RxJava Operator Details

Creating Observables

Operators that originate new Observables.

- **Create** — create an Observable from scratch by calling observer methods programmatically
- **Defer** — do not create the Observable until the observer subscribes, and create a fresh Observable for each observer
- **Empty/Never/Throw** — create Observables that have very precise and limited behavior
- **From** — convert some other object or data structure into an Observable
- **Interval** — create an Observable that emits a sequence of integers spaced by a particular time interval
- **Just** — convert an object or a set of objects into an Observable that emits that or those objects
- **Range** — create an Observable that emits a range of sequential integers
- **Repeat** — create an Observable that emits a particular item or sequence of items repeatedly
- **Start** — create an Observable that emits the return value of a function
- **Timer** — create an Observable that emits a single item after a given delay

Reactive Extensions (Rx)

Source: http://reactivex.io/

RxJava Operator Details

Transforming Observables

Operators that transform items that are emitted by an Observable.

- **Buffer** — periodically gather items from an Observable into bundles and emit these bundles rather than emitting the items one at a time
- **FlatMap** — transform the items emitted by an Observable into Observables, then flatten the emissions from those into a single Observable
- **GroupBy** — divide an Observable into a set of Observables that each emit a different group of items from the original Observable, organized by key
- **Map** — transform the items emitted by an Observable by applying a function to each item
- **Scan** — apply a function to each item emitted by an Observable, sequentially, and emit each successive value
- **Window** — periodically subdivide items from an Observable into Observable windows and emit these windows rather than emitting the items one at a time

Source: http://reactivex.io/

RxJava Operator Details

Filtering Observables

Operators that selectively emit items from a source Observable.

- **Debounce** — only emit an item from an Observable if a particular timespan has passed without it emitting another item
- **Distinct** — suppress duplicate items emitted by an Observable
- **ElementAt** — emit only item *n* emitted by an Observable
- **Filter** — emit only those items from an Observable that pass a predicate test
- **First** — emit only the first item, or the first item that meets a condition, from an Observable
- **IgnoreElements** — do not emit any items from an Observable but mirror its termination notification
- **Last** — emit only the last item emitted by an Observable
- **Sample** — emit the most recent item emitted by an Observable within periodic time intervals
- **Skip** — suppress the first *n* items emitted by an Observable
- **SkipLast** — suppress the last *n* items emitted by an Observable
- **Take** — emit only the first *n* items emitted by an Observable
- **TakeLast** — emit only the last *n* items emitted by an Observable

Source: http://reactivex.io/

Combining Observables

Operators that work with multiple source Observables to create a single Observable

- **CombineLatest** — when an item is emitted by either of two Observables, combine the latest item emitted by each Observable via a specified function and emit items based on the results of this function
- **Join** — combine items emitted by two Observables whenever an item from one Observable is emitted during a time window defined according to an item emitted by the other Observable
- **Merge** — combine multiple Observables into one by merging their emissions
- **StartWith** — emit a specified sequence of items before beginning to emit the items from the source Observable
- **Switch** — convert an Observable that emits Observables into a single Observable that emits the items emitted by the most-recently-emitted of those Observables
- **Zip** — combine the emissions of multiple Observables together via a specified function and emit single items for each combination based on the results of this function

Reactive Extensions (Rx)

Source: http://reactivex.io/

RxJava Operator Details

Observable Utility Operators

A toolbox of useful Operators for working with Observables

- **Delay** — shift the emissions from an Observable forward in time by a particular amount
- **Do** — register an action to take upon a variety of Observable lifecycle events
- **ObserveOn** — specify the scheduler on which an observer will observe this Observable
- **Serialize** — force an Observable to make serialized calls and to be well-behaved
- **Subscribe** — operate upon the emissions and notifications from an Observable
- **SubscribeOn** — specify the scheduler an Observable should use when it is subscribed to
- **TimeInterval** — convert an Observable that emits items into one that emits indications of the amount of time elapsed between those emissions
- **Timeout** — mirror the source Observable, but issue an error notification if a particular period of time elapses without any emitted items
- **Timestamp** — attach a timestamp to each item emitted by an Observable
- **Using** — create a disposable resource that has the same lifespan as the Observable

Source: http://reactivex.io/

RxJava Operator Details

Conditional and Boolean Operators

Operators that evaluate one or more Observables or items emitted by Observables

- **All** — determine whether all items emitted by an Observable meet some criteria
- **Amb** — given two or more source Observables, emit all of the items from only the first of these Observables to emit an item
- **Contains** — determine whether an Observable emits a particular item or not
- **DefaultIfEmpty** — emit items from the source Observable, or a default item if the source Observable emits nothing
- **SequenceEqual** — determine whether two Observables emit the same sequence of items
- **SkipUntil** — discard items emitted by an Observable until a second Observable emits an item
- **SkipWhile** — discard items emitted by an Observable until a specified condition becomes false
- **TakeUntil** — discard items emitted by an Observable after a second Observable emits an item or terminates
- **TakeWhile** — discard items emitted by an Observable after a specified condition becomes false

Reactive Extensions (Rx)

Source: http://reactivex.io/

RxJava Operator Details

Mathematical and Aggregate Operators

Operators that operate on the entire sequence of items emitted by an Observable

- **Average** — calculates the average of numbers emitted by an Observable and emits this average
- **Concat** — emit the emissions from two or more Observables without interleaving them
- **Count** — count the number of items emitted by the source Observable and emit only this value
- **Max** — determine, and emit, the maximum-valued item emitted by an Observable
- **Min** — determine, and emit, the minimum-valued item emitted by an Observable
- **Reduce** — apply a function to each item emitted by an Observable, sequentially, and emit the final value
- **Sum** — calculate the sum of numbers emitted by an Observable and emit this sum

Backpressure Operators

- **backpressure operators** — strategies for coping with Observables that produce items more rapidly than their observers consume them

Source: http://reactivex.io/

RxJava Operator Details

Connectable Observable Operators

Specialty Observables that have more precisely-controlled subscription dynamics

- **Connect** — instruct a connectable Observable to begin emitting items to its subscribers
- **Publish** — convert an ordinary Observable into a connectable Observable
- **RefCount** — make a Connectable Observable behave like an ordinary Observable
- **Replay** — ensure that all observers see the same sequence of emitted items, even if they subscribe after the Observable has begun emitting items

Operators to Convert Observables

- **To** — convert an Observable into another object or data structure

Error Handling Operators

Operators that help to recover from error notifications from an Observable

- **Catch** — recover from an onError notification by continuing the sequence without error
- **Retry** — if a source Observable sends an onError notification, re-subscribe to it in the hopes that it will complete without error

Reactive Extensions (Rx)

Source: http://reactivex.io/

Lab - Transformation