

手记 \ 我们为什么要在Android中使用RxJava

我们为什么要在Android中使用RxJava

2016-01-14 12:02:22 2777浏览 8评论

本文翻译来自-->[Why should we use RxJava on Android](#)

感觉RxJava最近风生水起，不学习一下都不好意思了，洒家也是初学RxJava，也是感觉代码好像更复杂更难懂了，本文简单介绍使用RxJava优势所在。但可能需要有一点RxJava基础，推荐先看一下抛物线的那篇经典的匠心写作。

-----华丽分割线，译文开始-----



Reactive Extensions (Rx) 是一系列接口和方法，为开发者提供了一种易懂且迅速简单易维护的方法。RxJava就是干这事儿的，提供一系列tools来帮你写出简洁的代码。

老实说，一开始我认为RxJava 写的代码理解起来很困难，并且引入一个库，单单就是为了用用这种新式的api，这困扰到了我。后来，我懂了。以传统的编码方式，随着app的发展，我需要重构代码、一遍一遍的重复样板代码，以满足用户不断变更的新需求，这让我苦不堪言。

我做的大量工作，其实是改写相关方法和接口，就是因为需求的变更（这是开发与产品间那些血案的原罪）或者需要改变展示的信息亦或是需要改变处理信息数据..这很抓狂。另外，这种代码让其他来维护的人来理解，通常是很耗时的。

举个栗子:我们需要从数据库获取一组用户的链表数据，并展示出来。我们可以用AsyncTask后台查询数据库，获得的结果给Ui的适配器展示出来。简单示例代码：

```
public SampleTask(SampleAdapter sampleAdapter) {
    mAdapterter = sampleAdapater;
}

@Override
protected List<Users> doInBackground(Void... voids) {
    //fetch there results from the database and return them to the onPostExecute
    List<Users> users = getUsersFromDatabase();
    return users;
}

@Override
protected void onPostExecute(List<Users> users) {
    super.onPostExecute(products);
    // Checking if there are users on the database
    if(users == null) {
        //No users, presenting a view saying there are no users
        showEmptyUsersMessageView();
        return;
    }
    for(User user : users){
        mAdapterter.add(user);
    }
    mAdapterter.notifyDataSetChanged();
}
}
```

现在有个新需求，要求只显示非guest的用户，我们处理的方法是，在添加到adapter前加个条件判断是不是guset，或者改变数据库查询的条件。更有甚者，你又被要求从数据库中获取另外的其他信息，跟user一并在这个adapter中显示出来呢？

这就是我们为什么要用RxJava了，把我们从这个泥潭中拉出来。换个姿势，我们Rx代码是这样子（假设您已学习过Rx基础用法）：

```

public Observable<List<User>> fetchUsersFromDatabase() {
    return Observable.create(new Observable.OnSubscribe<List<User>>(){
        @Override
        public void call(Subscriber<? super List<User>> subscriber){
            // Fetch information from database
            subscriber.onNext(getUserList());
            subscriber.onCompleted();
        }
    });
}

```

像这样被调用：

```

fetchUsersFromDatabase()
    .subscribeOn(Schedulers.io())
    //will process everything in a new thread
    .observeOn(AndroidSchedulers.mainThread())
    //will listen the results on the main thread
    .subscribe(new Subscriber<List<User>>() {
        @Override
        public void onCompleted() {
        }
        @Override
        public void onError(Throwable e) {
        }
        @Override
        public void onNext(List<User> users) {
            //Do whatever you want with each user
        }
    });

```

开始改需求了哈

怎么不显示guests呢，RxJava分分钟过滤掉这种不速之客：

```

fetchUsersFromDatabase()
    .filter(new Func1<User, Boolean>() {
        @Override
        public Boolean call(User user) {
            //only return the users which are not guests
            return !user.isGuest();
        }
    })
    .subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .subscribe(new Subscriber<User>() {
        @Override
        public void onCompleted() {
        }

        @Override
        public void onError(Throwable e) {
            /*Check if there was any error while retrieving from database*/
        }

        @Override
        public void onNext(User user) {
            //Do whatever you want with each user
        }
    })
    );

```

传统的方式，即便是个简单的变更，为了保持优雅接口化编程，我们得创建新接口，重构代码来实现过滤。但是使用RxJava让这一切变得优雅了，我们只需要一个被观察者用来获取所有的信息，让你就可以尽情的用这些方法来过滤获取你想要的数据。

可能你又会说了，ok，这是很好很易读的结构，但是这似乎使代码量变多了呢。well you are right，但是这就

是Retrolambda闪耀的时候了，这个库为我们兼容了以使用java8 lambda表达式，方法引用等等。

帮我们简化代码如下：

```

fetchUsersFromDatabase()
    .subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .subscribe(value -> {

```

```
        //Do whatever with the value
    },error -> {
        //do something with in case of error
    }
};
```

这个问题完美搞定，然后你又开始问了，我需要增加另外的查询结果和user一同显示在这个adapter中怎么破。这真不是事儿：

```
fetchUsersFromDatabase()
    .zipWith(fetchSomethingElseFromDatabase(), (users, somethingElse) -> {
        /*here combine users and something else into a new object*/
    })
    .subscribe( o -> {
        /*use the combine object from users and something else to fill the adapter */
    });
```

如上，我们可以轻松组合数据库查出来的其他数据和users给一个adapter一同显示。是不是更易维护，代码少，易读，清晰？

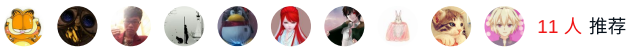
如果要更深入的学习RXJava可以看下面这篇文章，我看后受益匪浅。

Party tricks with RxJava, RxAndroid & Retrolambda

另外，这篇教程 [tutorial](#) 也帮我在RxJava路上进阶了很多。

本文为慕课网作者原创，转载请标明【原文作者及本文链接地址】。侵权必究，谢谢合作!

相关标签： JAVA Android



相关阅读

- 深入理解Java内部类
- Java高级特性之枚举
- 每个人都应该懂一点设计原则
- Android Activity状态保存
- 程序员，请向加班说No!!!



请登录，发表评论



全部评论 8条



微笑江湖

棒棒哒

8F

2016-01-16 21:15:05

赞同 0

回复



星风雪雨

正在学习RxJava和RxAndroid

7F

2016-01-15 18:17:16

赞同 0

回复




EricCui 回复 星风雪雨：

2016-04-08 17:43:55

你学的怎么样啊

回复



存折

收藏

6F

2016-01-14 21:39:02

赞同 0

回复



i爱慕客

点赞小能手顺便给你点个赞

5F

2016-01-14 16:37:34

赞同 0

回复



天才小喵

不错

4F

2016-01-14 16:21:48

赞同 0

回复



成都_小蚂蚁

最近经常听说，过来看看，楼主叨炸天

3F

2016-01-14 15:59:57

赞同 0

回复



微凉一季 回复 成都_小蚂蚁：

2016-01-14 17:08:05

666.

回复



椰子奶

2F

最近正好在研究,感谢楼主无私贡献,hello帅呆了

2016-01-14 15:59:29

赞同 0

回复



微凉一季 回复 椰子奶:

2016-01-14 17:08:30

校长的学生吗，学的真好

回复



椰子奶 回复 微凉一季:

2016-01-14 17:13:32

你猜~

回复



pkxutao

1F

正好有用到，感谢楼主的无私贡献，hello 吊炸天

2016-01-14 15:44:39

赞同 0

回复



微凉一季 回复 pkxutao:

2016-01-14 15:52:51

校长，财源滚滚，赢取白美富，出任CTO

回复