package redisTest.test;

import redis.clients.jedis.Jedis;

import redis.clients.jedis.JedisPool;

import redis.clients.jedis.JedisPoolConfig;

import redis.clients.jedis.ShardedJedis;

import redis.clients.jedis.ShardedJedisPool;

public class RedisUtil {

//非切片客户端连接（单机）

private Jedis jedis;

//非切片客户端连接池（单机）

private JedisPool jedisPool;

//切片客户端连接（集群）

private ShardedJedis shardedJedis;

//切片客户端连接池（集群）

private ShardedJedisPool shardedJedisPool;

//初始化非切片连接池

private void initJedisPool(){

// 池基本配置

        JedisPoolConfig config = new JedisPoolConfig();

        config.setMaxActive(20);

        config.setMaxIdle(5);

        config.setMaxWait(1000l);

        config.setTestOnBorrow(false);

        jedisPool = new JedisPool(config,"218.61.16.188",6379);

}

//初始化非切片客户端

private void initJedis(){

jedis = new Jedis("218.61.16.188",6379);

}

public RedisUtil(){

initJedisPool();

initJedis();

}

public Jedis getJedis() {

return jedis;

}

public void setJedis(Jedis jedis) {

this.jedis = jedis;

}

public JedisPool getJedisPool() {

return jedisPool;

}

public void setJedisPool(JedisPool jedisPool) {

this.jedisPool = jedisPool;

}

public ShardedJedis getShardedJedis() {

return shardedJedis;

}

public void setShardedJedis(ShardedJedis shardedJedis) {

this.shardedJedis = shardedJedis;

}

public ShardedJedisPool getShardedJedisPool() {

return shardedJedisPool;

}

public void setShardedJedisPool(ShardedJedisPool shardedJedisPool) {

this.shardedJedisPool = shardedJedisPool;

}

}

**package** redisTest.test;

**import** redis.clients.jedis.Jedis;

**public** **class** RedisSet {

**public** **void** SetOperate(Jedis jedis){

jedis.flushDB();

//添加

        jedis.sadd("user","liuling");

        jedis.sadd("user","xinxin");

        jedis.sadd("user","ling");

        jedis.sadd("user","zhangxinxin");

        jedis.sadd("user","who");

        //移除noname

        jedis.srem("user","who");

        System.***out***.println(jedis.smembers("user"));//获取所有加入的value

        System.***out***.println(jedis.sismember("user", "who"));//判断 who 是否是user集合的元素

        System.***out***.println(jedis.srandmember("user"));

        System.***out***.println(jedis.scard("user"));//返回集合的元素个数

}

**public** **static** **void** main(String[] a) {

RedisSet rm = **new** RedisSet();

RedisUtil ru = **new** RedisUtil();

rm.SetOperate(ru.getJedis());

}

}

package redisTest.test;

import java.util.Iterator;

import java.util.Set;

import redis.clients.jedis.Jedis;

public class RedisString {

public void KeyOperate(Jedis jedis){

//清空数据

jedis.flushDB();

//添加一个key

jedis.set("redisString", "我是redisString");

//获取一个key

String redisString = jedis.get("redisString");

System.out.println(redisString);

//判断某个key是否存在

System.out.println(jedis.exists("redisString"));

//获取所有key

Set<String> keySet = jedis.keys("\*");

Iterator<String> keyIter = keySet.iterator();

while(keyIter.hasNext()){

String key = keyIter.next();

System.out.println(key);

}

//查看key的类型

String type = jedis.type("redisString");

System.out.println(type);

//删除key

long r = jedis.del("redisString");

System.out.println(r);

//新增多个

jedis.mset("str1","str1","str2","str2");

//获取多个

jedis.mget("str1","str2");

//删除多个

jedis.del(new String[]{"str1","str2"});

}

public static void main(String[] a){

RedisUtil r = new RedisUtil();

RedisString rk = new RedisString();

rk.KeyOperate(r.getJedis());

}

}

**package** redisTest.test;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Map;

**import** redis.clients.jedis.Jedis;

**public** **class** RedisMap {

**public** **void** MapOperate(Jedis jedis) {

Map<String, String> map = **new** HashMap<String, String>();

map.put("name", "xinxin");

map.put("age", "22");

map.put("qq", "123456");

jedis.hmset("user", map);

// 取出user中的name，执行结果:[minxr]-->注意结果是一个泛型的List

// 第一个参数是存入redis中map对象的key，后面跟的是放入map中的对象的key，后面的key可以跟多个，是可变参数

List<String> rsmap = jedis.hmget("user", "name", "age", "qq");

System.***out***.println(rsmap);

// 删除map中的某个键值

jedis.hdel("user", "age");

System.***out***.println(jedis.hmget("user", "age")); // 因为删除了，所以返回的是null

System.***out***.println(jedis.hlen("user")); // 返回key为user的键中存放的值的个数2

System.***out***.println(jedis.exists("user"));// 是否存在key为user的记录 返回true

System.***out***.println(jedis.hkeys("user"));// 返回map对象中的所有key

System.***out***.println(jedis.hvals("user"));// 返回map对象中的所有value

Iterator<String> iter = jedis.hkeys("user").iterator();

**while** (iter.hasNext()) {

String key = iter.next();

System.***out***.println(key + ":" + jedis.hmget("user", key));

