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

```java
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](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()`

See [en.wikipedia.org/wiki/Factory_method_pattern](en.wikipedia.org/wiki/Factory_method_pattern)
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

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

See 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 instantiation time
      - i.e., it’s “eager”
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

\[
\text{static } \langle T \rangle \text{ Flux}\langle T \rangle \text{ just}(T... \text{ data})
\]

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

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

See Reactive/flux/ex1/src/main/java/FluxEx.java
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
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

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#just](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#just)
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

Create a stream of 4 BigFraction objects

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of](http://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of)
Key Factory Method Operators in the Flux Class

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

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

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#fromIterable
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)
```

### Interface `Iterable<T>`

**Type Parameters:**
- `T` - the type of elements returned by the iterator

**All Known Subinterfaces:**
- `BeanContext`, `BeanContextServices`, `BlockingDeque<E>`, `BlockingQueue<E>`, `Collection<E>`, `Deque<E>`, `DirectoryStream<T>`, `List<E>`, `NavigableSet<E>`, `Path`, `Queue<E>`, `SecureDirectoryStream<T>`, `Set<E>`, `SortedSet<E>`, `TransferQueue<E>`

See [docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html](https://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

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

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

See Reactive/flux/ex1/src/main/java/FluxEx.java
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)
```

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#fromIterable](http://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](http://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

```
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](http://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)
```
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

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

Create a Flux stream of Integer objects from a Java built-in 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

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

ObservableFromArray(array)
```

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](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.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
  - 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

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

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

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of](docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of)
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
- Similar to the `of()` method in Java Streams
  - Also, similar to the `stream()` method in Java Arrays

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

See [docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#stream](https://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

\[
\text{static } \langle T \rangle \ \text{Flux}<T> \ from \\
(Publisher<? \text{ extends } T> \ \text{source})
\]

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)
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
  - Returns a new Flux that decorates the source at runtime
    - i.e., it’s “lazy”

```java
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”

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](http://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)