

给初学者的RxJava2.0教程(五)



Season_zlc (/u/c50b715ccaeb) [+ 关注](#)

2016.12.13 12:19* 字数 1489 阅读 22409 评论 75 喜欢 303 赞赏 15

(/u/c50b715ccaeb)

Outline

[TOC]

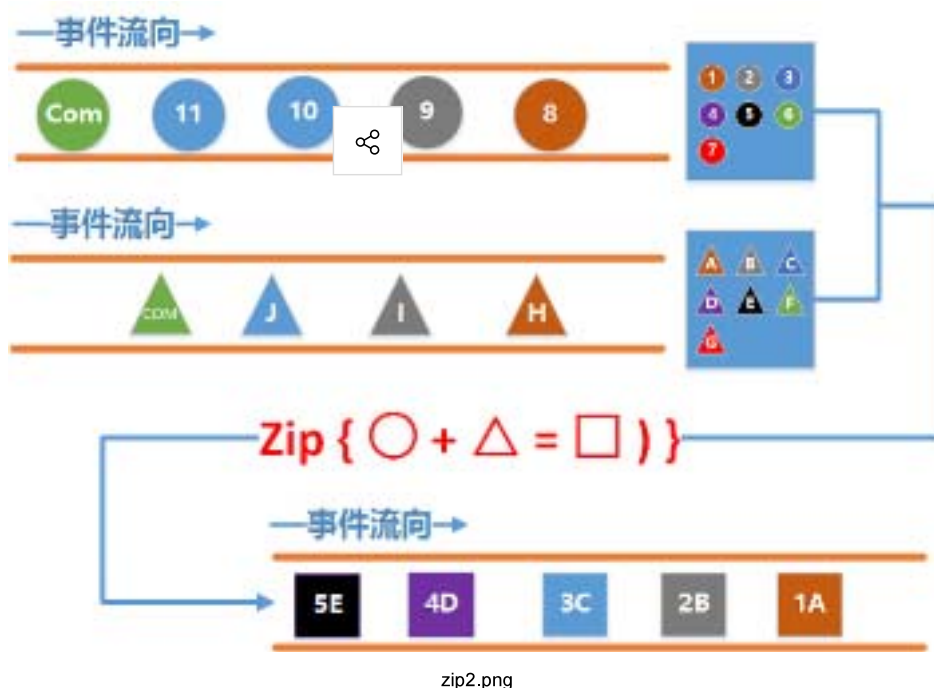
前言

大家喜闻乐见的 Backpressure 来啦。

这一节中我们将来学习 Backpressure，我看好多吃瓜群众早已坐不住了，别急，我们先来回顾一下上一节讲的 Zip。

正题

上一节中我们说到Zip可以将多个上游发送的事件组合起来发送给下游，那大家有没有想过一个问题，如果其中一个水管A发送事件特别快，而另一个水管B发送事件特别慢，那就可能出现这种情况，发得快的水管A已经发送了1000个事件了，而发的慢的水管B才发一个出来，组合了一个之后水管A还剩999个事件，这些事件需要继续等待水管B发送事件出来组合，那么这么多的事件是放在哪里的呢？总有一个地方保存吧？没错，Zip给我们的每一根水管都弄了一个水缸，用来保存这些事件，用通俗易懂的图片来表示就是：



如图中所示，其中蓝色的框框就是 zip 给我们的水缸！它将每根水管发出的事件保存起来，等两个水缸都有事件了之后就分别从水缸中取出一个事件来组合，当其中一个水缸是空的时候就处于等待的状态。

题外话：大家来分析一下这个水缸有什么特点呢？它是按顺序保存的，先进来的事件先取出来，这个特点是不是很熟悉呀？没错，这就是我们熟知的队列，这个水缸在Zip内部的实现就是用的队列，感兴趣的可以翻看源码查看。

