**Redis**

# Redis命令描述

Redis 命令用于在 redis 服务上执行操作。

要在 redis 服务上执行命令需要一个 redis 客户端。

Redis 客户端在我们之前下载的的 redis 的安装包中。

Redis支持五种数据类型：string（字符串），hash（哈希），list（列表），set（集合）及zset(sorted set：有序集合)等

## Redis键key

**Redis 键命令用于管理 redis 的键**

[DEL key](http://www.runoob.com/redis/keys-del.html)  
该命令用于在 key 存在时删除 key。

DUMP key  
序列化给定 key ，并返回被序列化的值。

[EXISTS key](http://www.runoob.com/redis/keys-exists.html)   
检查给定 key 是否存在。

[EXPIRE key](http://www.runoob.com/redis/keys-expire.html) seconds  
为给定 key 设置过期时间（以秒计）。

[PEXPIRE key milliseconds](http://www.runoob.com/redis/keys-pexpire.html)   
设置 key 的过期时间以毫秒计。

[TTL key](http://www.runoob.com/redis/keys-ttl.html)   
以秒为单位，返回给定 key 的剩余生存时间(TTL, time to live)  [PTTL key](http://www.runoob.com/redis/keys-pttl.html)   
以毫秒为单位返回 key 的剩余的过期时间。

[PERSIST key](http://www.runoob.com/redis/keys-persist.html)   
移除 key 的过期时间，key 将持久保持。

[KEYS pattern](http://www.runoob.com/redis/keys-keys.html)   
查找所有符合给定模式( pattern)的 key 。

keys 通配符 获取所有与pattern匹配的key,返回所有与该匹配

通配符：

\* 代表所有

？ 表示代表一个字符

[RENAME key newkey](http://www.runoob.com/redis/keys-rename.html)

修改Key的名称

[MOVE key db](http://www.runoob.com/redis/keys-move.html)   
将当前数据库的 key 移动到给定的数据库 db 当中

[TYPE key](http://www.runoob.com/redis/keys-type.html)   
返回 key 所储存的值的类型

## 应用场景

[**EXPIRE key**](http://www.runoob.com/redis/keys-expire.html)**seconds**  
1、限时的优惠活动信息

2、网站数据缓存（对于一些需要定时更新的数据，例如：积分排行榜）

3、手机验证码

4、限制网站访客访问频率（例如：1分钟最多访问10次）

## Key的命名建议

非关系数据库之间（NOSQL）

数据与数据之间是没有任何关联的。通过命令来解决。

redis单个key 存入512M大小

1.key不要太长，尽量不要超过1024字节，这不仅消耗内存，而且会降低查找的效率；  
2.key也不要太短，太短的话，key的可读性会降低；  
3.在一个项目中，key最好使用统一的命名模式（用冒号:来分割），

例如users:1:name 伍华锋 //这个key：表示的是Users表下id为1的name为伍华锋。

4.key名称区分大小写

# Redis数据类型

## String

### 简介

string是redis最基本的类型，一个key对应一个value。

string类型是二进制安全的（因为redis是c语言编写的。传输的时候是使用的二进制进行传输的 ）。意思是redis的string可以包含任何数据。比如jpg图片或者序列化的对象 。

string类型是Redis最基本的数据类型，一个键最大能存储512MB。

二进制安全是指，在传输数据时，保证二进制数据的信息安全，也就是不被篡改、破译等，如果被攻击，能够及时检测出来

**二进制安全特点：**

1、编码、解码发生在客户端完成，执行效率高

2、不需要频繁的编解码，不会出现乱码

### String命令

**赋值语法：**

**SET KEY\_NAME VALUE**

Redis SET 命令用于设置给定 key 的值。如果 key 已经存储值， SET 就覆写旧值，且无视类型

**SETNX key value //解决分布式锁 方案之一**

只有在 key 不存在时设置 key 的值。Setnx（**SET** if **N**ot e**X**ists） 命令在指定的 key 不存在时，为 key 设置指定的值

MSET key value [key value ...]

