

C 博客

🔍

★

🔔

✎

☰

THEONE10211024的专栏

📁 目录视图

📄 摘要视图

RSS 订阅

个人资料

THEONE10211024

+ 加关注

📧 发私信

🏆

访问：198414次

积分：2530

等级：BLOG<5

排名：第10435名

原创：69篇 转载：69篇

译文：6篇 评论：75条

文章搜索

文章分类

C/C++ (7)

操作系统 (4)

笔试面试 (17)

设计模式 (10)

数据结构与算法 (10)

零星知识 (13)

网络 (4)

等你来挑战 (1)

项目总结 (2)

android (71)

android开源项目 (21)

值得一读 (6)

数学之美 (1)

LeetCode经典解法 (2)

Python (2)

机器学习 (10)

文章存档

2016年08月 (9)

2016年07月 (3)

2016年06月 (2)

2016年05月 (4)

2016年04月 (1)

📄 展开

📄 可能是东半球最全的RxJava使用场景小结

标签：RxJava RxAndroid 响应式编程

2015-12-30 16:31 17892人阅读 评论(7) 收藏 举报

分类： android (70) android开源项目 (20)

快速回复

版权声明：本文为博主原创文章，未经博主允许不得转载。

一、Scheduler线程切换

这种场景经常会在“后台线程取数据，主线程展示”的模式中看见

```
[html]
01. Observable.just(1, 2, 3, 4)
02.     .subscribeOn(Schedulers.io()) // 指定 subscribe() 发生在 IO 线程
03.     .observeOn(AndroidSchedulers.mainThread()) // 指定 Subscriber 的回调发生在主线程
04.     .subscribe(new Action1<Integer>() {
05.         @Override
06.         public void call(Integer number) {
07.             Log.d(tag, "number:" + number);
08.         }
09.     });
```

二、使用debounce做textSearch

用简单的话讲就是当N个结点发生的时间太靠近（即发生的时间差小于设定的值T），debounce就会自动过滤掉前N-1个结点。

比如在做百度地址联想的时候，可以使用debounce减少频繁的网络请求。避免每输入（删除）一个字就做一次联想

```
[html]
01. RxTextView.textChangeEvents(inputEditText)
02.     .debounce(400, TimeUnit.MILLISECONDS)
03.     .observeOn(AndroidSchedulers.mainThread())
04.     .subscribe(new Observer<TextViewTextChangeEvent>() {
05.         @Override
06.         public void onCompleted() {
07.             log.d("onComplete");
08.         }
09.
10.         @Override
11.         public void onError(Throwable e) {
12.             log.d("Error");
13.         }
14.
15.         @Override
16.         public void onNext(TextViewTextChangeEvent onTextChangeEvent) {
17.             log.d(format("Searching for %s", onTextChangeEvent.text().toString()));
18.         }
19.     });
```

三、Retrofit结合RxJava做网络请求框架

阅读排行

- Google推荐的图片加载库 (32775)
- 可能是东半球最全的RxJava (17852)
- GetLastError()返回值列 (15290)
- fork出的子进程和父进程 (14247)
- Picasso: 一个专为Android (4657)
- Android App 线上热修复 (4360)
- 360 2013校园招聘笔试 (4263)
- 安卓调试神器-Stetho (Fa (4232)
- 58同城 2013研发一面面 (3997)
- android中finish()和onDe (3879)

评论排行

- Google推荐的图片加载库 (11)
- Freeline - Android平台 (9)
- 听说"双11"是这么解决线 (8)
- *运项目难点之ScrollView (8)
- 可能是东半球最全的RxJava (7)
- 360 2013校园招聘笔试 (5)
- GetLastError()返回值列 (4)
- 在Activity中使用Thread (3)
- Android后台进程保活之 (3)
- 奇虎360 2014后台研发二 (2)

推荐文章

- * Chromium扩展 (Extension) 机制简要介绍和学习计划
- * Android官方开发文档Training 系列课程中文版: APP的内存管理
- * 程序员, 别了校园入了江湖
- * RxJava 合并组合两个 (或多个) Observable数据源
- * 探索Android软键盘的疑难杂症

