幸福的懦夫

longdick

不在上帝那儿，就在去上帝那儿的路上

2009-09-24 通过网页

>>更多闲聊

浏览: 40771 次

性别:

来自: 0

看博客频道

详细资料 留言簿

搜索本博客

搜索

最近访客 >>更多访客

gelnyang y_franky

向各位介绍有意思的人工智能机器人ELBOT | activemq的消息存储机制

4顶
0踩

2009-08-20

最简实例说明wait、notify、notifyAll的使用方法

关键字: wait notify notifyall

```
/**
* 转载请注明作者longdick  http://longdick.javaeye.com
*
*/
```

wait()、notify()、notifyAll()是三个定义在Object类里的方法，可以用来控制线程的状态。
这三个方法最终调用的都是jvm级的native方法。随着jvm运行平台的不同可能有些许差异。

- 如果对象调用了wait方法就会使持有该对象的线程把该对象的控制权交出去，然后处于等待状态。
- 如果对象调用了notify方法就会通知某个正在等待这个对象的控制权的线程可以继续运行。
- 如果对象调用了notifyAll方法就会通知所有等待这个对象控制权的线程继续运行。

其中wait方法有三个over load方法：
wait()
wait(long)
wait(long,int)
wait方法通过参数可以指定等待的时长。如果没有指定参数，默认一直等待直到被通知。

以下是一个演示代码，以最简洁的方式说明复杂的问题：
简要说明下：
NotifyThread是用来模拟3秒钟后通知其他等待状态的线程的线程类；
WaitThread是用来模拟等待的线程类；
等待的中间对象是flag，一个String对象；
main方法中同时启动一个Notify线程和三个wait线程；

Java代码


shexh hellowei

博客分类

- 全部博客 (74)
- java全文检索 (2)
- 开心最重要 (5)
- Hibernate (2)
- DWR (2)
- Ubuntu (3)
- 杂项 (2)
- Oracle (4)
- Ajax (4)
- spring (3)
- 应用服务器 (3)
- JVM (8)
- webservice (1)
- IDE (1)
- 模式 (0)
- 简单心情 (14)
- OSGI (1)
- 版本管理 (1)
- jms (2)
- uml (6)
- java 基础 (7)
- 毋忘在莒 (1)

我的相册

```
1.  public class NotifyTest {
2.          private  String flag = "true";
3.
4.          class NotifyThread extends Thread{
5.                  public NotifyThread(String name) {
6.                          super(name);
7.                  }
8.                  public void run() {
9.                          try {
10.                                 sleep(3000);//推迟3秒钟通知
11.                         } catch (InterruptedException e) {
12.                                 e.printStackTrace();
13.                         }
14.
15.                                 flag = "false";
16.                                 flag.notify();
17.                 }
18.         };
19.
20.         class WaitThread extends Thread {
21.                 public WaitThread(String name) {
22.                         super(name);
23.                 }
24.
25.                 public void run() {
26.
27.                                 while (flag!="false") {
28.                                         System.out.println(getName() + " begin waiting!");
29.                                         long waitTime = System.currentTimeMillis();
30.                                         try {
31.                                                 flag.wait();
32.                                         } catch (InterruptedException e) {
33.                                                 e.printStackTrace();
34.                                         }
35.                                         waitTime = System.currentTimeMillis() - waitTime;
36.                                         System.out.println("wait time :"+waitTime);
37.                                 }
38.                                 System.out.println(getName() + " end waiting!");
39.
40.                 }
41.         }
42.
43.         public static void main(String[] args) throws InterruptedException {
```

HiYrh.jpg
共 3 张

我的留言簿 >>更多留言

- 楼主，你对JVM了解很深吧，我们能否联系一下，也许能合作哦！
  -- by linux1689
- 漂过
  -- by suitmefine
- 好厉害！
  -- by poson

其他分类

- 我的收藏 (38)
- 我的论坛帖子 (12)
- 我的精华良好贴 (0)

最近加入圈子

链接

- delicious
- reddit
- digg

存档

- 2009-11 (5)
- 2009-10 (11)
- 2009-09 (13)

```
44.                     System.out.println("Main Thread Run!");
45.                     NotifyTest test = new NotifyTest();
46.                     NotifyThread notifyThread =test.new NotifyThread("notify01");
47.                     WaitThread waitThread01 = test.new WaitThread("waiter01");
48.                     WaitThread waitThread02 = test.new WaitThread("waiter02");
49.                     WaitThread waitThread03 = test.new WaitThread("waiter03");
50.                     notifyThread.start();
51.                     waitThread01.start();
52.                     waitThread02.start();
53.                     waitThread03.start();
54.             }
55.
56.     }
```

OK，如果你拿这段程序去运行下的话， 会发现根本运行不了，what happened？满屏的java.lang.IllegalMonitorStateException。
没错，这段程序有很多问题，我们一个个来看。

首先，这儿要非常注意的几个事实是

1. 任何一个时刻，对象的控制权（monitor）只能被一个线程拥有。
2. 无论是执行对象的wait、notify还是notifyAll方法，必须保证当前运行的线程取得了该对象的控制权（monitor）
3. 如果在没有控制权的线程里执行对象的以上三种方法，就会报java.lang.IllegalMonitorStateException异常。
4. JVM基于多线程，默认情况下不能保证运行时线程的时序性

基于以上几点事实，我们需要确保让线程拥有对象的控制权。
也就是说在waitThread中执行wait方法时，要保证waitThread对flag有控制权；
在notifyThread中执行notify方法时，要保证notifyThread对flag有控制权。

线程取得控制权的方法有三：

1. 执行对象的某个同步实例方法。
2. 执行对象对应类的同步静态方法。
3. 执行对该对象加同步锁的同步块。

我们用第三种方法来做说明：
将以上notify和wait方法包在同步块中

Java代码

```
1.  synchronized (flag) {
2.                                      flag = "false";
```

- 更多存档...