同时设置一个或多个 key-value 对

**取值语法：**

**GET KEY\_NAME**

Redis GET命令用于获取指定 key 的值。如果 key 不存在，返回 nil （等同于java中的null）。如果key 储存的值不是字符串类型，返回一个错误。

GETRANGE key start end

用于获取存储在指定 key 中字符串的子字符串。字符串的截取范围由 start 和 end 两个偏移量决定(包括 start 和 end 在内)

GETBIT key offset

对 key 所储存的字符串值，获取指定偏移量上的位(bit)

MGET key1 [key2..]

获取所有(一个或多个)给定 key 的值

GETSET语法： GETSET KEY\_NAME VALUE

Getset 命令用于设置指定 key 的值，并返回 key 的旧值,当 key 不存在时，返回 nil

STRLEN key

返回 key 所储存的字符串值的长度

**删除语法：**

DEL KEY\_Name

删除指定的KEY，如果存在，返回值数字类型。

**自增/自减：**

**INCR KEY\_Name**

Incr 命令将 key 中储存的数字值增1。如果 key 不存在，那么 key 的值会先被初始化为 0 ，然后再执行 INCR 操作

**自增：INCRBY KEY\_Name 增量值**

 Incrby 命令将 key 中储存的数字加上指定的增量值

**自减：DECR KEY\_NAME 或 DECYBY KEY\_NAME 减值**

 decR 命令将 key 中储存的数字减1

字符串拼接：APPEND KEY\_NAME VALUE

 Append 命令用于为指定的 key 追加至未尾，如果不存在，为其赋值

### 应用场景：

**1、String通常用于保存单个字符串或JSON字符串数据**

2、**因String是二进制安全的**，所以你完全可以把一个图片文件的内容作为字符串来存储

3、**计数器**（常规key-value缓存应用。常规计数: 微博数, 粉丝数）

**INCR等指令本身就具有原子操作的特性**，所以我们完全可以利用redis的INCR、INCRBY、DECR、DECRBY等指令来实现原子计数的效果。假如，在某种场景下有3个客户端同时读取了mynum的值（值为2），然后对其同时进行了加1的操作，那么，最后mynum的值一定是5。

不少网站都利用redis的这个特性来实现业务上的统计计数需求。

## 哈希(Hash)

### 简介

Redis hash 是一个string类型的field和value的映射表，hash特别适合用于存储对象。 Redis 中每个 hash 可以存储 232 - 1 键值对（40多亿）

可以看成具有KEY和VALUE的MAP容器，该类型非常适合于存储值对象的信息，

如：uname,upass,age等。该类型的数据仅占用很少的磁盘空间（相比于JSON）

### Hash命令

**赋值语法：**

HSET KEY FIELD VALUE //为指定的KEY，设定FILD/VALUE

**HMSET KEY FIELD VALUE [FIELD1,VALUE1]…… 同时将多个 field-value (域-值)对设置到哈希表 key 中。** **（其实这里是一个javaBean对象）**



**取值语法：**

HGET KEY FIELD //获取存储在HASH中的值，根据FIELD得到VALUE

HMGET key field[field1] //获取key所有给定字段的值

HGETALL key //返回HASH表中所有的字段和值

HKEYS key //获取所有哈希表中的字段

HLEN key //获取哈希表中字段的数量

**删除语法：**

HDEL KEY field1[field2] //删除一个或多个HASH表字段

**其它语法：**

HSETNX key field value

只有在字段 field 不存在时，设置哈希表字段的值

HINCRBY key field increment

为哈希表 key 中的指定字段的整数值加上增量 increment 。

HINCRBYFLOAT key field increment

为哈希表 key 中的指定字段的浮点数值加上增量 increment 。

HEXISTS key field //查看哈希表 key 中，指定的字段是否存在

### 应用场景：

Hash的应用场景：（存储一个用户信息对象数据）

1. **常用于存储一个对象**
2. **为什么不用string存储一个对象？**

hash是最接近关系数据库结构的数据类型，可以将数据库一条记录或程序中一个对象转换成hashmap存放在redis中。

用户ID为查找的key，存储的value用户对象包含姓名，年龄，生日等信息，如果用普通的key/value结构来存储，主要有以下2种存储方式：

    第一种方式将用户ID作为查找key,把其他信息封装成一个对象以序列化的方式存储，这种方式的缺点是，增加了序列化/反序列化的开销，并且在需要修改其中一项信息时，需要把整个对象取回，并且修改操作需要对并发进行保护，引入CAS等复杂问题。