好了回到正题上来, 这个水缸有大小限制吗? 要是—直往里存会怎样? 我们来看个例子:

```
Observable<Integer> observable1 = Observable.create(new ObservableOnSubscribe<Integer>() {
    @Override
    public void subscribe(Observer<Integer> emitter) throws Exception {
        for (int i = 0; ; i++) {    //无限循环发事件

            emitter.onNext(i);
        }
    }
}).subscribeOn(Schedulers.io());

Observable<String> observable2 = Observable.create(new ObservableOnSubscribe<String>() {
    @Override
    public void subscribe(Observer<String> emitter) throws Exception {
        emitter.onNext("A");
    }
}).subscribeOn(Schedulers.io());

Observable.zip(observable1, observable2, new BiFunction<Integer, String, String>() {
    @Override
    public String apply(Integer integer, String s) throws Exception {
        return integer + s;
    }
}).observeOn(AndroidSchedulers.mainThread()).subscribe(new Consumer<String>() {
    @Override
    public void accept(String s) throws Exception {
        Log.d(TAG, s);
    }
}, new Consumer<Throwable>() {
    @Override
    public void accept(Throwable throwable) throws Exception {
        Log.w(TAG, throwable);
    }
});
```

在这个例子中, 我们分别创建了两根水管, 第一根水管用机器指令的执行速度来无限循环发送事件, 第二根水管随便发送点什么, 由于我们没有发送 Complete 事件, 因此第一根水管会一直发事件到它对应的水缸里去, 我们来看看运行结果是什么样.

运行结果GIF图:



zip2.gif

我勒个草, 内存占用以斜率为1的直线迅速上涨, 几秒钟就300多M, 最终报出了OOM:

```
zlc.season.rxjava2demo W/art: Throwing OutOfMemoryError "Failed to allocate a 28 byte allocation with
4194304 free bytes and 8MB until OOM;
zlc.season.rxjava2demo W/art: "main" prio=5 tid=1 Runnable
zlc.season.rxjava2demo W/art:   | group="main" sCount=0 dsCount=0 obj=0x75188710 self=0x7fc0efe7ba00
zlc.season.rxjava2demo W/art:   | sysTid=32686 nice=0 cgrp=default sched=0/0 handle=0x7fc0f37dc200
zlc.season.rxjava2demo W/art:   | state=R schedstat=( 0 0 0 ) utm=948 stm=120 core=1 HZ=100
zlc.season.rxjava2demo W/art:   | stack=0x7fff971e8000-0x7fff971ea000 stackSize=8MB

zlc.season.rxjava2demo W/art:   | held mutexes= "mutator lock"(shared held)
zlc.season.rxjava2demo W/art:   at java.lang.Integer.valueOf(Integer.java:742)
```

出现这种情况肯定是我们不想看见的, 这里就可以引出我们的 Backpressure 了, 所谓的 Backpressure 其实就是为了控制流量, 水缸存储的能力毕竟有限, 因此我们还得从 源头 去解决问题, 既然你发那么快, 数据量那么大, 那我就想办法不让你发那么快呗.

那么这个 源头 到底在哪里, 究竟什么时候会出现这种情况, 这里只是说的Zip这一个例子, 其他的地方会出现吗? 带着这个问题我们来探究一下.

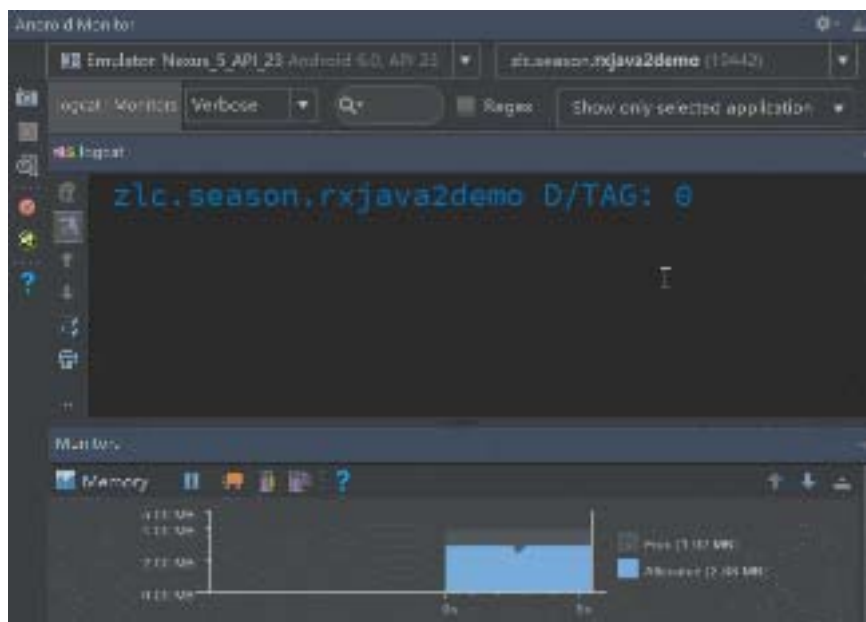
我们让事情变得简单一点, 从一个单一的 Observable 说起.

