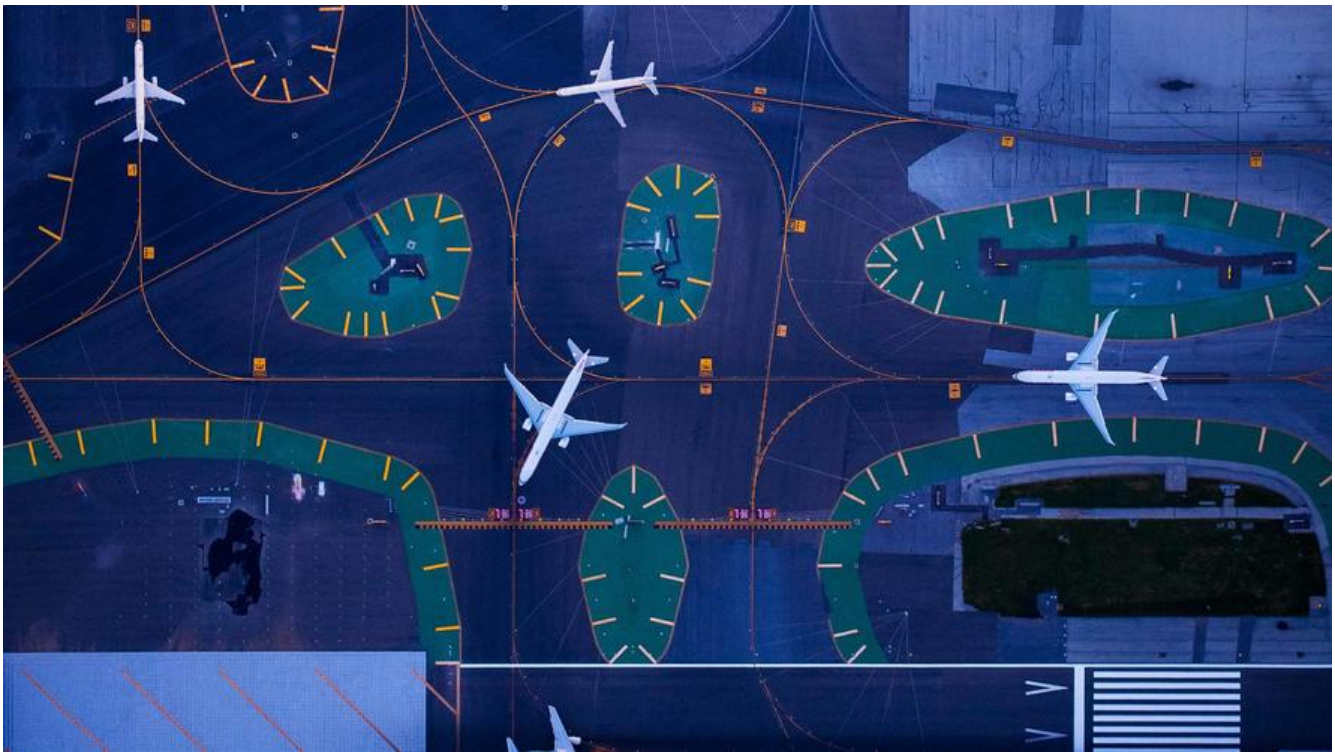
链滴

Java 线程监听，意外退出线程后自动重启

作者：woyehua
原文链接：https://ld246.com/article/1543287819148
来源网站：链滴



Java线程监听，意外退出线程后自动重启

某日，天朗气清，回公司，未到9点，刷微博，顿觉问题泛滥，惊恐万分！

前一天写了一个微博爬行程序，主要工作原理就是每隔2分钟爬行一次微博，获取某N个关注朋微博数量，然后将其保存起来，2分钟之后再次爬行，再取 其微博数量，与2分钟前保存的微博数量较，如果数量增加，说明该好友在此2分钟之内发布微博，如果数量减少，则是删除微博。最后将爬结果发送到指定手机上，作为通知！

今天看微博时发现自己关注的朋友发布了微博，然而自己手机却没有收到报警消息，查看爬行日发现，在凌晨6点钟时，公司网络曾经断网，导致网络堵 塞，程序在爬行的时候抛出网络异常**UnknonHostException**，此时线程就已经死掉，猝死。为解决此问题，现有如下需求：线程死掉之后会自重启。

常规解决办法有2种：

1，开启另外一个线程，监听爬行线程，使用线程之间的通信，一般是消费者模式，如果爬行线程死之后，监听线程会收到通知，但是如果监听线程先死掉，那么系统也就挂了！

2，使用心跳机制，爬行线程每隔一段时间往另一服务器进程发送数据包，如果服务器进程长时间没收到心跳包，则说明爬行线程已经死机！

两种方法都有合适的使用范围，但是对于监听线程猝死这种情况，个人觉得是使用观察者模式比合适！现说明一下观察者模式。

观察者模式：定义对象之间的一种一对多的依赖关系，当对象的状态发生改变时，所有依赖于它对象都得到通知并且被自动更新。

观察者模式在JDK中有现成的实现，**java.util.Obserable**，如何进行监听线程，请看例子：

被监听的线程类：

```
package  cn.std.test;
import  java.util.Observable;
import  cn.std.util.DateUtil;
public  class  RunThread  extends  Observable  implements  Runnable{
// 此方法一经调用，立马可以通知观察者，在本例中是监听线程
publicvoid doBusiness(){
if(true){
super.setChanged();
}
notifyObservers();
}
@Override
publicvoid run()
{
intc =  0;
while(true){
//模拟线程运行一段时间之后退出
System.out.println("Runing- "+c+" "+DateUtil.getStdDateTime());
try{
Thread.sleep(2000);
}catch (InterruptedException  e) {
e.printStackTrace();
doBusiness();
break;
}
c++;
//模拟抛出异常
```

```
try{
if(c==  4){
Stringstr =  null;
str.length();
//此处将会抛出空指针异常
}
}catch (Exception  e) {
e.printStackTrace();
doBusiness();
//在抛出异常时调用，通知观察者，让其重启线程
break;
//异常抛出之后，一定要跳出循环，保证将线程送进地狱
}
}
}
publicstatic void  main(String[] args) {
RunThreadrun =  new  RunThread();
Listener  listen  =  new  Listener();
un.addObserver(listen);
newThread(run).start();
//run.doBusiness();
}
}
```

被监听的线程需要继承**Observable**类，继承之后轻松变身为被观察者**doBusiness**方法，此处可以加你的业务处理内容，接下来是Listener

```
package  cn.std.test;
import  java.util.Observable;
import  java.util.Observer;
```

```
public class Listener implements Observer{
@Override
publicvoid update(Observable o, Object arg) {
System.out.println("RunThread死机");
RunThread run = new RunThread();
run.addObserver(this);
newThread(run).start();
System.out.println("RunThread重启");
}
}
```

很简单，只要实现Observer接口，实现其update方法，在方法内部重启线程