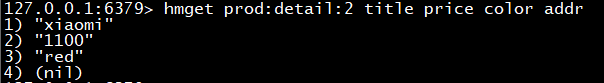
Hmset：

一步设置



Hmget:

一步获取，如果没有那个key(field)，则返回null

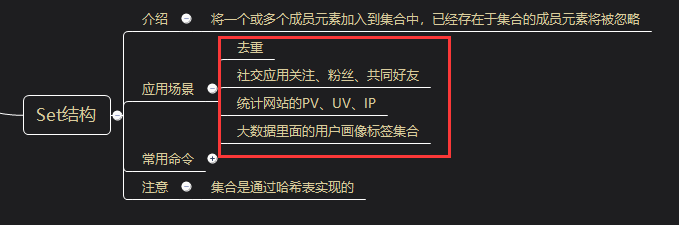


## Set:

主要是拿来去重，应用场景也很典型:

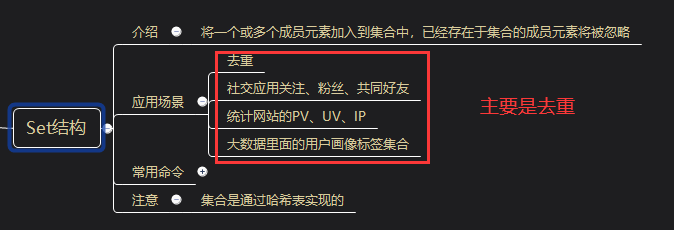
如Sdiff ：

标签差集可以用在内容推荐上，A有行为1，B没有行为1，但他们的差集非常小，两人很近似，所以可以给B推荐行为1，B搞不好也喜欢。



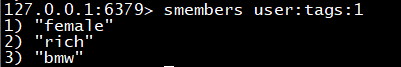
Sadd:

往集合里插元素:



Smember:

返回集合中所有元素



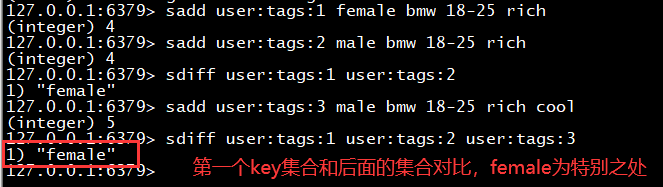
Scard:

返回集合内的元素数量:



Sdiff:

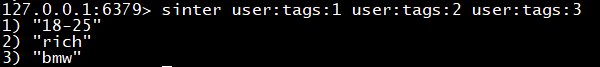
返回第一个key集合和后面所有key集合的差集（即返回第一个key的独特之处）



这个标签差集可以用在内容推荐上，A有行为1，B没有行为1，但他们的差集非常小，两人很近似，所以可以给B推荐行为1，B搞不好也喜欢。

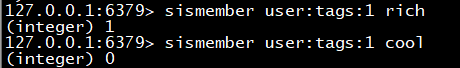
Sinter:

集合交集



Sismember:

集合是否存在某个元素



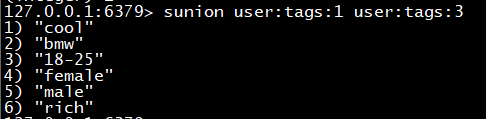
Srem:

移除集合某个元素



Sunion:

返回多个集合合并的结果（不影响集合本身）:



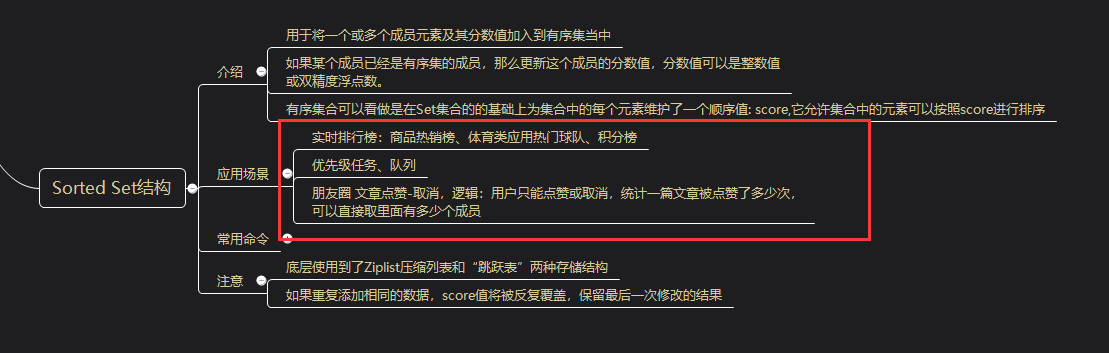
## Sortedset

带权值排序的set，维护了一个score属性，SortedSet通过该属性对集合进行排序。

也就是如果Set中的元素需要有一个优先级，那么用这个就可以。

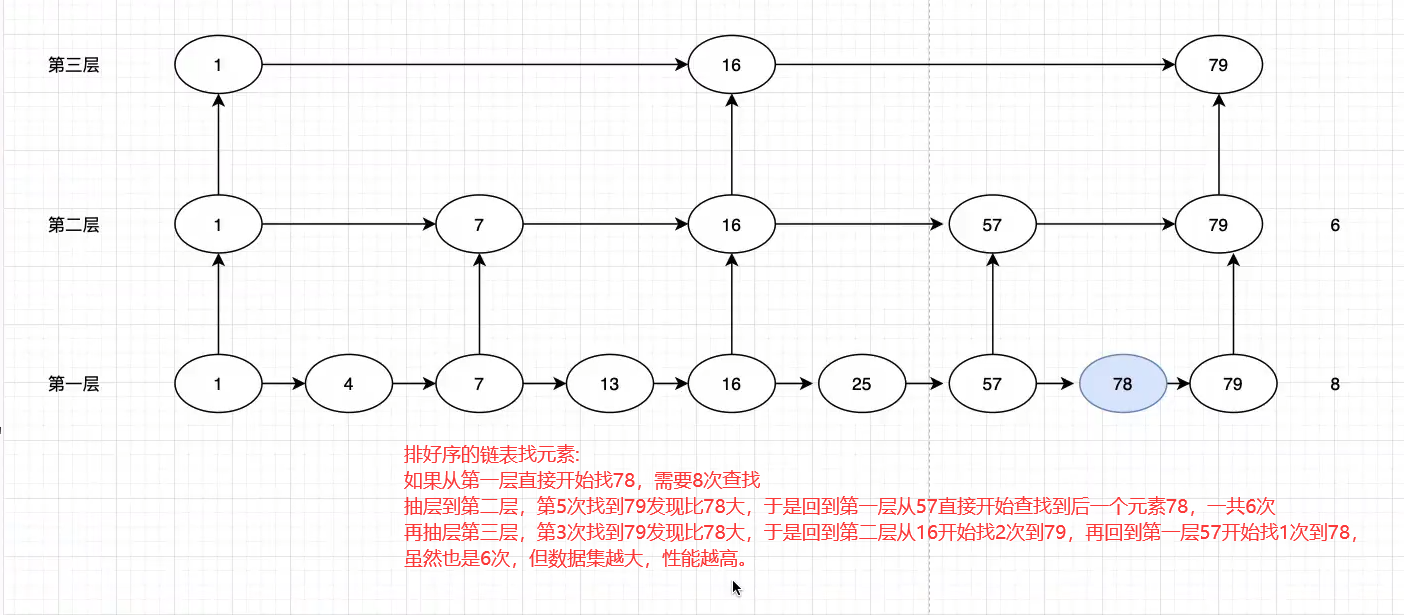
原本set是没有顺序的。

应用场景，带优先级考虑即可。



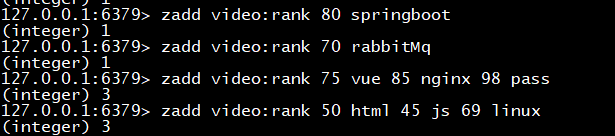
他的底层是ziplist压缩列表和跳跃表两种存储结构

这里讲一个跳跃表的数据结构,高性能查找



Zadd:

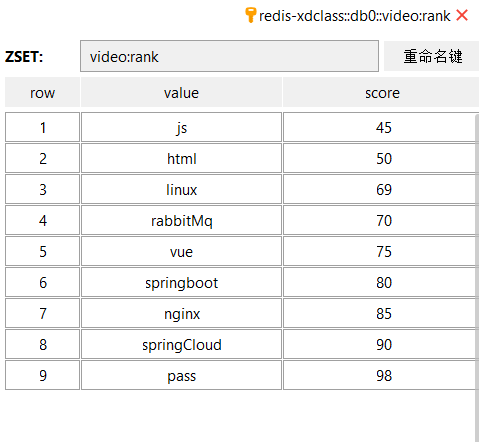
像sortedset中加入元素，格式为 分数 value，可一次性多加:



Zcard:

获取sortedset中的元素个数:(最上面有一个没有截图上)





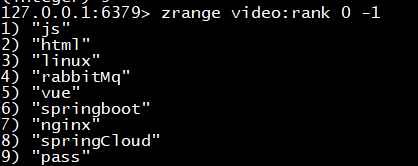
Zscore：

查看sortedset中某个元素的分数:

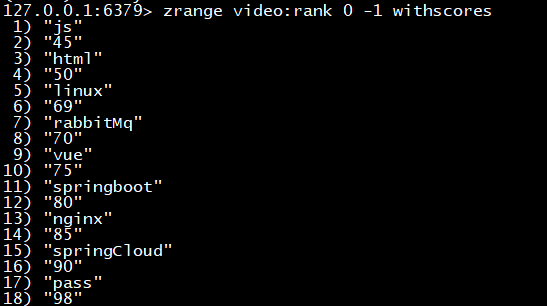


Zrange：

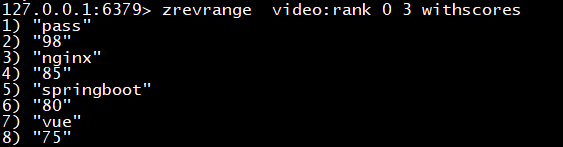
指定范围获取sortedset中的元素:（从小到大）



加上withscore就一起返回分数了

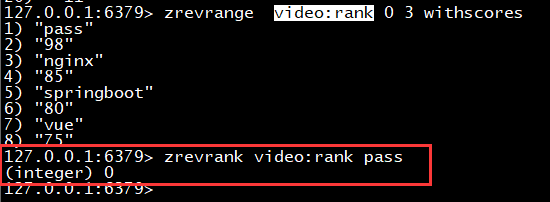


Zrevrange(从大到小):



Zrevrank：

返回指定元素在sortedset中从大到小的名次（排名表）:



Zcount:

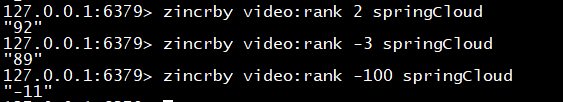
返回sortedset中指定分数范围的个数:

90-100,两个



Zincrby:

增加或减少sortedset集合中某个元素的score





Zrem：

删除sortedset中的指定元素:



# 4.Springboot连接redis的操作合集:

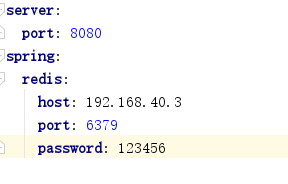


Spring官方为了方便整合这些热门数据中间件（redis，elk，mysql），于是自己封装了spring-data依赖，一个依赖管所有，就很方便。

Spring连接redis现版本用的是lettuce，老版本用的jedis，但线程不安全，所以淘汰了:



连接部分:



## 操作部分(序列化讲解):

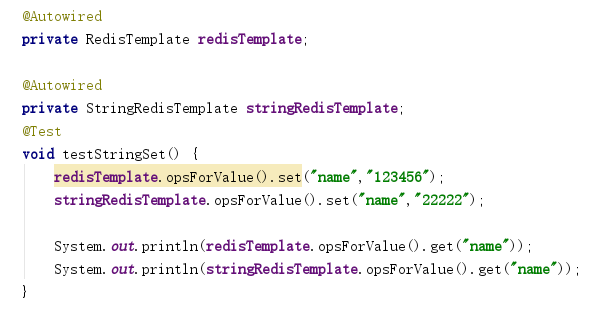
1. redisTemplate：

最经典的操作，相当于直接操作redis命令行:



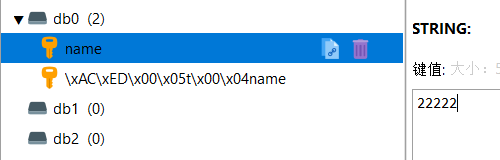
下面有一个StringRedisTemplate，这个是专门用来做String类型操作的。

注意一点是因为RedisTemplate和StringRedisTemplate使用的序列化不一样，他们产生的key值就不一样，如果不做序列化处理的话，他们的数据不互通（key值不一样，但后面做处理之后就一样了）:





存储如下:



RedisTemplate用的是java的序列化规则，所以会产生乱码，导致两个key名不一样。

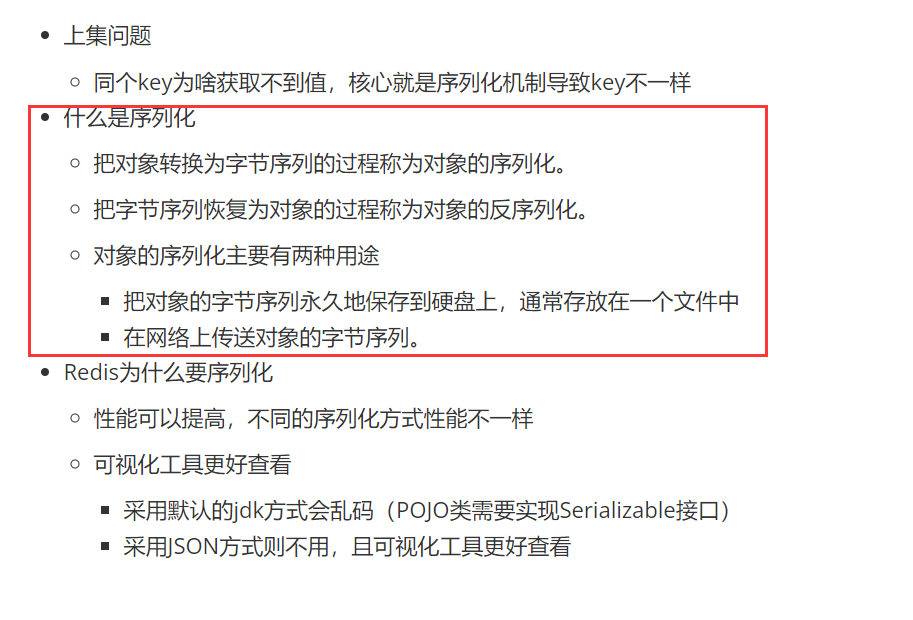
也就分开存储了。

### 1.序列化原理:

可以看到这里因为redisTemplate使用的是JDK的序列化策略，转成了字节数组，于是导致了redis展示时的乱码。但这种序列化格式对于jvm获取数据时是不影响的，因为序列规则，转回来罢了。

而StringredisTemplate使用的是String的序列格式，双方通用，于是就很完美的展示了出来。

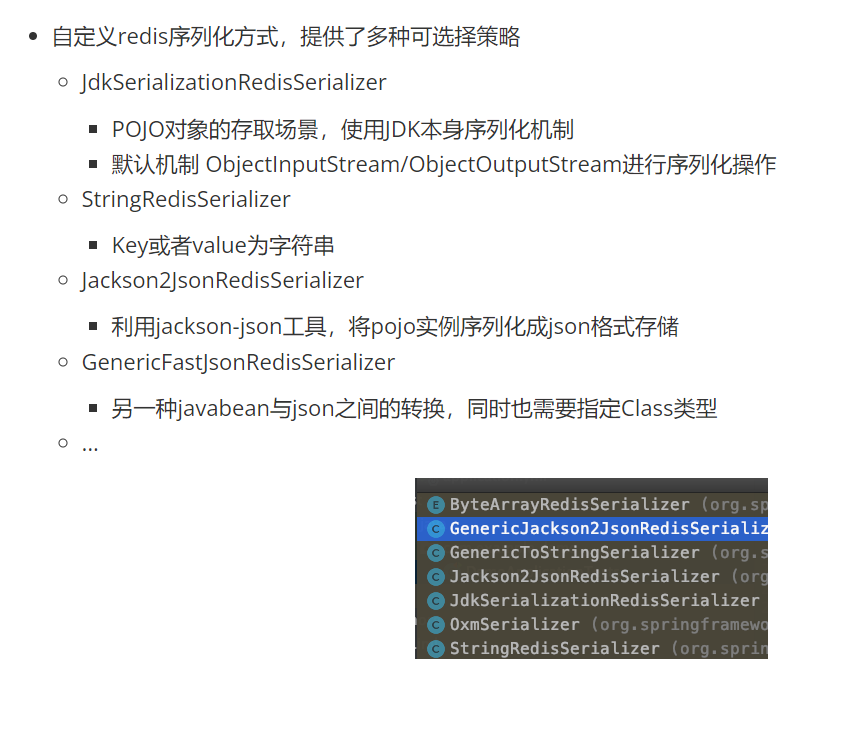
下面讲讲序列化:



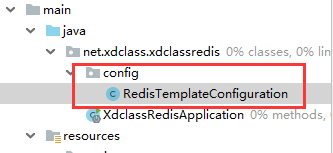
意思也就是说，不同的环境有各自的序列格式，比如java的对象传到go程序上让他变成go对象，那怎么样才能对象互通呢，就在这个序列化的过程，所有机制都离不开字节，字节是最基本的单位，于是java就将对象序列化为字节数组，然后将其传输，go程序获得的也是字节数组，然后用go的序列机制将其反序列化为对象，这样就实现了“对象互通”。（大体上是这个意思，但实际上还有细节可追究，比如java产生的字节数组是怎么能在go上反序列化的，可能是go中有java反序列化的方法？）

