RxJava2에서의 변화

RxJava2에서의 변경점

RxJava2.x의 릴리즈가 곧 있을 예정이므로, What's different in 2.0을 기반으로 `1.x` 와의 주요 변경점이 무엇인지 알아본다. 집필 시점에서의 최신 버전은 `2.0.0-RC2` 이다.

패키지명의 변경

`RxJava 2.x` 은 `ReactiveStream` 을 준수하기 때문에, 인터페이스명등이 크게 변경되었다. (예: `Subscription -> Disposable`) 그러기에 별도의 패키지에 의해 제공된다.

	1.x	2.x
Dependency notation	io.reactivex:rxjava:1.x.y	io.reactivex.rxjava2:rxjava:2.x.y
Java base package	`rx`	`io.reactivex`

Null 비허용

`onNext(null)` 은 `NullPointerException` 이 발생할 소지가 있다.

```
Observable.just(null);
BehaviorSubject.createDefault<String>(null);
behaviorSubject.onNext(null);
```

API의 response body가 비어있다는 의미로 `null` 을 반환하는 경우는, `Completable` 을 사용해야만 한다. 상태로서 널 허용의 `BehaviorSubject` 를 사용하는 경우에는 대체수단이 없어므로 치명적이다.

Observable와 Flowable

`1.x` 까지는 Backpressure는 `Observable` 에 구현되어있었지만,

Backpressure 없는 것이 `Observable`, 있는 것이 `Flowable`로 인터페이스가 변경되었다.

```
Observable.create<Int>(emitter -> emitter.onNext(1));
Flowable.create<Int>(emitter -> emitter.onNext(1), FlowableEmitter.
```

Backpressure를 지금까지 신경쓰지 않았던 경우는 문제가 없겠지만, 사용하고 있는 경우에는 `Flowable`로 변경할 필요가 있다.

Single

`onSuccess`나 `onError`중에 하나가 한번만 호출된다. `onCompleted`는 호출되지 않는다.

Completable

`onComplete`나 `onError`중에 하나가 한번만 호출된다.

Maybe

`Single`과 `Completable`을 합쳐놓은 것같은 모양이다. `onSuccess`, `onComplete`, `onError`중의 하나가 한번 호출된다.

Subject와 Processor

`Observable`과 `Flowable`처럼 Backpressure 없는 것이 `Subject`, 있는 것이 `Processor`이다.

함수형 인터페이스

`RxJava`에 정의되어있던 각종 함수형 인터페이스가 대부분 변경되었다. 기본적으로 람다를 사용하고 있다면 그렇게 큰 영향은 없을 것이다.

`1.x`	`2.x`
Action0	Action
Action1	Consumer

Action2	BiConsumer
ActionN	Consumer<Object[]>
Func0	삭제 ?
Func1	Function
Func2	BiFunction
Func3-9	Function3-9
FuncN	Function<Object[], R>
Func1<T, Boolean>	Predicate

아래와 같이, 함수형 인터페이스의 메소드명도 조금 바뀌었다.

`Interface name`	`Method name`
Action	run
Consumer	accept
Function	apply
Predicate	test

`Scheduler` 관련 함수형 인터페이스에는 `Runnable`이 사용되고있는듯 하다.

Subscriber

ReactiveStream 준수에 따라, `subscribe()`에도 변경이 있다.

`subscribe(Observer)`와 `subscribe(Subscriber)`를 사용하고 있었던 경우에는 특히 주의가 필요하다.

우선 `Observable#subscribe(Observer)`와 `Flowable#subscribe(Subscriber)`로 분리되었다. 그리고, 양쪽 모두 `void` 메소드로 되어있다.

이것을 `dispose(1.x에서는 unsubscribe)`하기위해서는 `Observer`와 `Subscriber` 자체에 그 구현을 하고, 외부로부터 `dispose`하지않으면 안된다.

또, `onCompleted`가 `onComplete`로 `onStart`가 `onSubscribe`로 변경되었

다.

람다의 'subscribe(onNext, onError, onComplete, onSubscribe)'를 사용하고있는 경우에는 큰 문제는 없어보인다.

Subscription

`Subscription`이 `Disposable`로 변경되었다.

1.x	2.x
Subscription	Disposable
isUnsubscribed()	isDisposed()
unsubscribe()	dispose()
CompositeSubscription	CompositeDisposable

Scheduler

`Schedulers.test()`가 삭제되었기에, `new TestScheduler()`을 사용해야만 한다.

Scheduler를 직접 구현하는 경우에, `Worker`가 폐지된듯하다. 그 외 경우에는 크게 영향은 없다.

Operator의 변경점

What's-different-in-2.0#1x-observable-to-2x-flowable

Operator 직접 구현

Writing-operators-for-2.0