Key Concurrency & Scheduler Operators in the Mono 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 Mono operators
  - Concurrency & scheduler operators
    - These operators arrange to run other operators in designated threads & thread pools
      - e.g., Mono.subscribeOn() & Schedulers.single()

Key Concurrency Operators in the Mono Class

Key Concurrency Operators in the Mono Class

- The subscribeOn() operator
  - Run subscribe(), onSubscribe(), & request() on the specified Scheduler worker

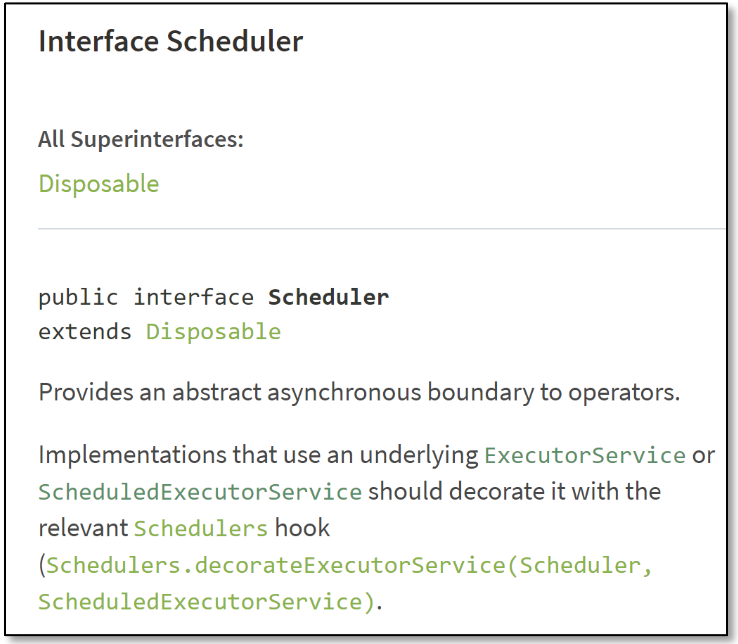
```
Mono<T> subscribeOn(Scheduler
                    scheduler)
```

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

Key Concurrency Operators in the Mono Class

- The subscribeOn() operator
  - Run subscribe(), onSubscribe(), & request() on the specified Scheduler worker
    - The scheduler param indicates what thread to perform the operation on

```
Mono<T> subscribeOn(Scheduler
                    scheduler)
```



Interface Scheduler

All Superinterfaces:

Disposable

```
public interface Scheduler
extends Disposable
```

Provides an abstract asynchronous boundary to operators.

Implementations that use an underlying `ExecutorService` or `ScheduledExecutorService` should decorate it with the relevant `Schedulers` hook (`Schedulers.decorateExecutorService(Scheduler, ScheduledExecutorService)`.

See projectreactor.io/docs/core/release/api/reactor/core/scheduler/Scheduler.html

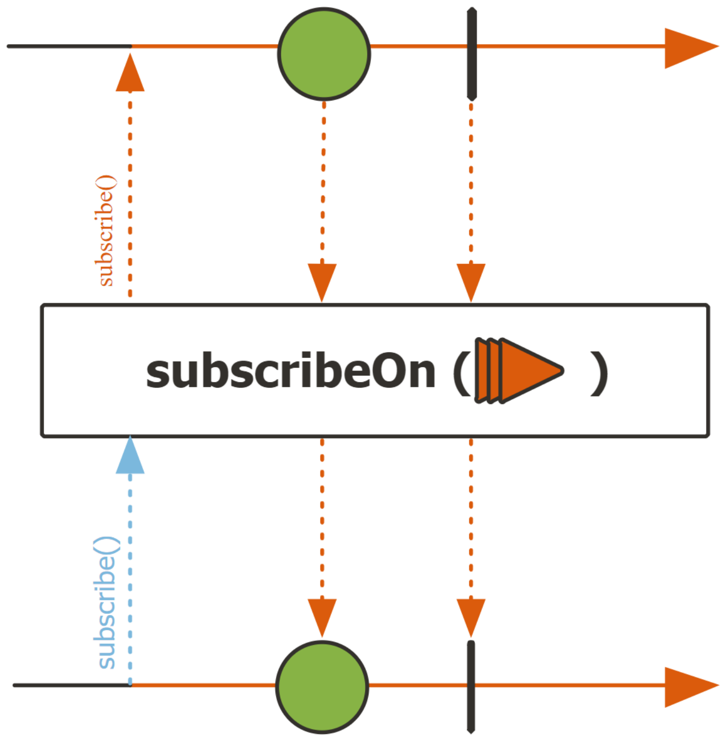
Key Concurrency Operators in the Mono Class