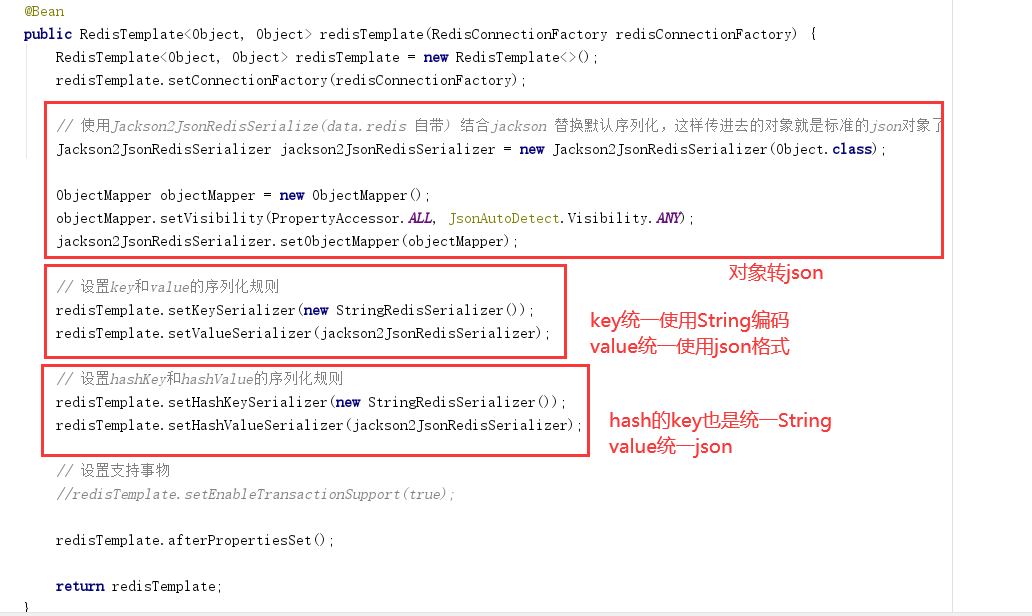
### 2.序列化改造

Redis中提供了很多种序列化方式,常用的可能就是pojo转json这个。



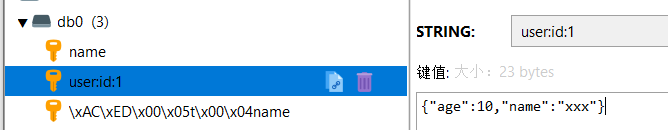
所以现在得对redisTemplate进行封装改造，指定他在存储时的序列化规则（原来默认的都是JDK序列化）:





这样也就不需要对pojo对象使用java自带的Serializable接口了，这里使用的都是json的序列化格式了

这样RedisTemplate再存储任何key进去，都是String格式，任何value进去，都是json格式了:



这样StringRedisTemplate和RedisTemplate的使用在Springboot中就是一模一样的了。

### 3.Luttuce(Jedis)连接池配置，切换：

、

对应解释:



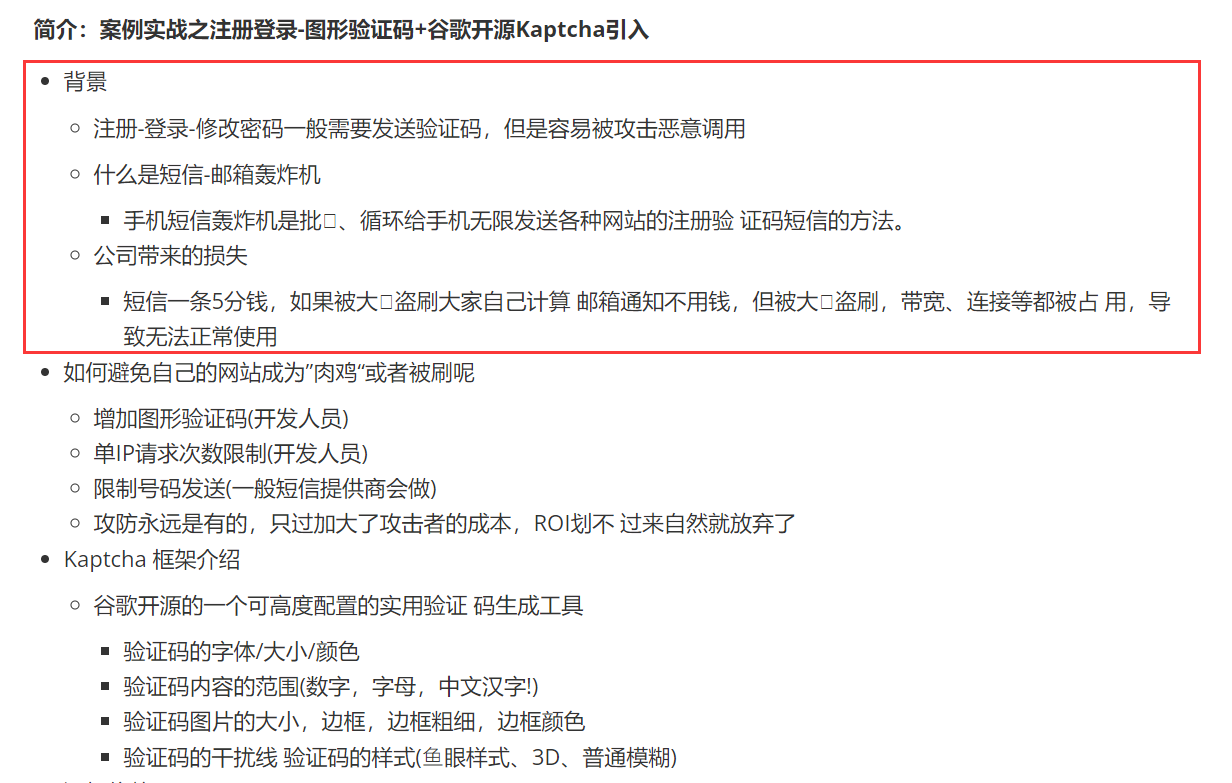
切换成jedis连接池配置:

exclusion中排除lettuce包，添加成jedis依赖即可

也可以配置文件中切换。



# 5.验证码实战



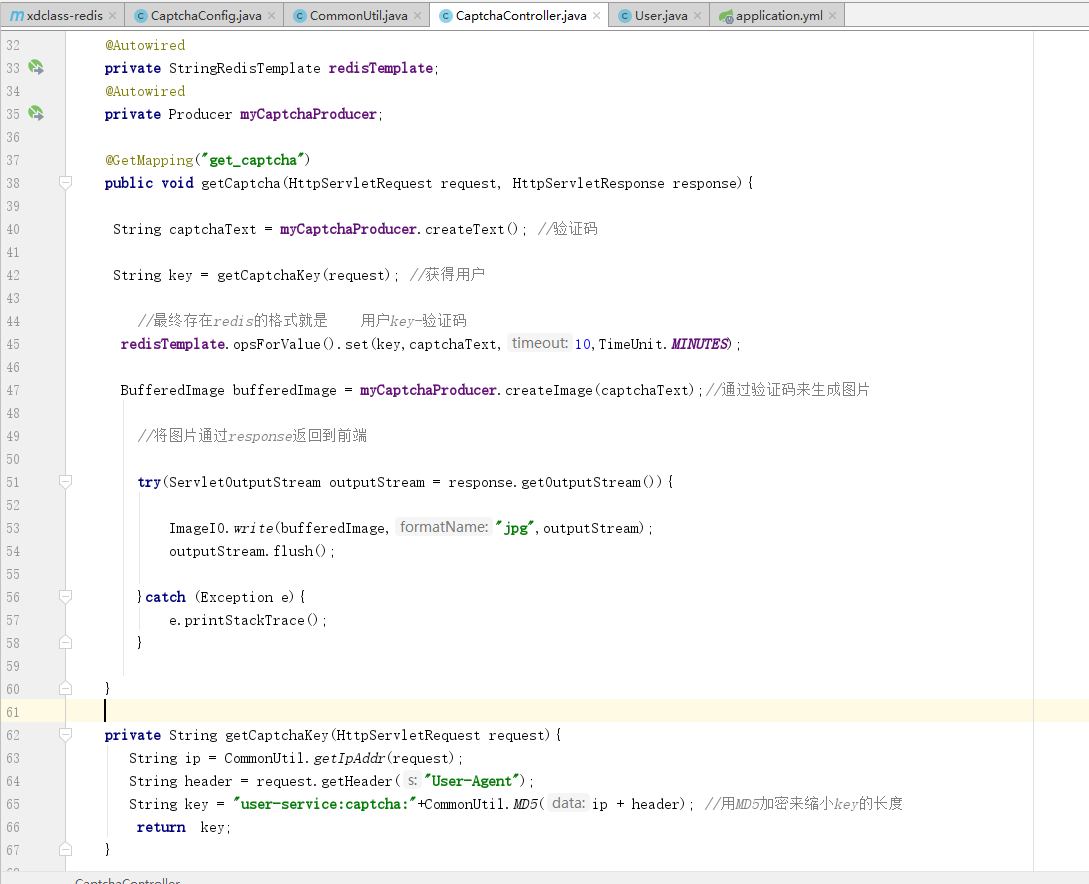