来看段代码:

```
Observable.create(new ObservableOnSubscribe<Integer>() {
    @Override
    public void subscribe(Observer<Integer> emitter) throws Exception {
        for (int i = 0; ; i++) {    //无限循环发事件

            emitter.onNext(i);
        }
    }
}).subscribe(new Consumer<Integer>() {
    @Override
    public void accept(Integer integer) throws Exception {
        Thread.sleep(2000);
        Log.d(TAG, "" + integer);
    }
});
```

这段代码很简单, 上游同样无限循环的发送事件, 在下游每次接收事件前延时2秒. 上下游工作在 同一个线程 里, 来看下运行结果:



哎卧槽, 怎么如此平静, 感觉像是走错了片场.

为什么呢, 因为上下游工作在 同一个线程 呀骚年们! 这个时候上游每次调用 `emitter.onNext(i)` 其实就相当于直接调用了Consumer中的:

```
public void accept(Integer integer) throws Exception {
    Thread.sleep(2000);
    Log.d(TAG, "" + integer);
}
```

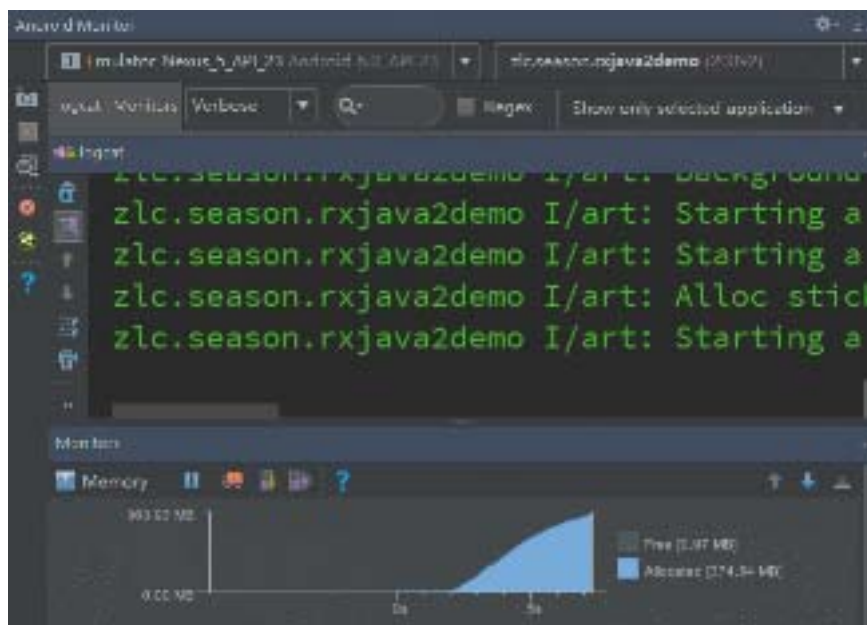
所以这个时候其实就是上游每延时2秒发送一次, 最终的结果也说明了这一切.

那我们加个线程呢, 改成这样:

```
Observable.create(new ObservableOnSubscribe<Integer>() {
    @Override
    public void subscribe(Observer<Integer> emitter) throws Exception {
        for (int i = 0; ; i++) {    //无限循环发事件

            emitter.onNext(i);
        }
    }
}).subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .subscribe(new Consumer<Integer>() {
        @Override
        public void accept(Integer integer) throws Exception {
            Thread.sleep(2000);
            Log.d(TAG, "" + integer);
        }
    });
```

这个时候把上游切换到了IO线程中去, 下游到主线程去接收, 来看看运行结果呢:



violence.gif

可以看到, 给上游加了个线程之后, 它就像脱缰的野马一样, 内存又爆掉了.

为什么不加线程和加上线程区别这么大呢, 这就涉及了 同步 和 异步 的知识了.

