Key Suppressing Operators in the Mono 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 Mono operators
  - Concurrency & scheduler operators
  - Transforming operators
  - Combining operators
  - Suppressing operators
    - These operators create a Mono that ignores its payload
      - e.g., materialize() & fromRunnable()

Key Suppressing Operators in the Mono Class

Key Suppressing Operators in the Mono Class

- The materialize() operator
  - Transforms incoming onNext, onError, & onComplete signals into Signal instances, thereby materializing these signals

```
Mono<Signal<T>>
  materialize()
```

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#materialize

Key Suppressing Operators in the Mono Class

- The materialize() operator
  - Transforms incoming onNext, onError, & onComplete signals into Signal instances, thereby materializing these signals
    - Returns a Mono that will first emit a Signal.complete() & then effectively complete the Mono

```
Mono<Signal<T>>
   materialize()
```

Interface Signal<T>

Type Parameters:

T - the value type

All Superinterfaces:

Consumer<Subscriber<? super T>>, Supplier<T>

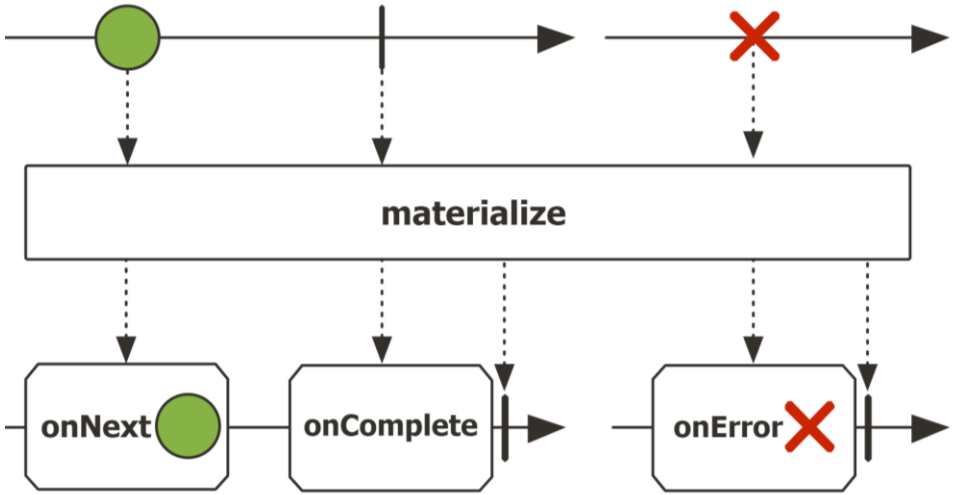
```
public interface Signal<T>
extends Supplier<T>, Consumer<Subscriber<? super T>>
```

A domain representation of a Reactive Stream signal. There are 4 distinct signals and their possible sequence is defined as such: onError | (onSubscribe onNext* (onError | onComplete)?)

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Signal.html

Key Suppressing Operators in the Mono Class

- The materialize() operator
  - Transforms incoming onNext, onError, & onComplete signals into Signal instances, thereby materializing these signals
  - This "data-suppressing" operator ignores its payload



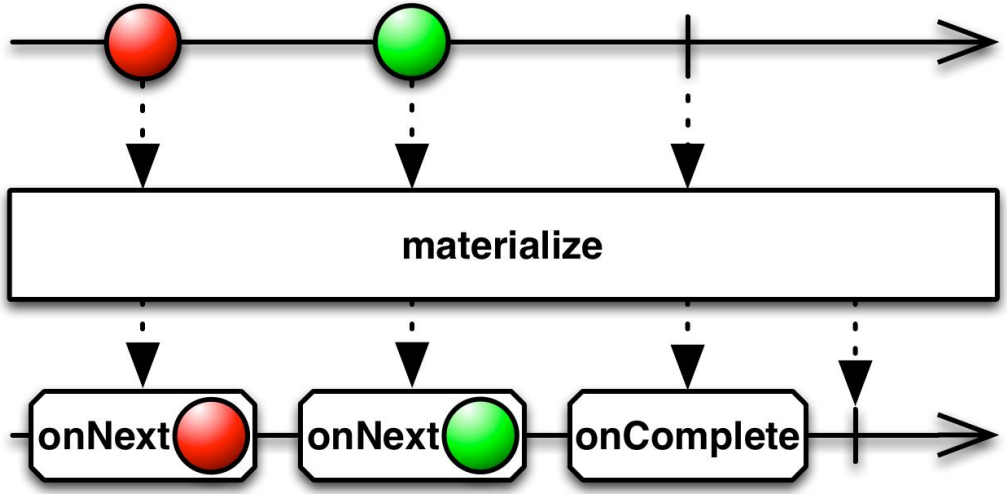

```
return monos -> Mono
   .when(monos)
   .materialize()
   .flatMap(v -> Flux.fromIterable(monos)
                     .map(Mono::block)
                     .collect(toList()));
```

