


Java多线程基础（十二）——Two-phase Termination模式

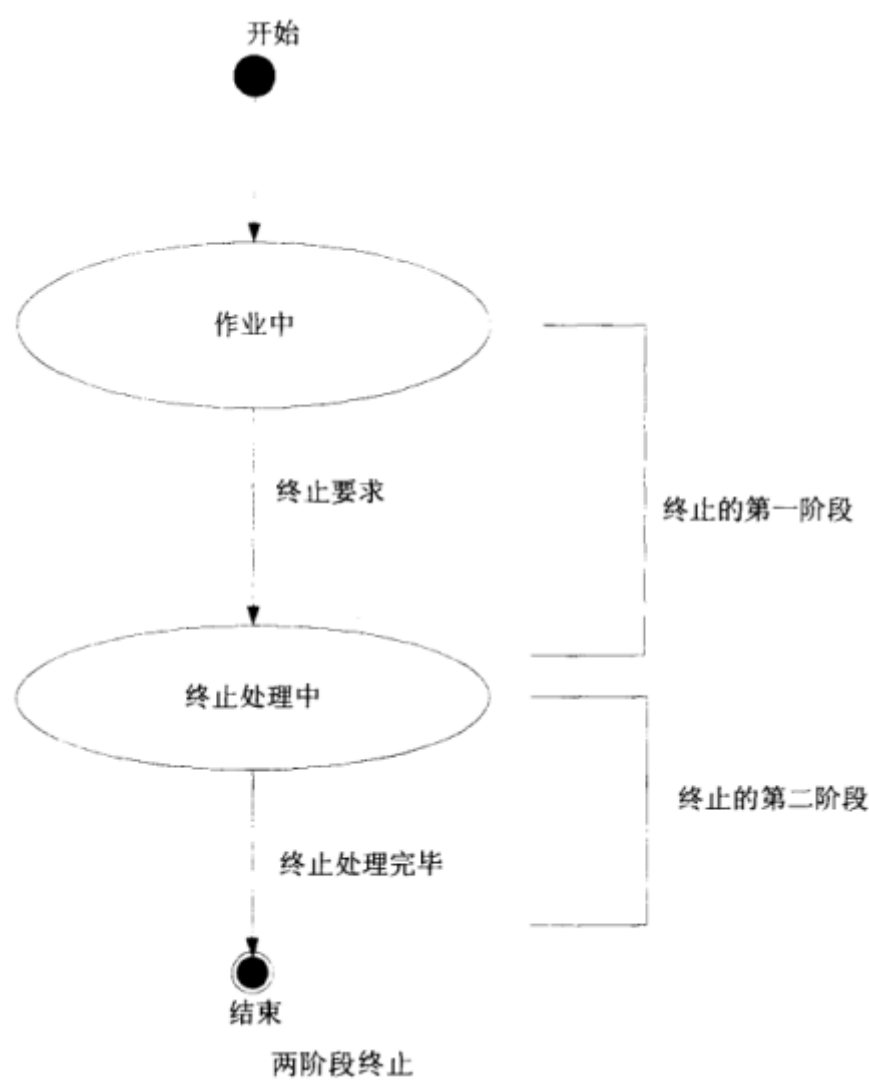

Ressmix 发布于 2018-07-07

一、定义

我们将线程的正常处理状态称为“作业中”，当希望结束这个线程时，则送出“终止请求”。接着，这个线程并不会立刻结束，而是进入“终止处理中”状态，此时线程还是运行着的，可能处理一些释放资源等操作。直到终止处理完毕，才会真正结束。

Two-phase Termination主要考虑以下问题：

- 安全地结束（安全性）；
- 一定会进行终止处理（生命性）；
- 收到“终止请求”后，要尽快进行终止处理（响应性）；



二、模式案例

该案例中，线程每隔500ms将计数器增加1，在大约10s后结束。

计数线程类：

```

public class CountupThread extends Thread {
    private long counter = 0;
    private volatile boolean shutdownRequested = false; //状态位
    public void shutdownRequest() {
        shutdownRequested = true;
        interrupt(); //终止请求
    }
    public boolean isShutdownRequested() {
        return shutdownRequested;
    }
    public final void run() {
        try {
            while (!shutdownRequested) {
                doWork(); //工作内容
            }
        } catch (InterruptedException e) {
        } finally {
            doShutdown(); //最后工作
        }
    }
    private void doWork() throws InterruptedException {
        counter++;
        System.out.println("doWork: counter = " + counter);
        Thread.sleep(500);
    }
    private void doShutdown() {
        System.out.println("doShutdown: counter = " + counter);
    }
}

```

执行:

```

public class Main {
    public static void main(String[] args) {
        System.out.println("main: BEGIN");
        try {
            CountupThread t = new CountupThread();
            t.start();
            Thread.sleep(10000);
            System.out.println("main: shutdownRequest");
            t.shutdownRequest();
            System.out.println("main: join");
            t.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("main: END");
    }
}

```

三、模式讲解

Two-phase Termination模式的角色如下:

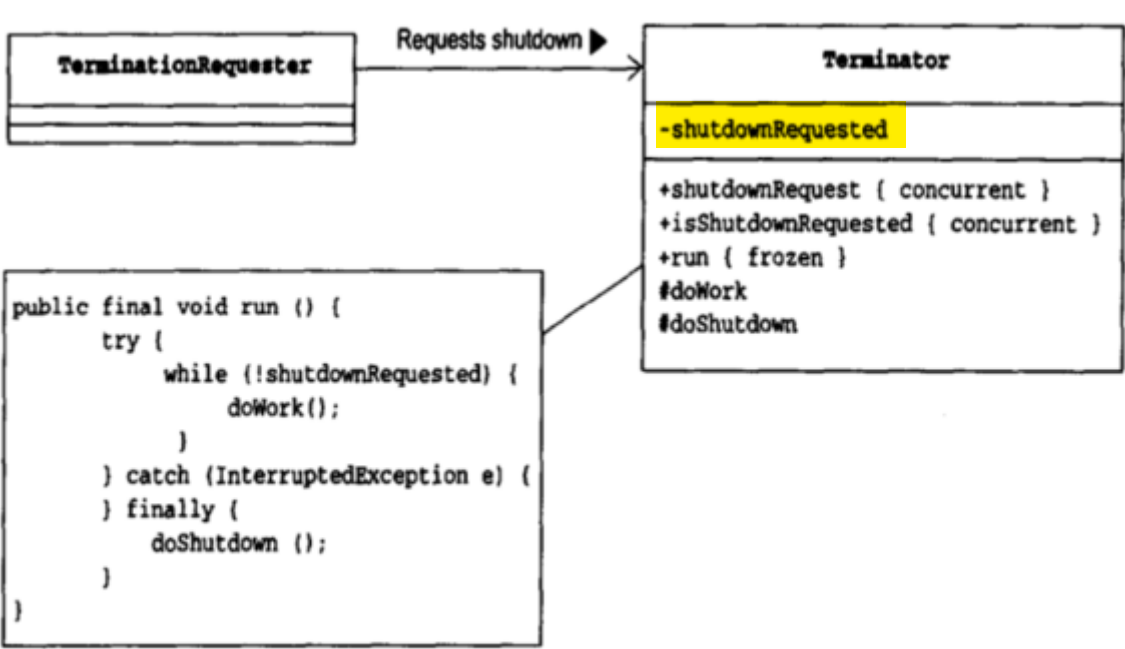
- TerminationRequester参与者

TerminationRequester参与者用于调用终止请求。(案例中的Main类)

- Terminator参与者

Terminator参与者接受终止请求, 进行实际的终止处理, 提供shutdownRequest终止方法供外界调用。

Terminator参与者拥有一个标识 (门闩) 表示是否已经收到终止请求



Two-Phase Termination Pattern 的类图

java 多线程

阅读 2.2k • 更新于 2018-08-02

赞 3

收藏 2

分享

本作品系原创，采用《署名-非商业性使用-禁止演绎 4.0 国际》许可协议



透彻理解Java并发编程

Java并发编程是整个Java开发体系中最难以理解但也是最重要的知识点，也是各类开源分布式框架中各...

关注专栏



Ressimix

1.2k 声望 1.3k 粉丝

关注作者

0 条评论

得票数 最新



撰写评论 ...



提交评论

你知道吗？

不要站着调试程序，那会使得你的耐心减半，你需要的是全神贯注。

注册登录

继续阅读

Java多线程基础（十一）——Future模式

Future模式用来获取线程的执行结果。在Thread-Per-Message模式中，如果调用一个线程异步执行任务，没有办法获取到返回值，...
Ressmix · 阅读 8.5k · 9 赞

Java多线程基础（三）——Single Threaded Execution模式

一、定义 Single Threaded Execution 是指“以1个线程执行”的意思，有时也称为Critical Section（临界区）。 二、模式案例 案例：...
Ressmix · 阅读 4.7k · 5 赞 · 1 评论

Java多线程基础（九）——Thread-Per-Message模式

Thread-Per-Message模式是指每个message一个线程，message可以理解成“消息”、“命令”或者“请求”。每一个message都会分配一...
Ressmix · 阅读 3.5k · 2 赞

Java多线程：Java多线程基础

也就是说，进程从系统那里获取到了一定的CPU占用时间片、内存单元和IO等等资源，然后线程将这些资源利用起来执行程序，线...
JinhaoPlus · 阅读 3.1k · 1 赞

Java多线程-线程状态

线程状态 6个状态定义： java.lang.Thread.State New: 尚未启动的线程的线程状态。 Runnable: 可运行线程的线程状态，等待CPU调...
小码农薛尧 · 阅读 484 · 1 赞

JAVA多线程编程

Java 给多线程编程提供了内置的支持。 一条线程指的是进程中一个单一顺序的控制流，一个进程中可以并发多个线程，每条线程...
已注销 · 阅读 180

java的线程的实现方式

继承 Java并发很大程度上应用了继承. 关键字： extends {代码...} Java的线程 获得当前线程的名称 Thread.currentThread().getName(...
oneboi · 阅读 665

多线程基础

多线程有什么作用 多线程不是为了提高执行速度，而是提高应用程序的使用率。 线程和线程共享“堆内存和方法区内存”， 栈内存...
Autonomy · 阅读 461

产品	课程	资源	合作	关注	条款
热门问答	Java 开发课程	每周精选	关于我们	产品技术日志	服务协议
热门专栏	PHP 开发课程	用户排行榜	广告投放	社区运营日志	隐私政策
热门课程	Python 开发课程	徽章	职位发布	市场运营日志	下载 App
最新活动	前端开发课程	帮助中心	讲师招募	团队日志	
技术圈	移动开发课程	声望与权限	联系我们	社区访谈	
酷工作		社区服务中心	合作伙伴		
		建议反馈			