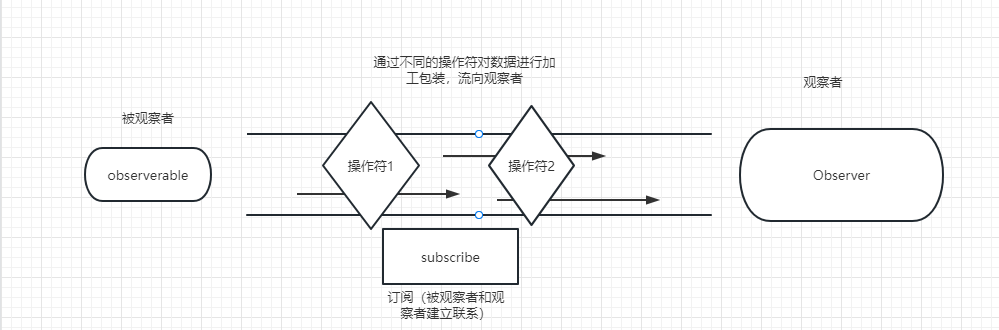
浅谈RXJAVA

一.Rxjava介绍

RxJava是近几年网上非常流行的一种响应式编程框架，核心思想是观察者模式，然后通过自己提供的一系列强大的操作符，对事件进行加工包装，并且可以轻松实现线程调度的一个框架。

二．基本概念

RxJava的核心思想是观察者模式，使用各种操作符对被观察者发送的事件进行包装，最终流向观察者。如下图



Observable ：被观察者，用来生产发送事件；

Observer：观察者，接收被观察者传来的事件；

Subscribe：被观察者和观察者通过订阅产生关系后，才具备事件发送和接收能力；

三．基本用法

如下图:



步骤：

1. 创建被观察者observable ,简单提一嘴ObservableEmitter，顾名思义，这个是发射器，被观察者和观察者建立联系后，可以通过这个发射器发射数据，观察者就能接收到数据。
2. 创建观察者 observer。其中onSubscribe,onNext,onError,onComplete是必要的实现方法
   1. onSubscribe:它会最早调用，用来做一些准备操作，比如弹出的对话框之类，Disposable参数则是用来切断上下游的关系的，调用dispose()方法后，观察者将不会再接收数据
   2. onNext:普通事件
   3. onError:异常事件, 在事件处理过程中出现异常情况时，此方法会被调用。同时队列将会终止，也就是不允许在有事件发出。
   4. onComplete：完成事件。当不再有onNext事件发出时，需要触发onComplete方法作为完成标识。
3. 通过subscribe 订阅方法，被观察者和观察者建立联系

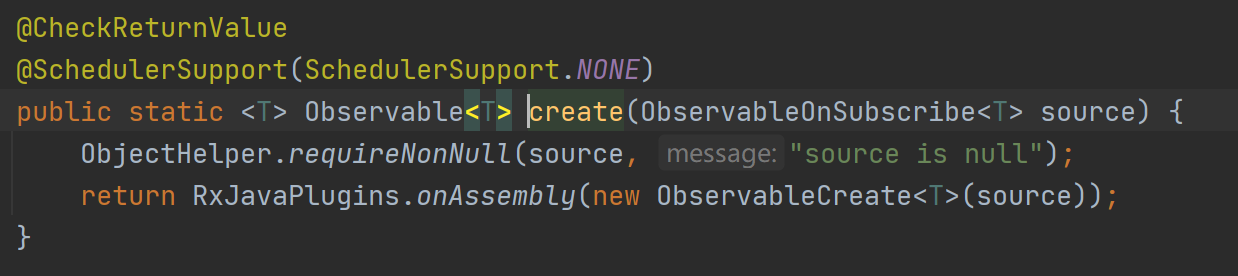
运行结果如下:

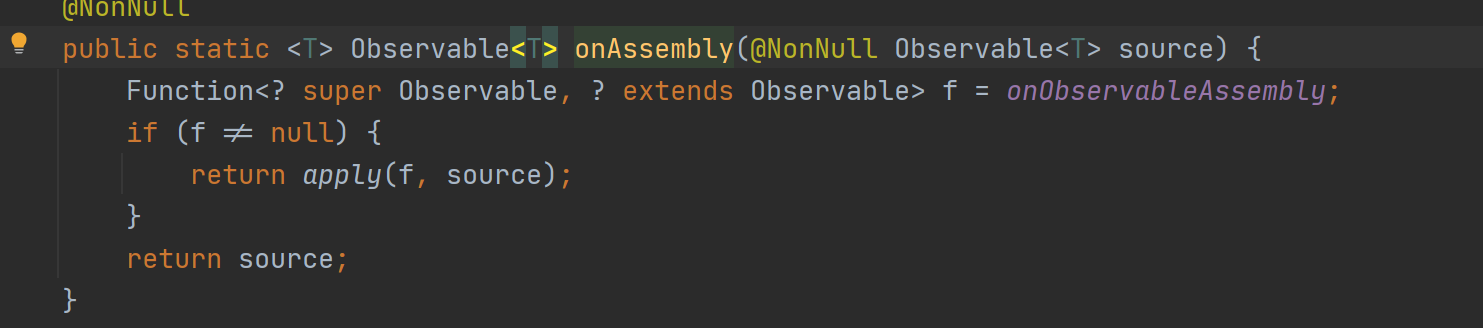


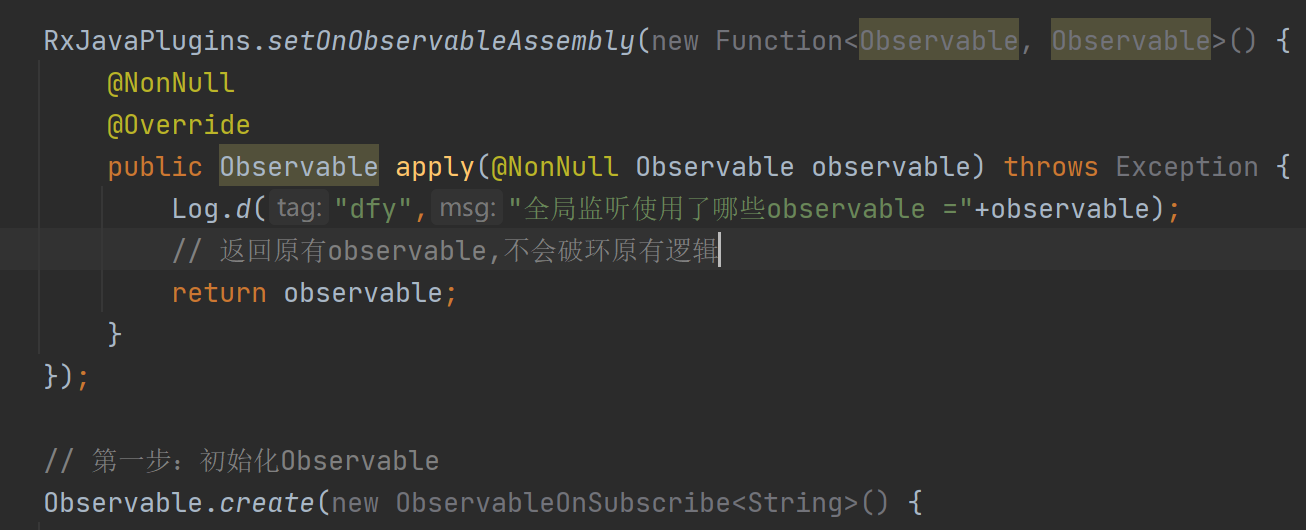
需要注意的是，在一个正确运行的事件序列中, onCompleted() 和 onError() 有且只有一个，并且是事件序列中的最后一个。onCompleted() 和 onError() 二者也是互斥的，即在队列中调用了其中一个，就不应该再调用另一个。

【冷知识点】

Observable.create方法中有个RxJavaPlugins.onAssembly,在我们调用setOnObservableAssembly,手动给onObservableAssembly赋值后，就能全局监听你用了哪些Observable



