

RxJava2 vs RxJava1



作者 今天是个大晴天 (/u/8284989f4f51) + 关注

2016.09.07 10:56* 字数 1741 阅读 13244 评论 16 喜欢 45

(/u/8284989f4f51)

官方WIKI What's different in 2.0 (<https://github.com/ReactiveX/RxJava/wiki/What's-different-in-2.0>)

RxJava2已经发布了两周了，相比RxJava1，它的改动还是很大的：

Observable and Flowable

在前一个版本里 `backpressure` 被集成到了 `Observable` 中，官方也提供了很多方法让我们来处理 `backpressure` 问题。但是有一些特殊的场景根本无法用其来解决，最常见的例如 UI 事件。而不处理 `backpressure` 有可能导致 `MissingBackpressureException` 的出现。

关于 `backpressure` 的概念可以看一下RxJava中backpressure这个概念的理解 (<http://www.dundunwen.com/article/275b1d92-f9da-4bb8-b111-3aa8a6ace245.html>)

为了解决这个问题，在RxJava2里，引入了Flowable这个类：`Observable` 不包含 `backpressure` 处理，而Flowable包含。

例如：

```
Flowable<Long> flowable =
    Flowable.create((FlowableOnSubscribe<Long>) e -> {
        Observable.interval(10, TimeUnit.MILLISECONDS)
            .take(Integer.MAX_VALUE)
            .subscribe(e::onNext);
    }, FlowableEmitter.BackpressureMode.DROP);

Observable<Long> observable =
    Observable.create((ObservableOnSubscribe<Long>) e -> {
        Observable.interval(10, TimeUnit.MILLISECONDS)
            .take(Integer.MAX_VALUE)
            .subscribe(e::onNext);
    });
```

两个对象都以10毫秒一次派发数据，假设订阅他们的方法都是：

```
i -> {
    Thread.sleep(100);
    Log.v("TEST", "out : " + i);
}
```

以100毫秒一次消费数据，消费数据的效率是生产的1/10。那么

- 对于observable

他会按照 0,1,2,3,4... 的顺序依次消费，并输出log，而没有消费的数据将会都存在内存中。如果在RxJava1中，内存数据超过128个时将会抛出

`MissingBackpressureException` 错误；而在RxJava2中并不会报错，数据会一直放到内存中，直到发生 `OutOfMemoryError`。



- 对于flowable, 在创建时我们设定了 `FlowableEmitter.BackpressureMode.DROP` , 一开始他会输出 `0,1,2,3...127` 但之后会忽然跳跃到 `966,967,968 ...` 。中间的部分数据由于缓存不了, 被抛弃掉了。

Single

和Observable, Flowable一样会发送数据, 不同的是订阅后只能接受到一次:

```
Single<Long> single = Single.just(11);

single.subscribe(new SingleObserver<Long>() {
    @Override
    public void onSubscribe(Disposable d) {
    }

    @Override
    public void onSuccess(Long value) {
        // 和onNext是一样的
    }

    @Override
    public void onError(Throwable e) {
    }
});
```

普通Observable可以使用toSingle转换: `Observable.just(1).toSingle()`

Completable

与Single类似, 只能接受到完成(`onComplete`)和错误(`onError`)

同样也可以由普通的Observable转换而来: `Observable.just(1).toCompletable()`

可订阅的对象在RxJava1中只有Observable一种, 之前我们经常会直接把数据源称作Observable。而在RxJava2中扩充成了4种, 因此在之后还是把他们统称为数据源为宜

Base reactive interfaces

和Flowable的接口Publisher类似, Observable、Single、Completable也有类似的基类

```
interface ObservableSource<T> {
    void subscribe(Observer<? super T> observer);
}

interface SingleSource<T> {
    void subscribe(SingleObserver<? super T> observer);
}

interface CompletableSource {
    void subscribe(CompletableObserver observer);
}
```