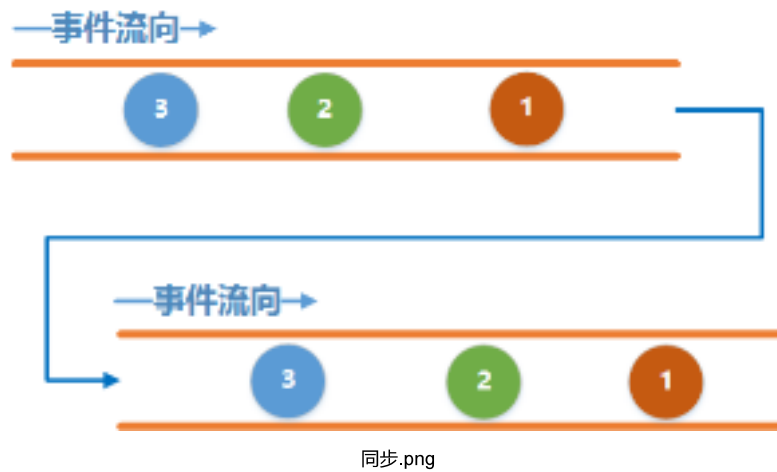
当上下游工作在 同一个线程 中时, 这时候是一个 同步 的订阅关系, 也就是说 上游 每发送一个事件 必须 等到 下游 接收处理完了以后才能接着发送下一个事件.

当上下游工作在 不同的线程 中时, 这时候是一个 异步 的订阅关系, 这个时候 上游 发送数据 不需要 等待 下游 接收, 为什么呢, 因为两个线程并不能直接进行通信, 因此上游发送的事件并不能直接到下游里去, 这个时候就需要一个田螺姑娘来帮助它们俩, 这个田螺姑娘就是我

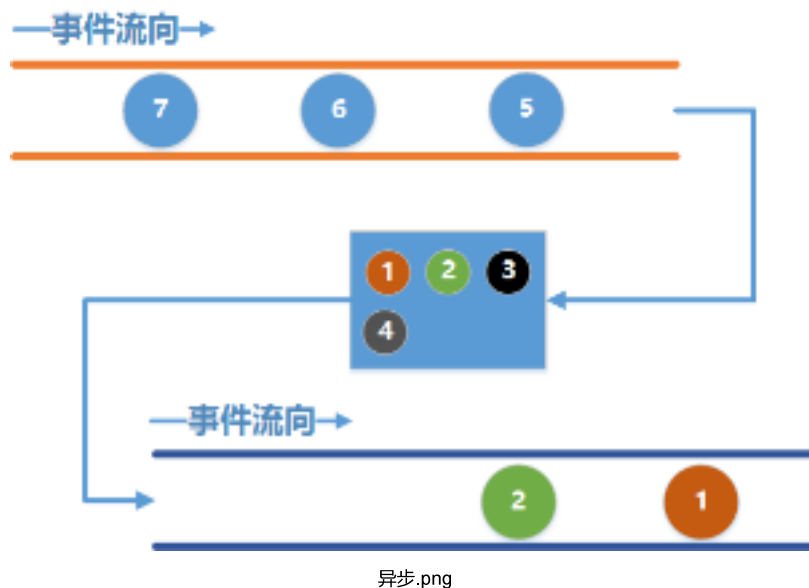
们刚才说的 水缸 ！上游把事件发送到水缸里去，下游从水缸里取出事件来处理，因此，当上游发事件的速度太快，下游取事件的速度太慢，水缸就会迅速装满，然后溢出来，最后就OOM了。

这两种情况用图片来表示如下：

同步：



异步：



从图中我们可以看出，同步和异步的区别仅仅在于是否有 水缸 。

相信通过这个例子大家对线程之间的通信也有了比较清楚的认知和理解。

源头找到了，只要有 水缸，就会出现上下游发送事件速度不平衡的情况，因此当我们以后遇到这种情况时，仔细思考一下水缸在哪里，找到水缸，你就找到了解决问题的办法。

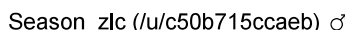
既然源头找到了，那么下一节我们就要来学习如何去解决了。下节见。

打赏功能还是要有的，万一谁手抖了呢

赞赏支持



举报文章 © 著作权归作者所有



+关注

喜欢 303



更多分享



下载简书 App ▶
随时随地发现和创作内容

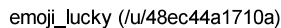


(/apps/download?utm_source=nbc)



