

# 线程池--拒绝策略RejectedExecutionHandler

 方毅超 [关注](#)

 2 2017.03.11 15:32:25 字数 676 阅读 25,189

当线程池的任务缓存队列已满并且线程池中的线程数目达到maximumPoolSize，如果还有任务到来就会采取任务拒绝策略，通常有以下四种策略：

```
1 | ThreadPoolExecutor.AbortPolicy:丢弃任务并抛出RejectedExecutionException异常。
2 | ThreadPoolExecutor.DiscardPolicy: 也是丢弃任务，但是不抛出异常。
3 | ThreadPoolExecutor.DiscardOldestPolicy: 丢弃队列最前面的任务，然后重新尝试执行任务（重复此过程）
4 | ThreadPoolExecutor.CallersRunsPolicy: 由调用线程处理该任务
```

当Executor已经关闭（即执行了executorService.shutdown()方法后），并且Executor将有限边界用于最大线程和工作队列容量，且已经饱和时，在方法execute()中提交的新任务将被拒绝。在以上述情况下，execute方法将调用其RejectedExecutionHandler的RejectedExecutionHandler.rejectedExecution(java.lang Runnable, java.util.concurrent.ThreadPoolExecutor)方法。线程池默认会采用的是defaultHandler策略。首先看defaultHandler的定义：

```
1 | private static final RejectedExecutionHandler defaultHandler = new AbortPolicy(); // 使用默认的拒绝策略

1 | //丢弃任务并抛出RejectedExecutionException异常。
2 | public static class AbortPolicy implements RejectedExecutionHandler {
3 |     public AbortPolicy() {}
4 |     // 抛出异常
5 |     public void rejectedExecution(Runnable r, ThreadPoolExecutor e) {
6 |         throw new RejectedExecutionException("Task " + r.toString() + " rejected from " + e.toString());
7 |     }
8 | }
```

如下是一个测试任务的例子，下面编写4个测试用例来测试。

```
1 | class MyRunnable implements Runnable {
2 |     private String name;
3 |     public MyRunnable(String name) {
4 |         this.name = name;
5 |     }
6 |     @Override
7 |     public void run() {
8 |         try {
9 |             System.out.println(this.name + " is running.");
10 |             Thread.sleep(100);
11 |         } catch (Exception e) {
12 |             e.printStackTrace();
13 |         }
14 |     }
15 | }
```

看一下其他拒绝策略的具体实现。

## 1.DiscardPolicy 示例(也是丢弃任务，但是不抛出异常。)

```
1 | public class DiscardPolicyDemo {
2 |
3 |     private static final int THREADS_SIZE = 1;
4 |     private static final int CAPACITY = 1;
5 | }
```

告别死记硬背  
用公式带你突破英语



有道精品课

 方毅超 [关注](#)

总资产54 (约4.72元)

JAVA使用Gson排除特定字段【转发】

阅读 60

Android 悬浮窗--无需权限

阅读 1,515

推荐阅读

闭关修炼21天，“啃完”283页pdf，我终于4面拿下字节跳动offer

阅读 48,853

离开菜鸟&新的面试体验

阅读 588

Java并发高频面试题

阅读 2,992

项目-无侵入代码方式使用Redis实现缓存功能

阅读 4,013

面试官：为什么java中静态方法不能调用非静态方法和变量？

阅读 3,259



```
11 pool.setRejectedExecutionHandler(new ThreadPoolExecutor.DiscardPolicy());
12
13 // 新建10个任务，并将它们添加到线程池中。
14 for (int i = 0; i < 10; i++) {
15     Runnable myrun = new MyRunnable("task-"+i);
16     pool.execute(myrun);
17 }
18 // 关闭线程池
19 pool.shutdown();
20 }
21 }
```