最新评论

- *运项目难点之ScrollView中嵌套qq_34066768: 当我看到 '我就反编译了百度地图的jar包时' 我震惊了 真滴屌
- Freeline - Android平台上的秒级: 失落夏天: 您好, 现在遇到一问题. 由于公司网络限制, 所以打开center之后无法正常的更新freeline模块...
- Google推荐的图片加载库Glide: 超级宇宙无敌冬瓜: Glide怎么清除缓存
- 基于sklearn 的one hot encoding gaodahongCSDN: 应用独热编码, 我个人还有一点理解, 不知对错, 请指教: 比如SVM在求解的时候, 中间用到了拉格朗日公式求...
- Freeline - Android平台上的秒级: THEONE10211024: @u012390044:需要什么样的教程? 或者有什么不明白的可以直接留言或者去github上: http...
- Freeline - Android平台上的秒级: 失落夏天: 赞一个, 最近研究LayoutCast, 本想继续去维护起LayoutCast的, 就顺着找到这了。

这里不作详解, 具体的介绍可以看扔物线的[这篇文章](#), 对RxJava的入门者有很大的启发。其中也讲到了RxJava和Retrofit如何结合来实现更简洁的代码

四、RxJava代替EventBus 进行数据传递: RxBus

注意: RxBus并不是一个库, 而是一种模式, 是使用了RxJava的思想来达到EventBus的数据传递效果。[这篇文章](#)把RxBus讲的比较详细。

五、使用combineLatest 合并最近N个结点

例如: 注册的时候所有输入信息 (邮箱、密码、电话号码等) 合法才点亮注册按钮。

```
[html]
01. Observable<CharSequence> _emailChangeObservable = RxTextView.textChanges(_email).skip(1);
02. Observable<CharSequence> _passwordChangeObservable = RxTextView.textChanges(_password).skip(1);
03. Observable<CharSequence> _numberChangeObservable = RxTextView.textChanges(_number).skip(1);
04.
05. Observable.combineLatest(_emailChangeObservable,
06.     _passwordChangeObservable,
07.     _numberChangeObservable,
08.     new Func3<CharSequence, CharSequence, CharSequence, Boolean>() {
09.         @Override
10.         public Boolean call(CharSequence newEmail,
11.             CharSequence newPassword,
12.             CharSequence newNumber) {
13.
14.             Log.d("xiayong",newEmail+" "+newPassword+" "+newNumber);
15.             boolean emailValid = !isEmpty(newEmail) &&
16.                 EMAIL_ADDRESS.matcher(newEmail).matches();
17.             if (!emailValid) {
18.                 _email.setError("Invalid Email!");
19.             }
20.
21.             boolean passValid = !isEmpty(newPassword) && newPassword.length() > 8;
22.             if (!passValid) {
23.                 _password.setError("Invalid Password!");
24.             }
25.
26.             boolean numValid = !isEmpty(newNumber);
27.             if (numValid) {
28.                 int num = Integer.parseInt(newNumber.toString());
29.                 numValid = num > 0 && num <= 100;
30.             }
31.             if (!numValid) {
32.                 _number.setError("Invalid Number!");
33.             }
34.
35.             return emailValid && passValid && numValid;
36.
37.         }
38.     })//
39.     .subscribe(new Observer<Boolean>() {
40.         @Override
41.         public void onCompleted() {
42.             log.d("completed");
43.         }
44.
45.         @Override
46.         public void onError(Throwable e) {
47.             log.d("Error");
48.         }
49.
50.         @Override
51.         public void onNext(Boolean formValid) {
52.             _btnValidIndicator.setEnabled(formValid);
53.         }
54.     });
```

Freeline - Android平台上的秒级:
CZ_DSY: 博主能发个使用教程吗?

Freeline - Android平台上的秒级:
THEONE10211024: @yishifu:什么问题?

