

大头鬼Bruce 程序人生

目录视图

摘要视图

RSS 订阅

我的微博

hi大头鬼hi

我的其他资料

我的Github

文章分类

Android (15)

Animation (1)

Gesture (1)

RxJava (7)

Square (2)

Gradle (8)

open resty (1)

阅读排行

深入浅出RxJava (一: [基础](#))

(65440)

深入浅出RxJava四-在An

(22651)

深入浅出RxJava(二: 操

(18653)

深入浅出RxJava三-响应

(14643)

Android实现类似QQ的滑

(7964)

Android热更新实现原理

(7873)

RxJava使用场景小结

(6718)

深入浅出Android Gradle

(4261)

Otto使用入门

(4199)

RxJava基本流程和lif

源码 (3894)

评论排行

深入浅出RxJava (一: [基础](#))

(56)

深入浅出RxJava(二: 操

(26)

深入浅出RxJava四-在An

(18)

深入浅出RxJava三-响应

(16)

RxJava基本流程和lif

源码 (9)

RxJava使用场景小结

(8)

使用动画和fragment改善

(6)

Android实现类似QQ的滑

(6)

Android热更新实现原理

(5)

深入浅出Android Gradle

(5)

个人资料

深入浅出RxJava三--响应式的好处

标签: RxJava android 响应式编程

2015-04-05 21:52 14646人阅读 评论(16) 收藏 举报

分类: RxJava (6) ▾

目录(?)

[+]

原文链接

在第一篇中，我介绍了RxJava的基础知识。第二篇中，我向你展示了操作符的强大。但是你可能仍然没被说服。这篇里面，我讲向你展示RxJava的其他的一些好处，相信这篇足够让你去使用Rxjava.

错误处理

到目前为止，我们都还没怎么介绍onComplete()和onError()函数。这两个函数用来通知订阅者，被观察的对象将停止发送数据以及为什么停止（成功的完成或者出错了）。

下面的代码展示了怎么使用这两个函数：

```

1 Observable.just("Hello, world!")
2     .map(s -> potentialException(s))
3     .map(s -> anotherPotentialException(s))
4     .subscribe(new Subscriber<String>() {
5         @Override
6         public void onNext(String s) { System.out.println(s); }
7
8         @Override
9         public void onComplete() { System.out.println("Completed!"); }
10
11        @Override
12        public void onError(Throwable e) { System.out.println("Ouch!"); }
13    });

```

代码中的potentialException() 和 anotherPotentialException()有可能会抛出异常。每一个Observable对象在终结的时候都会调用onCompleted()或者onError()方法，所以Demo中会打印“ Completed!” 或者“ Ouch! ”。

这种模式有以下几个优点：

1.只要有异常发生onError()一定会被调用

这极大的简化了错误处理。只需要在一个地方处理错误即可以。

2.操作符不需要处理异常

将异常处理交给订阅者来做，Observable的操作符调用链中一旦有一个抛出了异常，就会直接执行onError()方法。

3.你能够知道什么时候订阅者已经接收了全部的数据。

知道什么时候任务结束能够帮助简化代码的流程。（虽然有可能Observable对象永远不会结束）

我觉得这种错误处理方式比传统的错误处理更简单。传统的错误处理中，通常是在每个回调中处理错误。这不仅导致了重复的代码，并且意味着每个回调都必须知道如何处理错误，你的回调代码将和调用者紧耦合在一起。



hi大头鬼 hi



访问: 285484次

积分: 1589

等级: 4

排名: 第15832名

原创: 14篇 转载: 0篇

译文: 12篇 评论: 174条

文章搜索

文章存档

2015年12月 (1)

2015年11月 (4)

2015年10月 (1)

2015年09月 (1)

2015年07月 (3)

展开

推荐文章

- * HDFS如何检测并删除多余副本块
- * Project Perfect让Swift在服务器端跑起来—让Perfect更Rails (五)
- * 数据库性能优化之SQL语句优化
- * Animation动画详解(七)——ObjectAnimator基本使用
- * 机器学习系列(7)_机器学习路线图 (附资料)
- * 大数据三种典型云服务模式

最新评论

深入浅出RxJava (一: 基础篇)
AndroidSummer: 看了很多相关的文章，这个入门不错

深入浅出RxJava(二: 操作符)
鱼塘鱼汤: @bfbx5173:个人觉得能不用就最好别用那玩意儿。如果要做一些热修复类似的事情，就悲剧了。

深入浅出RxJava (一: 基础篇)
放风筝的骚年: 不明觉厉

深入浅出RxJava(二: 操作符)
4805凯盛: @qiantujava.onNext里面 不应该是List集合吗 为什么是“XXX”

深入浅出RxJava (一: 基础篇)
walfud: 感谢抛物线大神和Bruce!我也写了一系列源码分析，欢迎交流: '彻底搞懂 RxJava ----'

