

木兮同学

VIP

码龄5年

企业员工

184

948

6897

21万+

原创

周排名

总排名

访问

等级

3551

1738

597

116

800

积分

粉丝

获赞

评论

收藏

in

脉

1024

私信

已关注

搜博主文章

- 热门文章
- Variable used in lambda expression should be final or effectively final

68653
- 实现短信验证码登录

2902
- 如何写出一手漂亮的代码?

1805
- 单例模式的 9 种写法

1673
- 调优案例分析与实战

1563

分类专栏

JVM	★ 深入理解 Java 虚拟机	20篇
spring	★ Spring Security	15篇
MySQL	★ MySQL 实战	24篇
redis	★ Redis 开发与运维	16篇
Gof	★ Head First 设计模式	22篇
	★ Java 并发编程	27篇

- 最新评论
- Variable used in lambda expression sho...

木兮同学: [code=java] System.out.println ("原子类 Lamda 使用情况"); AtomicInte...
- Variable used in lambda expression sho...

qq_562293298: 为什么在lamda中可以使用原子类
- DataLink 数据同步平台

Gauss松鼠会: 好文，干货满满，已收藏，初来乍到，希望多多关注，期待大佬回...
- Variable used in lambda expression sho...

一格状语: 这种帖子需要多一些
- DataLink 数据同步平台

木兮同学: 😊

如何正确停止线程之 interrupt 的使用

原创

木兮同学

2020-03-03 10:04:54

755

★ 收藏 4

版权

分类专栏:

★ Java 并发编程

文章标签:

interrupt

停止线程

Java

★ Java 并发编程 专栏收录该内容

1 订阅

27 篇文章

订阅专栏

文章目录

原理介绍

Java 中停止线程的原则是什么

正确的停止方法：interrupt

通常线程会在什么情况下停止普通情况？

线程可能被阻塞

如果线程在每次迭代后都阻塞

while 内 try/catch 的问题

实际开发中的两种最佳实践

响应中断的方法总结列表

原理介绍

- 使用 interrupt 来通知，而不是强制停止

Java 中停止线程的原则是什么

- 在 Java 中，最好的停止线程的方式是使用中断 interrupt，但是 这仅仅是会通知到被终止的线程“你该停止了”，被终止的线程自身拥有决定权（决定是否、以及何时停止），这依赖于请求停止方和被停止方都遵守一种约定好的编码规范。
- 任务和线程的启动很容易。 在大多数时候，我们都会让它们运行直到结束，或者让它们自行停止。然而有时候我们希望提前结束任务或线程或许是因为用户取消了操作或者服务需要被快速关闭，或者是运行超时或出错了。
- 要使任务和线程能安全、快速、可靠地停止下来，并不是一件容易的事。Java 没有提供任何机制来安全地终止线程；但它提供了中断（Interruption），这是一种协作机制，能够使一个线程终止另一个线程的当前工作。
- 这种协作式的方法是必要的，我们很少希望某个任务、线程或服务立即停止，因为这种立即停止会使共享的数据结构处于不一致的状态。相反，在编写任务和服务时可以使用种协作的方式当需要停止时它们首先会清除当前正在执行的工作然后再结束。这提供了更好的灵活性，因为任务本身的代码比发出取消请求的代码更清楚如何执行清除工作。
- 生命周期结束（End- of-Lifecycle）的问题会使任务、服务以及程序的设计和实现等过程变得复杂，而这个在程序设计中非常重要的要素却经常被忽略。一个在行为良好的软件与勉强运的软件之间的最主要区别就是，行为良好的软件能很完善地处理失败、关闭和取消等过程。
- 接下来将给出各种实现取消和中断的机制，以及如何编写任务和服务，使它们能对取消请求做出响应。

正确的停止方法：interrupt

举报

🙄

强烈不推荐

😞

不推荐

😐

一般般

😊

推荐

😄

强烈推荐

最新文章

Spring Security OAuth：源码解析之还是内味儿

DataLink 数据同步平台

Spring Security OAuth：客户端模式超简单实现

2021

09月1篇

08月2篇

07月6篇

06月6篇

05月6篇

04月9篇

03月22篇

02月1篇

2020年72篇

2019年60篇

目录

文章目录

原理介绍

Java 中停止线程的原则是什么

正确的停止方法：interrupt

通常线程会在什么情况下停止普通情...