}

}

**public** **static** **void** main(String[] a) {

RedisMap rm = **new** RedisMap();

RedisUtil ru = **new** RedisUtil();

rm.MapOperate(ru.getJedis());

}

}

**package** redisTest.test;

**import** redis.clients.jedis.Jedis;

**public** **class** RedisList {

**public** **void** ListOperate(Jedis jedis) {

// 先向key java framework中存放三条数据

//lpush 在最前插入

jedis.lpush("java framework", "spring");

jedis.lpush("java framework", "struts");

jedis.lpush("java framework", "hibernate");

// 再取出所有数据jedis.lrange是按范围取出，

// 第一个是key，第二个是起始位置，第三个是结束位置，jedis.llen获取长度 -1表示取得所有

System.***out***.println(jedis.lrange("java framework", 0, -1));

jedis.del("java framework");

//rpush在最后插入

jedis.rpush("java framework", "spring");

jedis.rpush("java framework", "struts");

jedis.rpush("java framework", "hibernate");

System.***out***.println(jedis.lrange("java framework", 0, -1));

}

**public** **static** **void** main(String[] a) {

RedisList rm = **new** RedisList();

RedisUtil ru = **new** RedisUtil();

rm.ListOperate(ru.getJedis());

}

}

**package** redisTest.test;

**import** java.io.ByteArrayInputStream;

**import** java.io.ByteArrayOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** SerializableUtil {

/\*\*

 \* 序列化

 \*/

**public** **static** **byte**[] serializable(Object o){

ObjectOutputStream oos = **null**;

ByteArrayOutputStream baos = **null**;

**try**{

baos = **new** ByteArrayOutputStream();

oos = **new** ObjectOutputStream(baos);

oos.writeObject(o);

**byte**[] bytes = baos.toByteArray();

**return** bytes;

}**catch**(Exception e){

e.printStackTrace();

}

**return** **null**;

}

/\*\*

 \* 反序列化

 \*/

**public** **static** Object unserializable(**byte**[] b){

ByteArrayInputStream bais = **null**;

**try** {

//反序列化

bais = **new** ByteArrayInputStream(b);

ObjectInputStream ois = **new** ObjectInputStream(bais);

**return** ois.readObject();

} **catch** (Exception e) {

}

**return** **null**;

}

/\*\*

 \* 序列化集合

 \*/

**public** **static** **byte**[] serializeList(List<?> list) {

        ObjectOutputStream oos = **null**;

        ByteArrayOutputStream baos = **null**;

**byte**[] bytes = **null**;

**try** {

            baos = **new** ByteArrayOutputStream();

            oos = **new** ObjectOutputStream(baos);

**for** (Object obj : list) {

                oos.writeObject(obj);

            }

            bytes = baos.toByteArray();

        } **catch** (Exception e) {

            e.printStackTrace();

        } **finally** {

**try** {

oos.close();

baos.close();

} **catch** (IOException e) {

e.printStackTrace();

}

        }

**return** bytes;

    }

/\*\*

 \* 反序列化集合

 \*/

**public** **static** List<?> unserializeList(**byte**[] bytes) {

**if** (bytes == **null**) {

**return** **null**;

        }

        List<Object> list = **new** ArrayList<Object>();

        ByteArrayInputStream bais = **null**;

        ObjectInputStream ois = **null**;

**try** {

            // 反序列化

            bais = **new** ByteArrayInputStream(bytes);

            ois = **new** ObjectInputStream(bais);

**while** (bais.available() > 0) {

                Object obj = (Object) ois.readObject();

**if** (obj == **null**) {

**break**;

                }

                list.add(obj);

            }

        } **catch** (Exception e) {

            e.printStackTrace();

        } **finally** {

**try** {

        ois.close();

        bais.close();

} **catch** (IOException e) {

e.printStackTrace();

}

        }

**return** list;

    }

}

**package** redisTest.test;

**import** java.util.ArrayList;

**import** java.util.List;

**import** redis.clients.jedis.Jedis;

**public** **class** RedisObject {

**public** **void** ObjectOperate(Jedis jedis){

jedis.flushDB();

Stu s = **new** Stu();

s.setId(1);

s.setName("穆天扬");

jedis.set("stu\_1".getBytes(), SerializableUtil.*serializable*(s));

Stu newStu =  (Stu)SerializableUtil.*unserializable*(jedis.get("stu\_1".getBytes()));

System.***out***.println(newStu.getId());

System.***out***.println(newStu.getName());

List<Stu> list = **new** ArrayList<>();

Stu stu1 = **new** Stu();

stu1.setId(1);

stu1.setName("穆天扬");

Stu stu2 = **new** Stu();

stu2.setId(2);

stu2.setName("王猛");

Stu stu3 = **new** Stu();

stu3.setId(3);

stu3.setName("刘会俊");

list.add(stu1);

list.add(stu2);

list.add(stu3);

jedis.set("stuList".getBytes(), SerializableUtil.*serializeList*(list));

List<Stu> list1 = (List<Stu>)

SerializableUtil.*unserializeList*(jedis.get("stuList".getBytes()));

**for** (Stu stu : list1) {

System.***out***.println(stu);

}

}

**public** **static** **void** main(String[] a){

RedisObject rm = **new** RedisObject();

RedisUtil ru = **new** RedisUtil();

rm.ObjectOperate(ru.getJedis());

}

}