因此许多操作符接受的参数从以前的具体对象, 变成了现在的接口:



```
Flowable<R> flatMap(
    Function<? super T, ? extends Publisher<? extends R>> mapper
);

Observable<R> flatMap(
    Function<? super T, ? extends ObservableSource<? extends R>> mapper
);

-----
// 以前
Observable<R> flatMap(Func1<? super T, ? extends Observable<? extends R>> func) {
```

由于接收的都是接口，在使用其他遵循 Reactive-Streams 设计的第三方库的时候，就不需要把他自定义的 Flowable 转换成标准Flowable了。

Subjects and Processors

```
io.reactivex.subjects.AsyncSubject ,
io.reactivex.subjects.BehaviorSubject ,
io.reactivex.subjects.PublishSubject ,
io.reactivex.subjects.ReplaySubject ,
io.reactivex.subjects.UnicastSubject
```

在RxJava2中依然存在，但现在他们不支持 backpressure 。新出现的

```
io.reactivex.processors.AsyncProcessor ,
io.reactivex.processors.BehaviorProcessor ,
io.reactivex.processors.PublishProcessor ,
io.reactivex.processors.ReplayProcessor
io.reactivex.processors.UnicastProcessor
```

支持 backpressure

Other classes

```
rx.observables.ConnectableObservable 变成了
io.reactivex.observables.ConnectableObservable<T> 和
io.reactivex.flowables.ConnectableFlowable<T>
```

类似的还有 rx.observables.GroupedObservable 。

Functional interfaces

需要注意的一点是，现在RxJava2的接口方法里加上了 throws Exception：

```
ublic interface Consumer<T> {
    void accept(T t) throws Exception;
}
```

意味着在这些方法里调用一些会发生异常的方法不需要 try-catch 了

Actions

另外大部分接口方法都按照Java8的接口方法名进行了相应的修改，比如上面那个 Consumer<T> 接口原来叫 Action1<T> ，而 Action2<T> 改名成了 BiConsumer

Action3 - Action9 被删掉了，大概因为没人用。。



Functions

同上，基本就是名字的修改和不常用类的删除

Subscriber

RxJava1里 Subscriber 只是一个空接口，在新版里 Subscriber 被赋予了更多的作用，有几个实现类可以供我们使用，例如

```
ResourceSubscriber<Integer> subscriber = new ResourceSubscriber<Integer>() {  
    @Override  
    public void onStart() {  
        request(Long.MAX_VALUE);  
    }  
  
    @Override  
    public void onNext(Integer t) {  
        System.out.println(t);  
    }  
  
    @Override  
    public void onError(Throwable t) {  
        t.printStackTrace();  
    }  
  
    @Override  
    public void onComplete() {  
        System.out.println("Done");  
    }  
};
```

request() 方法可以控制当前subscriber需要接收几个事件。而且，还可以调用 subscriber.dispose() 来断开对信号的监听。

同时， onCompleted 方法改成了 onComplete。意味着完成时调用这个方法，而不是完成后 d

由于 Subscription 被改掉了(下面会讲到)。如果需要类似以前CompositeSubscription的用法，可以使用:

```
CompositeDisposable composite2 = new CompositeDisposable();  
composite2.add(Flowable.range(1, 5).subscribeWith(subscriber));
```

注意这里需要使用subscribeWith而不是subscribe，因为subscribe方法现在返回void

Subscription

在RxJava1里，Subscription起到的是订阅桥梁的作用。在2中，由于Subscription本身和 Reactive-Streams 里的另外一个同名概念冲突。因此把原本的Subscription改名成了 Disposable。

除了上一节里 subscribe(Subscriber) 方法返回 void，其他名为 subscribe 的方法都返回 Disposable

相应的，

- CompositeSubscription 改名成了 CompositeDisposable
- SerialSubscription 和 MultipleAssignmentSubscription 被合并到了 SerialDisposable 里。set() 方法会处理掉就的值，而replace()方法不会。
- RefCountSubscription 被移除了



