Overview of Key Classes in the RxJava API

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

- Understand key classes in the RxJava API
- Understand key classes in the RxJava API
Key Classes in the RxJava API
Key Classes in the RxJava API

• There are three key classes in the RxJava API
There are three key classes in the RxJava API

**Single** – Completes successfully or with failure, may or may not emit a single value

---

### Class Single<T>

```java
public abstract class Single<T extends java.lang.Object>
    extends java.lang.Object
    implements SingleSource<T>
```

The `Single` class implements the Reactive Pattern for a single value response.

`Single` behaves similarly to `Observable` except that it can only emit either a single successful value or an error (there is no `onComplete` notification as there is for an `Observable`).

The `Single` class implements the `SingleSource` base interface and the default consumer type it interacts with is the `SingleObserver` via the `subscribe(SingleObserver)` method.

Key Classes in the RxJava API

• There are three key classes in the RxJava API

• **Single** – Completes successfully or with failure, may or may not emit a single value

• Similar to a Java Completable Future or an async Optional<T>
Key Classes in the RxJava API

• There are three key classes in the RxJava API
  • **Single** – Completes successfully or with failure, may or may not emit a single value
    • Similar to a Java Completable Future or an async Optional<T>
    • Can be documented via a “marble diagram”

See [medium.com/@jshvarts/read-marble-diagrams-like-a-pro-3d72934d3ef5](medium.com/@jshvarts/read-marble-diagrams-like-a-pro-3d72934d3ef5)
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a “marble diagram”

This is the timeline of a Single, where time flows from left to right
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional\(<T>\)
    - Can be documented via a “marble diagram”
There are three key classes in the RxJava API

- **Single** – Completes successfully or with failure, may or may not emit a single value
  - Similar to a Java CompletableFuture, Future or an async Optional<T>
  - Can be documented via a “marble diagram”

These dotted lines & this box indicate that a transformation is being applied to the Single

The text inside the box indicates the type of transformation
There are three key classes in the RxJava API:

- **Single** – Completes successfully or with failure, may or may not emit a single value.
  - Similar to a Java Completable Future or an async Optional<T>.
  - Can be documented via a “marble diagram”.

This item is the result of the transformation.
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a “marble diagram”

This vertical line indicates the Single completed successfully
There are three key classes in the RxJava API

**Single** – Completes successfully or with failure, may or may not emit a single value

- Similar to a Java Completable Future or an async Optional<T>
- Can be documented via a “marble diagram”

If the Single terminates abnormally the vertical line is replaced by an X
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional\(\langle T\rangle\)
    - Can be documented via a "marble diagram"
  - Provides many operators

- Factory method operators
- Transforming operators
- Action operators
- Concurrency & scheduler operators
- Combining operators
- Suppressing operators
- Blocking operators
- etc.
There are three key classes in the RxJava API

**Single** – Completes successfully or with failure, may or may not emit a single value

- Similar to a Java Completable Future or an async Optional<T>
- Can be documented via a "marble diagram"
- Provides many operators
- Maybe is a variant of Single

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Maybe.html](http://reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Maybe.html)
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a “marble diagram”
    - Provides many operators
  - Maybe is a variant of Single
    - It may emit a single value, no value at all, or an exception

```java
BigInteger factorial(BigInteger n) {
    return Observable.rangeLong(1, n.longValue()).map(BigInteger::valueOf)
        .reduce(BigInteger::multiply)
        .blockingGet(BigInteger.ONE);
}
```

*reduce() returns a Maybe, which may contain no value at all if n is 0*
There are three key classes in the RxJava API

- **Single** – Completes successfully or with failure, may or may not emit a single value
- **Observable** – Emits an indefinite number of events (zero to infinite) & may complete successfully or fail

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Flowable.html
Key Classes in the RxJava API

• There are three key classes in the RxJava API

  • **Single** – Completes successfully or with failure, may or may not emit a single value

  • **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail

    • Similar to an async Java stream
      • i.e., completable futures used with a Java stream

      ```java
      return Observable
          .fromArray(bigFractionList)
          .subscribeOn(scheduler)
          .flatMap(reducedFraction -> Observable
              .fromCallable(() -> reducedFraction.multiply(sBigReducedFraction))
              .subscribeOn(scheduler))
          .reduce(BigFraction::add);
      ```
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram

See medium.com/@jshvarts/read-marble-diagrams-like-a-pro-3d72934d3ef5
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram

This is the timeline of an Observable, where time flows from left to right
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite number of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram

*These are the items emitted by the Observable*
There are three key classes in the RxJava API:

- **Single** – Completes successfully or with failure, may or may not emit a single value.
- **Observable** – Emits an indefinite number of events (zero to infinite) and may complete successfully or fail.
  - Similar to an async Java stream.
  - Can also be documented via a marble diagram.

These dotted lines & this box indicate that a transformation is being applied to the Observable.

The text inside the box indicates the type of transformation.
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite number of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram

These items are the result of the transformation
There are three key classes in the RxJava API

- **Single** – Completes successfully or with failure, may or may not emit a single value

- **Observable** – Emits an indefinite number of events (zero to infinite) & may complete successfully or fail
- Similar to an async Java stream
- Can also be documented via a marble diagram
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram

If an Observable terminates abnormally the vertical line is replaced by an X
Key Classes in the RxJava API

• There are three key classes in the RxJava API
  • **Single** – Completes successfully or with failure, may or may not emit a single value
  • **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    • Similar to an async Java stream
    • Can also be documented via a marble diagram
  • Provides many operators

• Factory method operators
• Transforming operators
• Action operators
• Concurrency & scheduler operators
• Combining operators
• Terminal operators
• Suppressing operators
• Blocking operators
• etc.
Key Classes in the RxJava API

- There are three key classes in the RxJava API
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
  - **Flowable** – Generalizes Observable to support backpressure

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Flowable.html](http://reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Flowable.html)
There are three key classes in the RxJava API:

- **Single** – Completes successfully or with failure, may or may not emit a single value

- **Observable** – Emits an indefinite number of events (zero to infinite) & may complete successfully or fail

- **Flowable** – Generalizes Observable to support backpressure
  - The subscriber indicates to the publisher how much data it can consume

See [www.baeldung.com/rxjava-backpressure](http://www.baeldung.com/rxjava-backpressure)
Key Classes in the RxJava API

• There are three key classes in the RxJava API
  • **Single** – Completes successfully or with failure, may or may not emit a single value
  • **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
  • **Flowable** – Generalizes Observable to support backpressure
    • The subscriber indicates to the publisher how much data it can consume
    • A Flowable can be converted to a ParallelFlowable

```
return Flowable
  .fromArray(bigFractions)
  .parallel()
  .runOn(scheduler)
  .flatMap(bigFraction ->
    bigFraction.multiply(sBigReducedFraction))
  .sequential()
  .reduce(BigFraction::add)
  ...
```

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/parallel/ParallelFlowable.html](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/parallel/ParallelFlowable.html)
End of Overview of Key Classes in the RxJava API