



¥ 259.00

联系我们



请扫描二维码联系

✉ webmaster@

☎ 400-660-0

🗨 QQ客服 🗨 更多

关于 招聘 广告服务

©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

他的最新文章

- view上的定时器如何销毁
- NSSortDescriptor 的使用-----排序
- 委托和协议的区别
- ios 转场动画
- 数组如何一边遍历一边删除元素

文章分类

ios

文章存档

- 2018年2月
- 2017年8月
- 2017年6月
- 2017年5月
- 2017年4月

他的热门文章

- 队列和线程的关系
👁 2898
- ios 核心动画-----跳动效果的转效果的实现
👁 1324
- 16位卡号输入框，每4位添加一个
👁 671

队列和线程的关系

转载

2017年04月27日 13:32:46

👁 2912

本文转载至<http://www.cnblogs.com/dsxniubility/p/4296937.html>

iOS多线程中，队列和执行的排列组合结果分析

本文是对以往学习的多线程中知识点的一个整理。

多线程中的队列有：**串行队列，并发队列，全局队列，主队列。**

执行的方法有：**同步执行和异步执行**。那么两两一组合会有哪些注意事项呢？

如果不是在董铂然博客园看到这边文章请 [点击查看原文](#)

提到多线程，也就是四种，pthread，NSThread，GCD，NSOperation

其中pthread是**跨平台**的。GCD和NSOperation都是常用的，**后者是基于前者的**。

但是两者区别：GCD的核心概念是**将一个任务添加到队列**，指定任务执行的方法，然后执行。NSOperation则是直接**将一个操作添加到队列**中。

为了整体结构更加清晰，我是用GCD来做此排列组合的实验。实验主要是通过循环内打印和主线程的打印先后顺序来判断结果，最后再加以总结

1.串行队列，同步执行

```
1 dispatch_queue_t q = dispatch_queue_create("dantesx", NULL);
2
3 // 执行任务
4 for (int i = 0; i<10; i++) {
5     dispatch_sync(q, ^{
6         NSLog(@"%@ %d", [NSThread currentThread], i);
7     });
8 }
9
10 NSLog(@"董铂然 come here");
```

运行效果：

```
2015-02-21 12:34:51.821 排列组合[15554:139347] <NSThread: 0x7fab785257e0>{number = 1, name = main} 0
2015-02-21 12:34:51.822 排列组合[15554:139347] <NSThread: 0x7fab785257e0>{number = 1, name = main} 1
2015-02-21 12:34:51.822 排列组合[15554:139347] <NSThread: 0x7fab785257e0>{number = 1, name = main} 2
2015-02-21 12:34:51.822 排列组合[15554:139347] <NSThread: 0x7fab785257e0>{number = 1, name = main} 3
2015-02-21 12:34:51.822 排列组合[15554:139347] <NSThread: 0x7fab785257e0>{number = 1, name = main} 4
2015-02-21 12:34:51.822 排列组合[15554:139347] <NSThread: 0x7fab785257e0>{number = 1, name = main} 5
2015-02-21 12:34:51.823 排列组合[15554:139347] <NSThread: 0x7fab785257e0>{number = 1, name = main} 6
2015-02-21 12:34:51.823 排列组合[15554:139347] <NSThread: 0x7fab785257e0>{number = 1, name = main} 7
2015-02-21 12:34:51.823 排列组合[15554:139347] <NSThread: 0x7fab785257e0>{number = 1, name = main} 8
2015-02-21 12:34:51.823 排列组合[15554:139347] <NSThread: 0x7fab785257e0>{number = 1, name = main} 9
2015-02-21 12:34:51.823 排列组合[15554:139347] 董铂然 come here
```

执行结果可以清楚的看到全在主线程执行，并且是按照数序执行，循环结束之后主线程的打印才输出。

2.串行队列，异步执行

```
1 dispatch_queue_t q = dispatch_queue_create("dantesx", NULL);
2
3 for (int i = 0; i<10; i++) {
4     dispatch_async(q, ^{
5         NSLog(@"%@ %d", [NSThread currentThread], i);
6     });
7 }
8 // [NSThread sleepForTimeInterval:0.001];
9 NSLog(@"董铂然 come here");
```