- The subscribeOn() operator
  - Run subscribe(), onSubscribe(), & request() on the specified Scheduler worker
    - The scheduler param indicates what thread to perform the operation on
    - Returns the Mono requesting async processing

```
Mono<T> subscribeOn(Scheduler
                    scheduler)
```

Key Concurrency Operators in the Mono Class

- The subscribeOn() operator
  - Run subscribe(), onSubscribe(), & request() on the specified Scheduler worker
  - The semantics of subscribeOn() are a bit unusual



Key Concurrency Operators in the Mono Class

- The subscribeOn() operator
  - Run subscribe(), onSubscribe(), & request() on the specified Scheduler worker
  - The semantics of subscribeOn() are a bit unusual
    - Placing this operator in a chain impacts the execution context of the onNext(), onError(), & onComplete() signals

```
return Mono
    .fromCallable(() -> BigFraction
                   .reduce(unreducedFrac))

    .subscribeOn(Schedulers.single())

    .doOnSuccess(bf -> logBigFraction
                  (unreducedFrac, bf, sb))

    .map(BigFraction::toMixedString)

    .doOnSuccess(bf ->
           displayBigFraction(bf, sb))

    .then();
```

Run all this processing in a background thread

See [Reactive/mono/ex2/src/main/java/MonoEx.java]

Key Concurrency Operators in the Mono Class

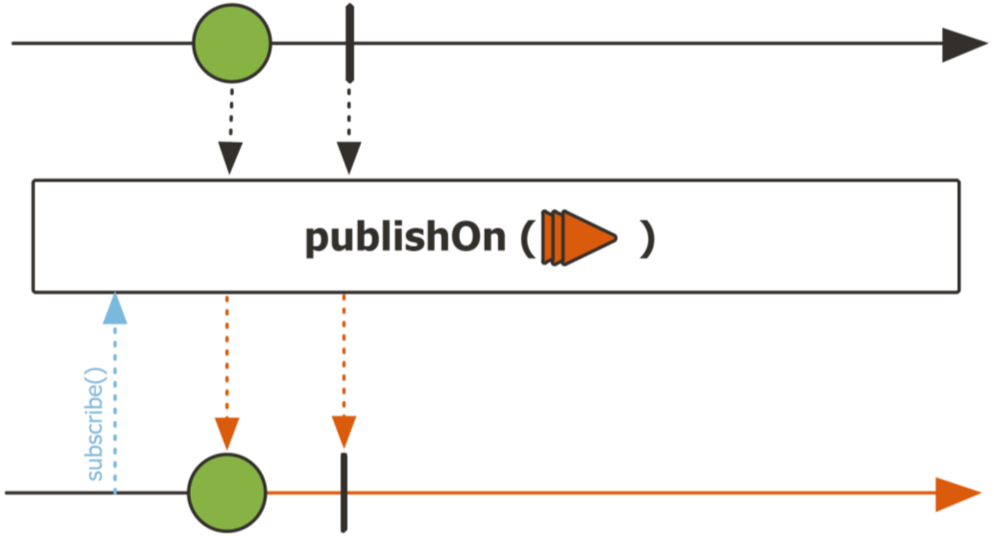
- The subscribeOn() operator
  - Run subscribe(), onSubscribe(), & request() on the specified Scheduler worker
  - The semantics of subscribeOn() are a bit unusual
    - Placing this operator in a chain impacts the execution context of the onNext(), onError(), & onComplete() signals

```
return Mono
   .fromCallable(() -> BigFraction
                .reduce(unreducedFrac))

   .doOnSuccess(bf -> logBigFraction
              (unreducedFrac, bf, sb))

   .map(BigFraction::toMixedString)

   .doOnSuccess(bf ->
          displayBigFraction(bf, sb))

   .subscribeOn(Schedulers.single())

   .then();
```