Backpressure

在第一节Observable and Flowable里已经说到了这个问题，在2中，Observable将不会处理backpressure，也就不会发生 `MissingBackpressureException` 问题，但内存仍然会缓存多余的数据。

而在使用Flowable时，如果配置Backpressure有问题，那么 `MissingBackpressureException` 依然存在

Schedulers

RxJava2里仍然包含了 `computation` , `io` , `newThread` 和 `trampoline` 这些默认线程调度。而 `immediate` 被移除了，因为他经常被人错误使用。同时 `Schedulers.test` 也被移除了。

Entering the reactive world

将普通方法转换成RxJava的数据源，在RxJava1中，提供了 `Observable.create()` 方法，但是这个方法过于强大，但使用时需要注意的东西太多经常会发生错误。

因此在RxJava2中，把原来的 `fromAsync` 重命名成了 `create` ， `fromAsync` 是一个和 `create` 类似但更为简单和安全的方法。这样大部分旧代码都能够继续使用。

Leaving the reactive world

之前如果想把数据源转换成普通的数据对象，需要先转换成 `BlockingObservable` 。而在2中，可以调用 `blockingXXX` 方法直接把数据源转换成对象：

```
List<Integer> list = Flowable.range(1, 100).toList().blockingFirst();
```

有一点需要特别注意，在RxJava2里，不建议在 `Subscriber` 里抛出错误，这意味着下面的代码可能有一天就不能继续运行了：

```
Subscriber<Integer> subscriber = new Subscriber<Integer>() {
    @Override
    public void onSubscribe(Subscription s) {
        s.request(Long.MAX_VALUE);
    }

    public void onNext(Integer t) {
        if (t == 1) {
            throw new IllegalArgumentException();
        }
    }

    public void onError(Throwable e) {
        if (e instanceof IllegalArgumentException) {
            throw new UnsupportedOperationException();
        }
    }

    public void onComplete() {
        throw new NoSuchElementException();
    }
};

Flowable.just(1).subscribe(subscriber);
```

由于上面类似的代码实际中出现得很多，因此在2中提供了 `safeSubscribe` 方法，使用它就可以继续在 `subscriber` 里抛出错误。

当然，你可以绕过 `subscribe(subscriber)` 这个方法，使用类似：



```
Flowable.just(1).subscribe(subscriber::onNext, subscriber::onError, subscriber::onComplete)
```

这样的方法，之前的代码仍然可以继续throw错误。

Operator differences

操作符的改动不大，大部分是扩充了参数数量。
或者是加入 prefetch 代表可以加入预置数据。

总结

可以明显的看到，RxJava2最大的改动就是对于 backpressure 的处理，为此将原来的 Observable 拆分成了新的 Observable 和 Flowable，同时其他相关部分也同时进行了拆分。

除此之外，他和我们最熟悉和喜爱的RxJava~

引用

- 官方WIKI What's different in 2.0 (<https://github.com/ReactiveX/RxJava/wiki/What's-different-in-2.0>)
- RxJava中backpressure这个概念的理解 (<http://www.dundunwen.com/article/275b1d92-f9da-4bb8-b111-3aa8a6ace245.html>)

📖 日记本 (/nb/3469416) 举报文章 © 著作权归作者所有



今天是个大晴天 (/u/8284989f4f51)
写了 5101 字，被 41 人关注，获得了 72 个喜欢 (/u/8284989f4f51)

+ 关注

今天是个大晴天

如果觉得我的文章对您有用，请随意打赏。您的支持将鼓励我继续创作！

赞赏支持

♡ 喜欢 (/sign_in) | 45




更多分享

(<http://cwb.assets.jianshu.io/notes/images/566694f>)



登录 (/sign_in) 后发表评论

16条评论 只看作者 按喜欢排序 按时间正序 按时间倒序



alighters (/u/6321040dd140)
2楼 · 2016.09.08 10:55
(/u/6321040dd140)