Return a mono that signals when all monos complete

See Reactive/flux/ex3/src/main/java/utils/MonosCollector.java

Key Suppressing Operators in the Mono Class

- The materialize() operator
  - Transforms incoming onNext, onError, & onComplete signals into Signal instances, thereby materializing these signals
  - This "data-suppressing" operator ignores its payload
  - RxJava's materialize() operator works the same



See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Single.html#materialize

Key Suppressing Operators in the Mono Class

- The fromRunnable() operator
  - Executes the given Runnable & returns a Mono

```
static <T> Mono<T>
   fromRunnable
      (Runnable runnable)
```

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#fromRunnable

Key Suppressing Operators in the Mono Class

- The fromRunnable() operator
  - Executes the given Runnable & returns a Mono
    - The Runnable param is executed before emitting the completion signal

```
static <T> Mono<T>
   fromRunnable
      (Runnable runnable)
```

Interface Runnable

All Known Subinterfaces:

RunnableFuture<V>, RunnableScheduledFuture<V>

All Known Implementing Classes:

AsyncBoxView.ChildState, ForkJoinWorkerThread, FutureTask, RenderableImageProducer, SwingWorker, Thread, TimerTask

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/lang/Runnable.html

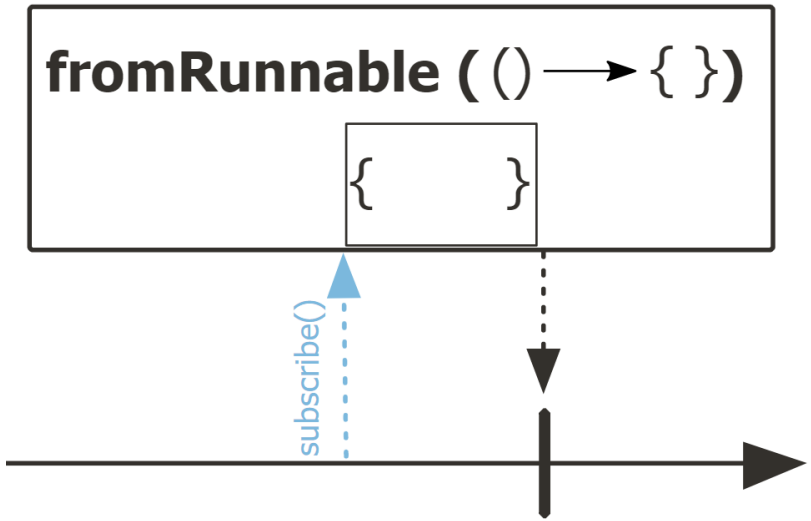
Key Suppressing Operators in the Mono Class

- The fromRunnable() operator
  - Executes the given Runnable & returns a Mono
    - The Runnable param is executed before emitting the completion signal
    - Returns a Mono that completes empty

```
static <T> Mono<T>
   fromRunnable
      (Runnable runnable)
```

Key Suppressing Operators in the Mono Class

- The fromRunnable() operator
  - Executes the given Runnable & returns a Mono
  - Can be used to trigger some final processing before completion



