

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

深入浅出RxJava四-在An

深入浅出RxJava(二: 操

深入浅出RxJava三--响应

Android实现类似QQ的消

Android热更新实现原理

RxJava使用场景小结

深入浅出Android Gradle

Otto使用入门

RxJava基本流程和lift源

评论排行

深入浅出RxJava (一: 基

深入浅出RxJava(二: 操

深入浅出RxJava四-在An

深入浅出RxJava三--响应

RxJava基本流程和lift源

RxJava使用场景小结

使用动画和fragment改善

Android实现类似QQ的消

Android热更新实现原理

深入浅出Android Gradle

个人资料

深入浅出RxJava四-在Android中使用响应式编程

标签: android RxJava RxAndroid 响应式编程

2015-04-13 22:41 22657人阅读 评论(18) 收藏 举报

分类: RxJava (6)

目录(?) [+]

原文链接

在第1, 2, 3篇中, 我大概介绍了RxJava是怎么使用的。下面我会介绍如何在Android中使用RxJava.

RxAndroid

RxAndroid是RxJava的一个针对Android平台的扩展。它包含了一些能够简化Android开发的工具。

首先, AndroidSchedulers提供了针对Android的线程系统的调度器。需要在UI线程中运行某些代码?很简单, 只需要使用AndroidSchedulers.mainThread():

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

如果你已经创建了自己的Handler, 你可以使用HandlerThreadScheduler1将一个调度器链接到你的handler上。

接着要介绍的就是AndroidObservable, 它提供了跟多的功能来配合Android的生命周期。bindActivity()和bindFragment()方法默认使用AndroidSchedulers.mainThread()来执行观察者代码, 这两个方法会在Activity或者Fragment结束的时候通知被观察者停止发出新的消息。

```
1 AndroidObservable.bindActivity(this, retrofitService.getImage(url))
2   .subscribeOn(Schedulers.io())
3   .subscribe(bitmap -> myImageView.setImageBitmap(bitmap));
```

我自己也很喜欢AndroidObservable.fromBroadcast()方法, 它允许你创建一个类似BroadcastReceiver的Observable对象。下面的例子展示了如何在网络变化的时候被通知到:

```
1 IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
2 AndroidObservable.fromBroadcast(context, filter)
3   .subscribe(intent -> handleConnectivityChange(intent));
```

最后要介绍的是ViewObservable,使用它可以给View添加了一些绑定。如果你想在每次点击view的时候都收到一个事件, 可以使用ViewObservable.clicks(), 或者你想监听TextView的内容变化, 可以使用ViewObservable.text()。

```
1 ViewObservable.clicks(mCardNameEditText, false)
2   .subscribe(view -> handleClick(view));
```

Retrofit

大名鼎鼎的Retrofit库内置了对RxJava的支持。通常调用发可以通过使用一个Callback对象来获取异步的结果:

```
1 @GET("/user/{id}/photo")
```



hi大头鬼 hi



访问: 285483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: 感谢扔物线大神和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lift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...

```
2 | void getUserPhoto(@Path("id") int id, Callback<Photo> cb);
```

使用RxJava，你可以直接返回一个Observable对象。

```
1 | @GET("/user/{id}/photo")  
2 | Observable<Photo> getUserPhoto(@Path("id") int id);
```

现在你可以随意使用Observable对象了。你不仅可以获取数据，还可以进行变换。

Retrofit对Observable的支持使得它可以很简单的将多个REST请求结合起来。比如我们有一个请求是获取照片的，还有一个请求是获取元数据的，我们就可以将这两个请求并发的发出，并且等待两个结果都返回之后再做处理：

```
1 | Observable.zip(  
2 |     service.getUserPhoto(id),  
3 |     service.getPhotoMetadata(id),  
4 |     (photo, metadata) -> createPhotoWithData(photo, metadata))  
5 | .subscribe(photoWithData -> showPhoto(photoWithData));
```

在第二篇里我展示过一个类似的例子（使用flatMap()）。这里我只是想展示以下使用RxJava+Retrofit可以多么简单地组合多个REST请求。

### 遗留代码，运行极慢的代码

Retrofit可以返回Observable对象，但是如果你使用的别的库并不支持这样怎么办？或者说一个内部的内码，你想把他们转换成Observable的？有什么简单的办法没？

