redis【锁】

作者：haxLook
原文链接：https://ld246.com/article/1617284746798
来源网站：链滴

工具类

RedisLockHandler.java

```
package com.bohee.module.gl.voucher.redisCommon.lock;

import com.alibaba.excel.util.CollectionUtils;
import com.bohee.module.gl.voucher.entity.Voucher;
import com.bohee.module.gl.voucher.redisCommon.redisUtils.JedisClientPools;
import com.bohee.utils.common.LoggerUtils;
import com.bohee.utils.common.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPool;
import redis.clients.jedis.Pipeline;
import redis.clients.jedis.exceptions.JedisConnectionException;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CopyOnWriteArrayList;
import java.util.concurrent.TimeUnit;

/**
 * @author  hax redis锁实现高并发
 * Created by Administrator on 2020/9/4.
 */
@Component
public class RedisLockHandler {

    private static final Logger LOGGER = LoggerFactory.getLogger(RedisLockHandler.class);

    private static final int DEFAULT_SINGLE_EXPIRE_TIME = 3;

    private static final int DEFAULT_BATCH_EXPIRE_TIME = 6;

    private static final String LOCK_SUCCESS = "OK";

    private static final String SET_IF_NOT_EXIST = "NX";

    private static final String SET_WITH_EXPIRE_TIME = "PX";

    @Autowired
    private JedisClientPools jedisClientPool;

    /**
     * 获取锁  如果锁可用   立即返回true，  否则返回false
     * @param billIdentify
     * @return
     */
    public boolean tryLock(TSuperclass billIdentify) {
```

```
        return tryLock(billIdentify, 0L, null);
    }

    public void lock(TSuperclass billIdentify) {
        this.voidLock(billIdentify);
    }


   /**
    * 锁在给定的等待时间内空闲，则获取锁成功 返回true， 否则返回false
     * @param billIdentify
     * @param timeout
     * @param unit
     * @return
     */
    public boolean tryLock(TSuperclass billIdentify, long timeout, TimeUnit unit) {
        String $_lockKey = (String) billIdentify.getTSuperclassKey();
        try {
                String $_lockValue= StringUtils.uuid();
               long nano = System.nanoTime();
            do {
                LOGGER.debug("try lock key: " + $_lockKey);
                Long i = jedisClientPool.setnx($_lockKey, $_lockValue);
                if (i == 1) {
                    jedisClientPool.expire($_lockKey, DEFAULT_SINGLE_EXPIRE_TIME);
                    LOGGER.debug("get lock, key: " + $_lockKey + " , expire in " + DEFAULT_SINGL
_EXPIRE_TIME + " seconds.");
                    return Boolean.TRUE;
                } else { // 存在锁
                    if (LOGGER.isDebugEnabled()) {
                        String desc = jedisClientPool.get($_lockKey);
                        LOGGER.debug("key: " + $_lockKey + " locked by another business：" + des
);
                    }
                }
                if (timeout == 0) {
                    break;
                }
                Thread.sleep(300);
            } while ((System.nanoTime() - nano) < unit.toNanos(timeout));
            return Boolean.FALSE;
        } catch (JedisConnectionException je) {
            LOGGER.error(je.getMessage(), je);
            returnBrokenResource(jedisClientPool.getJedis());
        } catch (Exception e) {
            LOGGER.error(e.getMessage(), e);
        } finally {
            returnResource(jedisClientPool.getJedis());
        }
        return Boolean.FALSE;
    }

    /**
     * 如果锁空闲立即返回  获取失败 一直等待
```

```
     * @param billIdentify
     */
    public  void voidLock(TSuperclass billIdentify) {
        String key = (String) billIdentify.getTSuperclassKey();
        try {
            do {
                LOGGER.info("lock key: " + key);
                Long i = jedisClientPool.setnx(key, key);
                if (i == 1) {
                    jedisClientPool.expire(key, DEFAULT_SINGLE_EXPIRE_TIME);
                    LOGGER.info("get lock, key: " + key + " , expire in " + DEFAULT_SINGLE_EXPIRE
TIME + " seconds.");
                    return;
                } else {
                    if (LOGGER.isDebugEnabled()) {
                        String desc = jedisClientPool.get(key);
                        LOGGER.info("key: " + key + " locked by another business:  " + desc);
                    }
                }
                Thread.sleep(300);
            } while (true);
        } catch (JedisConnectionException je) {
            LOGGER.error(je.getMessage(), je);
            returnBrokenResource(jedisClientPool.getJedis());
        } catch (Exception e) {
            LOGGER.error(e.getMessage(), e);
        } finally {
            returnResource(jedisClientPool.getJedis());
        }
    }

