### 1. 背景

##### 1.1 功能概述

此依赖库用于处理线程的暂停、等待和唤醒操作。

##### 1.2 例如

1.2.1：当你需要在执行某个任务前让当前线程暂停一段时间，例如，模拟网络延迟、等待某个资源变得可用或简单地控制任务执行的速率。

1.2.2：在生产者-消费者模式中，消费者线程可能会调用此方法等待生产者生产数据。当生产者在锁对象上调用notify或notifyAll时，等待的消费者线程将被唤醒。

### 2. 使用说明

### 1. threadSleep(Long millisecond)

****使当前执行的线程暂停指定的时间长度。****

****参数:**** Integer类型

millisecond: 线程暂停的时间，以毫秒为单位。

****返回值:****

true: 如果线程成功暂停指定的时间。

false: 如果参数无效或线程在暂停期间被中断。

****异常处理：****

InterruptedException: 如果线程在睡眠期间被中断，会捕获此异常，并重新设置中断状态。

****日志记录:****

info: 当线程成功休眠时，记录休眠的时间。

error: 当传入的毫秒数无效时，记录错误信息。

### 2. waitThread(String lock)

使当前线程在指定的锁对象上等待，直到被其他线程唤醒或线程被中断。

****参数:**** String 类型

lock: 用作同步锁的对象。

返回值: Boolean类型

true: 如果线程成功被唤醒。

false: 如果等待过程中线程被中断。

****异常处理:****

InterruptedException: 如果线程在等待期间被中断，会捕获此异常，并重新设置中断状态。

****日志记录:****

info: 当线程成功唤醒时，记录唤醒信息。

error: 当锁对象为null或等待被中断时，记录错误信息。

### 3. waitThreadWithTimeout(String lock, Long timeout)

****使当前线程在指定的锁对象上等待，直到被唤醒、线程被中断或超过指定的超时时间。****

****参数:**** String 类型

lock: 用作同步锁的对象。

timeout: 超时时间，以毫秒为单位。

****返回值:**** Boolean类型

true: 如果线程在超时前被唤醒。

false: 如果等待过程中线程被中断或超时。

****异常处理:****

InterruptedException: 如果线程在等待期间被中断，会捕获此异常，并重新设置中断状态。

日志记录:

info: 当线程在超时前被唤醒时，记录唤醒信息。

error: 当锁对象为null、超时时间设置不正确或等待被中断时，记录错误信息。

### 4. notifyThread(String lock)

****唤醒在指定锁对象上等待的第一个线程。****

****参数:**** String 类型

lock: 用作同步锁的对象。

****返回值:**** Boolean类型

true: 如果成功唤醒等待的线程。

false: 如果监控状态不合法，即当前线程不持有锁。

****异常处理:****

IllegalMonitorStateException: 如果当前线程不持有锁，会捕获此异常。

****日志记录:****

info: 当成功唤醒等待的线程时，记录唤醒信息。

error: 当锁对象为null或监控状态不合法时，记录错误信息。

### 5. notifyAllThreads(String lock)

****唤醒在指定锁对象上等待的所有线程。****

****参数:**** String类型

lock: 用作同步锁的对象。 String类型

****返回值:**** Boolean类型

true: 如果成功唤醒所有等待的线程。

false: 如果锁对象为null或监控状态不合法，即当前线程不持有锁。

****异常处理:****

IllegalMonitorStateException: 如果当前线程不持有锁，会捕获此异常。

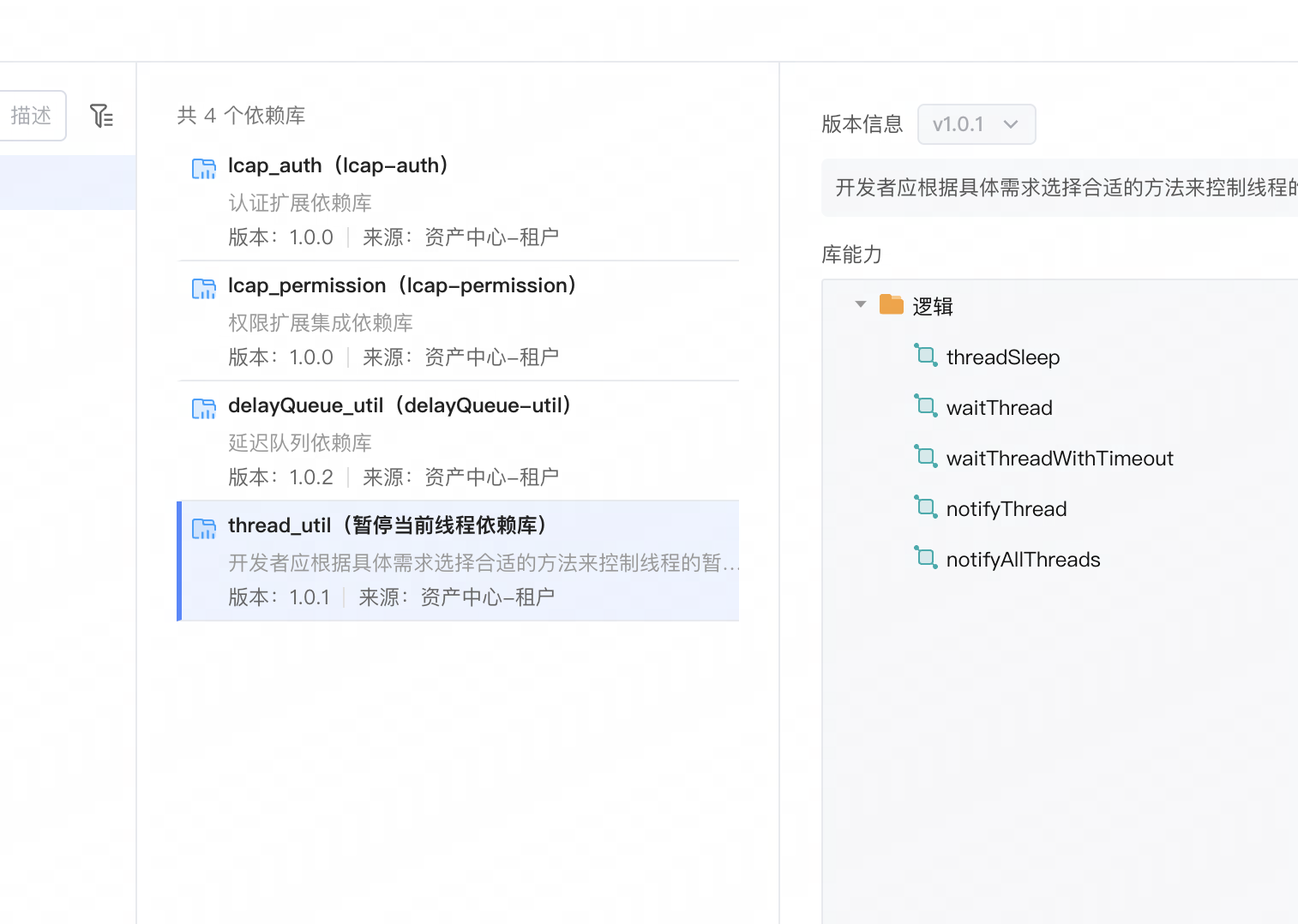
****日志记录****:

info: 当成功唤醒所有等待的线程时，记录唤醒信息。

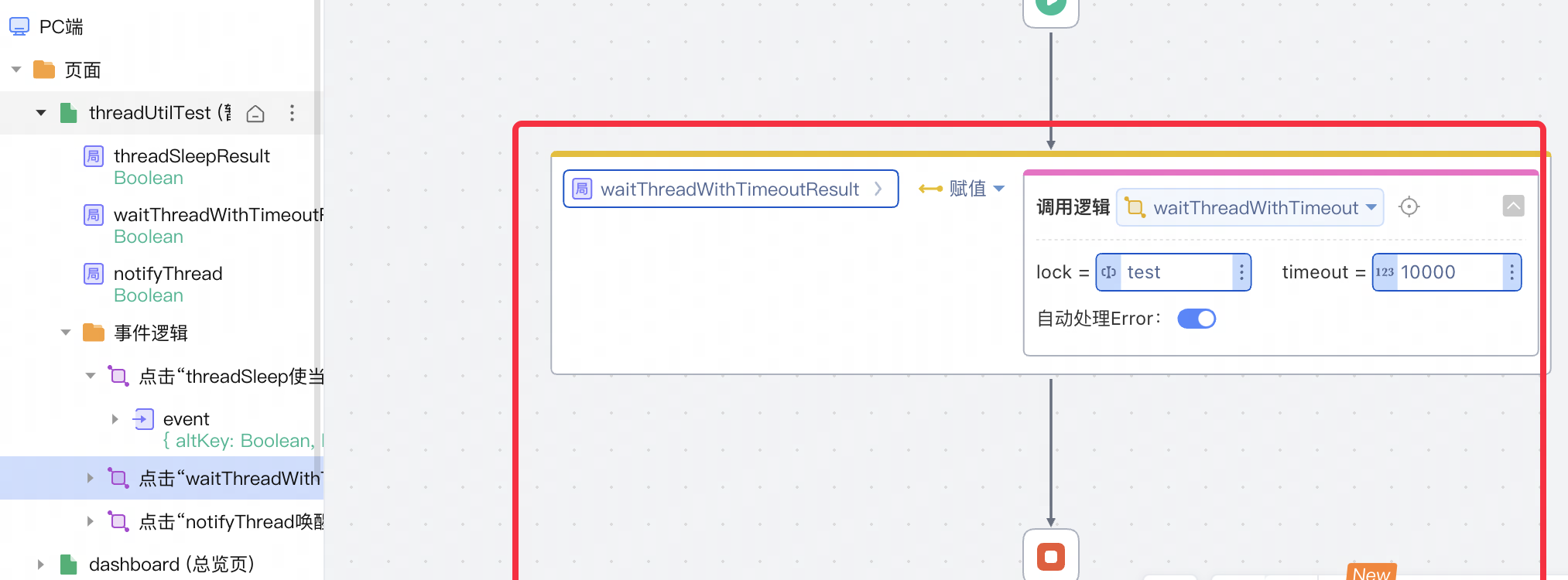
error: 当锁对象为null或监控状态不合法时，记录错误信息。

### 使用案例

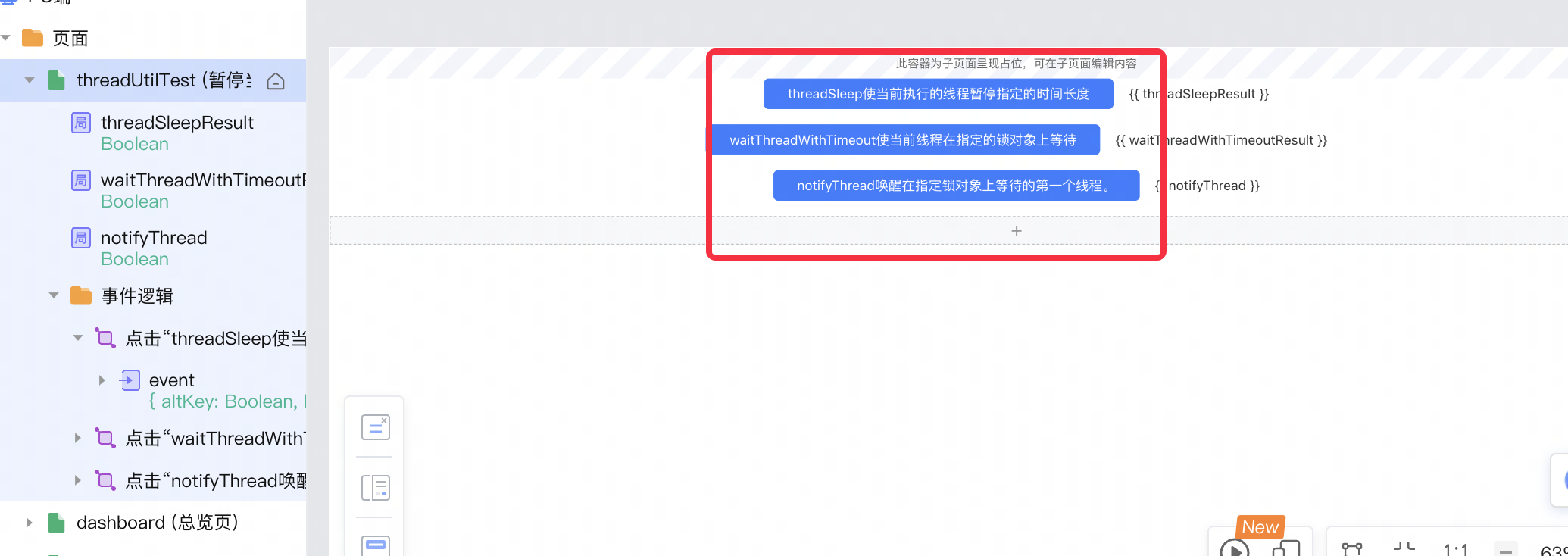
3.1引入依赖库



3.2调用方法传入参数



3.4IDE页面内调用显示（测试）



3.5查看结果

