Key Factory Method 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

Class Flux<T>

java.lang.Object
   reactor.core.publisher.Flux<T>

Type Parameters:
T - the element type of this Reactive Streams Publisher

All Implemented Interfaces:
Publisher<T>, CorePublisher<T>

Direct Known Subclasses:
ConnectableFlux, FluxOperator, FluxProcessor, GroupedFlux

public abstract class Flux<T>
   extends Object
   implements CorePublisher<T>

A Reactive Streams Publisher with rx operators that emits 0 to N elements, and then completes (successfully or with an error).

The recommended way to learn about the Flux API and discover new operators is through the reference documentation, rather than through this javadoc (as opposed to learning more about individual operators). See the “which operator do I need?” appendix.

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html
Learning Objectives in this Part of the Lesson

- Recognize key Flux operators
- Factory method operators
  - These operators create Flux streams in various ways
  - e.g., just(), fromArray(), fromIterable(), & from()
Key Factory Method
Operators in the Flux Class
Key Factory Method Operators in the Flux Class

- The `just()` operator

  - Create a Flux that emits the given element(s) & then completes

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

- The just() operator
  - Create a Flux that emits the given element(s) & then completes
    - The param(s) are the elements to emit, as a vararg

```
static <T> Flux<T> just(T... data)
```

See [www.baeldung.com/java-varargs](http://www.baeldung.com/java-varargs)
Key Factory Method Operators in the Flux Class

- The just() operator
  - Create a Flux that emits the given element(s) & then completes
    - The param(s) are the elements to emit, as a vararg
    - Returns a new Flux that’s captured at “assembly time”
      - i.e., it’s “eager”
  
```
static <T> Flux<T> just(T... data)
```

“Assembly time” is when the Flux object is instantiated, rather than when it “runs”
Key Factory Method Operators in the Flux Class

- **The just() operator**
  - Create a Flux that emits the given element(s) & then completes
    - The param(s) are the elements to emit, as a vararg
    - Returns a new Flux that’s captured at instantiation time
  - Multiple elements can be emitted, unlike the Mono.just() operator

```java
static <T> Flux<T> just(T... data)
```

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

• The just() operator
  • Create a Flux that emits the given element(s) & then completes
  • This factory method operator adapts non-reactive input sources to the reactive model

Create a Flux stream of four BigFraction objects

 Flux .just(BigFraction.valueOf(100, 3),
         BigFraction.valueOf(100, 4),
         BigFraction.valueOf(100, 2),
         BigFraction.valueOf(100, 1))
...
Key Factory Method Operators in the Flux Class

- The `just()` operator
  - Create a Flux that emits the given element(s) & then completes
  - This factory method operator adapts non-reactive input sources to the reactive model
  - Since `just()` is evaluated eagerly at “assembly time” it runs in the thread where assembly is performed

The `fromIterable()` & `fromArray()` factory method operators also evaluate eagerly.
Key Factory Method Operators in the Flux Class

- The just() operator
  - Create a Flux that emits the given element(s) & then completes
  - This factory method operator adapts non-reactive input sources to the reactive model
- RxJava’s Observable.just() works the same

Create an Observable stream of four BigFraction objects

```java
Observable.just(BigFraction.valueOf(100,3),
    BigFraction.valueOf(100,4),
    BigFraction.valueOf(100,2),
    BigFraction.valueOf(100,1))
```

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#just](http://reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#just)
Key Factory Method Operators in the Flux Class

• The `just()` operator
  • Create a Flux that emits the given element(s) & then completes
  • This factory method operator adapts non-reactive input sources to the reactive model
  • RxJava’s `Observable.just()` works the same
  • Similar to the `Stream.of()` operator in Java Streams

```
of
```  
@SafeVarargs
static <T> Stream<T> of(T... values)

Returns a sequential ordered stream whose elements are the specified values.

Type Parameters:
  T - the type of stream elements

Parameters:
  values - the elements of the new stream

Returns:
  the new stream

Stream
.of(BigFraction.valueOf(100,3),
    BigFraction.valueOf(100,4),
    BigFraction.valueOf(100,2),
    BigFraction.valueOf(100,1))
...
Key Factory Method Operators in the Flux Class

- The fromIterable() method
- Create a Flux that emits items contained in the given Iterable

```java
static <T> Flux<T> fromIterable
    (Iterable <? extends T> it)
```

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

- The `fromIterable()` method
- Create a Flux that emits items contained in the given `Iterable`
- The `Iterable.iterator()` method will be invoked at least once & at most twice for each subscriber

```java
static <T> Flux<T> fromIterable(
    Iterable<? extends T> it)
```

---

See [docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html](docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html)
Key Factory Method Operators in the Flux Class

- The `fromIterable()` method
  - Create a Flux that emits items contained in the given `Iterable`
  - This factory method operator also adapts non-reactive input sources into the reactive model
    - e.g., Java collections like List & Set

Create a Flux stream of Integer objects from a Java List collection

```java
List<Integer> list = List.of(0,1,1,2,3,5,8,13,21);
Flux.fromIterable(list) ...
```

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

- The `fromIterable()` method
  - Create a Flux that emits items contained in the given Iterable
  - This factory method operator also adapts non-reactive input sources into the reactive model
  - RxJava’s method `Observable.fromIterable()` works the same

```java
List<Integer> list = List.of(0,1,1,2,3,5,8,13,21);
Observable.fromIterable(list)
```

Create an Observable stream of Integer objects from a List collection

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

- The fromIterable() method
  - Create a Flux that emits items contained in the given Iterable
  - This factory method operator also adapts non-reactive input sources into the reactive model
  - RxJava’s method Observable. fromIterable() works the same
  - Similar to the stream() method in Java Collection

Create a stream of Integer objects

List<Integer> list = List.of(0,1,1,2,3,5,8,13,21);

list.stream()...

