

Applying Key Operators in Project Reactor: Case Study ex4 (Part 2)

Douglas C. Schmidt

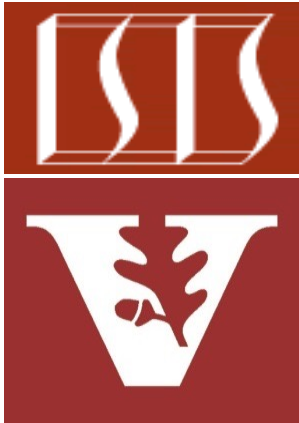
d.schmidt@vanderbilt.edu

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

- Part 2 of case study ex4 applies Flux operators `subscribe()`, `flatMap()`, & `fromArray()` to create, multiply, & display `BigFraction` objects asynchronously

```
Mono
    .fromSupplier(() ->
        .makeBigFraction
            (sRANDOM, true))
    .flatMapMany(bf1 -> Flux
        .fromArray(bigFractionArray)
        .flatMapMany(bf2 -> Mono
            .fromCallable(() -> bf2)
            .subscribeOn(scheduler)
            .map(____ -> bf2
                .multiply(bf1))))
    .subscribe(blockingSubscriber);

blockingSubscriber.await();
```

This example does not apply backpressure at all

Learning Objectives in this Part of the Lesson

- Part 2 of case study ex4 applies Flux operators `subscribe()`, `flatMap()`, & `fromArray()` to create, multiply, & display `BigFraction` objects asynchronously
- It also shows how to use `Mono` operators `fromSupplier()`, `map()`, `flatMapMany()`, & `subscribeOn()`

Mono

```
.fromSupplier(() ->
    .makeBigFraction
        (sRANDOM, true))
.flatMapMany(bf1 -> Flux
    .fromArray(bigFractionArray)
    .flatMapMap(bf2 -> Mono
        .fromCallable(() -> bf2)
        .subscribeOn(scheduler)
        .map(____ -> bf2
            .multiply(bf1))))

.subscribe(blockingSubscriber);

blockingSubscriber.await();
```

Learning Objectives in this Part of the Lesson

- Part 1 of case study ex4 applies Flux operators `subscribe()`, `flatMap()`, & `fromArray()` to create, multiply, & display `BigFraction` objects asynchronously
 - It also shows how to use Mono operators `fromSupplier()`, `map()`, `flatMapMany()`, & `subscribeOn()`
 - In addition, it shows how to use the generic blocking Subscriber in a “backpressure-unaware” manner

Mono

```
.fromSupplier(() ->
    .makeBigFraction
        (sRANDOM, true))
.flatMapMany(bf1 -> Flux
    .fromArray(bigFractionArray)
    .flatMapMany(bf2 -> Mono
        .fromCallable(() -> bf2)
        .subscribeOn(scheduler)
        .map(____ -> bf2
            .multiply(bf1))))
.subscribe(blockingSubscriber);
```

```
blockingSubscriber.await();
```

Applying Key Operators in Project Reactor to ex4

Applying Key Operators in Project Reactor to ex4

```
59 Mono
60 // Generate a random large BigFraction.
61 .fromSupplier(() -> BigFractionUtils
62     .makeBigFraction(sRANDOM, true))
63
64 // Transform the item emitted by this Mono into a
65 // Publisher and then forward its emissions into the
66 // returned Flux.
67 .flatMapMany(bf1 -> Flux
68     // Generate a stream of BigFractions.
69     .fromArray(bigFractionArray)
70
71     // Perform the Project Reactor
72     // flatMap() concurrency idiom.
73     .flatMap(bf2 ->
74         multiplyFraction(bf1,
75             bf2,
76             Schedulers.parallel(),
77             sb)))
```

See github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/flux/ex4

End of Applying Key Methods in Project Reactor: Case Study ex4 (Part 2)