Key Action Operators in the Flux Class

(Part 2)

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
  - Concurrency & scheduler operators
  - Factory method operators
- Action operators
  - These operators don’t modify a Flux, but instead use it for side effects
    - e.g., doFinally()
Key Action Operators in the Flux Class
Key Action Operators in the Flux Class

- The `doFinally()` operator
- Add a behavior triggered after the Flux terminates for any reason

```java
Flux<T> doFinally
(Consumer<SignalType> onFinally)
```

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#doFinally](projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#doFinally)
Key Action Operators in the Flux Class

- The `doFinally()` operator
- Add a behavior triggered after the Flux terminates for any reason
- The param is called when the Flux signals `onError()` or `onComplete()` or is disposed by the downstream

```java
Flux<T> doFinally
(Consumer<SignalType> onFinally)
```

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

- The `doFinally()` operator
- Add a behavior triggered after the Flux terminates for any reason
  - The param is called when the Flux signals `onError()` or `onComplete()` or is disposed by the downstream
  - i.e., it is a “callback”

```java
Flux<T> doFinally(Consumer<SignalType> onFinally)
```

See en.wikipedia.org/wiki/Callback_(computer_programming)
Key Action Operators in the Flux Class

- The `doFinally()` operator
- Add a behavior triggered after the Flux terminates for any reason
  - The param is called when the Flux signals `onError()` or `onComplete()` or is disposed by the downstream
  - Returns the new unchanged Flux instance
    - i.e., it only has a “side-effect”

See en.wikipedia.org/wiki/Side_effect_(computer_science)
Key Action Operators in the Flux Class

• The `doFinally()` operator
  • Add a behavior triggered after the Flux terminates for any reason
  • Does not operate by default on a particular Scheduler
Key Action Operators in the Flux Class

- The `doFinally()` operator
  - Add a behavior triggered after the Flux terminates for any reason
  - Does not operate by default on a particular Scheduler
  - Can’t change the type or value of elements it processes

```java
Scheduler subscriber = Schedulers.newParallel("subscriber", 1);
return Flux
  .create(makeAsyncFluxSink())
  .publishOn(subscriber)
  ...
  .doFinally(___ -> subscriber.dispose()) ...
```

This operator is called after the Flux completes

See Reactive/flux/ex2/src/main/java/FluxEx.java
Key Action Operators in the Flux Class

• The `doFinally()` operator
  • Add a behavior triggered after the Flux terminates for any reason
  • Does not operate by default on a particular Scheduler
  • Can’t change the type or value of elements it processes

Scheduler subscriber = Schedulers.newParallel("subscriber", 1);
return Flux
   .create(makeAsyncFluxSink())
   .publishOn(subscriber)
   ...
   .doFinally(__ -> subscriber.dispose()) ...

This callback disposes the subscriber thread
Key Action Operators in the Flux Class

- The `doFinally()` operator
  - Add a behavior triggered after the Flux terminates for any reason
  - Does not operate by default on a particular Scheduler
  - Can’t change the type or value of elements it processes

- RxJava’s operator Observable
  .`doFinally()` works the same

```java
Observable
  .create(ObservableEx::emitAsync)
  .observeOn(Schedulers.newThread()) ...
  .doFinally(() -> BigFractionUtils.display(sb.toString()))
```

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

Print BigIntegers to aid debugging
End of Key Action
Operators in the Flux Class
(Part 2)