    /**
     * 释放锁
     * @param billIdentify
     */
    public void unLock(TSuperclass billIdentify) {
        List<TSuperclass> list = new ArrayList<TSuperclass>();
        list.add(billIdentify);
        unLock(list);
    }

    /**
     * 获取所有的锁数据
     * @param ids
     * @return
     */
    public List<TSuperclass> queryLocks(List<String> ids) {
        List<TSuperclass> list=new ArrayList<>();
        ids.forEach(id->{
            list.add(TSuperclass.getVoucher(id));
        });
        return list;
    }
```

```
/**
 * 一键释放锁
 * @param ids
 * @return
 */
public boolean  unLocks(List<String> ids) {
    List<TSuperclass> list = this.queryLocks(ids);
    boolean lock = this.tryLock(list);
    return lock;
}
 /**
* 批量获取锁  如果全部获取   立即返回true, 部分获取失败 返回false
 * @param billIdentifyList
 * @return
 */
public boolean tryLock(List<TSuperclass> billIdentifyList) {
    return tryLock(billIdentifyList, 0L, null);
}

/**
 * 锁在给定的等待时间内空闲，则获取锁成功 返回true， 否则返回false
 * @param billIdentifyList
 * @param timeout
 * @param unit
 * @return
 */
public boolean tryLock(List<TSuperclass> billIdentifyList, long timeout, TimeUnit unit) {

    try {
        List<String> needLocking = new CopyOnWriteArrayList<String>();
        List<String> locked = new CopyOnWriteArrayList<String>();
        long nano = System.nanoTime();
        do {
            // 构建pipeline，批量提交
            Pipeline pipeline = jedisClientPool.getJedis().pipelined();
            for (TSuperclass identify : billIdentifyList) {
                String key = (String) identify.getTSuperclassKey();
                needLocking.add(key);
                pipeline.setnx(key, key);
            }
            LOGGER.debug("try lock keys: " + needLocking);
            // 提交redis执行计数
            List<Object> results = pipeline.syncAndReturnAll();
            for (int i = 0; i < results.size(); ++i) {
                Long result = (Long) results.get(i);
                String key = needLocking.get(i);
                if (result == 1) {    // setnx成功，获得锁
                    jedisClientPool.expire(key, DEFAULT_BATCH_EXPIRE_TIME);
                    locked.add(key);
                }
            }
            needLocking.removeAll(locked);    // 已锁定资源去除

            if (CollectionUtils.isEmpty(needLocking)) {
```

```
                return true;
            } else {
                // 部分资源未能锁住
                LOGGER.debug("keys: " + needLocking + " locked by another business：");
            }

            if (timeout == 0) {
                break;
            }
            Thread.sleep(500);
        } while ((System.nanoTime() - nano) < unit.toNanos(timeout));

        // 得不到锁，释放锁定的部分对象，并返回失败
        if (!CollectionUtils.isEmpty(locked)) {
            jedisClientPool.delbath(locked.toArray(new String[0]));
        }
        return false;
    } catch (JedisConnectionException je) {
        LOGGER.error(je.getMessage(), je);
        returnBrokenResource(jedisClientPool.getJedis());
    } catch (Exception e) {
        LOGGER.error(e.getMessage(), e);
    } finally {
        returnResource(jedisClientPool.getJedis());
    }
    return true;
}

/**
 * 批量释放锁
 * @param billIdentifyList
 */
public void unLock(List<TSuperclass> billIdentifyList) {
    List<String> keys = new CopyOnWriteArrayList<String>();
    for (TSuperclass identify : billIdentifyList) {
        String key = (String) identify.getTSuperclassKey();
        keys.add(key);
    }
    try {
        jedisClientPool.delbath(keys.toArray(new String[0]));
        LOGGER.debug("release lock, keys :" + keys);
    } catch (JedisConnectionException je) {
        LOGGER.error(je.getMessage(), je);
        returnBrokenResource(jedisClientPool.getJedis());
    } catch (Exception e) {
        LOGGER.error(e.getMessage(), e);
    } finally {
        returnResource(jedisClientPool.getJedis());
    }
}
/**
 * 销毁连接
 * @param jedis
 */
```

```
    private void returnBrokenResource(Jedis jedis) {
        if (jedis == null) {
            return;
        }
        try {
            //容错
            jedisClientPool.getJedisPool().returnBrokenResource(jedis);
        } catch (Exception e) {
            LOGGER.error(e.getMessage(), e);
        }
    }

    /**
     * @param jedis
     */
    private void returnResource(Jedis jedis) {
        if (jedis == null) {
            return;
        }
        try {
            jedisClientPool.getJedisPool().returnResource(jedis);
        } catch (Exception e) {
            LOGGER.error(e.getMessage(), e);
        }
    }

}
```

持续更新中.....