只看作者

按喜欢排序 按时间正序 按时间倒序




27楼 · 2017.06.14 17:10

(/u/48ec44a1710a)


rxjava2.x的Observable是不存在背压的概念的，首先博主都没有完全理解什么是背压，背压是下游控制上游流速的一种手段。在rxjava1.x的时代，上游会给下游set一个producer，下游通过producer向上游请求n个数据，这样上游就有记录下游请求了多少个数据，然后下游请求多少个上游就给多少个，这个就是背压。一般来讲，每个节点都有缓存，比如说缓存的大小是64，这个时候下游可以一次性向上游request 64个数据。rxjava1.x的有些操作符不支持背压，也就是说这些操作符不会给下游set一个producer，也就是上游根本不理睬下游的请求，一直向下游丢数据，如果下游的缓存爆了，那么下游就会抛出MissingBackpressureException，也就是背压失效了。在rxjava2.x时代，上述的背压逻辑全部搬到Flowable里了，所以说Flowable支持背压。而2.x时代的Observable是没有背压的概念的，Observable如果来不及消费会死命的缓存直到OOM，所以rxjava2.x的官方文档里面有讲，大数据流用Flowable，小数据流用Observable

23人赞 回复


唐诗淳 (/u/e979a8a00c84): 老哥,你先把博主这系列看完再来喷好吗~

2017.06.25 23:51  回复

我是高手 (/u/cde131ae7ab0) : @唐诗淳 (/users/e979a8a00c84) 人家没喷啊, 首先博主写的文章很好, 但rxjava 2.0 observable确实没有背压的概念, 你可以看看其他文章

2017.07.10 17:16  回复


我的QQ宠物 (/u/923b3cd71955): 这是学术的课堂, 不要什么喷不喷的就上来了

2017.07.17 19:54  回复


daocaowe (/u/8cf7fd79ddf7) : 我到底听谁的


2017.08.01 17:04 回复


FF4081颜色的大野牛 (/u/6c3fd4cc938f) : @daocaowe (/users/8cf7fd79ddf7) 初学者先看水管图, 把基本概念理解。

2017.08.07 14:01  回复


爱哭的笨小孩 (/u/cb6fc788160c) : @唐诗淳 (/users/e979a8a00c84) 这叫学术交流

2017.08.11 16:43  回复


ChangQin (/u/601bff1a5d52) : @唐诗淳 (/users/e979a8a00c84) 搞笑, 你看看人回那么多, 是喷? 

2017.08.16 17:29  回复


沒事偷著樂 (/u/cde86b15d3d6) : 这一章节 只是解释 背压 下一章节刘Flowabl了

2017.08.16 18:01  回复


天门一只猿 (/u/b1938fc462f9) : 恩恩, 都有道理, 感谢补充。这里没有不尊敬作者的地方, 学术交流很正常

2017.09.06 11:44  回复


狠喜欢 (/u/6b3e44548091) : 博主也没说observable 有背压啊 只是举例


2017.09.20 17:25  回复

de2078abd1f9 (/u/de2078abd1f9) : 博主说的没有问题, 博主说了, 背压是控制水流嘛, 这篇只是引出背压的概念了

2017.09.26 18:14  回复

ITlan (/u/2034ae2eb9b8) : 评论里的人, 人家这叫喷吗? 提出理性质疑是很好的事, 另外层主, 博主并没有说observable有背压, 下一节说不定就说了

2017.10.27 18:00  回复

 添加新评论 | 收起



小小小池 (/u/067d829266d9)


7楼 · 2016.12.14 17:13

(/u/067d829266d9)

楼主, 2.0 Flowable 取消订阅这块怎么弄? 1.x subscribe 订阅是会返回一个 Subscription, 然后mCompositeSubscription.add(Subscription), 在 Activity 退出时取消订阅 mCompositeSubscription.unsubscribe(), 2.x, subscribe订阅 返回值是 void 了, 2.0把Subscription 放到 onSubscribe(Subscription s)中去了, 那现在是在直接在 onSubscribe(Subscription s)方法中在 s.request(1)后s.cancel()?

2人赞  回复

小小小池 (/u/067d829266d9) : @小小小池 (/users/067d829266d9) 楼主, 我的问题另外一个大神帮解决了。以下是他的回答:Flowable 提供了 subscribeWith 方法可以返回当前订阅的观察者, 并且通过 ResourceSubscriber 这种类来提供 Disposable 接口 按照你之前的做法, 现在应该是这样 CompositeDisposable composite2 = new CompositeDisposable(); composite2.add(Flowable.range(1, 5).subscribeWith(subscriber)); subscriber 应该是 ResourceSubscriber 或者 DisposableSubscriber 的实例

2016.12.14 19:28  回复

 添加新评论




沒事偷著樂 (/u/cde86b15d3d6)


10楼 · 2016.12.23 14:41

(/u/cde86b15d3d6)


额 打赏都出现bug了

1人赞  回复


Season_zlc (/u/c50b715ccaeb) : @沒事偷著樂 (/users/cde86b15d3d6) 骗人! 