RxJava基本流程和ifit源码分析
walfud: 感谢!看过你的文章深受启发，我也写了一系列源码分析，请多指教: 基础 <http://diordn...>

使用RxJava，Observable对象根本不需要知道如何处理错误！操作符也不需要处理错误状态-一旦发生错误，就会跳过当前和后续的操作符。所有的错误处理都交给订阅者来做。

调度器

假设你编写的Android app需要从网络请求数据（感觉这是必备的了，还有单机么？）。网络请求需要花费较长的时间，因此你打算在另外一个线程中加载数据。那么问题来了！

编写多线程的Android应用程序是很难的，因为你必须确保代码在正确的线程中运行，否则的话可能会导致app崩溃。最常见的就是在非主线程更新UI。

使用RxJava，你可以使用subscribeOn()指定观察者代码运行的线程，使用observeOn()指定订阅者运行的线程：

```
1 myObservableServices.retrieveImage(url)
2   .subscribeOn(Schedulers.io())
3   .observeOn(AndroidSchedulers.mainThread())
4   .subscribe(bitmap -> myImageView.setImageBitmap(bitmap));
```

是不是很简单？任何在我的Subscriber前面执行的代码都是在I/O线程中运行。最后，操作view的代码在主线程中运行。

最棒的是我可以把subscribeOn()和observeOn()添加到任何Observable对象上。这两个也是操作符！。我不需要关心Observable对象以及它上面有哪些操作符。仅仅运用这两个操作符就可以实现在不同的线程中调度。

如果使用AsyncTask或者其他类似的，我将不得不仔细设计我的代码，找出需要并发执行的部分。使用RxJava，我可以保持代码不变，仅仅在需要并发的时候调用这两个操作符就可以。

订阅 (Subscriptions)

当调用Observable.subscribe()，会返回一个Subscription对象。这个对象代表了被观察者和订阅者之间的联系。

```
1 subscription = Observable.just("Hello, World!")
2   .subscribe(s -> System.out.println(s));
```

你可以在后面使用这个Subscription对象来操作被观察者和订阅者之间的联系。

```
1 subscription.unsubscribe();
2 System.out.println("Unsubscribed=" + subscription.isUnsubscribed());
3 // Outputs "Unsubscribed=true"
```

RxJava的另外一个好处就是它处理unsubscribing的时候，会停止整个调用链。如果你使用了一串很复杂的操作符，调用unsubscribe将会在他当前执行的地方终止。不需要做任何额外的工作！

总结

记住这个系列仅仅是对RxJava的一个入门介绍。RxJava中有更多的我没介绍的功能等你探索（比如backpressure）。当然我也不是所有的代码都使用响应式的方式—仅仅当代码复杂到我想将它分解成简单的逻辑的时候，我才使用响应式代码。

最初，我的计划是这篇文章作为这个系列的总结，但是我收到许多请求我介绍在Android中使用RxJava，所以你可以继续阅读第四篇了。我希望这个介绍能让你开始使用RxJava。如果你想学到更多，我建议你阅读RxJava的官方wiki。

顶 践

8 0

上一篇 [深入浅出RxJava\(二 : 操作符\)](#)

下一篇 [深入浅出RxJava四-在Android中使用响应式编程](#)

我的同类文章

RxJava基本流程和lift源码分析
walfud: 感谢 bruce!看过你的文章深受启发,我也写了一系列源码分析,请多指教: <http://dio...>

深入浅出RxJava(二: 操作符)
wuxiaoming1992: Subscriber实现了Observer, 多出了几个方法, onstart之类的

深入浅出RxJava (一: 基础篇)
林深: 赞, 学习了!

Gradle Tips#1-tasks
zhaojianand: 不错, 可以基础学习

RxJava (6)

- RxJava使用场景小结 2015-11-30 阅读 6625 • RxJava基本流程和lift源码分析 2015-11-30 阅读 3841
- 如何升级到RxAndroid 1.0 2015-10-19 阅读 3096 • 深入浅出RxJava四-在Andro... 2015-04-13 阅读 22315
- 深入浅出RxJava(二: 操作符) 2015-03-06 阅读 18474 • 深入浅出RxJava (一: 基础... 2014-12-09 阅读 64614

[主题推荐](#) [函数](#) [响应式](#) [对象](#) [数据](#) [pre](#) [class](#)

猜你在找

[Android开源项目实践之UI篇](#)

[深入浅出RxJava四-在Android中使用响应式编程](#)

[威哥全套Android开发课程【基础与UI技术】](#)

[深入浅出RxJava四响应式安卓开发](#)

[Web APP设计与实现](#)

[使用响应式编程RxJava开发Android App](#)

[高级UI设计师养成记](#)

[26Bootstrap学习之工具class响应式工具](#)

[移动端游戏UI设计-二部曲](#)

[函数响应式编程--资料收集](#)

[查看评论](#)

12楼 [dingshuhong_](#) 2016-01-18 10:55发表



你好,感谢的你的分享,我想问题几个问题, Observable到底应该怎样的理解, 还有就是PublicSubject.onNext() 跟 subscriber.onNext 的区别是什么. 非常感谢

11楼 [android无聊大神](#) 2016-01-18 10:55发表



引用“hr8951”的评论:

如何在Observable.just("Hello, world!")