绝大多数时候Observable.just() 和 Observable.from() 能够帮助你从遗留代码中创建 Observable 对象：

```
1 | private Object oldMethod() { ... }  
2 |  
3 | public Observable<Object> newMethod() {  
4 |     return Observable.just(oldMethod());  
5 | }
```

上面的例子中如果oldMethod()足够快是没有什么问题的，但是如果很慢呢？调用oldMethod()将会阻塞住他所在的线程。

为了解决这个问题，可以参考我一直使用的方法—使用defer()来包装缓慢的代码：

```
1 | private Object slowBlockingMethod() { ... }  
2 |  
3 | public Observable<Object> newMethod() {  
4 |     return Observable.defer(() -> Observable.just(slowBlockingMethod()));  
5 | }
```

现在，newMethod()的调用不会阻塞了，除非你订阅返回的observable对象。

### 生命周期

我把最难的不分留在了最后。如何处理Activity的生命周期？主要就是两个问题：

1.在configuration改变（比如转屏）之后继续之前的Subscription。

比如你使用Retrofit发出了一个REST请求，接着想在listview中展示结果。如果在网络请求的时候用户旋转了屏幕怎么办？你当然想继续刚才的请求，但是怎么搞？

2.Observable持有Context导致的内存泄露

这个问题是因为创建subscription的时候，以某种方式持有了context的引用，尤其是当你和view交互的时候，这太容易发生！如果Observable没有及时结束，内存占用就会越来越大。

不幸的是，没有银弹来解决这两个问题，但是这里有一些指导方案你可以参考。

第一个问题的解决方案就是使用RxJava内置的缓存机制，这样你就可以对同一个Observable对象执行unsubscribe/resubscribe，却不用重复运行得到Observable的代码。cache() (或者 replay())会继续执行网络请求（甚至你调用了unsubscribe也不会停止）。这就是说你可以在Activity重新创建的时候从cache()的返回值中创建一个新的Observable对象。

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

```
1 Observable<Photo> request = service.getUserPhoto(id).cache();
2 Subscription sub = request.subscribe(photo -> handleUserPhoto(photo));
3
4 // ...When the Activity is being recreated...
5 sub.unsubscribe();
6
7 // ...Once the Activity is recreated...
8 request.subscribe(photo -> handleUserPhoto(photo));
```

注意，两次sub是使用的同一个缓存的请求。当然在哪里去存储请求的结果还是要你自己来做，和所有其他的生命周期相关的解决方案一延虎，必须在生命周期外的某个地方存储。（retained fragment或者单例等等）。

第二个问题的解决方案就是在生命周期的某个时刻取消订阅。一个很常见的模式就是使用CompositeSubscription来持有所有的Subscriptions，然后在onDestroy()或者onDestroyView()里取消所有的订阅。

```
1 private CompositeSubscription mCompositeSubscription
2     = new CompositeSubscription();
3
4 private void doSomething() {
5     mCompositeSubscription.add(
6         AndroidObservable.bindActivity(this, Observable.just("Hello, World!"))
7         .subscribe(s -> System.out.println(s)));
8 }
9
10 @Override
11 protected void onDestroy() {
12     super.onDestroy();
13
14     mCompositeSubscription.unsubscribe();
15 }
```

你可以在Activity/Fragment的基类里创建一个CompositeSubscription对象，在子类中使用它。

注意! 一旦你调用了 CompositeSubscription.unsubscribe()，这个CompositeSubscription对象就不可用了, 如果你还想使用CompositeSubscription，就必须在创建一个新的对象了。

两个问题的解决方案都需要添加额外的代码，如果谁有更好的方案，欢迎告诉我。

总结

RxJava还是一个很新的项目，RxAndroid更是。RxAndroid目前还在活跃开发中，也没有多少好的例子。我打赌一年之后我的一些建议就会被看做过时了。

顶

17

踩

2

上一篇

深入浅出RxJava三--响应式的好处

下一篇

Gradle Tips#1-tasks

我的同类文章

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三--响应式...	2015-04-05	阅读 14525
• 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

android

url

ui

响应式

android平台

android开发

线程

猜你在找

威哥最新Android开发课程【核心组件】  
Android底层技术：Linux驱动框架与开发  
Android开源项目实践之UI产  
Android开发—在线支付  
Android开发—广告植入