运行结果

```
2015-02-21 12:44:55.824 排列组合[16293:146979] 董铂然 come here
2015-02-21 12:44:55.824 排列组合[16293:147053] <NSThread: 0x7fb34963c370>{number = 2, name = (null)} 0
2015-02-21 12:44:55.826 排列组合[16293:147053] <NSThread: 0x7fb34963c370>{number = 2, name = (null)} 1
2015-02-21 12:44:55.826 排列组合[16293:147053] <NSThread: 0x7fb34963c370>{number = 2, name = (null)} 2
2015-02-21 12:44:55.826 排列组合[16293:147053] <NSThread: 0x7fb34963c370>{number = 2, name = (null)} 3
2015-02-21 12:44:55.826 排列组合[16293:147053] <NSThread: 0x7fb34963c370>{number = 2, name = (null)} 4
2015-02-21 12:44:55.827 排列组合[16293:147053] <NSThread: 0x7fb34963c370>{number = 2, name = (null)} 5
2015-02-21 12:44:55.827 排列组合[16293:147053] <NSThread: 0x7fb34963c370>{number = 2, name = (null)} 6
2015-02-21 12:44:55.827 排列组合[16293:147053] <NSThread: 0x7fb34963c370>{number = 2, name = (null)} 7
2015-02-21 12:44:55.827 排列组合[16293:147053] <NSThread: 0x7fb34963c370>{number = 2, name = (null)} 8
2015-02-21 12:44:55.827 排列组合[16293:147053] <NSThread: 0x7fb34963c370>{number = 2, name = (null)} 9
```

结果显示，系统开了1条异步线程，因此全部在线程2执行，并且是顺序执行。主线程打印虽然在最上面，但是这个先后顺序是不确定，如果睡个0.001秒，主线程的打印会混在中间。

3.并发队列，异步执行

```
1 // 1. 队列
2 dispatch_queue_t q = dispatch_queue_create("dantesx", DISPATCH_QUEUE_CONCURRENT);
3
4 // 2. 异步执行
5 for (int i = 0; i<10; i++) {
6     dispatch_async(q, ^{
7         NSLog(@"%@ %d", [NSThread currentThread], i);
8     });
9 }
10 // [NSThread sleepForTimeInterval:2.0];
11 NSLog(@"董铂然 come here");
```

运行结果

```
2015-02-21 12:49:35.575 排列组合[16565:149686] <NSThread: 0x7fd042d04350>{number = 2, name = (null)} 0
2015-02-21 12:49:35.575 排列组合[16565:149687] <NSThread: 0x7fd042f13710>{number = 4, name = (null)} 3
2015-02-21 12:49:35.575 排列组合[16565:149611] 董铂然 come here
2015-02-21 12:49:35.575 排列组合[16565:149705] <NSThread: 0x7fd042f15f10>{number = 6, name = (null)} 4
2015-02-21 12:49:35.576 排列组合[16565:149705] <NSThread: 0x7fd042f15f10>{number = 6, name = (null)} 7
2015-02-21 12:49:35.577 排列组合[16565:149705] <NSThread: 0x7fd042f15f10>{number = 6, name = (null)} 8
2015-02-21 12:49:35.577 排列组合[16565:149705] <NSThread: 0x7fd042f15f10>{number = 6, name = (null)} 9
2015-02-21 12:49:35.576 排列组合[16565:149687] <NSThread: 0x7fd042f13710>{number = 4, name = (null)} 6
2015-02-21 12:49:35.575 排列组合[16565:149685] <NSThread: 0x7fd042f0e2f0>{number = 5, name = (null)} 2
2015-02-21 12:49:35.575 排列组合[16565:149688] <NSThread: 0x7fd042f0e030>{number = 3, name = (null)} 1
2015-02-21 12:49:35.576 排列组合[16565:149686] <NSThread: 0x7fd042d04350>{number = 2, name = (null)} 5
```