subscribeOn() can appear later in the chain & have the same effect

Key Concurrency Operators in the Mono Class

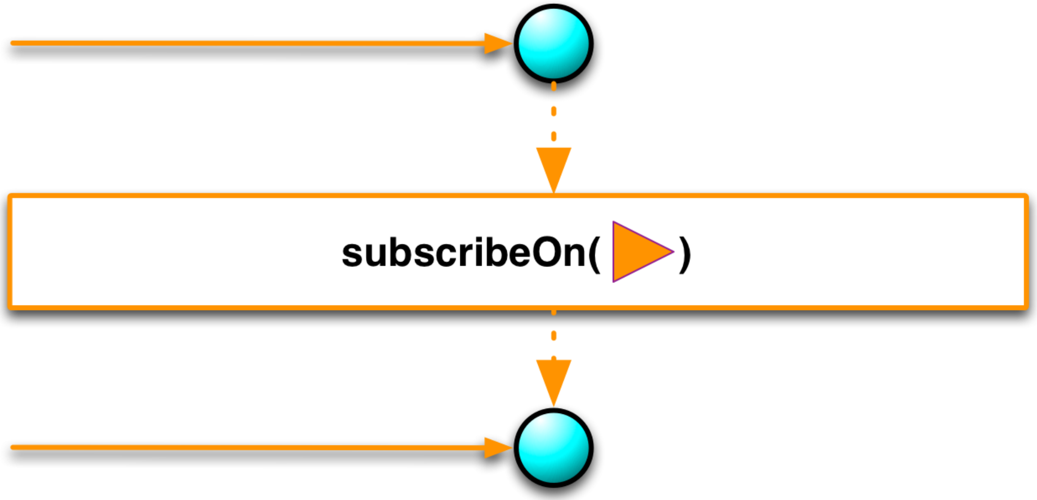
- The subscribeOn() operator
  - Run subscribe(), onSubscribe(), & request() on the specified Scheduler worker
  - The semantics of subscribeOn() are a bit unusual
    - Placing this operator in a chain impacts the execution context of the onNext(), onError(), & onComplete() signals
      - However, if a publishOn() operator appears later in the chain that will change the threading context where the rest of the operators in the chain below it execute (publishOn() can appear multiple times)



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

Key Concurrency Operators in the Mono Class

- The subscribeOn() operator
  - Run subscribe(), onSubscribe(), & request() on the specified Scheduler worker
  - The semantics of subscribeOn() are a bit unusual
  - RxJava's Single.subscribeOn() works the same way



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

Key Scheduler Operators in the Mono Class

Key Scheduler Operators in the Mono Class

- The Schedulers.single() operator
  - Hosts a single-threaded Executor Service-based worker that runs concurrently wrt the caller

```
static Scheduler single()
```

See projectreactor.io/docs/core/release/api/reactor/core/scheduler/Schedulers.html#single

Key Scheduler Operators in the Mono Class

- The Schedulers.single() operator

```
static Scheduler single()
```

  - Hosts a single-threaded Executor Service-based worker that runs concurrently wrt the caller

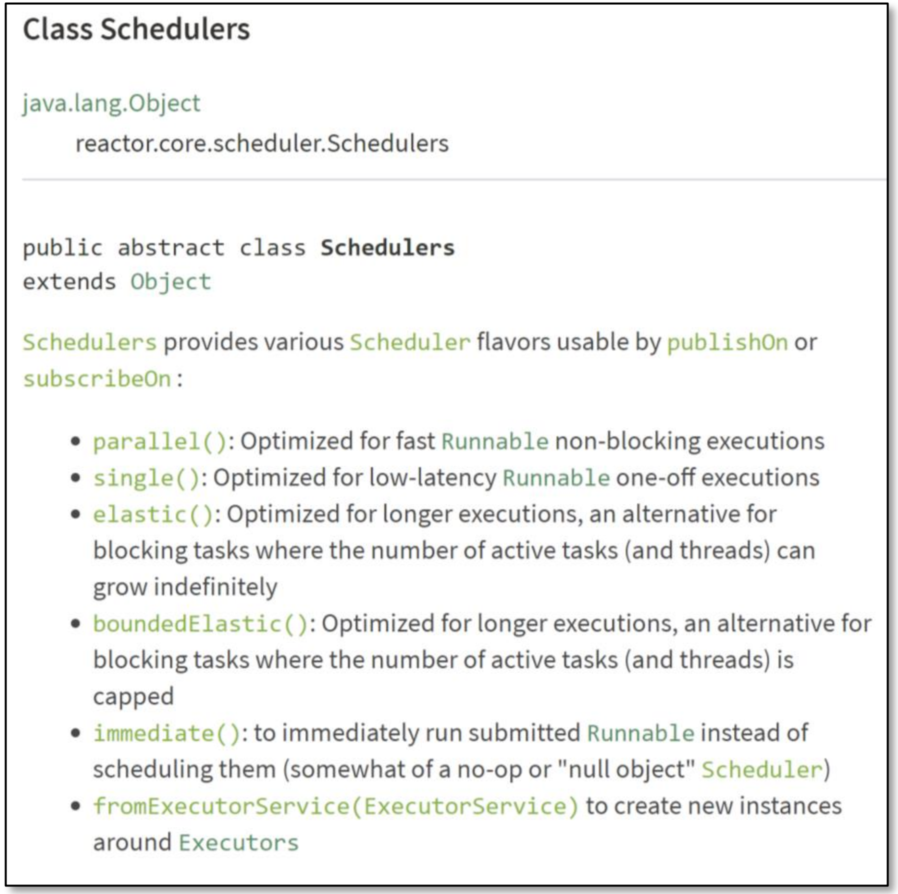
```
return Mono
   .fromCallable(() -> BigFraction.reduce(unreducedFrac))

   .subscribeOn(Schedulers.single())


   .doOnSuccess(bf -> logBigFraction(unreducedFrac, bf, sb))
   .map(BigFraction::toMixedString)
   .doOnSuccess(bf -> displayBigFraction(bf, sb))
   .then();
```