    第二种方法是这个用户信息对象有多少成员就存成多少个key-value对儿，用用户ID+对应属性的名称作为唯一标识来取得对应属性的值，虽然省去了序列化开销和并发问题，但是用户ID为重复存储，如果存在大量这样的数据，内存浪费还是非常可观的。

**总结：**

**Redis提供的Hash很好的解决了这个问题，Redis的Hash实际是内部存储的Value为一个HashMap，并提供了直接存取这个Map成员的接口。**

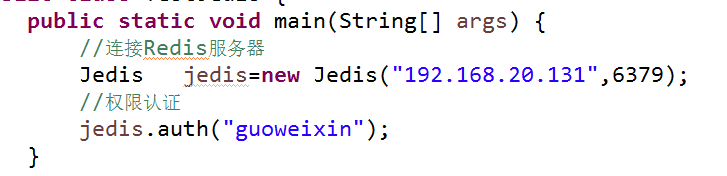
# Java连接Redis

在官方网站列一些Java客户端访问，有：Jedis/Redisson/Jredis/JDBC-Redis等，其中官方推荐使用Jedis和Redisson。常用**Jedis**。

开始在 Java 中使用 Redis 前， 我们需要确保已经安装了 redis 服务及 Java redis 驱动，且你的机器上能正常使用 Java。 Java的安装配置可以参考我们的 Java开发环境配置 接下来让我们安装 Java redis 驱动

安装相应JAR：

|  |
| --- |
| <!-- Redis -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>2.4.2</version>  </dependency> |



上面的这个权限：是4.0版本之后需要进行密码认证。

成功之后返回“Pong”

## 开启端口防火墙

**开放端口（如下命令只针对Centos7以上）**

查看已经开放的端口：firewall-cmd --list-ports

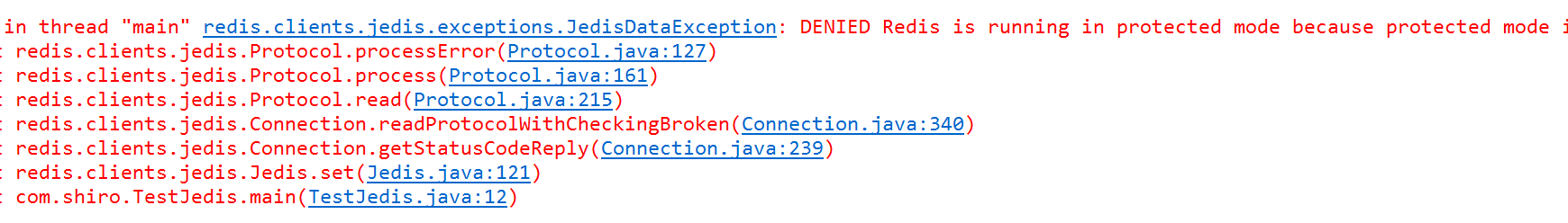
开启端口:

firewall-cmd --zone=public --add-port=6379/tcp --permanent

重启防火墙

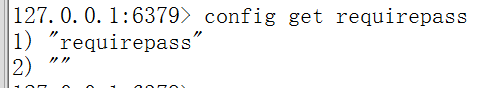
firewall-cmd --reload #重启

## Java操作Redis 设置密码

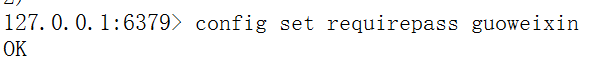


这个问题是由于**[Redis](http://lib.csdn.net/base/redis)**没有配置密码的原因导致的，只需要为**[redis](http://lib.csdn.net/base/redis)**设置密码即可

