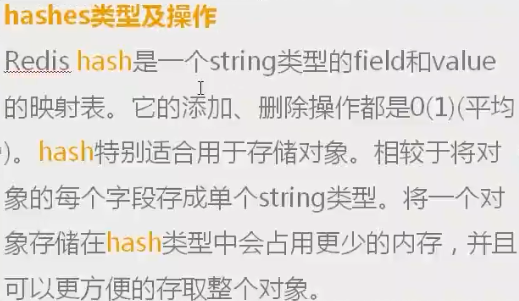
Redis的数据类型之hash

1. 第二种数据类型：hash：

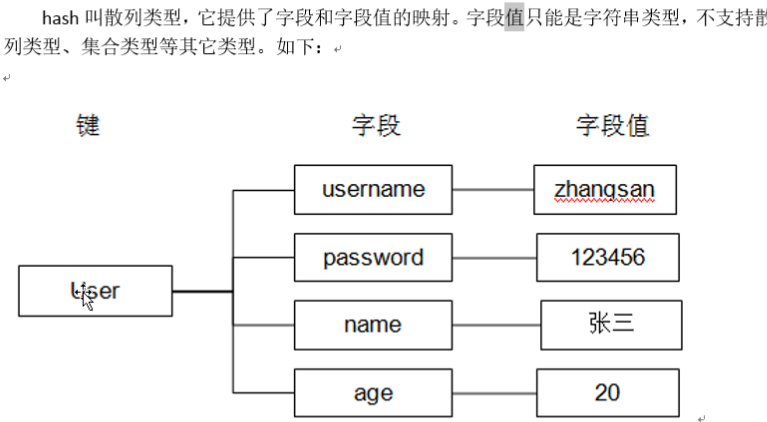


1. **hash类型，又称散列类型：**

假设有User对象以JSON序列化的形式存储到Redis中，User对象有id，username、password、age、name等属性，存储的过程如下：

**保存、更新： User对象 –> json(string) -> redis**

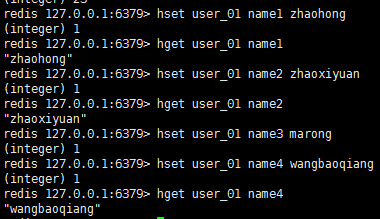
如果在业务上只是更新age属性，其他的属性并不做更新我应该怎么做呢？ 如果仍然采用上边的方法在**传输、处理时会造成资源浪费，下边讲的hash可以很好的解决这个问题。**



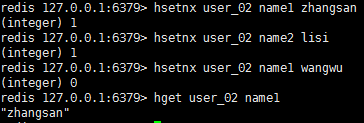
1. **关于set的方法：**

（1）**hset** : 设置hash field 为指定值，如果key不存在，则先创建。设置成功则返回1，否则返回0。

**hset hashtablename field1 value**



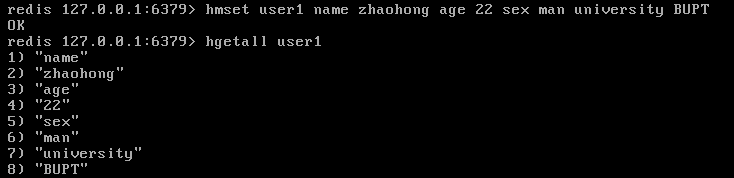
（2）**hsetnx ： 设置成功返回1，否则返回0.**



（3）**hmset** ： 批处理，同时设置多个。

**hmset hashtablename field1 value field2 value2 field3 value3 …**

**一个POJO类，首先封装成json，然后变成hash表。**



注意没有 hmsetnx这个命令。

1. **关于get的方法**：
2. **hget** ： 获取key对应的值。

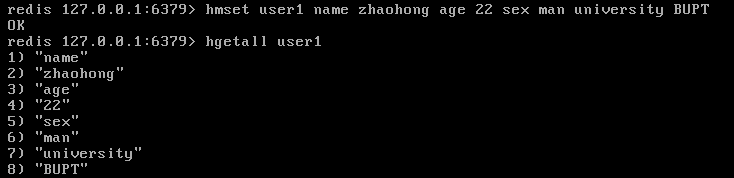


1. **hmget** ： 获取多个key对应的值。



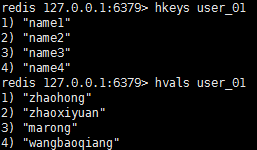
(3) hgetall ：**获取某个hash中全部的field和value。**

**一个POJO类，首先封装成json，然后变成hash表。**

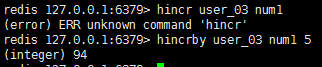


**（4）hkeys：返回hash中所有的key。**

**hvals：返回hash中所有的value。**

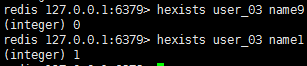


1. **hincrby** ：增量。对hash上某个field对应的value增加一定值。返回增量后的value值。



**没有hincr方法。**

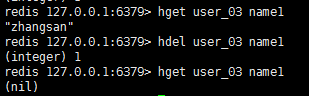
1. **hexists**： 判断hash中某个field是否存在。存在则返回1，否则返回0.



1. **hlen** ： 获取hash中几个field，返回个数。

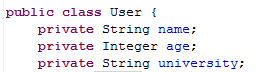


1. **hdel** ： 删除hash中指定的field。删除成功返回1，否则返回0.



1. 注意：hash中没有 hmsetnx 、hsetex、hsetrange、hincr等方法。
2. **hash类型的应用**：

在java中创建一个POJO类，如User：



中间经过封装成json，然后如何存到redis中呢？

命令 ：添加和获取

hmset user1 name zhaohong age 22 university BUPT

hgetall user1

修改某一个属性：

hset user1 name zhaoxiyuan