Freeline - Android平台上的秒级:
L 托尼: 配置成功了么? 能请教一下吗

Freeline - Android平台上的秒级:
THEONE10211024:
@mengyy110:目前已经开源到github上了

六、使用merge合并两个数据源。

例如一组数据来自网络，一组数据来自文件，需要合并两组数据一起展示。

[html] [icon] [icon]

```
01. Observable.merge(getDataFromFile(), getDataFromNet())
02.             .observeOn(AndroidSchedulers.mainThread())
03.             .subscribe(new Subscriber<String>() {
04.                 @Override
05.                 public void onCompleted() {
06.                     log.d("done loading all data");
07.                 }
08.
09.                 @Override
10.                 public void onError(Throwable e) {
11.                     log.d("error");
12.                 }
13.
14.                 @Override
15.                 public void onNext(String data) {
16.                     log.d("all merged data will pass here one by one!");
17.                 }
17.             });
```

快速回复

七、使用concat和first做缓存

依次检查memory、disk和network中是否存在数据，任何一步一旦发现数据后面的操作都不执行。

[html] [icon] [icon]

```
01. Observable<String> memory = Observable.create(new Observable.OnSubscribe<String>() {
02.     @Override
03.     public void call(Subscriber<? super String> subscriber) {
04.         if (memoryCache != null) {
05.             subscriber.onNext(memoryCache);
06.         } else {
07.             subscriber.onCompleted();
08.         }
09.     }
10. });
11. Observable<String> disk = Observable.create(new Observable.OnSubscribe<String>() {
12.     @Override
13.     public void call(Subscriber<? super String> subscriber) {
14.         String cachePref = rxPreferences.getString("cache").get();
15.         if (!TextUtils.isEmpty(cachePref)) {
16.             subscriber.onNext(cachePref);
17.         } else {
18.             subscriber.onCompleted();
19.         }
20.     }
21. });
22.
23. Observable<String> network = Observable.just("network");
24.
25. //依次检查memory、disk、network
26. Observable.concat(memory, disk, network)
27.     .first()
28.     .subscribeOn(Schedulers.newThread())
29.     .subscribe(s -> {
30.         memoryCache = "memory";
31.         System.out.println("-----subscribe: " + s);
32.     });
```

八、使用timer做定时操作。当有“x秒后执行y操作”类似的需求的时候，想到使用timer

例如：2秒后输出日志“hello world”，然后结束。

[html] [icon] [icon]

```
01. Observable.timer(2, TimeUnit.SECONDS)
02.             .subscribe(new Observer<Long>() {
03.                 @Override
```

```
04.         public void onCompleted() {
05.             log.d ("completed");
06.         }
07.
08.         @Override
09.         public void onError(Throwable e) {
10.             log.e("error");
11.         }
12.
13.         @Override
14.         public void onNext(Long number) {
15.             log.d ("hello world");
16.         }
17.     });
```

九、使用**interval**做周期性操作。当有“每隔xx秒后执行yy操作”类似的需求的时候，想到使用**interval**

例如：每隔2秒输出日志“helloworld”。

 快速回复



```
[html]
01. Observable.interval(2, TimeUnit.SECONDS)
02.     .subscribe(new Observer<Long>() {
03.         @Override
04.         public void onCompleted() {
05.             log.d ("completed");
06.         }
07.
08.         @Override
09.         public void onError(Throwable e) {
10.             log.e("error");
11.         }
12.
13.         @Override
14.         public void onNext(Long number) {
15.             log.d ("hello world");
16.         }
17.     });
```

十、使用**throttleFirst**防止按钮重复点击

ps: **debounce**也能达到同样的效果


```
[html]
01. RxView.clicks(button)
02.     .throttleFirst(1, TimeUnit.SECONDS)
03.     .subscribe(new Observer<Object>() {
04.         @Override
05.         public void onCompleted() {
06.             log.d ("completed");
07.         }
08.
09.         @Override
10.         public void onError(Throwable e) {
11.             log.e("error");
12.         }
13.
14.         @Override
15.         public void onNext(Object o) {
16.             log.d("button clicked");
17.         }
18.     });
```

十一、使用**schedulePeriodically**做轮询请求



```
[html]    
01. Observable.create(new Observable.OnSubscribe<String>() {  
02.     @Override  
03.     public void call(final Subscriber<? super String> observer) {  
04.  
05.         Schedulers.newThread().createWorker()  
06.             .schedulePeriodically(new Action0() {  
07.                 @Override  
08.                 public void call() {  
09.                     observer.onNext(doNetworkCallAndGetStringResult());  
10.                 }  
11.             }, INITIAL_DELAY, POLLING_INTERVAL, TimeUnit.MILLISECONDS);  
12.         }  
13.     }).subscribe(new Action1<String>() {  
14.         @Override  
15.         public void call(String s) {  
16.             log.d("polling...");  
17.         }  
18.     })
```

