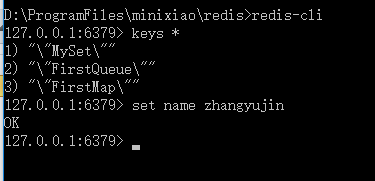
# 1.连接redis

## 1.添加测试数据name



## 2.java连接代码测试 jedis

**public class** TestPing {  
 **public static void** main(String[] args)   
 {  
 Jedis jedis = **new** Jedis(**"127.0.0.1"**,6379);  
 System.***out***.println(jedis.ping());  
 System.***out***.print(jedis.get(**"name"**));  
 }  
}

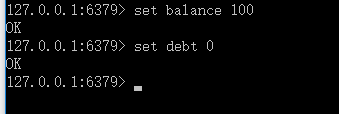
# 2.测试API

## 1.保存key 和取出

**public class** TestAPI {  
 **public static void** main(String[] args)   
 {  
 Jedis jedis = **new** Jedis(**"127.0.0.1"**,6379);  
   
 jedis.set(**"k1"**,**"v1"**);  
 jedis.set(**"k2"**,**"v2"**);  
 jedis.set(**"k3"**,**"v3"**);  
   
   
 System.***out***.println(jedis.get(**"k3"**));  
   
 Set<String> sets = jedis.keys(**"\*"**);  
 System.***out***.println(sets.size());  
   
 *//后续请参考脑图，家庭作业，敲一遍......* }  
}

# 3.事务

## 1.添加测试数据



## 2.进行测试代码

### 1.wacth

*/\*\*  
 \* 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"**);

### 2.开启事务

Transaction transaction = jedis.multi();

### 3.代码

**public class** TestTX {  
  
 /\*\*  
 \* 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 {  
 TestTX test = **new** TestTX();  
 **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","5");//此句不该出现，讲课方便。模拟其他程序已经修改了该条目* 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  
 // transaction.discard(); 丢弃* balance = Integer.*parseInt*(jedis.get(**"balance"**));  
 debt = Integer.*parseInt*(jedis.get(**"debt"**));  
  
 System.***out***.println(**"\*\*\*\*\*\*\*"** + balance);  
 System.***out***.println(**"\*\*\*\*\*\*\*"** + debt);  
 **return true**;  
 }  
 }

# 4.主从复制（不会）

# 5 jedisPool



## 1.单例模式获取

**public class** JedisPoolUtil   
{  
 **private static volatile** JedisPool *jedisPool* = **null**;  
   
 **private** JedisPoolUtil(){}  
   
 **public static** JedisPool getJedisPoolInstance()  
 {  
 **if**(**null** == *jedisPool*)  
 {  
 **synchronized** (JedisPoolUtil.**class**)  
 {  
 **if**(**null** == *jedisPool*)  
 {  
 JedisPoolConfig poolConfig = **new** JedisPoolConfig();  
 *// poolConfig.setMaxActive(1000);* poolConfig.setMaxIdle(32);  
 *// poolConfig.setMaxWait(100\*1000);* poolConfig.setTestOnBorrow(**true**);  
  
 *jedisPool* = **new** JedisPool(poolConfig,**"127.0.0.1"**,6379);  
 }  
 }  
 }  
 **return** *jedisPool*;  
 }  
  
 **public static void** release(JedisPool jedisPool,Jedis jedis)  
 {  
 **if**(**null** != jedis)  
 {  
 jedisPool.returnResourceObject(jedis);  
 }  
 }  
   
}

## 2.测试

**public class** TestPool {  
  
 **public static void** main(String[] args) {  
 JedisPool jedisPool = JedisPoolUtil.*getJedisPoolInstance*();  
 JedisPool jedisPool2 = JedisPoolUtil.*getJedisPoolInstance*();  
   
 System.***out***.println(jedisPool == jedisPool2);  
  
   
 Jedis jedis = **null**;  
 **try** {  
 jedis = jedisPool.getResource();  
 jedis.set(**"aa"**,**"bb"**);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 JedisPoolUtil.*release*(jedisPool, jedis);  
 }  
 }  
}