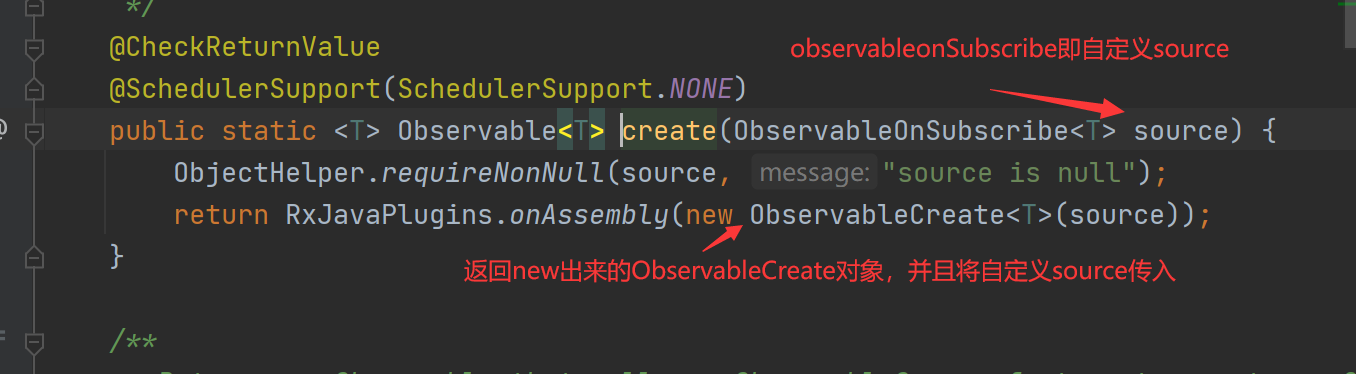




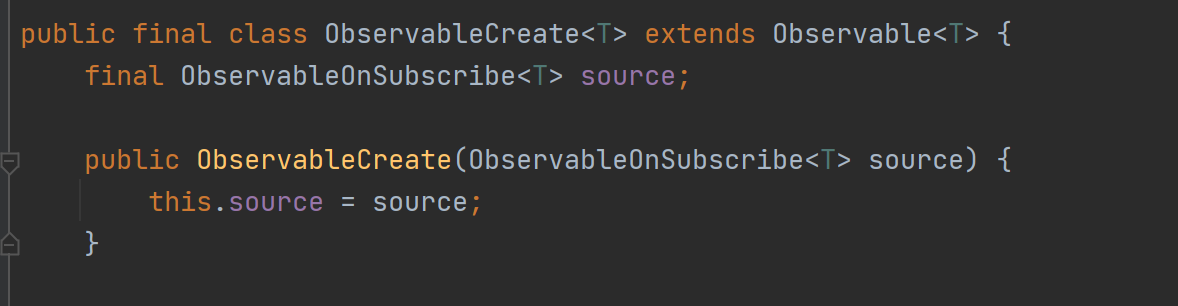
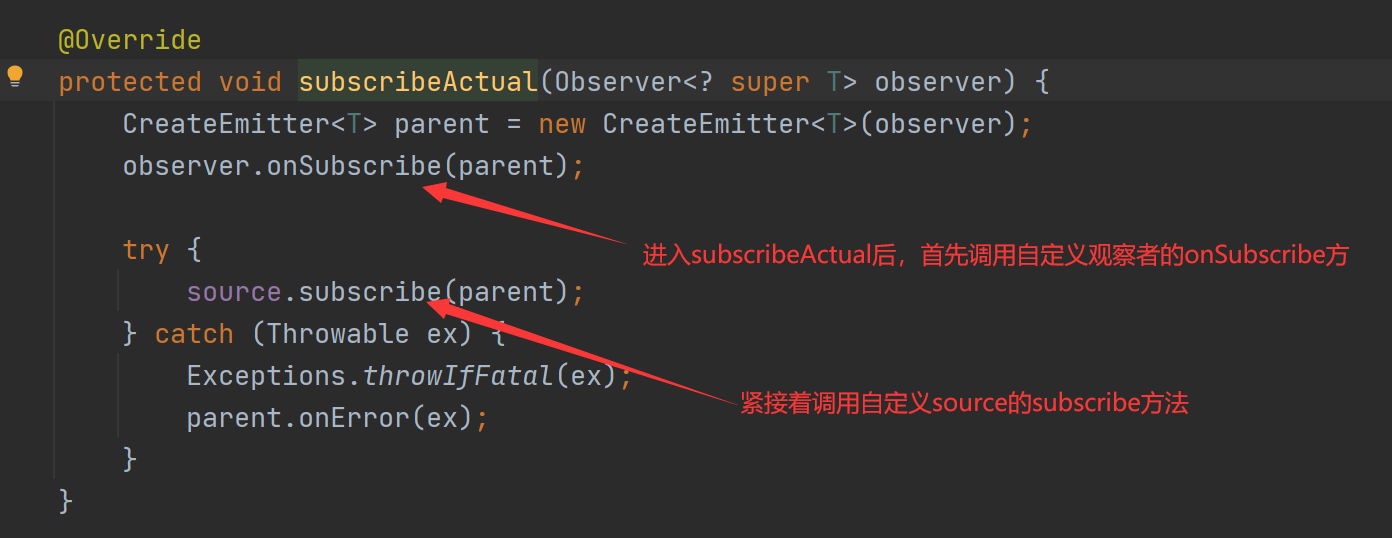
四．基本原理解析