线程池pool的”最大池大小”和”核心池大小”都为1(THREADS\_SIZE)，这意味着”线程池能同时运行的任务数量最大只能是1”。

线程池pool的阻塞队列是ArrayBlockingQueue，ArrayBlockingQueue是一个有界的阻塞队列，ArrayBlockingQueue的容量为1。这也意味着线程池的阻塞队列只能有一个线程池阻塞等待。

根据”中分析的execute()代码可知：线程池中运行了2个任务。第1个任务直接放到Worker中，通过线程去执行；第2个任务放到阻塞队列中等待。其他的任务都被丢弃了！

**2.DiscardOldestPolicy 示例**(丢弃队列最前面的任务，然后重新尝试执行任务（重复此过程）)

```
1 public class DiscardOldestPolicyDemo {
2
3     private static final int THREADS_SIZE = 1;
4     private static final int CAPACITY = 1;
5
6     public static void main(String[] args) throws Exception {
7
8         // 创建线程池。线程池的"最大池大小"和"核心池大小"都为1(THREADS_SIZE)，"线程池"的阻塞队列容量为1(CAPACITY)
9         ThreadPoolExecutor pool = new ThreadPoolExecutor(THREADS_SIZE, THREADS_SIZE, 0, TimeUnit.SECONDS,
10             new ArrayBlockingQueue<Runnable>(CAPACITY));
11         // 设置线程池的拒绝策略为"DiscardOldestPolicy"
12         pool.setRejectedExecutionHandler(new ThreadPoolExecutor.DiscardOldestPolicy());
13
14         // 新建10个任务，并将它们添加到线程池中。
15         for (int i = 0; i < 10; i++) {
16             Runnable myrun = new MyRunnable("task-"+i);
17             pool.execute(myrun);
18         }
19         // 关闭线程池
20         pool.shutdown();
21     }
22 }
```

运行结果：

```
1 task-0 is running.
2 task-9 is running.
```

将”线程池的拒绝策略”由DiscardPolicy修改为DiscardOldestPolicy之后，当有任务添加到线程池被拒绝时，线程池会丢弃阻塞队列中末尾的任务，然后将被拒绝的任务添加到末尾。

**3.AbortPolicy 示例**(丢弃任务并抛出RejectedExecutionException异常。)

```
1 public class AbortPolicyDemo {
2
3     private static final int THREADS_SIZE = 1;
4     private static final int CAPACITY = 1;
5
6     public static void main(String[] args) throws Exception {
7
8         // 创建线程池。线程池的"最大池大小"和"核心池大小"都为1(THREADS_SIZE)，"线程池"的阻塞队列容量为1(CAPACITY)
9         ThreadPoolExecutor pool = new ThreadPoolExecutor(THREADS_SIZE, THREADS_SIZE, 0, TimeUnit.SECONDS,
10             new ArrayBlockingQueue<Runnable>(CAPACITY));
```

写下你的评论...

评论4

赞35

...

```
16 // 新建10个任务，并将它们添加到线程池中。
17 for (int i = 0; i < 10; i++) {
18     Runnable myrun = new MyRunnable("task-"+i);
19     pool.execute(myrun);
20 }
21 } catch (RejectedExecutionException e) {
22     e.printStackTrace();
23     // 关闭线程池
24     pool.shutdown();
25 }
26 }
27 }
```

(某一次)运行结果：

```
1 java.util.concurrent.RejectedExecutionException
2   at java.util.concurrent.ThreadPoolExecutor$AbortPolicy.rejectedExecution(ThreadPoolExecutor.java:1774)
3   at java.util.concurrent.ThreadPoolExecutor.reject(ThreadPoolExecutor.java:768)
4   at java.util.concurrent.ThreadPoolExecutor.execute(ThreadPoolExecutor.java:656)
5   at AbortPolicyDemo.main(AbortPolicyDemo.java:27)
6 task-0 is running.
7 task-1 is running.
```

将”线程池的拒绝策略”由DiscardPolicy修改为AbortPolicy之后，当有任务添加到线程池被拒绝时，会抛出RejectedExecutionException。