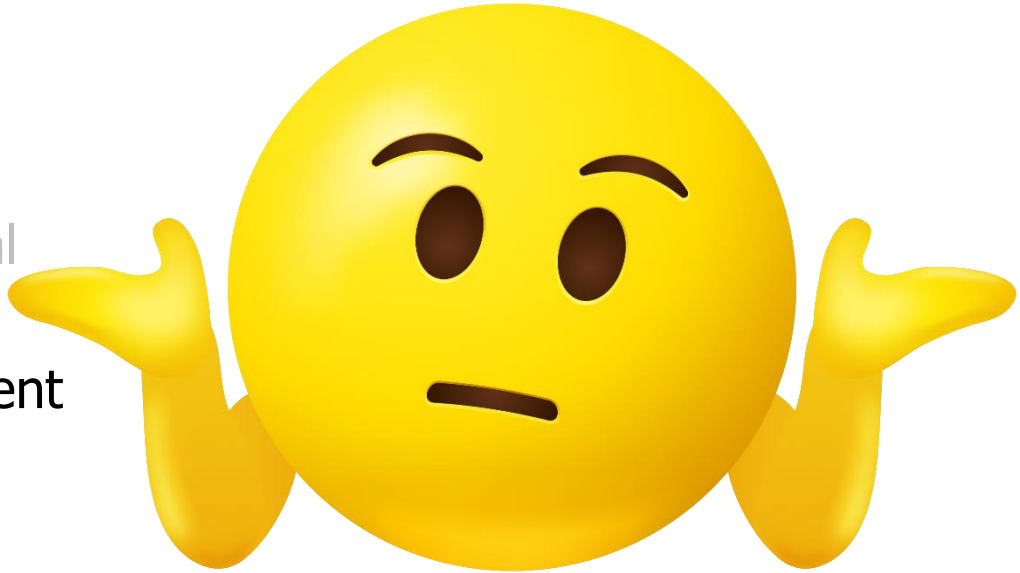

```
return Flux
   .create(makeAsyncFluxSink())
   ...
   .map(bigInteger ->
         FluxEx.checkIfPrime(bigInteger, sb))
   ...
   .then(Mono
         .fromRunnable(() ->
               BigFractionUtils.display(sb.toString())));
```

Display results & return an empty mono to sync with AsyncTaskBarrier

See Reactive/flux/ex2/src/main/java/FluxEx.java

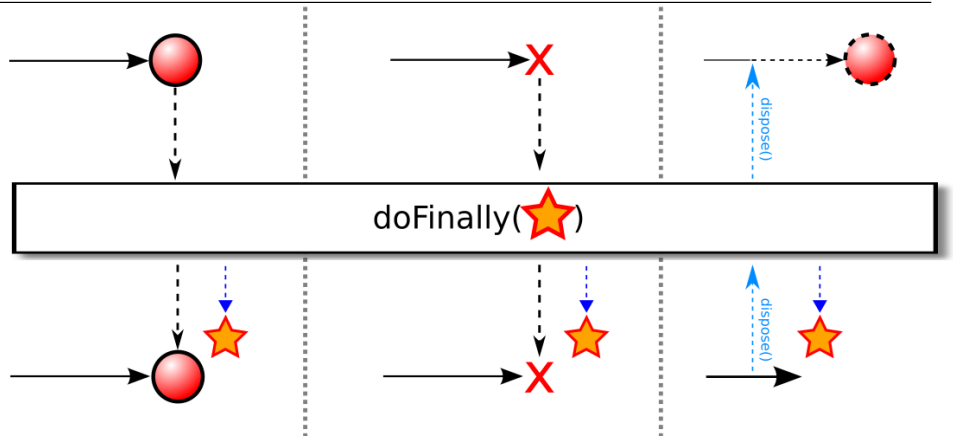
Key Suppressing Operators in the Mono Class

- The fromRunnable() operator
  - Executes the given Runnable & returns a Mono
  - Can be used to trigger some final processing before completion
  - There's no direct RxJava equivalent



Key Suppressing Operators in the Mono Class

- The fromRunnable() operator
  - Executes the given Runnable & returns a Mono
  - Can be used to trigger some final processing before completion
  - There's no direct RxJava equivalent




```
return Observable
   .create(ObservableEx::emitAsync)
   ...
   .map(bigInteger -> ObservableEx
         .checkIfPrime(bigInteger, sb))
   ...
   .doFinally(() -> BigFractionUtils.display(sb.toString()))
   ...
```

However, doFinally() can play a similar role

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Single.html#doFinally

Key Suppressing Operators in the Mono Class

- The fromRunnable() operator
  - Executes the given Runnable & returns a Mono
  - Can be used to trigger some final processing before completion
  - There's no direct RxJava equivalent
  - Similar to thenRun() in Java completable futures

thenRun

```
public CompletableFuture<Void> thenRun(Runnable action)
```

Description copied from interface: CompletionStage

Returns a new CompletionStage that, when this stage completes normally, executes the given action. See the CompletionStage documentation for rules covering exceptional completion.

Specified by:

thenRun in interface CompletionStage<T>

Parameters:

action - the action to perform before completing the returned CompletionStage

Returns:

the new CompletionStage

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#thenRun

End of Key Suppressing Operators in the Mono Class (Part 2)