Key Transforming Operators
in the Flux Class (Part 1)

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

- Recognize key Flux operators
- Factory method operators
- Transforming operators
  - Transform the values and/or types emitted by a Flux
    - e.g., map() & mapNotNull()
Key Transforming Operators in the Flux Class
Key Transforming Operators in the Flux Class

- The map() operator
- Transform the item(s) emitted by this Flux

\[ \text{<V> Flux<V> map (Function<? super T, ? extends V> mapper)} \]

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#map](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#map)
Key Transforming Operators in the Flux Class

- The map() operator
  - Transform the item(s) emitted by this Flux
  - Applies a synchronous function to transform each item

```java
<V> Flux<V> map
    (Function<? super T, ? extends V> mapper)
```

See [docs.oracle.com/javase/8/docs/api/java/util/function/Function.html](docs.oracle.com/javase/8/docs/api/java/util/function/Function.html)
Key Transforming Operators in the Flux Class

- The map() operator
- Transform the item(s) emitted by this Flux
  - Applies a synchronous function to transform each item
  - map() can terminate if mapper throws an exception

```
<V> Flux<V> map
(Function<? super T, ? extends V> mapper)
```
The map() operator

- Transform the item(s) emitted by this Flux
  - Applies a synchronous function to transform each item
- Returns a transformed Flux

\[
\text{Map: } \text{Flux}<\text{V}> \text{ map} \\
(\text{Function}\langle? \text{ super T, } ? \text{ extends V}\rangle \text{ mapper})
\]
Key Transforming Operators in the Flux Class

- The map() operator
  - Transform the item(s) emitted by this Flux
- The # of output items must match the # of input items

Flux

```java
.fromIterable
  (bigFractionList)
... 
.map(fraction -> fraction
  .multiply(sBigReducedFrac))
... 
```

*Multiply each element in the Flux stream by a constant*

See Reactive/flux/ex1/src/main/java/FluxEx.java
Key Transforming Operators in the Flux Class

- The map() operator
  - Transform the item(s) emitted by this Flux
- The # of output items must match the # of input items
  - map() can transform the type and/or value of elements it processes
Key Transforming Operators in the Flux Class

- The `map()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
- RxJava’s `Observable.map()` operator works the same

```java
Observable.fromIterable(bigFractionList)
  .map(fraction -> fraction.multiply(sBigReducedFrac))
```

Multiply each element in the Observable stream by a constant

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#map](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#map)
Key Transforming Operators in the Flux Class

- The map() operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
  - RxJava’s Observable.map() operator works the same
  - Similar to Stream.map() in Java Streams

```java
List<String> collect = List.of("a", "b", "c").stream()
    .map(String::toUpperCase)
    .collect(toList());
```

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#map

map

```java
<R> Stream<R> map(Function<? super T,? extends R> mapper)
```

Returns a stream consisting of the results of applying the given function to the elements of this stream.

This is an intermediate operation.

Type Parameters:
- R - The element type of the new stream

Parameters:
- mapper - a non-interfering, stateless function to apply to each element

Uppercase strings & collect into a List
• The mapNotNull() operator
• Transform the item(s) emitted by this Flux

\[
\text{mapNotNull}(\text{Function}\langle?\text{ super T, }?\text{ extends V}\rangle\text{ mapper})
\]

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#mapNotNull
Key Transforming Operators in the Flux Class

- The `mapNotNull()` operator
- Transform the item(s) emitted by this Flux
- Applies a synchronous function to transform each item

\[
<V> \text{ Flux}<V> \text{ mapNotNull} \\
(\text{Function}<\text{? super T, ? extends V}> \text{ mapper})
\]

**Interface Function\text{T,R>**

Type Parameters:
- T - the type of the input to the function
- R - the type of the result of the function

All Known Subinterfaces:
- UnaryOperator\text{T>

Functional Interface:
This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

See [docs.oracle.com/javase/8/docs/api/java/util/function/Function.html](http://docs.oracle.com/javase/8/docs/api/java/util/function/Function.html)
The `mapNotNull()` operator

Transform the item(s) emitted by this Flux

- Applies a synchronous function to transform each item
- The mapper function may produce null values

```java
<V> Flux<V> mapNotNull (Function<? super T, ? extends V> mapper)
```
Key Transforming Operators in the Flux Class