线程可能被阻塞

如果线程在每次迭代后都阻塞

while 内 try/catch 的问题

实际开发中的两种最佳实践

响应中断的方法总结列表

通常线程会在什么情况下停止普通情况？

示例一：run 方法内没有 sleep 或 wait 方法时，停止线程

```
1 public class RightWayStopThreadWithoutSleep implements Runnable {
2
3     @Override
4     public void run() {
5         int num = 0;
6         while (!Thread.currentThread().isInterrupted() && num <= Integer.MAX_VALUE / 2) {
7             if (num % 10000 == 0) {
8                 System.out.println(num + "是10000的倍数");
9             }
10            num++;
11        }
12        System.out.println("任务运行结束了");
13    }
14
15    public static void main(String[] args) throws InterruptedException {
16        Thread thread = new Thread(new RightWayStopThreadWithoutSleep());
17        thread.start();
18        Thread.sleep(2000);
19        thread.interrupt();
20    }
21 }
```

打印结果：

```
1 ...
2 279980000是10000的倍数
3 279990000是10000的倍数
4 280000000是10000的倍数
5 280010000是10000的倍数
6 任务运行结束了
```

虽然主线程调用了 thread.interrupt() 方法，但他只是把子线程标记为中断， 代码中还需要不停地去判断是否被中断，即：Thread.currentThread().isInterrupted()

线程可能被阻塞

示例二：带有 sleep 的中断线程的写法

```
1 public class RightWayStopThreadWithSleep {
2
3     public static void main(String[] args) throws InterruptedException {
4         Runnable runnable = () -> {
5             int num = 0;
6             try {
7                 while (num <= 300 && !Thread.currentThread().isInterrupted()) {
8                     if (num % 100 == 0) {
9                         System.out.println(num + "是100的倍数");
10                    }
11                    num++;
12                }
13                Thread.sleep(1000);
14            } catch (InterruptedException e) {
15                e.printStackTrace();
16            }
17        };
18        Thread thread = new Thread(runnable);
19        thread.start();
20        Thread.sleep(500);
21        thread.interrupt();
22    }
```

举报

打印结果：

```
1 0是100的倍数
2 100是100的倍数
3 200是100的倍数
4 300是100的倍数
5 java.lang.InterruptedException: sleep interrupted
6     at java.lang.Thread.sleep(Native Method)
7     at com.hd.thread.stop.RightWayStopThreadWithSleep.lambda$main$0(RightWayStopThreadWithS
8     at java.lang.Thread.run(Thread.java:745)
```

`Thread.sleep()` 阻塞过程中能够检测到中断标记，而抛出中断异常。

如果线程在每次迭代后都阻塞

示例三：如果在执行过程中，每次循环都会调用 `sleep` 或 `wait` 等方法，那么不需要每次迭代都检查是否已中断，`sleep` 会响应中断

```
1 public class RightWayStopThreadWithSleepEveryLoop {
2
3     public static void main(String[] args) throws InterruptedException {
4         Runnable runnable = () -> {
5             int num = 0;
6             try {
7                 while (num <= 10000) {
8                     if (num % 100 == 0) {
9                         System.out.println(num + "是100的倍数");
10                    }
11                    num++;
12                    Thread.sleep(10);
13                }
14            } catch (InterruptedException e) {
15                e.printStackTrace();
16            }
17        };
18        Thread thread = new Thread(runnable);
19        thread.start();
20        Thread.sleep(5000);
21        thread.interrupt();
22    }
23 }
```

打印结果：

```
1 0是100的倍数
2 100是100的倍数
3 200是100的倍数
4 300是100的倍数
5 java.lang.InterruptedException: sleep interrupted
6     at java.lang.Thread.sleep(Native Method)
7     at com.hd.thread.stop.RightWayStopThreadWithSleepEveryLoop.lambda$main$0(RightWayStopTh
8     at java.lang.Thread.run(Thread.java:745)
```