config get requirepass: 这是查询redis是否配置密码,如果返回为空，则表明未配置密码



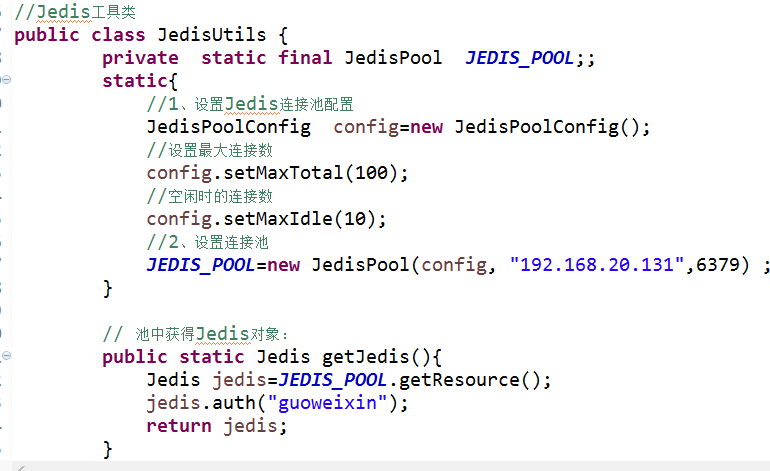
config set requirepass “guoweixin”这是将redis的密码设置为“guoweixin”



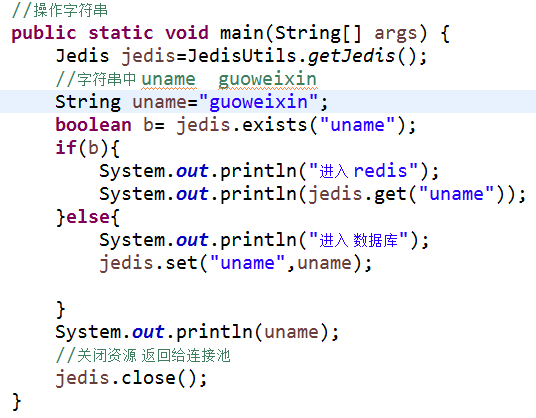
**客户端登录**：用redis-cli 密码登陆（**redis-cli -a  password**）