2016.12.23 15:02  回复

沒事偷著樂 (/u/cde86b15d3d6) : @Season_zlc (/users/c50b715ccaeb) 木有骗你啊

2016.12.23 15:26  回复

d8184ca3c970 (/u/d8184ca3c970) : @沒事偷著樂 (/users/cde86b15d3d6) 为什么我没有bug啊

2017.02.11 14:12  回复

 添加新评论



KingJA (/u/8a1a8ed656e8)


23楼 · 2017.05.25 09:52


(/u/8a1a8ed656e8)

博主的无限发送事件的过程中怎么没进行垃圾自动回收？我测试过程中会经常性被JVM回收，内存不会直线上升

1人赞  回复

王蛋蛋的father (/u/8d31dba9b5c4)：我也没有oom😓

2017.12.30 14:30  回复

 添加新评论



小新GG (/u/89e34b9ce1d4)


2楼 · 2016.12.13 17:41

(/u/89e34b9ce1d4)

楼主讲的很详细,,,,希望速度快点啊,,,,还有希望深入一点 针对Flowable 和Rxbus 多写点啊

赞  回复


Season_zlc (/u/c50b715ccaeb)：@zhouxin1233 (/users/89e34b9ce1d4) 我每天还要上班呀，都是抽空写的，还要画图，各位多多理解。□

2016.12.13 17:45  回复

追风917 (/u/8cb49b5ad78b)：@Season_zlc (/users/c50b715ccaeb) 图画的相当棒啊,小学生都看明白了,嘿嘿

2017.05.13 12:43  回复

Sun_951d (/u/3124e231c257)：@Season_zlc (/users/c50b715ccaeb) 请问一下，这个是什么画图工具？好nice的样子

2017.11.20 16:32  回复

 添加新评论 | 还有1条评论，[展开查看](#)



小爨 (/u/9acf5e154ab0)

3楼 · 2016.12.14 01:49

(/u/9acf5e154ab0)

不错

赞  回复



小vv小池 (/u/067d829266d9)


4楼 · 2016.12.14 08:47

(/u/067d829266d9)


楼主，快更新啊，吃不饱啊，多更新些Flowable 啊

赞  回复


Season_zlc (/u/c50b715ccaeb)：@小vv小池 (/users/067d829266d9) 哼，你们给的赞太少了，本宝宝生气□

2016.12.14 08:52  回复


追风917 (/u/8cb49b5ad78b)：@Season_zlc (/users/c50b715ccaeb) 哈哈哈哈哈，撒娇，

2017.05.13 12:43  回复

 添加新评论

 starCoder (/u/7fd1c374bf10)
5楼 · 2016.12.14 09:38
(/u/7fd1c374bf10)
一直跟着看，讲的不错

赞 回复

 junerver (/u/5246e1822cf5)
6楼 · 2016.12.14 10:24
(/u/5246e1822cf5)
楼主我恨你，天天跟追小说似地等更新 😞

赞 回复


Season_zlc (/u/c50b715ccaeb) : @junerver (/users/5246e1822cf5) 哈哈, 就是喜欢看你们着急的样子~(≥▽≤)/~

2016.12.14 11:20 回复

追风917 (/u/8cb49b5ad78b) : @Season_zlc (/users/c50b715ccaeb) 撩的一手好妹 😏

2017.05.13 12:44 回复

添加新评论


 墨痕坊 (/u/d2ab3859e18f)
8楼 · 2016.12.19 14:33
(/u/d2ab3859e18f)
猝不及防，结尾卖的一手好萌

赞 回复


Season_zlc (/u/c50b715ccaeb) : @墨痕坊 (/users/d2ab3859e18f) □

2016.12.19 17:42 回复


添加新评论

 久山崎 (/u/c31d01cd6aff)
9楼 · 2016.12.22 10:09
(/u/c31d01cd6aff)
😊看到结尾....