这个案例中我们不再需要检查是否中断，`sleep()` 阻塞时候会自动检测中断，即不再需要：
`Thread.currentThread().isInterrupted()`

while 内 try/catch 的问题

示例四：如果 `while` 里面放 `try/catch`，会导致中断失效

```
1 public class CantInterrupt {
2
3     public static void main(String[] args) throws InterruptedException {
4
5     }
```



木兮同学

已关注



4



0



4



专栏目录



举报

```
5         int num = 0;
6         while (num <= 10000 && !Thread.currentThread().isInterrupted()) {
7             if (num % 100 == 0) {
8                 System.out.println(num + "是100的倍数");
9             }
10            num++;
11            try {
12                Thread.sleep(10);
13            } catch (InterruptedException e) {
14                e.printStackTrace();
15            }
16        }
17    };
18    Thread thread = new Thread(runnable);
19    thread.start();
20    Thread.sleep(5000);
21    thread.interrupt();
22 }
23 }
```

打印结果：

```
1 0是100的倍数
2 100是100的倍数
3 200是100的倍数
4 300是100的倍数
5 java.lang.InterruptedException: sleep interrupted
6     at java.lang.Thread.sleep(Native Method)
7     at com.hd.thread.stop.CantInterrupt.lambda$main$0(CantInterrupt.java:20)
8     at java.lang.Thread.run(Thread.java:745)
9 400是100的倍数
10 500是100的倍数
11 600是100的倍数
12 ...
```

Thread.sleep() 检测到的中断信号抛出的异常被捕获了。

实际开发中的两种最佳实践

- 优先选择：传递中断

示例五：最佳实践：catch 了 InterruptedException 之后的优先选择：在方法签名中抛出异常 那么在 run() 就会强制 try/catch

```
1 public class RightWayStopThreadInProd implements Runnable {
2
3     @Override
4     public void run() {
5         while (true && !Thread.currentThread().isInterrupted()) {
6             System.out.println("go");
7             try {
8                 throwInMethod();
9             } catch (InterruptedException e) {
10                Thread.currentThread().interrupt(); 再次中断，程序才能终止
11                //保存日志、停止程序
12                System.out.println("保存日志");
13                e.printStackTrace();
14            }
15        }
16    }
17
18    private void throwInMethod() throws InterruptedException {
19        Thread.sleep(2000);
20    }
```



举报



木兮同学

已关注

👍 4

💬 0

🌟 4



专栏目录

InterruptedException {

```
23         Thread thread = new Thread(new RightWayStopThreadInProd());
24         thread.start();
25         Thread.sleep(1000);
26         thread.interrupt();
27     }
28 }
```

打印结果：

```
1 go
2 java.lang.InterruptedException: sleep interrupted
3     at java.lang.Thread.sleep(Native Method)
4     at com.hd.thread.stop.RightWayStopThreadInProd.throwInMethod(RightWayStopThreadInProd.j
5     at com.hd.thread.stop.RightWayStopThreadInProd.run(RightWayStopThreadInProd.java:16)
6     at java.lang.Thread.run(Thread.java:745)
7 保存日志
```

处理中断的最好方法是什么？

优先选择在方法上抛出异常。用 throws Interrupted Exception 标记你的方法，不采用try语句块捕获异常，以便于该异常可以传递到顶层，让 run 方法可以捕获这一异常，例如

```
1     private void throwInMethod() throws InterruptedException {
2         Thread.sleep(2000); 封装成方法，抛出去
3     }
```

由于 run 方法内无法抛出 checked Exception（只能用 try catch），顶层方法必须处理该异常，避免了漏掉或者被吞掉的情况，增强了代码的健壮性。