## Java Jedis连接池优化

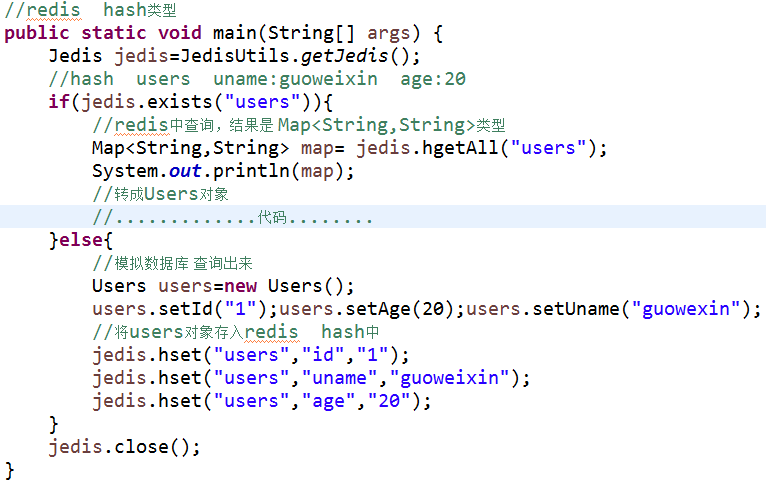




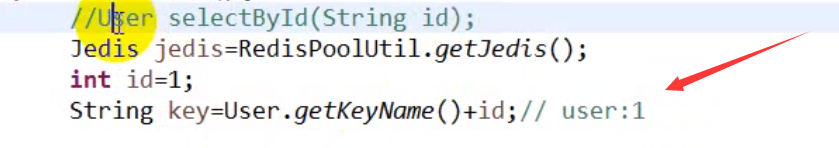
## 案例：Jedis操作Redis string 和hash

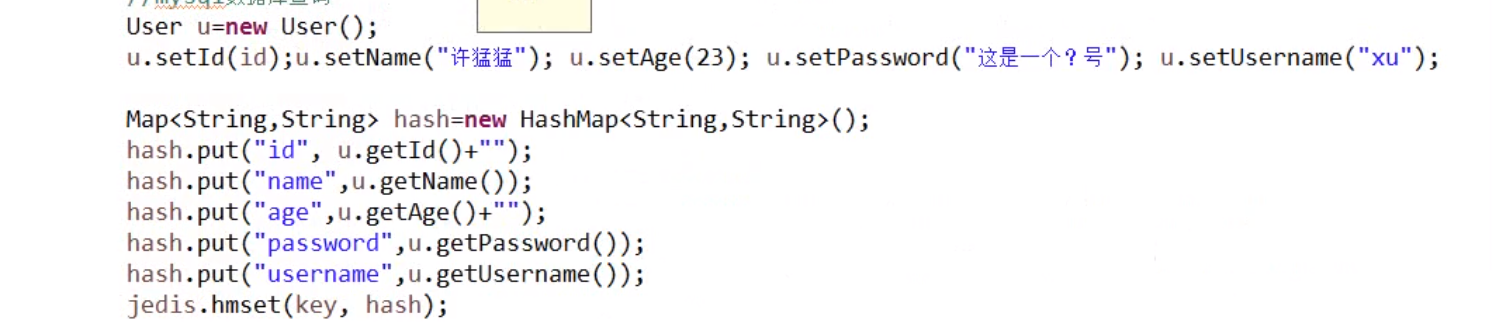


----------------------------



上面是一个一个的存，下面是一下存多个。





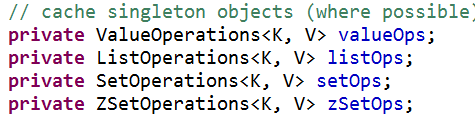
# RedisTemplate

## 简介

Spring data 提供了RedisTemplate模版

它封装了redis连接池管理的逻辑，业务代码无须关心获取，释放连接逻辑；spring redis同时支持了Jedis，Jredis,rjc 客户端操作；

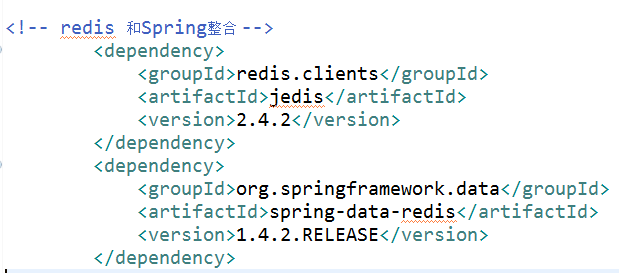
在RedisTemplate中提供了几个常用的接口方法的使用，分别是





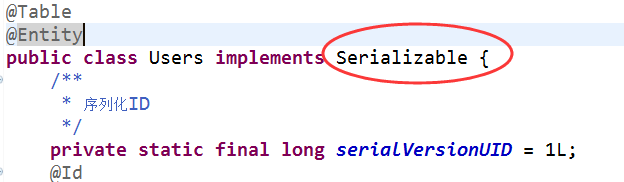
## Spring data 使用RedisTemplate模版

1、jar：Redis和Spring整合



|  |
| --- |
| <!-- redis 和Spring整合 -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>2.4.2</version>  </dependency>  <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-redis</artifactId>  <version>1.4.2.RELEASE</version>  </dependency> |