 快速回复

十二、RxJava进行数组、list的遍历



```
[html]    
01. String[] names = {"Tom", "Lily", "Alisa", "Sheldon", "Bill"};  
02. Observable  
03.     .from(names)  
04.     .subscribe(new Action1<String>() {  
05.         @Override  
06.         public void call(String name) {  
07.             log.d(name);  
08.         }  
09.     });
```

十三、解决嵌套回调（callback hell）问题

```
[html]    
01. NetworkService.getToken("username", "password")  
02.     .flatMap(s -> NetworkService.getMessage(s))  
03.     .subscribe(s -> {  
04.         System.out.println("message: " + s);  
05.     })
```

十四、响应式的界面

比如勾选了某个checkbox，自动更新对应的preference

```
[html]    
01. SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);  
02. RxSharedPreferences rxPreferences = RxSharedPreferences.create(preferences);  
03.  
04. Preference<Boolean> checked = rxPreferences.getBoolean("checked", true);  
05.  
06. CheckBox checkBox = (CheckBox) findViewById(R.id.cb_test);  
07. RxCompoundButton.checkedChanges(checkBox)  
08.     .subscribe(checked.asAction());
```

最后，由于个人能力有限，文章难免有疏漏之处，如果您有任何疑问，请让我知道，谢谢！本文所有的例子已经上传到[github](#)上

致谢：这篇文章的绝大多数例子是从[这里](#)总结的，还有部分例子来自[这里](#)。对作者的无私贡献表示感谢！



顶

15

踩

0

我的同类文章



android（70）		android开源项目（20）	
• 巧用接口解耦分离实现	2016-08-18 阅读 65	• Freeline - Android平台上的...	2016-08-04 阅读 777
• Android 上使用 iconfont 的...	2016-06-30 阅读 2410	• 章鱼店长Android端启动优...	2016-06-08 阅读 1122
• 为android的apk动态写入信...	2016-05-31 阅读 462	• Android后台进程保活之黑...	2016-05-31 阅读 698
• 那些年我们踩过AsyncTask...	2016-05-30 阅读 302	• jvm参数设置 -vmargs -Xms...	2016-04-07 阅读 730
• Android面试回忆录	2016-03-31 阅读 434	• Android 开发绕不过的坑：...	2016-03-04 阅读 1654
• 常见的Android优化方案	2016-03-03 阅读 240		

更多文章

猜你在找

- 嵌入式Linux网络编程
 - 拥抱开源： Github使用指南
 - 网络基础-linux应用编程和网络编程第8部分
 - linux网络编程实践-linux应用编程和网络编程第9部分
 - MySQL数据库管理
- RxJava使用场景小结
 - RxJava多种常用场景使用方法
 - RxJava使用场景搜集
 - RxJava zip操作符在Android中的实际使用场景
 - RxJava使用场景小结

查看评论

6楼 万力_ 2016-06-13 12:55发表

C

感谢分享

5楼 Dawish_大D 2016-06-04 18:46发表

很全面很好，谢谢大神分享

4楼 [游客] 2016-05-11 17:26发表

C

不，我认为还是有区别的。想象一下暴力测试拿着按钮疯狂的点然后告诉你怎么点击都没有反应（手动滑稽

3楼 鲍协浩 2016-01-20 11:58发表

C

十、使用throttleFirst防止按钮重复点击

ps： debounce也能达到同样的效果


这个有误，debounce和throttleFirst不同，debounce在连续点击的时候会只取最后一个，而throttleFirst会去第一个。

Re: THEONE10211024 2016-01-20 12:33发表

回复鲍协浩：是这样子的，但是一般来说在按钮点击的时候取第一个点击和最后一个点击并没有差别。这里只是多提供了一种思路~

2楼 singsong 2015-12-31 13:42发表


 很全面,支持.

1楼 qq137712630 2015-12-31 12:44发表 

 谢谢分享

发表评论

用 户 名: avalonx7

评论内容: 

 快速回复






提交

* 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 © 1999-2016, CSDN.NET, All Rights Reserved 