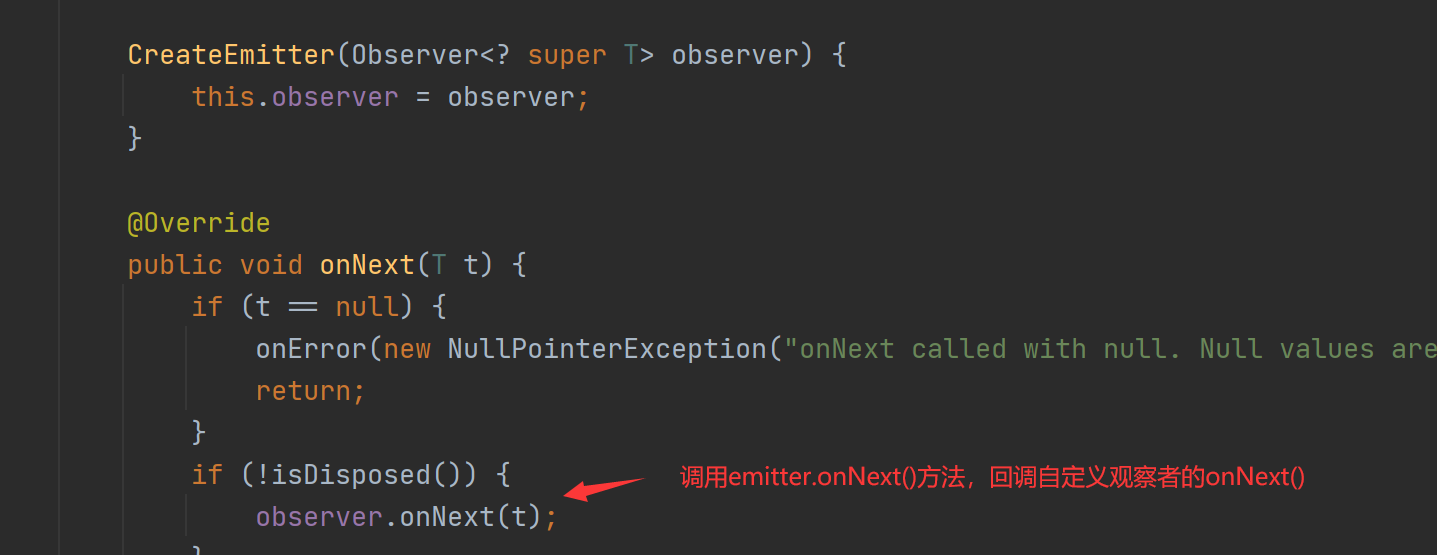
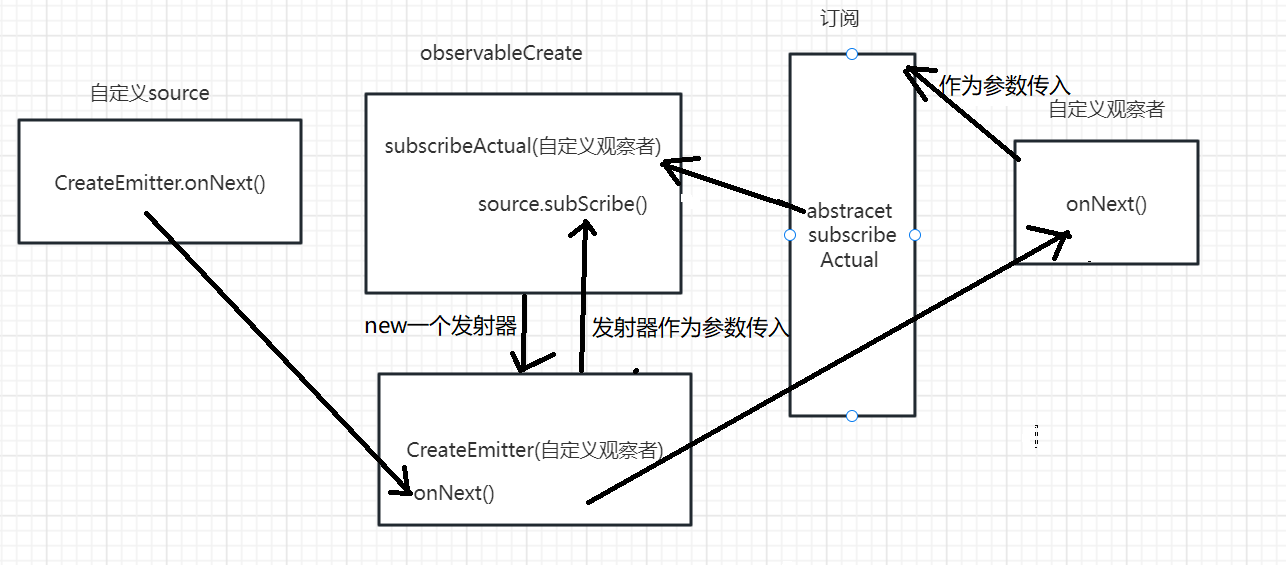
* Observable.create创建了一个ObservableCreate对象，并且将自定义source传入ObservableCreate中