验证码的第一目标是为了防止被攻击。

现在有谷歌的验证码框架，生成很简单，框架介绍中有参数调整方式。



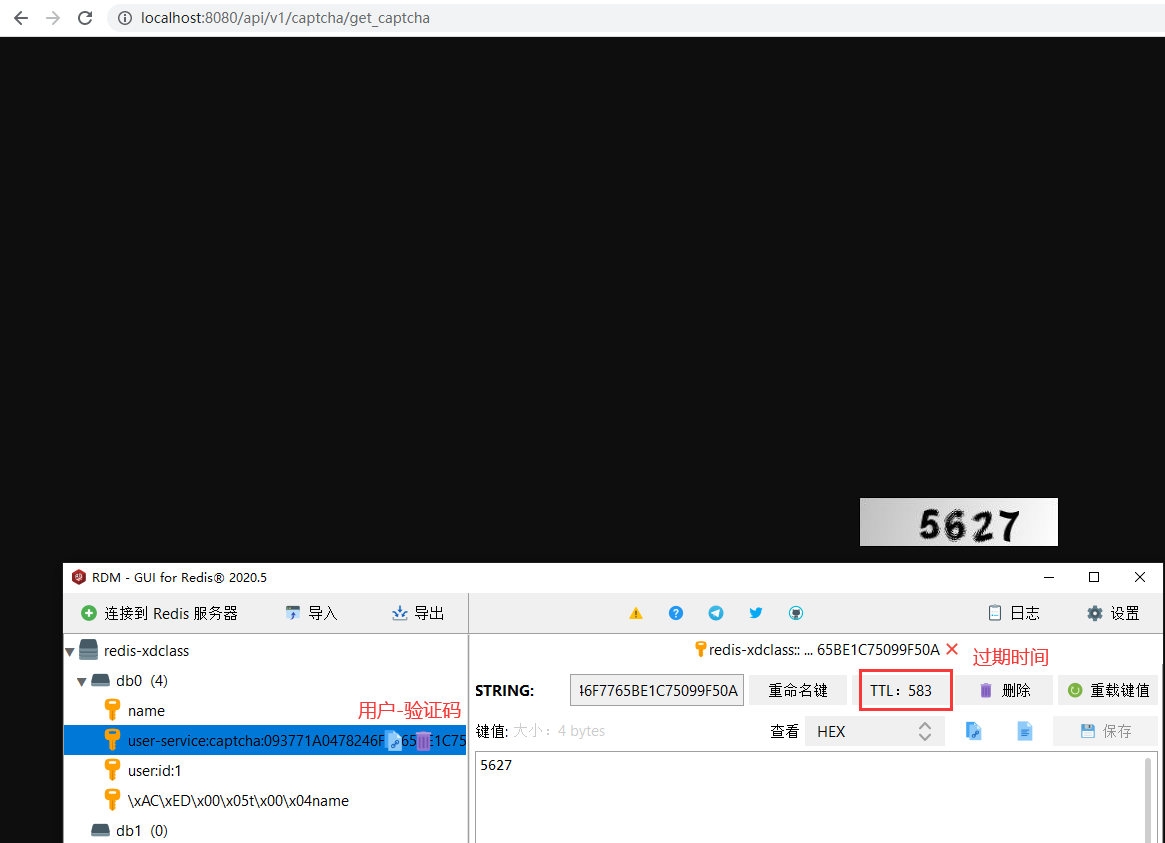
实现一个生成验证码的controller:

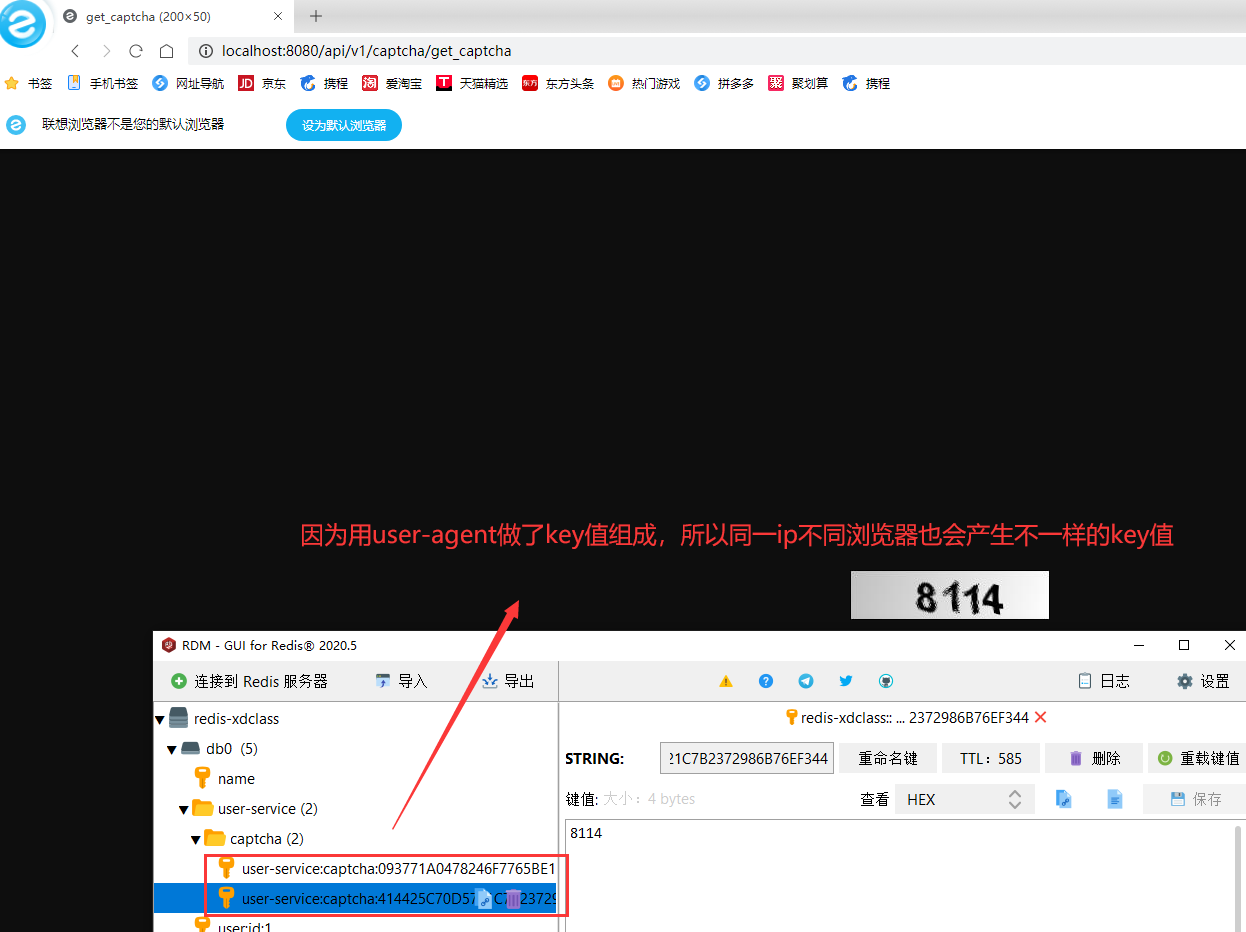
这里的key值是由浏览器user-agent+ip组成的



User-agent做了key值组成部分，同一ip不同浏览器生成的key不一样。

最终就是redis中存一个 用户key-验证码 ，返回前端一个验证码的图片

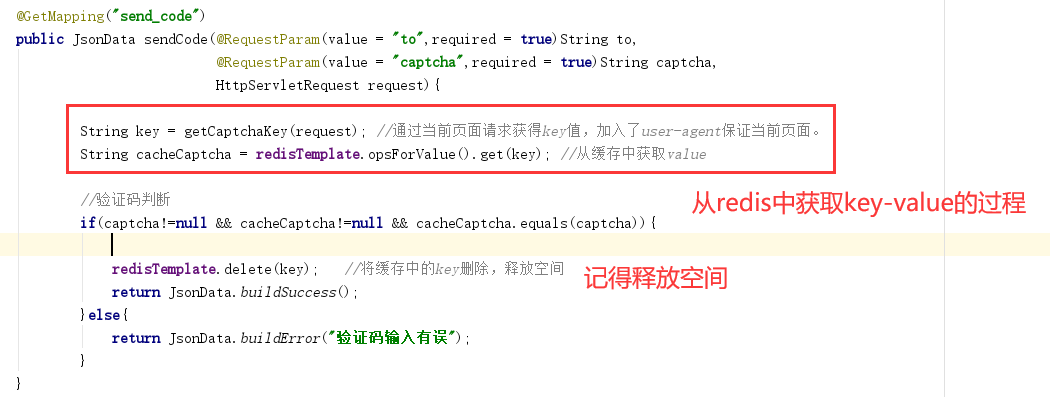




现在到了判断验证码正确环节，

这里的场景是只有第一层验证码通过了，才能发送第二层注册验证码:





注意上面是get的方式获取。RequestParam是获取?后面的参数，required设置必须有，否则请求不到。

验证，正确为4304.