See docs.oracle.com/javase/8/docs/api/java/util/Collection.html#stream
Key Factory Method Operators in the Flux Class

- The `fromArray()` method
- Create a Flux that emits items in the given Java built-in array

```java
static <T> Flux<T> fromArray(T[] array)
```

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#fromArray
Key Factory Method Operators in the Flux Class

- The `fromArray()` method
  - Create a Flux that emits items in the given Java built-in array
  - The param provides the array to read the data from

```java
static <T> Flux<T> fromArray(T[] array)
```

See [docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html](docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html)
The fromArray() method

Create a Flux that emits items in the given Java built-in array

- The param provides the array to read the data from
- The returned Flux emits the items from the array

```java
static <T> Flux<T> fromArray(T[] array)
```
The `fromArray()` method

- Create a Flux that emits items in the given Java built-in array.
- This factory method operator also adapts non-reactive input sources into the reactive model.

```java
Integer[] array = {0, 1, 1, 2, 3, 5, 8, 13, 21};
Flux.fromArray(array);
```

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

- The fromArray() method
  - Create a Flux that emits items in the given Java built-in array
  - This factory method operator also adapts non-reactive input sources into the reactive model
  - RxJava’s method Observable.fromArray() works the same

```
Integer[] array = {0, 1, 1, 2, 3, 5, 8, 13, 21};
```

Create an Observable stream of Integer objects from a built-in array

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#fromArray](http://reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#fromArray)
• The `fromArray()` method
  • Create a Flux that emits items in the given Java built-in array
  • This factory method operator also adapts non-reactive input sources into the reactive model
  • RxJava’s method Observable. `fromArray()` works the same
  • Similar to the `of()` method in Java Streams

```
Integer[] array = {0, 1, 1, 2, 3, 5, 8, 13, 21};
Stream.of(array)
```

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of](https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of)
• The `fromArray()` method
• Create a Flux that emits items in the given Java built-in array
• This factory method operator also adapts non-reactive input sources into the reactive model
• RxJava’s method Observable. `fromArray()` works the same
• Similar to the of() method in Java Streams
  • Also, similar to the stream() method in Java Arrays

```
Integer[] array = {0, 1, 1, 2, 3, 5, 8, 13, 21};
Arrays.
.stream(array)
...```

See docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#stream
Key Factory Method Operators in the Flux Class

• The from() method
• Decorate the specified Publisher with the Flux API

static <T> Flux<T> from
(Publisher<? extends T> source)

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#from
Key Factory Method Operators in the Flux Class

- The `from()` method
- Decorate the specified Publisher with the Flux API
- The param provides the source to decorate

```java
static <T> Flux<T> from
    (Publisher<? extends T> source)
```

See [www.reactive-streams.org/reactive-streams-1.0.3-javadoc/org/reactivestreams/Publisher.html](http://www.reactive-streams.org/reactive-streams-1.0.3-javadoc/org/reactivestreams/Publisher.html)
• The from() method
  • Decorate the specified Publisher with the Flux API
    • The param provides the source to decorate
    • Returns a new Flux that decorates the source at runtime
    • i.e., it’s “lazy”

static <T> Flux<T> from
  (Publisher<? extends T> source)
• The from() method
  • Decorate the specified Publisher with the Flux API
• This factory method operator adapts non-Flux publishers into the Flux API

Create a Flux containing a single BigFraction object from a Mono

Flux
  .from(Mono
    .fromCallable
      (() ->
        BigFractionUtils
          .makeBigFraction(random, true)))

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

- The from() method
  - Decorate the specified Publisher with the Flux API
- This factory method operator adapts non-Flux publishers into the Flux API
  - from() is “lazy”

```java
Flux.from(Mono.fromCallable(() -> BigFractionUtils.makeBigFraction(random, true)))
```

*It invokes the Publisher param at the time of subscription & separately for each subscriber*
Key Factory Method Operators in the Flux Class

- The `from()` method
  - Decorate the specified Publisher with the Flux API
- This factory method operator adapts non-Flux publishers into the Flux API
  - `from()` is “lazy”

```java
Flux
  .from(Mono
    .fromCallable
      (() ->
        BigFractionUtils
          .makeBigFraction(random, 
          true))))
```

Can be used as a workaround for Flux’s lack of a `fromCallable()` method
Key Factory Method Operators in the Flux Class

- The from() method
  - Decorate the specified Publisher with the Flux API
  - This factory method operator adapts non-Flux publishers into the Flux API
  - RxJava’s method Observable.fromCallable() is similar

Create an Observable containing a single BigFraction object

```java
Observable.fromCallable(() -> BigFractionUtils.makeBigFraction(random, true))
```

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#fromCallable
End of Key Factory Method
Operators in the Flux Class
(Part 1)