结果显示，主线程的打印还是混在中间不确定的，因为异步线程就是谁也不等谁。系统开了多条线程，并且执行的顺序也是乱序的

4.并发队列，同步执行

```
1 // 1. 队列
2 dispatch_queue_t q = dispatch_queue_create("dantesx", DISPATCH_QUEUE_CONCURRENT);
3
4 // 2. 同步执行
5 for (int i = 0; i<10; i++) {
6     dispatch_sync(q, ^{
7         NSLog(@"%@ %d", [NSThread currentThread], i);
8     });
9 }
10 // [NSThread sleepForTimeInterval:2.0];
11 NSLog(@"董铂然 come here");
```

运行结果

这个运行结果和第1种的串行队列，同步执行是一模一样的。因为同步任务的概念就是按顺序执行，后面都要等。言外之意就是不允许多开线程。同步和异步则是决定开一条还是开多条。

所以一旦是同步执行，前面什么队列已经没区别了。

5.主队列，异步执行

```
1 // 1. 主队列 - 程序启动之后已经存在主线程，主队列同样存在
2 dispatch_queue_t q = dispatch_get_main_queue();
```

JD.COM 京东



¥259.00

Objective-C Autorelease Pool...

联系我们



请扫描二维码联系
webmaster@csdn.net
400-660-0000
QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

```
5         dispatch_async(q, ^{
6             NSLog(@"%@ %d", [NSThread currentThread], i);
7         });
8     }
9     NSLog(@"睡会");
10    [NSThread sleepForTimeInterval:2.0];
11    NSLog(@"董铂然 come here");
```

运行结果

```
2015-02-22 00:03:42.630 排列组合[44708:413279] 睡会
2015-02-22 00:03:44.633 排列组合[44708:413279] 董铂然 come here
2015-02-22 00:03:44.643 排列组合[44708:413279] <NSThread: 0x7ff02250aef0>{number = 1, name = main} 0
2015-02-22 00:03:44.643 排列组合[44708:413279] <NSThread: 0x7ff02250aef0>{number = 1, name = main} 1
2015-02-22 00:03:44.643 排列组合[44708:413279] <NSThread: 0x7ff02250aef0>{number = 1, name = main} 2
2015-02-22 00:03:44.644 排列组合[44708:413279] <NSThread: 0x7ff02250aef0>{number = 1, name = main} 3
2015-02-22 00:03:44.644 排列组合[44708:413279] <NSThread: 0x7ff02250aef0>{number = 1, name = main} 4
2015-02-22 00:03:44.644 排列组合[44708:413279] <NSThread: 0x7ff02250aef0>{number = 1, name = main} 5
2015-02-22 00:03:44.644 排列组合[44708:413279] <NSThread: 0x7ff02250aef0>{number = 1, name = main} 6
2015-02-22 00:03:44.645 排列组合[44708:413279] <NSThread: 0x7ff02250aef0>{number = 1, name = main} 7
2015-02-22 00:03:44.645 排列组合[44708:413279] <NSThread: 0x7ff02250aef0>{number = 1, name = main} 8
2015-02-22 00:03:44.645 排列组合[44708:413279] <NSThread: 0x7ff02250aef0>{number = 1, name = main} 9
```

结果显示有点出人意料。主线程在睡会之后才打印，循环一直在等着。因为主队列的任务虽然会加到主线程中执行，但是如果**主线程里也有任务就必须等主线程任务执行完才轮到主队列的**。

6.主队列，同步执行

```
1    dispatch_queue_t q = dispatch_get_main_queue();
2
3    NSLog(@"卡死了吗? ");
4
5    dispatch_sync(q, ^{
6        NSLog(@"我来了");
7    });
8
9    NSLog(@"董铂然 come here");
```

运行结果为卡死

卡死的原因是**循环等待**，主队列的东西要等主线程执行完，而因为是同步执行不能开线程，所以下面的任务要等上面的任务执行完，所以卡死。这是排列组合中唯一一个会卡死的组合。

7.同步任务的使用场景

```
1    dispatch_queue_t q = dispatch_queue_create("dantesx", DISPATCH_QUEUE_CONCURRENT);
2    // 1. 用户登录，必须要第一个执行
3    dispatch_sync(q, ^{
4        [NSThread sleepForTimeInterval:2.0];
5        NSLog(@"用户登录 %@", [NSThread currentThread]);
6    });
7    // 2. 扣费
8    dispatch_async(q, ^{
9        NSLog(@"扣费 %@", [NSThread currentThread]);
10    });
11    // 3. 下载
12    dispatch_async(q, ^{
13        NSLog(@"下载 %@", [NSThread currentThread]);
14    });
15    NSLog(@"董铂然 come here");
```