最新评论

- 神秘的数字9

  quote="nethibernate"]1楼nb, 崇拜一下 ...

  -- by sunwenran

- 一个女孩向我抱怨

  -- by 解未知数

- 一个女孩向我抱怨

  人的伦理道德 改变比较难！！

  -- by HeroXuan

- 一个女孩向我抱怨

  ...

  -- by louies

- 一个女孩向我抱怨

  LZ说的真精

  辟！！！！！！！！！！

  -- by severusz

评论排行榜

- 一个女孩向我抱怨
- 给思维做体操的谜题--光有IQ还不够（二）
- 图解JVM在内存中申请对象及垃圾回收流程
- 完美解决甲醛问题的终极方案--只要人人都献 ...
- 给思维做体操的谜题--光有IQ还不够


```
                                flag.notify();
                        }
```

Java代码

```
synchronized (flag) {
                        while (flag!="false") {
                                System.out.println(getName() + " begin waiting!");
                                long waitTime = System.currentTimeMillis();
                                try {
                                        flag.wait();
                                } catch (InterruptedException e) {
                                        e.printStackTrace();
                                }
                                waitTime = System.currentTimeMillis() - waitTime;
                                System.out.println("wait time :"+waitTime);
                        }
                        System.out.println(getName() + " end waiting!");
                }
```

我们向前进了一步。
问题解决了吗？
好像运行还是报错java.lang.IllegalMonitorStateException。what happened？

这时的异常是由于在针对flag对象同步块中，更改了flag对象的状态所导致的。如下：
flag="false";
flag.notify();
对在同步块中对flag进行了赋值操作，使得flag引用的对象改变，这时候再调用notify方法时，因为没有控制权所以抛出异常。

我们可以改进一下，将flag改成一个JavaBean，然后更改它的属性不会影响到flag的引用。
我们这里改成数组来试试，也可以达到同样的效果：

Java代码

```
private   String flag[] = {"true"};
```

Java代码

```
synchronized (flag) {
                        flag[0] = "false";
```

```
                        flag.notify();
                }
```

Java代码

```
synchronized (flag) {
                                flag[0] = "false";
                                flag.notify();
                        }synchronized (flag) {
                                while (flag[0]!="false") {
                                        System.out.println(getName() + " begin waiting!");
                                        long waitTime = System.currentTimeMillis();
                                        try {
                                                flag.wait();

                                        } catch (InterruptedException e) {
                                                e.printStackTrace();
                                        }
```

这时候再运行，不再报异常，但是线程没有结束是吧，没错，还有线程堵塞，处于wait状态。

原因很简单，我们有三个wait线程，只有一个notify线程，notify线程运行notify方法的时候，是随机通知一个正在等待的线程，所以，现在应该还有两个线程在waiting。

我们只需要将NotifyThread线程类中的flag.notify()方法改成notifyAll()就可以了。notifyAll方法会通知所有正在等待对象控制权的线程。

最终完成版如下：

Java代码

```
public class NotifyTest {
        private String flag[] = { "true" };

        class NotifyThread extends Thread {
                public NotifyThread(String name) {
                        super(name);
                }

                public void run() {
                        try {
```

```
                        sleep(3000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    synchronized (flag) {
                        flag[0] = "false";
                        flag.notifyAll();
                    }
                }
            };

            class WaitThread extends Thread {
                public WaitThread(String name) {
                    super(name);
                }

                public void run() {
                    synchronized (flag) {
                        while (flag[0] != "false") {
                            System.out.println(getName() + " begin waiting!");
                            long waitTime = System.currentTimeMillis();
                            try {
                                flag.wait();

                            } catch (InterruptedException e) {
                                e.printStackTrace();
                            }
                            waitTime = System.currentTimeMillis() - waitTime;
                            System.out.println("wait time :" + waitTime);
                        }
                        System.out.println(getName() + " end waiting!");
                    }
                }
            }

            public static void main(String[] args) throws InterruptedException {
                System.out.println("Main Thread Run!");
                NotifyTest test = new NotifyTest();
                NotifyThread notifyThread = test.new NotifyThread("notify01");
                WaitThread waitThread01 = test.new WaitThread("waiter01");
                WaitThread waitThread02 = test.new WaitThread("waiter02");
                WaitThread waitThread03 = test.new WaitThread("waiter03");
                notifyThread.start();
                waitThread01.start();
```

```
55.                     waitThread02.start();
56.                     waitThread03.start();
57.             }
58.
59.     }
```

4 顶　　0 踩

向各位介绍有意思的人工智能机器人ELBOT | activemq的消息存储机制

14:43 | 浏览 (843) | 评论 (1) | 分类: java 基础 | 相关推荐

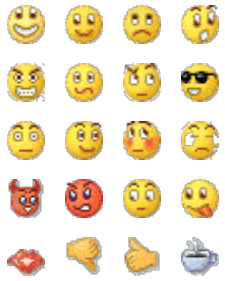
评论

1 楼 iammonster 2009-08-21 引用

例子有点偏,不过也正好来解释出现的问题

发表评论

表情图标



B I U Quote Code List Img URL Flash Table

字体颜色: 标准 字体大小: 标准 对齐: 标准

提示：选择您需要装饰的文字, 按上列按钮即可添加上相应的标签

您还没有登录，请登录后发表评论(快捷键 Alt+S / Ctrl+Enter) 提交