• The mapNotNull() operator
  • Transform the item(s) emitted by this Flux
  • Applies a synchronous function to transform each item
  • The mapper function may produce null values
  • These null values are not emitted

```
 Flux<V> mapNotNull
 (Function<? super T, ? extends V> mapper)
```
Key Transforming Operators in the Flux Class

- The mapNotNull() operator
  - Transform the item(s) emitted by this Flux
    - Applies a synchronous function to transform each item
      - The mapper function may produce null values
        - These null values are not emitted
      - Behaves like map(Function) followed by filter(Predicate)
        - However, null is not a supported value, so it can't be filtered out
Key Transforming Operators in the Flux Class

- The mapNotNull() operator
- Transform the item(s) emitted by this Flux
  - Applies a synchronous function to transform each item
    - The mapper function may produce null values
  - mapNotNull() can terminate if mapper throws an exception

$$\langle V \rangle \text{ Flux}<V> \text{ mapNotNull} \text{ (Function}\langle ? \text{ super } T, ? \text{ extends } V\rangle \text{ mapper})$$
Key Transforming Operators in the Flux Class

• The `mapNotNull()` operator
  • Transform the item(s) emitted by this Flux
    • Applies a synchronous function to transform each item
  • Returns a transformed Flux that emits no nulls

```
<V> Flux<V> mapNotNull
  (Function<? super T, ? extends V> mapper)
```
Key Transforming Operators in the Flux Class

• The `mapNotNull()` operator
• Transform the item(s) emitted by this Flux
• The # of output items must match the # of input items

Flux
  .just(BigFraction
   .valueOf(100, 3),
   BigFraction
   .valueOf(100, 10))
  .mapNotNull(bf -> bf
   .equals(BigFraction.TEN)
   ? null : bf)
  ...

Return null if bf equals 10, which is then ignored

See Reactive/flux/ex1/src/main/java/FluxEx.java
Key Transforming Operators in the Flux Class

- The `mapNotNull()` operator
  - Transform the item(s) emitted by this Flux
- The # of output items must match the # of input items
  - `mapNotNull()` can transform the type and/or value of elements it processes
Key Transforming Operators in the Flux Class

- The `mapNotNull()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
- RxJava’s Observable lacks a `mapNotNull()` operator

See [github.com/ReactiveX/RxJava/issues/4644](https://github.com/ReactiveX/RxJava/issues/4644)
Key Transforming Operators in the Flux Class

- The `mapNotNull()` operator
- Transform the item(s) emitted by this Flux
- The # of output items must match the # of input items
- RxJava’s Observable lacks a `mapNotNull()` operator
- Java Optional can be used in this case

```java
return Observable
    .fromCallable(() -> url)
    .subscribeOn(Schedulers.io())
    .map(__ -> Optional
        .ofNullable(download(url)))
    .filter(Optional::isPresent)
    .map(Optional::get);
```

See [joshfein.medium.com/handling-null-in-rxjava-2-0-10abd72afa0b](https://joshfein.medium.com/handling-null-in-rxjava-2-0-10abd72afa0b)
Key Transforming Operators in the Flux Class

• The `mapNotNull()` operator
  • Transform the item(s) emitted by this Flux
  • The # of output items must match the # of input items
  • RxJava’s Observable lacks a `mapNotNull()` operator
  • Java Optional can be used in this case
  • RxJava transformers can also be used

```java
static <T, R> ObservableTransformer<T, R>
mapNotNull(Function<? super T, ? extends R> mapper) {
  return upstream -> upstream
    .flatMap(it -> {
      R result = mapper.apply(it);
      if (result == null)
        return Observable.empty();
      else
        return Observable.just(result);
    });
}
```

Key Transforming Operators in the Flux Class

- The `mapNotNull()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
- RxJava’s `Observable` lacks a `mapNotNull()` operator
  - Java `Optional` can be used in this case
  - RxJava transformers can also be used

```java
Observable<Image> downloadImage(URL url) {
    return Observable.fromCallable(() -> url)
        .subscribeOn(Schedulers.io())
        .compose(mapNotNull(this::download));
}
```

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#compose](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#compose)
End of Key Transforming Operators in the Flux Class (Part 1)