运行结果

```
2015-02-22 00:12:14.857 排列组合[44794:417433] 用户登录 <NSThread: 0x7ff5eb414650>{number = 1, name = main}
2015-02-22 00:12:14.858 排列组合[44794:417433] 董铂然 come here
2015-02-22 00:12:14.858 排列组合[44794:417456] 下载 <NSThread: 0x7ff5eb611070>{number = 3, name = (null)}
2015-02-22 00:12:14.858 排列组合[44794:417458] 扣费 <NSThread: 0x7ff5eb7345f0>{number = 2, name = (null)}
```

结果显示，“用户登陆”在主线程打印，后两个在异步线程打印。上面的“用户登陆”使用同步执行，后面的扣费和下载都是异步执行。所以“用户登陆”必须第一个打印出来不管等多久，然后后面的两个异步和主线程打印会不确

JD.COM 京东



¥ 259.00

联系我们



请扫描二维码联系
webmaster@
400-660-0
QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

8.block异步任务包裹同步任务

```
1 dispatch_queue_t q = dispatch_queue_create("dantesx", DISPATCH_QUEUE_CONCURRENT);
2 void (^task)() = ^ {
3     // 1. 用户登录, 必须要第一个执行
4     dispatch_sync(q, ^{
5         NSLog(@"用户登录 %@", [NSThread currentThread]);
6     });
7     // 2. 扣费
8     dispatch_async(q, ^{
9         NSLog(@"扣费 %@", [NSThread currentThread]);
10    });
11    // 3. 下载
12    dispatch_async(q, ^{
13
14        NSLog(@"下载 %@", [NSThread currentThread]);
15    });
16 };
17
18 dispatch_async(q, task);
19 [NSThread sleepForTimeInterval:1.0];
20 NSLog(@"董铂然 come here");
```

运行结果

```
2015-02-22 00:19:05.310 排列组合[44844:421864] 用户登录 <NSThread: 0x7fd7b0f097f0>{number = 2, name = (null)}
2015-02-22 00:19:05.311 排列组合[44844:421864] 扣费 <NSThread: 0x7fd7b0f097f0>{number = 2, name = (null)}
2015-02-22 00:19:05.311 排列组合[44844:421864] 下载 <NSThread: 0x7fd7b0f097f0>{number = 2, name = (null)}
2015-02-22 00:19:06.311 排列组合[44844:421769] 董铂然 come here
```

因为整个block是在异步执行的，所以即使里面“用户登陆”是同步执行，那也无法在主线程中执行，只能开一条异步线程执行，因为是同步的所以必须等他先执行，后面的“扣费”和“下载”在上面同步执行结束之后，不确定顺序的打印。

9.全局队列

```
1 dispatch_queue_t q = dispatch_get_global_queue(0, 0);
2
3 for (int i = 0; i < 10; i++) {
4     dispatch_async(q, ^{
5         NSLog(@"%@ %d", [NSThread currentThread], i);
6     });
7 }
8 [NSThread sleepForTimeInterval:1.0];
9 NSLog(@"com here");
```

