#### Redis

## 优势和介绍：

支持持久化

能存数据结构（string、hash、list、set、sorted set）

支持数据备份

性能极高：读的速度是110000次/s,写的速度是81000次/s 。

所有操作都是原子性

## 基本配置

需要jar包：jedis-2.2.1-sources.jar

## 初始化

JedisPoolConfig config = new JedisPoolConfig();//资源池配置

config.setMaxIdle(5);//最大等待连接限制

config.setMaxActive(20);//最大连接限制

config.setMaxWait(10000L);//最大等待毫秒限制

JedisPool pool = new JedisPool(config, ip, port, timeout, password);//连接配置

## 设置和取值

//set

Jedis client = pool.getResource();//取得连接

try {

client.connect();//开始连接

return ("OK".equals(client.set(key, value)));//设置并返回结果

} finally {

pool.returnResource(client);//资源释放，3.0用jedis.close();

}

//get

Jedis client = pool.getResource();//取得连接

try {

client.connect();//开始连接

byte[] obj = client.get(key.getBytes());//返回结果

if (obj == null) {

return null;

} else {

return KryoPoolSerializerUtil.deserialize(obj);

}

} finally {

pool.returnResource(client);

}

public class SerializeUtil {

public static void main(String [] args){

Jedis jedis = new Jedis("172.16.135.2");

String keys = "name";

// 删数据

//jedis.del(keys);

// 存数据

jedis.set(keys, "zy");

// 取数据

String value = jedis.get(keys);

System.out.println(value);

//存对象

Person p=new Person(); //peson类记得实现序列化接口 Serializable

p.setAge(20);

p.setName("姚波");

p.setId(1);

jedis.set("person".getBytes(), serialize(p));

byte[] byt=jedis.get("person".getBytes());

Object obj=unserizlize(byt);

if(obj instanceof Person){//判断是否为一个实例

System.out.println(obj);

}

}

//序列化

public static byte [] serialize(Object obj){

ObjectOutputStream obi=null;

ByteArrayOutputStream bai=null;

try {

bai=new ByteArrayOutputStream();

obi=new ObjectOutputStream(bai);

obi.writeObject(obj);

byte[] byt=bai.toByteArray();

return byt;

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

//反序列化

public static Object unserizlize(byte[] byt){

ObjectInputStream oii=null;

ByteArrayInputStream bis=null;

bis=new ByteArrayInputStream(byt);

try {

oii=new ObjectInputStream(bis);

Object obj=oii.readObject();

return obj;

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

}