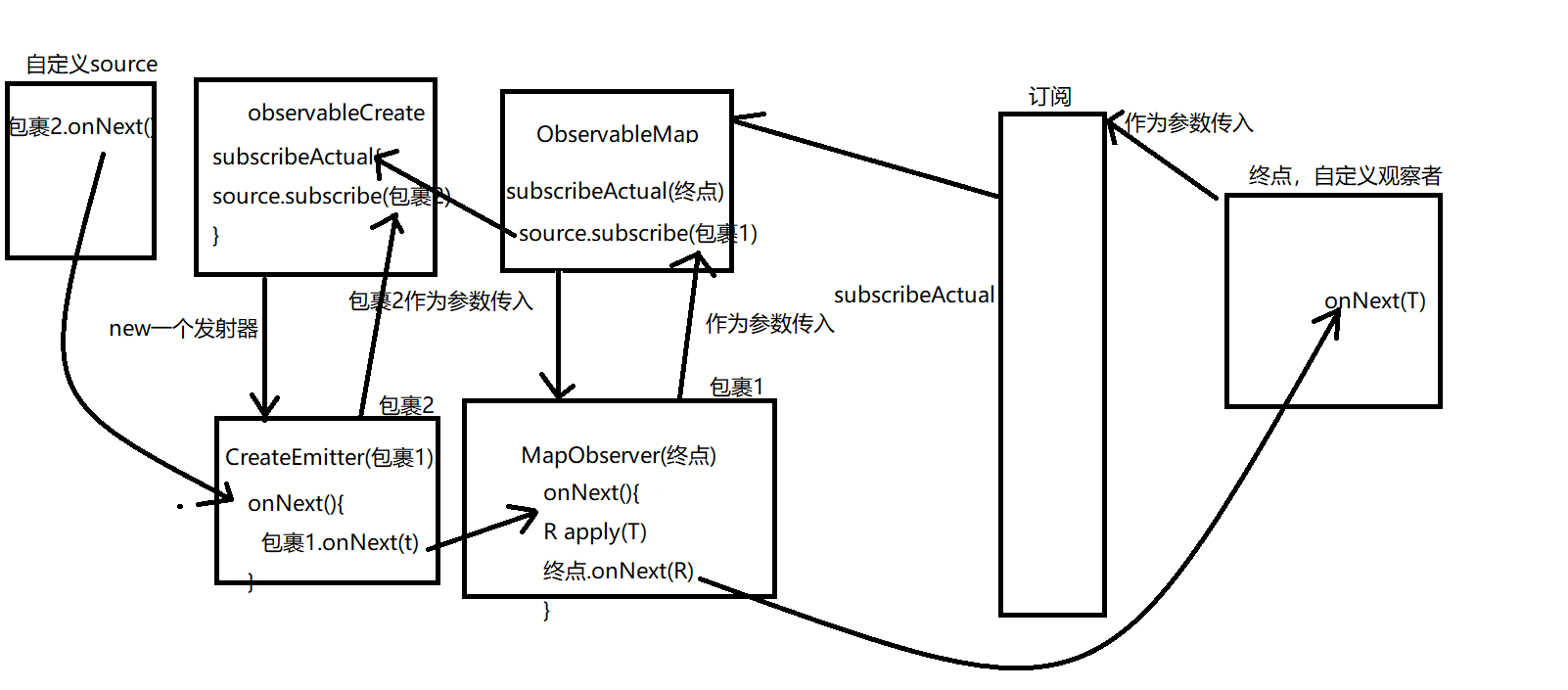




* 调用订阅方法subscribe(Observer)：即ObservableCreate.subscribe，接着调用subscribeActual方法，传入自定义观察者observer 

* 进入source.subscribe方法后，手动调用emitter.onNext方法，会让observer触发onNext方法
* 总结: 
* map操作符原理解析