运行结果

```
2015-02-22 00:27:40.371 排列组合[44928:425000] 董铂然 come here
2015-02-22 00:27:40.372 排列组合[44928:425028] <NSThread: 0x7fab0b417220>{number = 2, name = (null)} 0
2015-02-22 00:27:40.373 排列组合[44928:425028] <NSThread: 0x7fab0b417220>{number = 2, name = (null)} 5
2015-02-22 00:27:40.374 排列组合[44928:425028] <NSThread: 0x7fab0b417220>{number = 2, name = (null)} 6
2015-02-22 00:27:40.374 排列组合[44928:425028] <NSThread: 0x7fab0b417220>{number = 2, name = (null)} 7
2015-02-22 00:27:40.372 排列组合[44928:425027] <NSThread: 0x7fab0b71be80>{number = 5, name = (null)} 3
2015-02-22 00:27:40.374 排列组合[44928:425028] <NSThread: 0x7fab0b417220>{number = 2, name = (null)} 8
2015-02-22 00:27:40.374 排列组合[44928:425027] <NSThread: 0x7fab0b71be80>{number = 5, name = (null)} 9
2015-02-22 00:27:40.372 排列组合[44928:425029] <NSThread: 0x7fab0b70a1c0>{number = 3, name = (null)} 1
2015-02-22 00:27:40.372 排列组合[44928:425055] <NSThread: 0x7fab0b4796b0>{number = 6, name = (null)} 4
2015-02-22 00:27:40.372 排列组合[44928:425026] <NSThread: 0x7fab0b6036e0>{number = 4, name = (null)} 2
```

全局队列的本质就是**并发队列**，只是在后面加入了，“服务质量”，和“调度优先级”两个参数，这两个参数一般为为了系统间的适配，**最好直接填0和0**。

如果不是在董铂然博客园看到这边文章请 [点击查看原文](#)

总结：

- 1. 开不开线程，取决于执行任务的函数，同步不开，异步开。
- 2. 开几条线程，取决于队列，串行开一条，并发开多条(异步)

JD.COM 京东



¥ 259.00

联系我们



请扫描二维码联系
✉ webmaster@
☎ 400-660-0
🗣 QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

- 4. 如果主线程上当前正在有执行的任务，主队列暂时不会调度任务的执行！主队列同步任务，会造成死锁。原因是循环等待
- 5. 同步任务可以队列调度多个异步任务前，指定一个同步任务，让所有的异步任务，等待同步任务执行完成，这是依赖关系。
- 6. 全局队列：并发，能够调度多个线程，执行效率高，但是相对费电。串行队列效率较低，省电省流量，或者是任务之间需要依赖也可以使用串行队列。
- 7. 也可以通过判断当前用户的网络环境来决定开的线程数。WIFI下6条，3G/4G下2~3条。

分类: Objective-C相关

标签: OC, iOS, 多线程, GCD, 排列组合, 主队列, 异步线程

目前您尚未登录，请 [登录](#) 或 [注册](#) 后参与评论

队列在多线程中使用

u010951897 2017年10月21日 12:42 238

1. 概述：1.1 队列简介队列是一种特殊的线性表，特殊之处在于它只允许在表的前端（front）进行删除操作，而在表的后端（rear）进行插入操作，和栈一样，队列是一种操作受限制的线性表。进...

Java多线程总结之线程安全队列Queue

madun 2014年03月02日 22:20 157593

在Java多线程应用中，队列的使用率很高，多数生产消费模型的首选数据结构就是队列。Java提供的线程安全的Queue可以分为阻塞队列和非阻塞队列，其中阻塞队列的典型例子是BlockingQueue，非...

5分钟完成加壳，防止代码反编译，防调试

一键加密代码逻辑，驱动级别反调试,秒杀常见调试器，无法dump内存。

广告



IOS对于线程队列的一些理解

jacky_jin 2017年03月30日 15:59 1423

第一部分：线程与队列 线程是代码执行的路径，队列则是用于保存以及管理任务的，线程负责去队列中取任务进行执行。例如：在主线程调用如下代码 dispatch_sync(queue, ^{ ...

线程池与线程队列分析—优

mawming 2016年08月24日 14:00 4318

· 线程池是对象池的一个有用的例子，它能够节省在创建它们时候的资源开销。并且线程池对系统中的线程数量也起到了很好的限制作用。· 线程池中的线程数量必须仔细的设置，否则冒然增加线程数量只会带来性...

人才引进落户

大幅放宽人才落户政策

百度广告



Java多线程 之 同步队列BlockingQueue与管道（十五）

一.同步队列BlockingQueue前面的两篇博文： Java多线程 之 生产者、消费者（十三） Java多线程 之 lock与condition的使用（十四） 详细阐述了多个任务之...

fan2012huan 2016年07月17日 15:41 1878

JAVA多线程与队列

wanghuan203 2014年12月31日 11:06 10364

JAVA 已经给我们提供了比较好的队列实现Queue，继承于Collection，本次我使用的是BlockingQueue，继承于Queue...