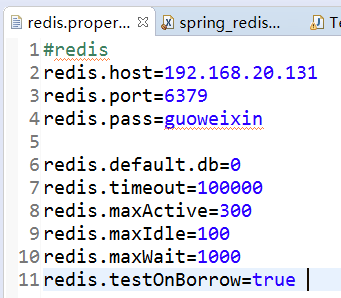
2、对应实体Bean进行序列化操作



3、编写相应配置文件  
利用RedisTemplate操作Redis



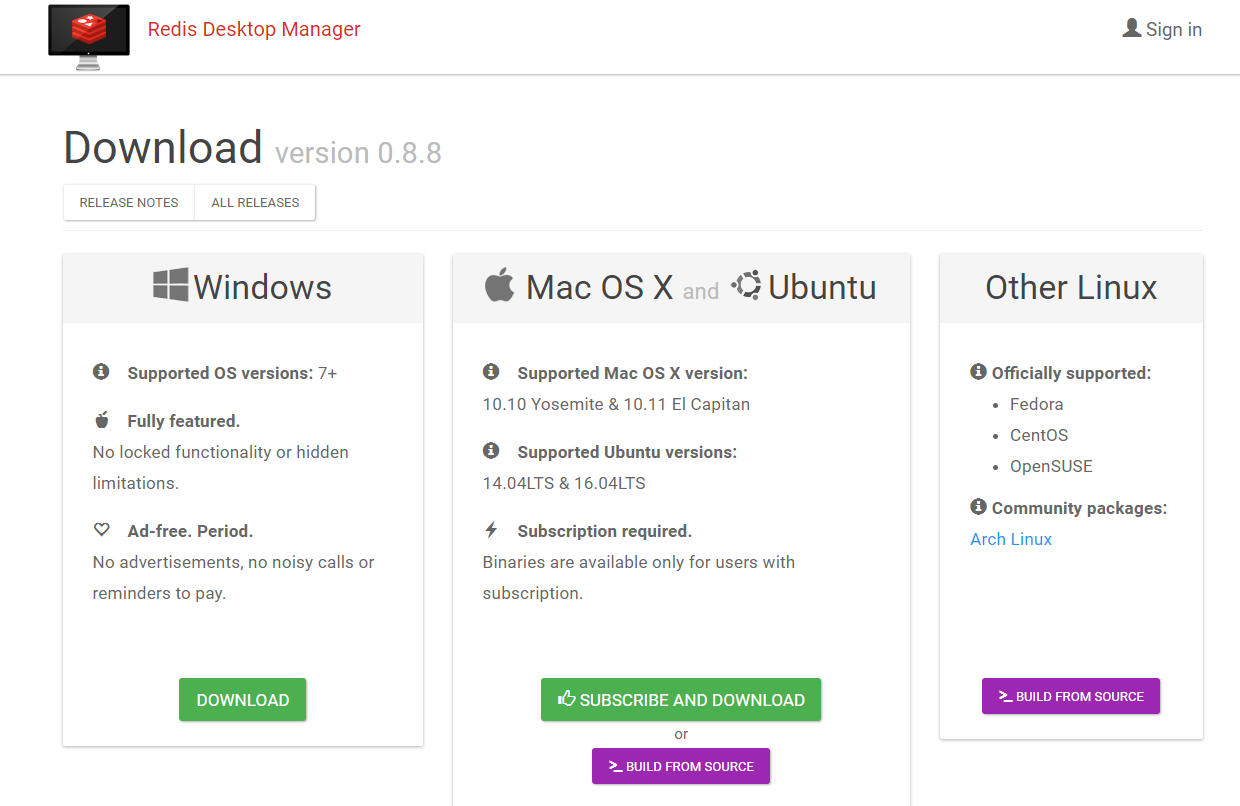
## JedisPoolConfig常用配置信息



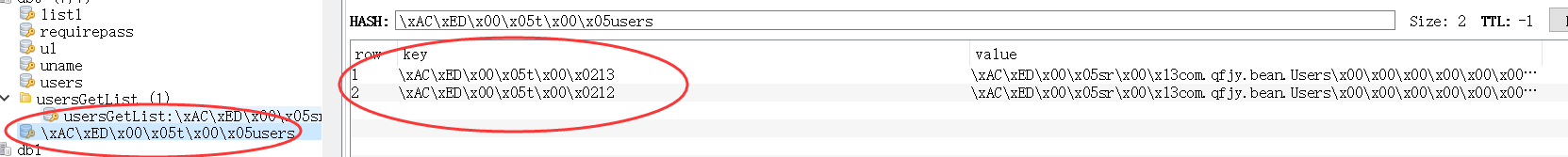
|  |
| --- |
| * #最大空闲数 * redis.maxIdle=300 * #连接池的最大数据库连接数。设为0表示无限制,如果是jedis 2.4以后用redis.maxTotal * #redis.maxActive=600 * #控制一个pool可分配多少个jedis实例,用来替换上面的redis.maxActive,如果是jedis 2.4以后用该属性 * redis.maxTotal=1000 * #最大建立连接等待时间。如果超过此时间将接到异常。设为-1表示无限制。 * redis.maxWaitMillis=1000 * #连接的最小空闲时间 默认1800000毫秒(30分钟) * redis.minEvictableIdleTimeMillis=300000 * #每次释放连接的最大数目,默认3 * redis.numTestsPerEvictionRun=1024 * #逐出扫描的时间间隔(毫秒) 如果为负数,则不运行逐出线程, 默认-1 * redis.timeBetweenEvictionRunsMillis=30000 * #是否在从池中取出连接前进行检验,如果检验失败,则从池中去除连接并尝试取出另一个 * redis.testOnBorrow=true * #在空闲时检查有效性, 默认false * redis.testWhileIdle=true |