  + 作用: map 可以将被观察者发送的数据类型转变成其他的类型

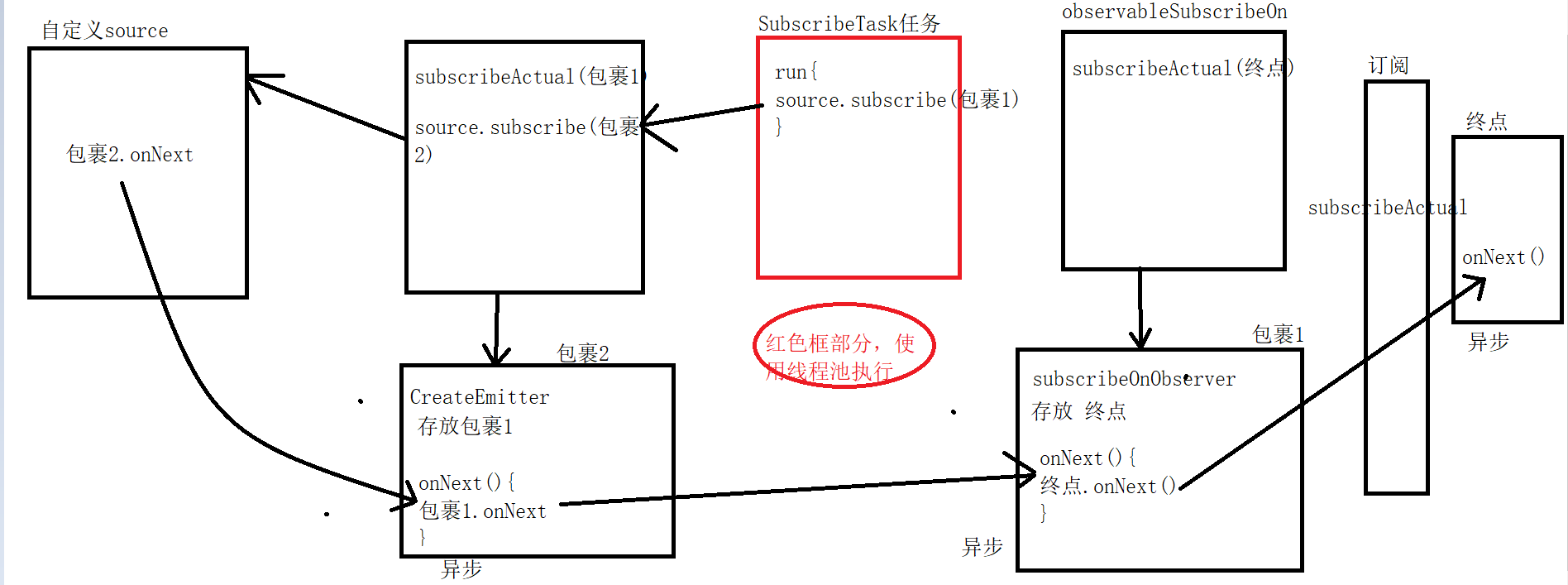
加了一个map操作符，在上面的流程上增加了一层observableMap，在这层对observer封装一层，将这个封装层再传给上层的observableCreate对象，再执行了emitter.onNext后，一层层的拆包装，并最终在map里面的onNext()方法中进行输入输出的转换，并最终传给observer的onNext