#### 4.CallerRunsPolicy 示例(由调用线程处理该任务)

```
1 public class CallerRunsPolicyDemo {
2
3     private static final int THREADS_SIZE = 1;
4     private static final int CAPACITY = 1;
5
6     public static void main(String[] args) throws Exception {
7
8         // 创建线程池。线程池的"最大池大小"和"核心池大小"都为1(THREADS_SIZE)，"线程池"的阻塞队列容量为1(CAPACITY)
9         ThreadPoolExecutor pool = new ThreadPoolExecutor(THREADS_SIZE, THREADS_SIZE, 0, TimeUnit.SECONDS,
10             new ArrayBlockingQueue<Runnable>(CAPACITY));
11         // 设置线程池的拒绝策略为"CallerRunsPolicy"
12         pool.setRejectedExecutionHandler(new ThreadPoolExecutor.CallerRunsPolicy());
13
14         // 新建10个任务，并将它们添加到线程池中。
15         for (int i = 0; i < 10; i++) {
16             Runnable myrun = new MyRunnable("task-"+i);
17             pool.execute(myrun);
18         }
19
20         // 关闭线程池
21         pool.shutdown();
22     }
23 }
```

(某一次)运行结果：

```
1 task-2 is running.
2 task-3 is running.
3 task-4 is running.
4 task-5 is running.
5 task-6 is running.
6 task-7 is running.
7 task-8 is running.
8 task-9 is running.
9 task-0 is running.
10 task-1 is running.
```

将”线程池的拒绝策略”由DiscardPolicy修改为CallerRunsPolicy之后，当有任务添加到线程池

写下你的评论...

评论4

赞35

...

"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下



方毅超  
总资产54 (约4.72元) 共写了3.8W字 获得288个赞 共67个粉丝

关注

# 买公网IP盒子，做自己的私有云 - 装公网IP盒子，得公网固定IP

普通宽带可用，不改变本地网络和应用程序 ipv4.me



写下你的评论...

全部评论 4 只看作者

按时间倒序 按时间正序



Xargs  
3楼 2019.09.04 10:40

写的挺好的


 赞  回复



方毅超 作者  
2019.09.04 10:51

谢夸奖 主要还是大家的智慧，我只是整理了一下

 回复

 添加新评论



wxainn  
2楼 2019.02.25 08:53

逻辑清晰，谢谢总结分享👍


 赞  回复



方毅超 作者  
2019.02.25 13:21

@wxainn 谢谢，我也只是在前人的智慧上小小的做了一点东西😊

 回复

 添加新评论

写下你的评论...

 评论4  赞35 ...

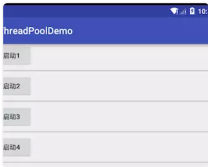
推荐阅读

更多精彩内容>

Android 关于线程池的理解

前言 线程池是Java中的一个重要概念，从Android上来说，当我们跟服务端进行数据交互的时候我们都知道主线程不...

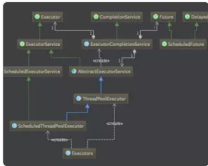
 老实任 阅读 491 评论 1 赞 6



深入理解Java线程池：ThreadPoolExecutor

博客链接：<http://www.ideabuffer.cn/2017/04/04/深入理解Java线程池：Thre...>

 ideabuffer 阅读 10,624 评论 14 赞 126



线程池

作者：肥肥鱼链接：<https://www.zhihu.com/question/30804052/answer/4...>

 TTTqiu 阅读 401 评论 0 赞 5

Java 基础面试宝典(自己总结)

一、多线程 说明下线程的状态 java中的线程一共有 5 种状态。NEW：这种情况指的是，通过 New 关键字创...

 Java旅行者 阅读 2,713 评论 0 赞 41

Java多线程笔记（三）：线程池

前言 多线程的软件设计方案确实可以最大限度地发挥现代多核处理器的计算能力，提高生产系列的吞吐量和性能。但是，若不加...

 泊浮目 阅读 207 评论 0 赞 1