赞 回复

 lyangc (/u/44e85cc61626)
11楼 · 2017.02.04 16:06
(/u/44e85cc61626)
实在太好了 给楼主打赏了下，感谢楼主的好文章，我接着往下看...还有，楼主我在前面问了一个问题 你空了看到 指点一下我啦。👏

赞 回复

 kuwork (/u/8741fa1e0393)
12楼 · 2017.02.08 15:12
(/u/8741fa1e0393)
写得真好，很详细。TOC无效，加一个目录形式的链接就更好了

赞 回复

 ling9400 (/u/44dd2c2ff016)
13楼 · 2017.02.10 17:23
(/u/44dd2c2ff016)
楼主，最后的异步发送的，我测试了内存也不会直接爆表啊！感觉还是逐步递增的，不知道是哪里的的问题？

赞 回复



d8184ca3c970 (/u/d8184ca3c970)

14楼 · 2017.02.14 07:59

(/u/d8184ca3c970)
求楼主建群

赞 回复

Season_zlc (/u/c50b715ccaeb) : @d8184ca3c970 (/users/d8184ca3c970) 你可以加我
RxDownload的讨论群603610731

2017.02.14 09:17 回复

添加新评论

1

2

3

下一页

被以下专题收入，发现更多相似内容



RxJava2.x (/c/299d0a51fdd4?utm_source=desktop&utm_medium=notes-included-collection)



RxJava (/c/a904f328163d?utm_source=desktop&utm_medium=notes-included-collection)



Android知识 (/c/3fde3b545a35?utm_source=desktop&utm_medium=notes-included-collection)



Android开发 (/c/d1591c322c89?utm_source=desktop&utm_medium=notes-included-collection)



Android... (/c/58b4c20abf2f?utm_source=desktop&utm_medium=notes-included-collection)



首页投稿 (/c/bDHhpK?utm_source=desktop&utm_medium=notes-included-collection)



Android... (/c/b5617afad677?utm_source=desktop&utm_medium=notes-included-collection)

展开更多

推荐阅读

更多精彩内容 > (/)

给初学者的RxJava2.0教程(四) (/p/bb58571cdb64?utm...

(/p/bb58571cdb64?

Outline [TOC] 前言 在上一节中, 我们提到了Flowable 和Backpressure背压, 本来这一节的确是讲这两个东西的, 可是写到一半感觉还是差点火候, 感觉...

utm_campaign=maleskine&utm_content=note&utm...

Season_zlc (/u/c50b715ccaeb?

utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

造轮子 - PracticalRecyclerView (/p/e1787fa7913d?utm_campaign=mal...

PracticalRecyclerView 标签 (空格分隔) : Android RecyclerView 对RecyclerView的一个封装,添加一些实用的功能 项目地址 PracticalRecyclerView 效果图与示例APK APK下载地址 主要功能: 下拉刷新,使用默认...

Season_zlc (/u/c50b715ccaeb?

utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

我才二十多岁, 就怕再也遇不到真爱了 (/p/d9cce2716...

(/p/d9cce2716bc2?

2018年1月27日 周六 大雪 文 # 阿呗 你可以不在乎我的感受, 也可以不在乎我的热情, 甚至不用理会我的沮丧与难过。 但你要知道, 每个人的付出都是有限...

utm_campaign=maleskine&utm_content=note&utm...

阿咻有个帅阿呗 (/u/bfb1aa483a03?

utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

快速行动：成为一个永不拖延的人，你需要这8本书 (/p/2dc061e5ea18?

(/p/2dc061e5ea18?

拖延是每个追求上进的人不得不面对和解决的问题，有些人认为，凡事都可以“等一会儿”，偶尔等一会没关系，认为船到桥头自然直，天无绝人之路，拖...

utm_campaign=maleskine&utm_content=note&utm

田宝谈写作 (/u/09c373f051cf?

utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

从韩寒到刘亦婷：中国式教育的冰与火之歌 (/p/c41991...

(/p/c4199181c732?

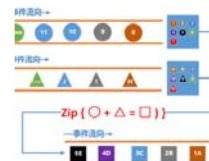
-壹- 就在前几天，刚刚在微博上做完访谈的韩寒发了一篇文章，叫《我所理解的教育》。截止到我下定决心写这篇文章的时候，这篇文章已经阅读量破1060...

utm_campaign=maleskine&utm_content=note&utm

黄不会 (/u/01229c4b4f5a?

utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

(/p/931fcf9c23f6?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

给初学者的RxJava2.0教程(五) (/p/931fcf9c23f6?utm_campaign=maleski...

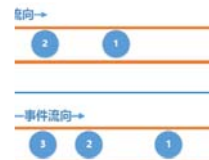
特此声明：本文为转载文章！尊重原创的劳动果实，严禁剽窃本文转载于：

http://www.jianshu.com/p/0f2d6c2387c9出自于：【Season_zlc】前言 大家喜闻乐见的Backpressure来...