* subscribeOn([Schedulers.io](http://schedulers.io/)())原理：
  + 作用: 给上面的代码分配IO线程
  + 如果连续调用多个subscribeOn，以离上面最近的代码的分配线程为主，例如下图，此时emitter.onNext执行在离他最近的subscribeOn(Schedukers.io)线程中

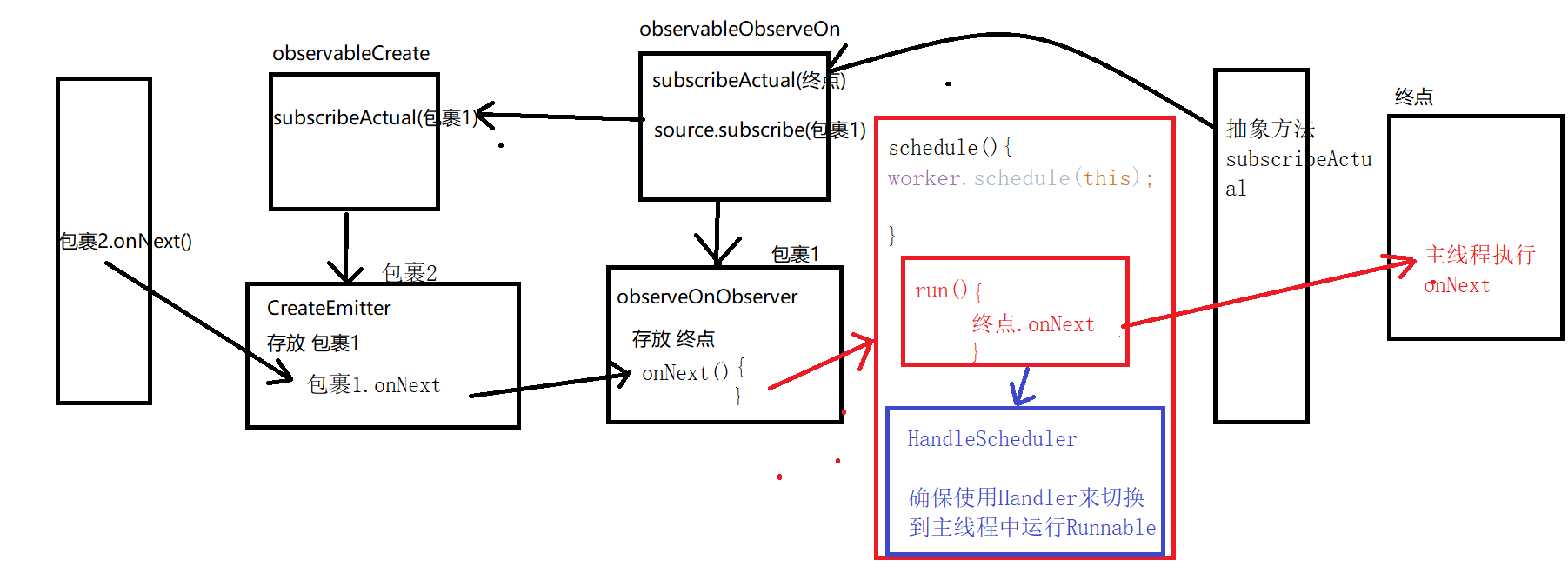


* + 如下图，整体流程和之前分析的并无太多差异，只是在调用source.subscribe的时候是放在一个异步任务中执行，导致后续的调用均在异步线程中



* observeOn(AndroidSchedulers.mainThread())原理：
  + 作用: 给下面的代码分配主线程
  + 如果连续调用多个，离下面最近的代码分配的线程为主，例如下图，Observer的onNext回调运行在IO线程中



* + 如下图，整体流程和之前分析的并无太多差异，只是在拆包的时候，调用onNext的时候是放在一个HandlerScheduler里面执行，里面用的是handler机制，保证最后的observer.onNext运行在主线程

 五．总结

本文简单介绍了下什么是RxJava以及RxJava中的create/map/subscribeon/observeon操作符的原理。RxJava还有其他的非常强大的操作符flatMap,buffer,zip等等，翻看他们的内部源码，会发现Rxjava是一种类似洋葱的模型设计，先是一层一层的包装，然后再一层一层的剥开，最后希望本文可以帮助各位读者更好的理解使用RxJava