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

See Reactive/flux/ex3/src/main/java/utils/MonosCollector.java
Key Supressing 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](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

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

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#fromRunnable](projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#fromRunnable)
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)

See docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html
• 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

```java
static <T> Mono<T> fromRunnable
  (Runnable runnable)
```
The `fromRunnable()` operator
- Executes the given Runnable & returns a Mono
- Can be used to trigger some final processing before completion

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

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

```java
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](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)