## Redis客户端工具

https://redisdesktop.com/download

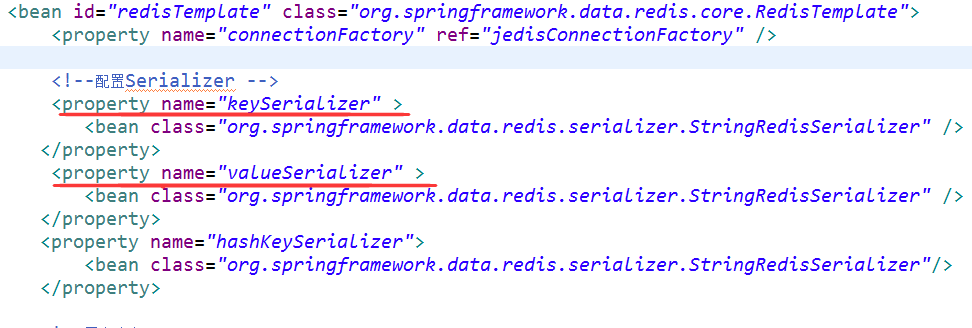


### 查看Redis客户端



原因 ：把任何数据保存到redis中时，都需要进行序列化，默认使用JdkSerializationRedisSerializer进行数据序列化。

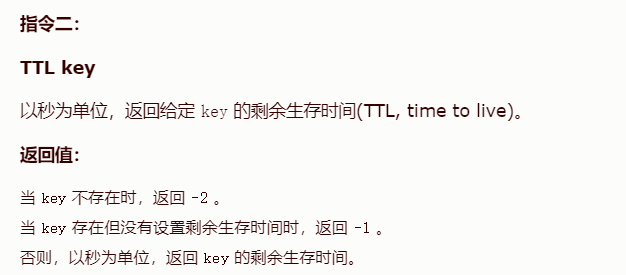
所有的key和value还有hashkey和hashvalue的原始字符前，都加了一串字符。