社会你鹏哥 (/u/d512e2781e94?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/f4ed455de5f0?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

给初学者的RxJava2.0教程(七) (/p/f4ed455de5f0?utm_campaign=malesk...

特此声明：本文为转载文章！尊重原创的劳动果实，严禁剽窃本文转载于：

http://www.jianshu.com/p/9b1304435564出自于：【Season_zlc】前言 上一节里我们学习了只使用...

社会你鹏哥 (/u/d512e2781e94?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

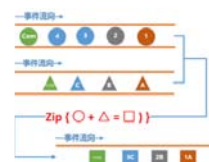
RxJava 2.x 学习（转载） (/p/b3a63548514e?utm_campaign=maleskine...

转载自：https://xiaobailong24.me/2017/03/18/Android-RxJava2.x/ 前言最近学习了一下RxJava，发现是个好东西，有点相见恨晚的感觉，一开始学习了RxJava 1.x，看了很多国内的博客，有点理解了，后来发现...

Young1657 (/u/64cccdaa9204?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/9da04136b5f6?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

给初学者的RxJava2.0教程(四) (/p/9da04136b5f6?utm_campaign=males...

特此声明：本文为转载文章！尊重原创的劳动果实，严禁剽窃本文转载于：

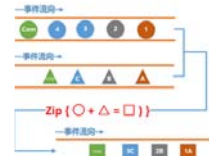
http://www.jianshu.com/p/bb58571cdb64出自于：【Season_zlc】前言 在上一节中，我们提到了 Flowabl...



社会你鹏哥 (/u/d512e2781e94?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/bb58571cdb64?)



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

给初学者的RxJava2.0教程(四) (/p/bb58571cdb64?utm_campaign=males...

Outline [TOC] 前言 在上一节中, 我们提到了Flowable 和Backpressure背压, 本来这一节的确是想讲这两个东西的, 可是写到一半感觉还是差点火候, 感觉时机未到, 因此, 这里先来做个准备工作, 先带大家学习zi...



Season_zlc (/u/c50b715ccaeb?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

Southfields (/p/c374e06df87c?utm_campaign=maleskine&utm_conten...

August30, 2007 其实回来也有两三天了, 我住在伦敦西南部靠近温布尔登的小镇Southfields, 一片非常宁静美丽的地方, 寓所叫Struan House, 是天主教的修女管辖的产业, 这座三层尖顶小楼里住的全是女孩子, ...



seg (/u/9c8b7850cf4a?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/053b066f94e0?)



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

三十岁女人丰胸是一种什么体验? (/p/053b066f94e0?utm_campaign=mal...

其实我小时候, 就没太注意这种东西, 但是后来慢慢的就发现自己这方面不太好, 女孩子吗, 都是爱美的, 也对这些事情敏感, 眼看着人家发育的都比我好, 心里就不是滋味, 但也不知道到底是啥原因, 它就是不...



一人行者! (/u/17e841b2549c?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

“政治”杂谈 2017/1/22 星期日 19:36 (/p/ecbaccb0abf0?utm_campaign=m...

(一) 今天写的日记是有题目的, 也就说明今天的这篇日记我想写一些东西, 而非像之前的一样仅仅是记录, 记录当然是有用的, 以后可以从中得到想法啊, 教训啊或者启示, 写东西也很重要, 就像以前的知识...



LeBronJames (/u/afd44853f115?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/38b1f2da857c?)



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

销售高手支招: “小恩小惠”助你开单 (/p/38b1f2da857c?utm_campaign=...

销售高手包里总是装着外国香烟和日本美食, 见了男的就是烟熊火燎, 见了女的就是糖衣炮弹。正所谓: 吃人的嘴短, 拿人的手短。他吃了你的东西或者是拿了你的东西, 他总得对你有点表示啊。这是我们中国多...



骑毛驴儿上京 (/u/c07010d894b9?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

因为喜欢，我变成了嗤之以鼻的那种自己 (/p/a6d9ea81185e?utm_campai...

有种说法是，姑娘说的分手都是在抗议“你还不够爱我”。桃子的身边有好多对，分分合合无数次，却至今在一起。她曾经对这种类型的情侣嗤之以鼻，幼稚、没事找事，此类的标签被无情地贴在他们身上。桃子或...



Cynthiaaaaaa (/u/a4cde10a40b8?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)