深入浅出RxJava三--响应式的好处  
深入浅出RxJava三--响应式的好处  
深入浅出RxJava三--响应式的好处  
深入浅出RxJava三响应式的好处  
[使用ReactiveCocoa实现iOS平台响应式编程](#)

查看评论

10楼 野生ChaoS 2015-12-21 11:15发表



看了抛物线的文章再来看大头哥的，感觉就像是复习。不过为什么要用 Lambda 表达式。。。蛋疼啊。。。

Re: Joe\_n 2016-01-11 17:53发表



回复野生ChaoS: lambda 确实简化了代码，可读性不好

9楼 Zane964 2015-12-07 22:37发表



楼上的都不会用github！？

8楼 修 2015-09-18 15:16发表



感觉好像是eventbus

7楼 nitaoge 2015-07-22 20:33发表



请问一下，RxAndroid在哪里下载

Re: Justlove\_DK 2015-08-20 10:25发表



回复nitaoge: 目前使用的还是rxjava 吧，你去谷歌一下就可以了，但是我还没弄清楚rxjava和rxAndroid到底的区别是神马（使用的时候）

6楼 zlc2200 2015-07-22 14:26发表



新版本已经没有AndroidObservable这个类了，用AppObservable和ContentObservable 等替代了

Re: 麟麟麟麟麟麟麟麟麟麟麟麟麟麟麟麟 2015-12-03 17:29发表



回复zlc2200: 可是我找不到 AppObservable和ContentObservable 啊!!?!?

Re: 红烧茄子 2016-01-19 14:44发表



回复麟麟麟麟麟麟麟麟麟麟麟麟麟麟麟麟: AndroidObservable，ViewObservable，AppObservable都找不到，只要ContentObservable能找到，但是确实android包下的。。。不是rx包下的

5楼 xtaowang 2015-07-15 10:10发表



怎么在android 使用？

Re: Justlove\_DK 2015-08-20 10:21发表



回复xtaowang: 下载jar包就可以了

4楼 vicp7764 2015-05-13 11:33发表



终于等到你把这一系列给翻译完了,在你翻译之前,我自己蹩脚的英语按照自己的理解也翻译了一遍,我做后端开发的,不太了解Android的东西。国内的rx系列的东西太少了,学习都好伤，从面向对象到理解函数式编程也是一种挑战，我是新手，希望大神你能继续为国内的rx系列做出贡献。

3楼 安卓 2015-04-28 11:26发表



简直太帅了 谢谢楼主

2楼 瘦纸好过夏 2015-04-20 10:41发表




RxJava wiki 上各种操作符 的api 怎么使用可以翻译吗，大神。

Re: hi大头鬼hi 2015-04-23 15:57发表




回复瘦纸好过夏: 这个比较费时间，是有这个打算，但是估计要很久


Re: [hi大头鬼hi](#) 2015-04-23 15:56发表

 回复瘦纸好过夏：这个比较费时间，是有这个打算，但是估计要很久

1楼 [SchemIng13](#) 2015-04-18 23:08发表

 RxJava和eventbus有什么区别么？？？

Re: [弹一曲Happy颂](#) 2015-11-18 14:03发表

 回复SchemIng13：问的好啊，你这么问，说明了一个问题：作者并没有阐述清楚真正的响应式思想，其实响应式编程重要的是事件流的传输和变换，而文中讲到的这些工具api只是为实现“响应式思想”而服务的。因此只有真正理解思想，才能够用好它。那推荐你一篇文章，学习下思想：<http://www.infoq.com/cn/articles/functional-reactive-programming#10006-weixin-1-52626-6b3bfd01fdde4900130bc5a2751b6d1>

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](#)

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服

杂志客服

微博客服

webmaster@csdn.net

400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