...

我也有同样的需求,

比如, 网络请求返回 dataModel 对象为 {code , message , data}

我需要在 map 中转换 dataModel 对象为 List<data> 对象

如果 code == 1

则返回正常的Observable<List<data>>给 onNext

否则 code != 1

则将 String 类型的 message 传给 onError , toast 给用户看

请问这如何实现呢?

Re: [android无聊大神](#) 2016-01-18 12:02发表



回复android无聊大神: 看了下 <https://github.com/ribot/ribot-app-android> 的源码是可以 throw 一个自定义的 RuntimeException 来搞定

10楼 [w7421](#) 2016-01-14 11:18发表



subscribeOn和ObservableOn没有理解

9楼 [追云似梦](#) 2015-12-31 15:43发表



骚年, 弱弱的问一下, 在RxJava中, 如果在Schedulers.io()中执行多个网络请求任务, 那么如何取消所有的网络请求任务呢?

8楼 [追云似梦](#) 2015-12-31 15:42发表



骚年, 弱弱的问一下, 在RxJava中, 如果在Schedulers.io()中执行多个网络请求任务, 那么如何取消所有的网络请求任务呢?

7楼 [newuserxx](#) 2015-12-25 09:05发表



赞赞赞

6楼 [hr8951](#) 2015-11-25 11:09发表



如何在Observable.just("Hello, world!")
.map(s -> potentialException(s))
.map(s -> anotherPotentialException(s)), 比如anotherPotentialException中主动抛出异常, 让Subscriber的onError来接管异常处理。

5楼 [androidstackoverflow](#) 2015-10-06 14:57发表

两处, 错别字。网络请求需要话费较长的时间, 因此你打算在另外一个线程中加载数据。为问题来了! “话费”->“花费”, “为问



题来了”→“那么问题来了”

Re: hi大头鬼hi 2015-10-08 20:09发表

回复androidstackoverflow: 已修改, 多谢

4楼 在下雨了 2015-09-09 16:49发表



现在android中很多网络请求的框架，都有很好的封装了线程等待时的子线程，包括子线程的回收和重用，在获取到数据后，设置了回调提供在主线程中使用。包括谷歌自己出的volley大概也是这样，如果使用rxjava来实现数据请求，在观察者中请求设置子线程，然后获取到以后通知订阅者在主线程中实现界面的改变，也就是说其中的网络部分自己去实现，而不能在使用比较好的网络框架了？是这样吗？楼主！

3楼 Justlove_DK 2015-08-20 10:15发表



楼主，那个subscription.unsubscribe()方法加不加后面的输出好像都是true

2楼 Mr_wangyong 2015-08-18 11:29发表

**[java]**

```
01. .subscribeOn(Schedulers.io())
02. .observeOn(AndroidSchedulers.mainThread())
```

是否有误

subscribeOn代表观察者所在的线程，子线程

observeOn代表事件源所在的线程，主线程

最后subscribe更新UI，子线程更新UI？？

还请大神详解！！

1楼 Mr_wangyong 2015-08-18 11:29发表

**[java]**

```
01. .subscribeOn(Schedulers.io())
02. .observeOn(AndroidSchedulers.mainThread())
```

是否有误

subscribeOn代表观察者所在的线程，子线程

observeOn代表事件源所在的线程，主线程

最后subscribe更新UI，子线程更新UI？？

还请大神详解！！

Re: androidstackoverflow 2015-10-06 15:02发表

回复Mr_wangyong: 你自己弄反了。.subscribeOn(Schedulers.io()) // 指定 subscribe() 发生在 IO 线程
.observeOn(AndroidSchedulers.mainThread()) // 指定 Subscriber 的回调发生在主线程

Re: hi大头鬼hi 2015-08-19 17:44发表



回复Mr_wangyong: 反了

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题	Hadoop	AWS	移动游戏	Java	Android	iOS	Swift	智能硬件	Docker	OpenStack		
VPN	Spark	ERP	IE10	Eclipse	CRM	JavaScript	数据库	Ubuntu	NFC	WAP	jQuery	
BI	HTML5	Spring	Apache	.NET	API	HTML	SDK	IIS	Fedora	XML	LBS	Unity
Splashtop	UML	components	Windows Mobile	Rails	QEMU	KDE	Cassandra	CloudStack				
FTC	coremail	OPhone	CouchBase	云计算	iOS6	Rackspace	Web App	SpringSide	Maemo			
Compuware	大数据	aptech	Perl	Tornado	Ruby	Hibernate	ThinkPHP	HBase	Pure	Solr		
Angular	Cloud Foundry	Redis	Scala	Django	Bootstrap							