- 不想或无法传递：恢复中断

示例六：最佳实践 2：在 catch 子语句中调用 Thread.currentThread().interrupt() 来恢复设置中断状态，以便于在后续的执行中，依然能够检查到刚才发生了中断回到刚才 RightWayStopThreadInProd 补上中断，让它跳出。

```
1 public class RightWayStopThreadInProd2 implements Runnable {
2
3     @Override
4     public void run() {
5         while (true) {
6             if (Thread.currentThread().isInterrupted()) {
7                 System.out.println("Interrupted，程序运行结束");
8                 break;
9             }
10            reInterrupt();
11        }
12    }
13
14    private void reInterrupt() {
15        try {
16            Thread.sleep(2000);
17        } catch (InterruptedException e) {
18            Thread.currentThread().interrupt(); 再次中断，程序才能停止
19            e.printStackTrace();
20        }
21    }
22
23    public static void main(String[] args) throws InterruptedException {
24        Thread thread = new Thread(new RightWayStopThreadInProd2());
25        thread.start();
26        Thread.sleep(1000);
27        thread.interrupt();
```



举报

打印结果：

```
1 | Interrupted, 程序运行结束
2 | java.lang.InterruptedException: sleep interrupted
3 |     at java.lang.Thread.sleep(Native Method)
4 |     at com.hd.thread.stop.RightWayStopThreadInProd2.reInterrupt(RightWayStopThreadInProd2.j
5 |     at com.hd.thread.stop.RightWayStopThreadInProd2.run(RightWayStopThreadInProd2.java:19)
6 |     at java.lang.Thread.run(Thread.java:745)
```

如果不能抛出中断，要怎么做？

如果不想或无法传递 InterruptedException（例如用 run 方法的时候，就不让该方法 throws InterruptedException），那么应该选择在 catch子句中调用 `Thread.currentThread().interrupt()` 来恢复设置中断状态，以便于在后续的执行依然能够检查到刚才发生了中断。看上面代码，线程在 sleep 期间被中断，并且由 catch 捕获到该中断并重新设置了中断状态，以便于可以在下一个循环的时候检测到中断状态，正常退出。

- 不应屏蔽中断


响应中断的方法总结列表

- Object.wait() / wait(long) / wait(long, int)
- Thread.sleep(long) / sleep(long, int)
- Thread.join() / join(long) / join(long, int)
- java.util.concurrent.BlockingQueue.take() / put (E)
- java.util.concurrent.locks.Lock.lockInterruptibly()
- java.util.concurrent.CountDownLatch.await
- java.util.concurrent.CyclicBarrier.await
- java.util.concurrent.Exchanger.exchange(v)
- java.nio.channels.InterruptibleChannel相关方法
- java.nio.channels.Selector的相关方法

笔记来源：慕课网悟空老师视频《Java并发核心知识体系精讲》

Thread.**interrupt()**方法的使用以及使用它退出**线程** 12-18
Thread.**interrupt()**方法的使用以及使用它退出**线程**

Java停止线程的四种方法 fanrendale的博客 3087
一、**线程停止**基础知识 **interrupted()**: 测试当前**线程**是否已经中断。该方法为静态方法，调用后会返回boolean值。不过调用...



优质评论可以帮助作者获得更高权重

抢沙发

评论

相关推荐 更多相似内容

停止Java线程,小心**interrupt()**方法_学习笔记 8-16
甚至,在Thread.**interrupt()**被调用后,**线程**仍然继续运行。 真正地中断一个**线程** 中断**线程**最好的,最受推荐的方式是,使用共享...

Java多线程之如何**正确**的中断**线程**(**Interrupt/Stop**的使用)含示例代码 7-28
//**不正确**的**停止**方法 thread.stop(); //value a = 1|value b = 0 task.print(); **正确的线程停止 Interrupt InterruptedException** 当目标...

一、使用**interrupt ()** 中断**线程** varyall的专栏 5166
当一个**线程**运行时，另一个**线程**可以调用对应的Thread对象的**interrupt ()** 方法来中断它，该方法只是在目标**线程**中设置一...

如何用 **interrupt 停止线程** 一叶之秋的博客 51
首先介绍一下**线程**的六大状态 就像生物从出生到长大、最终死亡的过程一样，**线程**也有自己的生命周期，在 **Java** 中**线程**的...



举报