赞赞👍。。。

👍 赞 💬 回复


今天是个大晴天 (/u/8284989f4f51): @alighters (/users/6321040dd140) 谢谢支持 😊

2016.09.08 16:02 💬 回复

XMLUbanu (/u/addc2c87899f): @今天是个大晴天 (/users/8284989f4f51) 一直用Rxjava1 , 看了楼主Rxjava2 ,也想学一下, 求加个QQ好友, 751372658


2017.01.22 11:40 💬 回复

✍️ 添加新评论

慕容昭言 (/u/f1f44e3dad0c)
3楼 · 2016.09.08 19:05
(/u/f1f44e3dad0c)


👍

👍 赞 💬 回复

wotaylor (/u/5213233293e7)
4楼 · 2016.09.09 10:19
(/u/5213233293e7)


🙏

👍 赞 💬 回复

7ad395dd7dbe (/u/7ad395dd7dbe)
5楼 · 2016.12.05 18:42
(/u/7ad395dd7dbe)

👍👍👏👏

👍 赞 💬 回复

夏淑影 (/u/b250640dbeeb)
6楼 · 2016.12.22 09:10
(/u/b250640dbeeb)


嗯.....最后一句话是不是想说: 除此之外, 他**还是**我们最熟悉和喜爱的RxJava~

👍 赞 💬 回复

夏淑影 (/u/b250640dbeeb): @夏淑影 (/users/b250640dbeeb) 还有说一句 RxJava中 backpressure这个概念的理解 这篇文章翻译的太烂了,建议直接看文章开头提供的英文原版

2016.12.22 10:31 💬 回复

✍️ 添加新评论

r17171709 (/u/470afcb80dc2)
7楼 · 2017.02.07 09:07
(/u/470afcb80dc2)


请教一个问题, RxJava2与RxJava共存会不会有问题?

👍 赞 💬 回复

hayukleung (/u/2471a8f0e7a5): 必须有问题, 至少有些命名冲突了

2017.04.02 15:52 💬 回复

✍️ 添加新评论

王元_Trump (/u/358688e13ed6)
8楼 · 2017.02.21 16:12
(/u/358688e13ed6)

数据源 在RxJava中有 Observable Flowable Single Completable。那Maybe这个类呢



👍 赞 💬 回复

今天是个大晴天 (/u/8284989f4f51): Maybe是2新出的类哦

2017.02.27 14:19 💬 回复

王元_Trump (/u/358688e13ed6): @王元_Trump (/users/358688e13ed6) 额 我的意思是 数据源不是应该加上这个一共5个么

2017.02.27 15:09 💬 回复

今天是个大晴天 (/u/8284989f4f51): @王元_Trump (/users/358688e13ed6) 哦哦,谢谢指正哈

2017.02.27 15:12 💬 回复

✎ 添加新评论



EasonDev (/u/7eb1b0c7e6b6)

9楼 · 2017.03.22 19:41

(/u/7eb1b0c7e6b6)

你好。我更新到rxjava2后没有找到FlowableEmitter.BackpressureMode.DROP。只有BackpressureStrategy.DROP。还有我用这个的话依然会报MissingBackpressureException错误, 请问您有遇到吗

👍 赞 💬 回复



北极星周广亚 (/u/479e39fc919d)

10楼 · 2017.03.28 11:14

(/u/479e39fc919d)

Single<Long> single = Single.just(11L); // 你这个地方写错了吧?

```
single.subscribe(new SingleObserver<Long>()
```

```
{
    @Override (/users/d55323762b08)
    public void onSubscribe(Disposable d)
    {

    }
}
```

```
@Override (/users/d55323762b08)
public void onSuccess(Long aLong)
{

}
}
```

```
@Override (/users/d55323762b08)
public void onError(Throwable e)
{

}
}
});
```

👍 赞 💬 回复



被以下专题收入，发现更多相似内容