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册

JD.COM 京东



¥ 259.00

联系我们



请扫描二维码联系

✉ webmaster@

☎ 400-660-0

🗣 QQ客服 📞 电

关于 招聘 广告服务 🐾

©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

java多线程队列BlockingQueue的用法

landebug

2015年07月30日 12:29

2185

认识BlockingQueue阻塞队列，顾名思义，首先它是一个队列，而一个队列在数据结构中所起的作用大致如下图所示：从上图我们可以很清楚看到，通过一个共享的队列，可以使得数据由队列的一端输入，从另外...

Android中的消息队列和线程队列机制

zhoupenglei

2017年06月13日 10:02

1562

ndroid通过Looper、Handler来实现消息循环机制。Android的消息循环是针对线程的，每个线程都可以有自己的消息队列和消息循环。Android系统中的Looper负责管理线程的消息队列...

Java多线程总结之线程安全队列Queue

doutao6677

2017年02月07日 13:22

5705

BlockingQueue 阻塞算法ConcurrentLinkedQueue，非阻塞算法

iOS–多线程编程学习之GCD——串行队列和并发队列(五)

linyousong

2016年02月27日 09:51

2478

Grand Central Dispatch(GCD)有很多部分构成，例如有很好的语言特性，运行库，还提供了系统的、高效的方式来支持具有多核处理器的iOS和OS X设备进行并发事件处理。 BSD...

恭喜：获3支牛股√

股票别乱买！牛人教你正确选股赚钱

领

秒杀多线程第十六篇 多线程十大经典案例之一 双线程读写队列数据

MoreWindows

2013年03月13日 02:28

61149

本文配套程序下载地址为：http://download.csdn.net/detail/morewindows/5136035转载请标明出处，原文地址：http://blog.csdn.net/mor...

主队列，全局队列，并发队列关系比较

FocusOnLovingFreedom

2015年11月17日 15:46

3374

多线程（2） – ios关于线程调度的三种方式之GCD GCD（链接：http://blog.csdn.net/linzhiji/article/details/6863972） – 概念...

队列的线程安全

u014788227

2016年10月20日 15:45

1145

在Java多线程应用中，队列的使用率很高，多数生产消费模型的首选数据结构就是队列(先进先出)。Java提供的线程安全的Queue可以分为阻塞队列和非阻塞队列，其中阻塞队列的典型例子是BlockingQ...

14.[个人]C++线程入门到进阶（14）-----双线程读写队列数据

shuimanting520

2017年04月25日 23:40

516

本文配套程序下载地址为：http://download.csdn.net/detail/morewindows/5136035 转载请标明出处，原文地址：http://blog.csdn.net/m...

秒杀多线程第十六篇 多线程十大经典案例之一 双线程读写队列数据

2013年03月13日 10:47

39KB

下载

RAR

笔记本租赁

网上的出租笔记本电脑可信吗

百度广告

笔记本租赁

给定程序 使用两个线程 一个队列 并由一个线程从键盘读取数据

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录 注册

JD.COM 京东

¥ 259.00

联系我们

微信扫一扫联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

iOS中多线程知识总结：进程、线程、GCD、串行队列、并行队列、全局队列、主线程队...

iOS中多线程知识总结：进程、线程、GCD、串行队列、并行队列、全局队列、主线程队列、同步任务、异步任务等（有示例代码）进程正在运行中的程序被称作进程，负责程序运行的内存分配;每一个进程都有自己独立的...

 JacobKong 2015年08月22日 12:38  2896

Java多线程总结之线程安全队列Queue

 bieleyang 2017年09月19日 10:21  2856

在Java多线程应用中，队列的使用率很高，多数生产消费模型的首选数据结构就是队列。Java提供的线程安全的Queue可以分为阻塞队列和非阻塞队列，其中阻塞队列的典型例子是BlockingQueue，非...

多线程队列

2013年08月22日 10:54 8KB [下载](#)



JD.COM 京东



¥ 259.00

联系我们



请扫描二维码联系
 webmaster@
 400-660-0
 QQ客服 

关于 招聘 广告服务 
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心