



链滴

用 Redis 构建分布式锁

作者: [ibut](#)

原文链接: <https://ld246.com/article/1542011794566>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)



用Redis构建分布式锁

在不同进程需要互斥地访问共享资源时，分布式锁是一种非常有用的技术手段。有很多三方库和文章述如何用Redis实现一个分布式锁管理器，但是这些库实现的方式差别很大，而且很多简单的实现其只需采用稍微增加一点复杂的设计就可以获得更好的可靠性。

最近自己基于Redis做了一个实现，发出来希望大家一起讨论下，看是否还有不可靠情况，能否再完。

```
@Value("${配置服务器标识}")
private String index;
/**
 * 获取锁再试时间ms
 */
private static final int DEFAULT_ACQUIRY_RETRY_MILLIS = 97;
/**
 * 获取锁的超时时间3s
 */
private int TIMEOUT_MSECS = 3 * 1000;

/**
 * 获取锁之后最长拥有时间，防止死锁 默认7s
 */
private long EXPIRE_MSECS = 7 * 1000L;

/** @Description: 获取锁 */
public synchronized String lockByRedis(final String lockKey, final long expireMsecs) throws InterruptedException {

    int timeout = TIMEOUT_MSECS;
```

```

while (timeout >= 0) {
    long expires = System.currentTimeMillis() + expireMsecs + 1;
    String expiresStr = index+String.valueOf(expires); //服务器标识 + 锁到期时间
    if (this.setNX(lockKey, expiresStr)) {
        return expiresStr;
    }
    //redis里key的时间(表示失效时间)
    String redisValue = this.get(lockKey);
    Long redisValueTime = Long.parseLong(redisValue.substring(redisValue.length()-13));
    //判断锁是否已经过期，过期则重新设置并获取
    if (redisValue != null && redisValueTime < System.currentTimeMillis()) {
        //设置锁并返回旧值
        String oldValue = this.getSet(lockKey, expiresStr);
        //比较锁的时间值，如果不一致则可能是其他锁已经修改了值并获取锁
        if (oldValue != null && oldValue.equals(redisValue)) {
            return expiresStr;
        }
    }
    timeout -= DEFAULT_ACQUIRY_RETRY_MILLIS;
    //延时
    Thread.sleep(DEFAULT_ACQUIRY_RETRY_MILLIS);
}
return StringUtils.EMPTY;
}

```

```

/* @Description: 释放锁 *>finally */
public synchronized void unlockByRedis(final String lockKey,final String lockValue) {

    String redisValue = this.get(lockKey);

    //确认是否是要删除的锁,不相同可能锁已经过期
    if (redisValue != null && lockValue.equals(lockValue)) {
        Boolean delStatus = stringRedisTemplate.delete(lockKey);
        if(!delStatus) {
            logger.error("锁释放失败! 不存在的KEY: {}",lockKey);
        }
    }
}
}

```