此文章已于 17:50:36 2017/12/5 发布到 HealerJean梦想博客

1、java redis

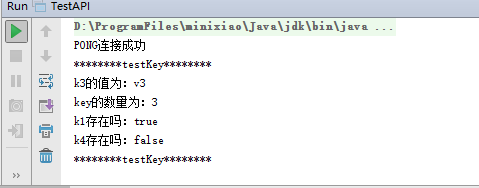
类别 redis

# 1、java redis连接

## 1、jedis 连接 ip 端口

|  |
| --- |
| **public class** TestAPI {     **public static void** main(String[] args) {  TestAPI testAPI = **new** TestAPI();  *//testAPI.ConnectRedis();* testAPI.testKey();  }  **public void** testKey()  {  Jedis jedis =ConnectRedis();   jedis.set(**"k1"**,**"v1"**);  jedis.set(**"k2"**,**"v2"**);  jedis.set(**"k3"**,**"v3"**);  System.***out***.println(**"\*\*\*\*\*\*\*\*testKey\*\*\*\*\*\*\*\*"**);  System.***out***.println(**"k3的值为："**+jedis.get(**"k3"**));  Set<String> sets = jedis.keys(**"\*"**);  System.***out***.println(**"key的数量为："**+sets.size());  *//后续请参考脑图，家庭作业，敲一遍......* System.***out***.println(**"k1存在吗："**+jedis.exists(**"k1"**));  System.***out***.println(**"k4存在吗："**+jedis.exists(**"k4"**));  System.***out***.println(**"\*\*\*\*\*\*\*\*testKey\*\*\*\*\*\*\*\*"**);   }  **public** Jedis ConnectRedis(){  Jedis jedis = **new** Jedis(**"192.168.1.37"**,6379);  System.***out***.println(jedis.ping()+**"连接成功"**);  **return** jedis;  }  **public void** testString() {  Jedis jedis =ConnectRedis();   } } |

## 2、测试成功



# 2、watch 标记

|  |
| --- |
| **public** **class** JedisTwoWatch {    /\*\*  \* 通俗点讲，watch命令就是标记一个键，如果标记了一个键，  \* 在提交事务前如果该键被别人修改过，那事务就会失败，这种情况通常可以在程序中  \* 重新再尝试一次。  \* 首先标记了键balance，然后检查余额是否足够，不足就取消标记，并不做扣减；  \* 足够的话，就启动事务进行更新操作，  \* 如果在此期间键balance被其它人修改， 那在提交事务（执行exec）时就会报错，  \* 程序中通常可以捕获这类错误再重新执行一次，直到成功。  \* **@throws** InterruptedException  \*/  **public** **static** **void** main(String[] args) **throws** InterruptedException {  JedisTwoWatch test = **new** JedisTwoWatch();  **boolean** retValue = test.transMethod();  System.*out*.println("main retValue-------: " + retValue);  }    **public** **boolean** transMethod() **throws** InterruptedException {  Jedis jedis = **new** Jedis("127.0.0.1", 6379);  **int** balance;// 可用余额  **int** debt;// 欠额  **int** amtToSubtract = 10;// 实刷额度    /\*\*  \* 通俗点讲，watch命令就是标记一个键，如果标记了一个键，  \* 在提交事务前如果该键被别人修改过，那事务就会失败，这种情况通常可以在程序中  \* 重新再尝试一次。  \*/  jedis.watch("balance");  jedis.set("balance","12");//此句不该出现，讲课方便。模拟其他程序已经修改了该条目  System.*out*.println("我是其他人，我已经修改了balance 在你watch期间");    Thread.*sleep*(7000);  balance = Integer.*parseInt*(jedis.get("balance"));  **if** (balance < amtToSubtract) {  jedis.unwatch();  System.*out*.println("modify");  **return** **false**;  } **else** {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*transaction");  Transaction transaction = jedis.multi();  transaction.decrBy("balance", amtToSubtract);  transaction.incrBy("debt", amtToSubtract);    transaction.exec(); //提交事务  balance = Integer.*parseInt*(jedis.get("balance"));  debt = Integer.*parseInt*(jedis.get("debt"));  System.*out*.println("\*\*\*\*\*\*\*" + balance);  System.*out*.println("\*\*\*\*\*\*\*" + debt);  **return** **true**;  }  }  } |

# 3、JedisPool

## 1、获取JedisPool工具

|  |
| --- |
| **public** **class** JedisThreeJedisPoolUtil  {  **private** **static** **volatile** JedisPool *jedisPool* = **null**;    **private** JedisThreeJedisPoolUtil(){}    **public** **static** JedisPool getJedisPoolInstance()  {  **if**(**null** == *jedisPool*)  {  **synchronized** (JedisThreeJedisPoolUtil.**class**)  {  **if**(**null** == *jedisPool*)  {  JedisPoolConfig poolConfig = **new** JedisPoolConfig();  //配置最大jedis实例数  poolConfig.setMaxTotal(1000);  //配置资源池最大闲置数  poolConfig.setMaxIdle(200);  //等待可用连接的最大时间  poolConfig.setMaxWaitMillis(10000);    //在borrow一个jedis实例时，是否提前进行validate操作；如果为true，则得到的jedis实例均是可用的  poolConfig.setTestOnBorrow(**true**);    *jedisPool* = **new** JedisPool(poolConfig,"127.0.0.1",6379);  }  }  }  **return** *jedisPool*;  }  ////释放Jedis资源  **public** **static** **void** release(JedisPool jedisPool)  {  **if**(**null** != jedisPool)  {  jedisPool.close();  }  }    } |