# 案例作业

## 限制登录功能

用户在2分钟内，仅允许输入错误密码5次。如果超过次数，限制其登录1小时。（要求每登录失败时，都要给相应提式）



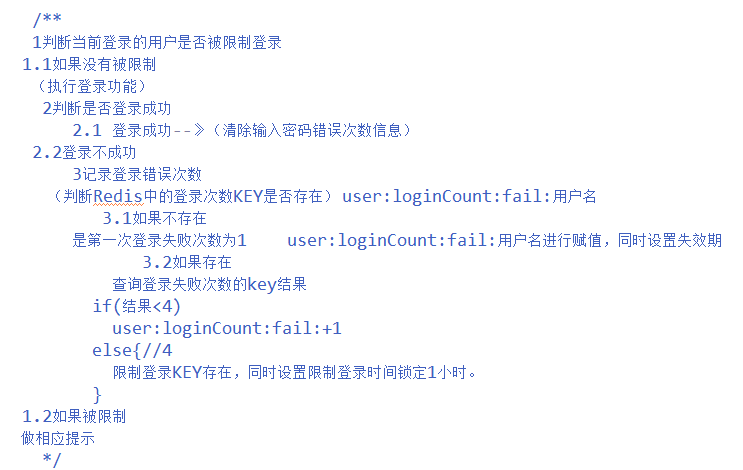
### 基本代码1：





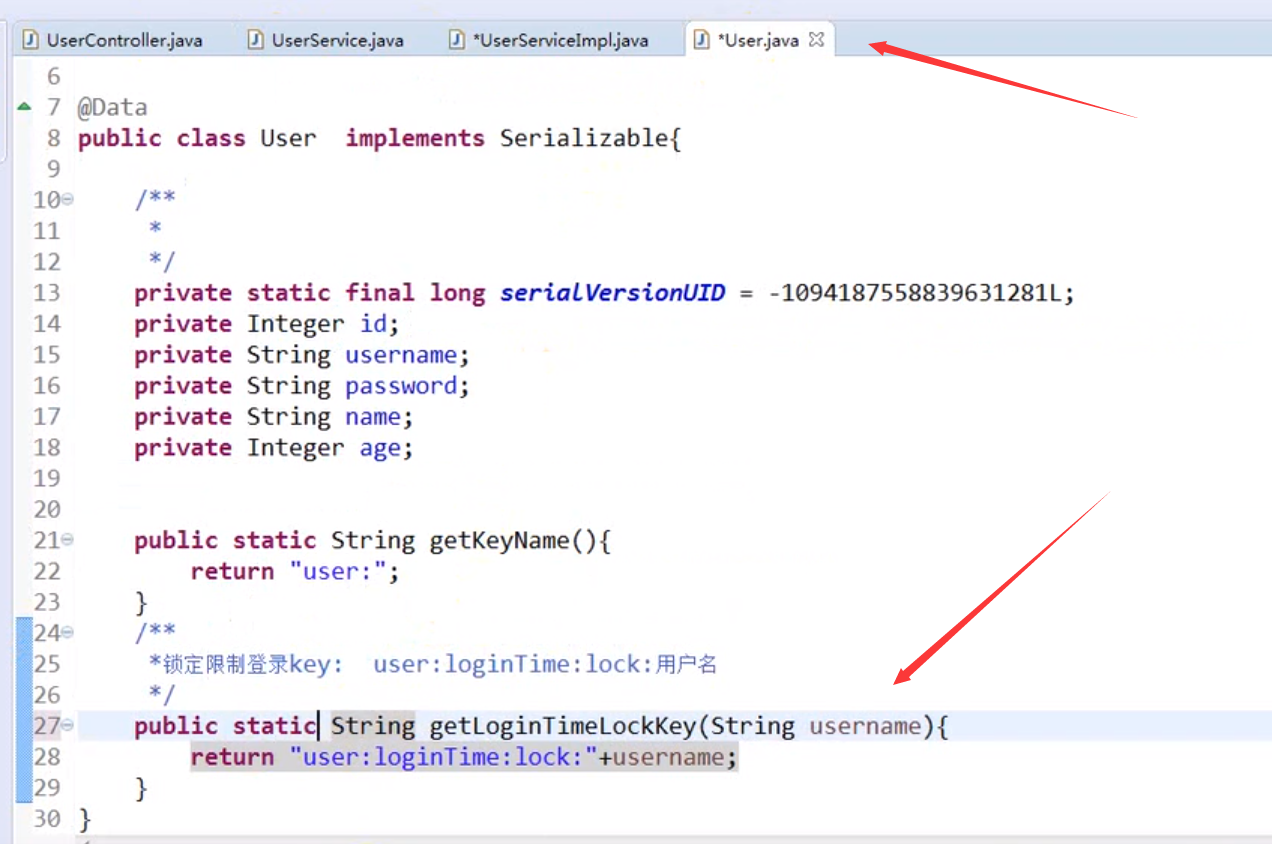
### 完整代码2：

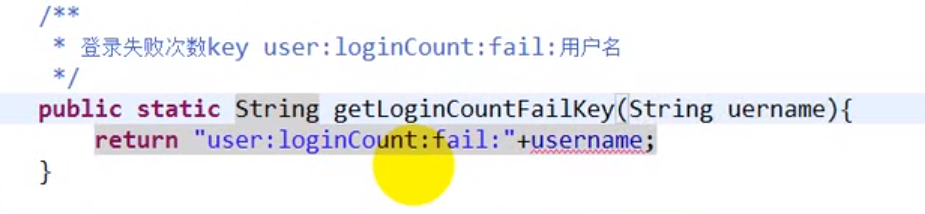
##### 需求分析：



##### 业务层编写

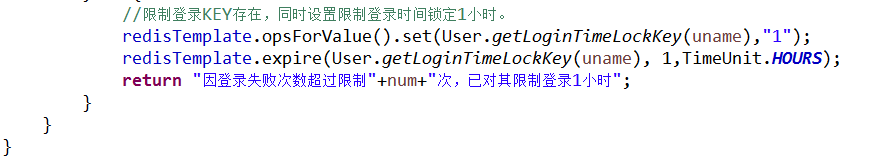
User实体类的代码：











##### Controller层

注意：下面的final：表示不能被修改。

验证码比较的代码省略了。



## 手机验证功能

用户在客户端输入手机号，点击发送后随机生成4位数字码。有效期为90秒。

输入验证码，点击验证，返回成功或者失败。且每个手机号在一天内只能验证3次。并给相应信息提示