## 2、操作类

|  |
| --- |
| **public** **class** JedisThreeMain {  **public** **static** **void** main(String[] args) {  JedisPool jedisPool = JedisThreeJedisPoolUtil.*getJedisPoolInstance*();  JedisPool jedisPool2 = JedisThreeJedisPoolUtil.*getJedisPoolInstance*();    System.*out*.println(jedisPool == jedisPool2);    Jedis jedis1 = **null**;  Jedis jedis2 = **null**;  **try** {  jedis1 = jedisPool.getResource();  jedis1.set("aa","bb");  jedis2 = jedisPool2.getResource();  System.*out*.println(jedis2.get("aa"));      } **catch** (Exception e) {  e.printStackTrace();  }**finally**{  JedisThreeJedisPoolUtil.*release*(jedisPool); //这个方法我自己编的不靠谱  }  }  } |

# 4、主从复制

|  |
| --- |
| **public** **class** JedisFourMaterSlaveMain {    **public** **static** **void** main(String[] args) {  *masterForSlave*();  }    @SuppressWarnings("resource")  **public** **static** **void** masterForSlave() {  //主从复制，配从不配主  Jedis jedis\_M = **new** Jedis("192.168.1.37",6000);  jedis\_M.auth("123456");  Jedis jedis\_S6001 = **new** Jedis("192.168.1.37",6001);  jedis\_S6001.auth("123456");  jedis\_S6001.slaveof("192.168.1.37",6000);      Jedis jedis\_S6002 = **new** Jedis("192.168.1.37",6002);  jedis\_S6002.auth("123456");  jedis\_S6002.slaveof("192.168.1.37",6000);    jedis\_M.set("class","1122V2");  //jedis\_M.del("class");  String result1 = jedis\_S6001.get("class");  String result2 = jedis\_S6002.get("class");  System.*out*.println(result1+result2);  } |

# 5、map、list、set

|  |
| --- |
| **public** **class** JedisMapList {  **static** Jedis *jedis* = **null**;  **static**{  *jedis* = **new** Jedis("127.0.0.1",6379);  }  /\*\*  \* **@param** args  \*/  **public** **static** **void** main(String[] args) {  JedisMapList t =**new** JedisMapList();  t.testString();  }      /\*\*  \* redis存储字符串  \*/  **public** **void** testString() {  //-----添加数据----------  *jedis*.set("name","xinxin");//向key-->name中放入了value-->xinxin  System.*out*.println(*jedis*.get("name"));//执行结果：xinxin    *jedis*.append("name", " is my lover"); //拼接  System.*out*.println(*jedis*.get("name"));    *jedis*.del("name"); //删除某个键  System.*out*.println(*jedis*.get("name"));  //设置多个键值对  *jedis*.mset("name","liuling","age","23","qq","476777XXX");  *jedis*.incr("age"); //进行加1操作  System.*out*.println(*jedis*.get("name") + "-" + *jedis*.get("age") + "-" + *jedis*.get("qq"));    }    /\*\*  \* redis操作Map  \*/  @Test  **public** **void** testMap() {  //-----添加数据----------  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));  }  }    /\*\*  \* jedis操作List  \*/  @Test  **public** **void** testList(){  //开始前，先移除所有的内容  *jedis*.del("java framework");  System.*out*.println(*jedis*.lrange("java framework",0,-1));  //先向key java framework中存放三条数据  *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");  *jedis*.rpush("java framework","spring");  *jedis*.rpush("java framework","struts");  *jedis*.rpush("java framework","hibernate");  System.*out*.println(*jedis*.lrange("java framework",0,-1));  }    /\*\*  \* jedis操作Set  \*/  **public** **void** testSet(){  //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** **void** test() **throws** InterruptedException {  //jedis 排序  //注意，此处的rpush和lpush是List的操作。是一个双向链表（但从表现来看的）  *jedis*.del("a");//先清除数据，再加入数据进行测试  *jedis*.rpush("a", "1");  *jedis*.lpush("a","6");  *jedis*.lpush("a","3");  *jedis*.lpush("a","9");  System.*out*.println(*jedis*.lrange("a",0,-1));// [9, 3, 6, 1]  System.*out*.println(*jedis*.sort("a")); //[1, 3, 6, 9] //输入排序后结果  System.*out*.println(*jedis*.lrange("a",0,-1));  }      } |