Run all this processing in a single background thread

See Reactive/mono/ex2/src/main/java/MonoEx.java

Key Scheduler Operators in the Mono Class

- The Schedulers.single() operator
  - Hosts a single-threaded Executor Service-based worker that runs concurrently wrt the caller
    - Optimized for low-latency calls that all run in one (& only one) background thread



Class Schedulers

java.lang.Object
    reactor.core.scheduler.Schedulers

```
public abstract class Schedulers
extends Object
```

Schedulers provides various Scheduler flavors usable by publishOn or subscribeOn:

- parallel(): Optimized for fast Runnable non-blocking executions
- single(): Optimized for low-latency Runnable one-off executions
- elastic(): Optimized for longer executions, an alternative for blocking tasks where the number of active tasks (and threads) can grow indefinitely
- boundedElastic(): Optimized for longer executions, an alternative for blocking tasks where the number of active tasks (and threads) is capped
- immediate(): to immediately run submitted Runnable instead of scheduling them (somewhat of a no-op or "null object" Scheduler)
- fromExecutorService(ExecutorService) to create new instances around Executors

See projectreactor.io/docs/core/release/api/reactor/core/scheduler/Schedulers.html

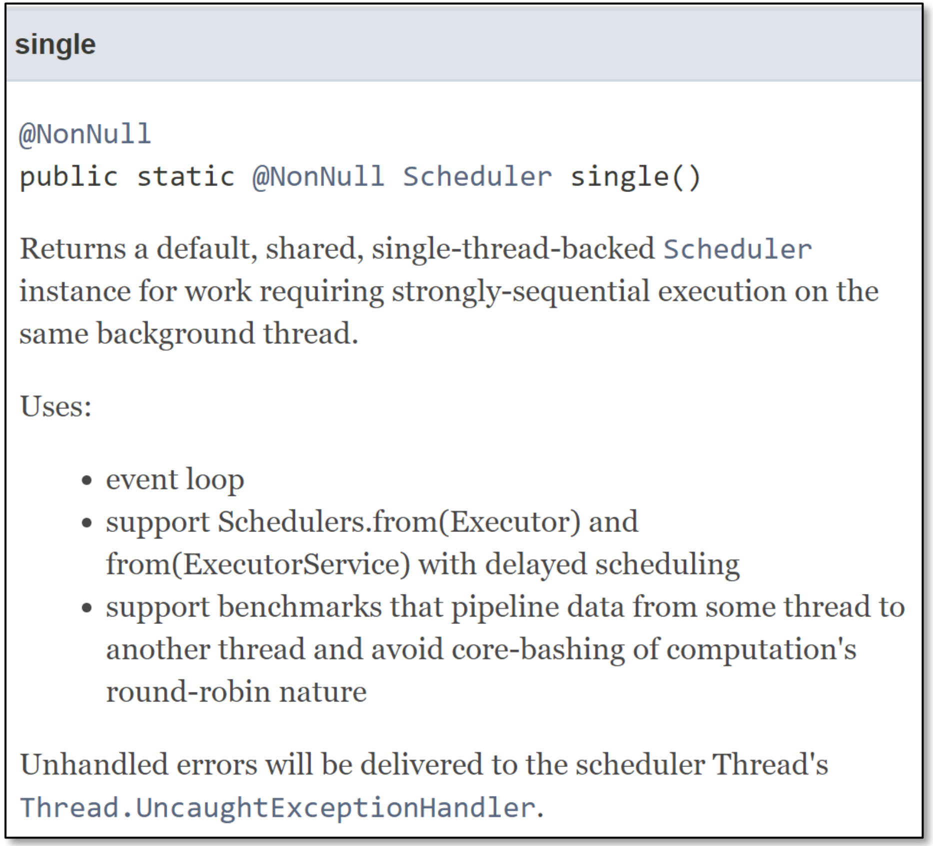
Key Scheduler Operators in the Mono Class

- The Schedulers.single() operator
  - Hosts a single-threaded Executor Service-based worker that runs concurrently wrt the caller
    - Optimized for low-latency calls that all run in one (& only one) background thread
    - Implemented via a "daemon thread"
      - i.e., won't prevent the app from exiting even if its work isn't done

See www.baeldung.com/java-daemon-thread

Key Scheduler Operators in the Mono Class

- The Schedulers.single() operator
  - Hosts a single-threaded ExecutorService-based worker that runs concurrently wrt the caller
  - RxJava's Schedulers.single() works the same way



single

```
@NonNull
public static @NonNull Scheduler single()
```

Returns a default, shared, single-thread-backed `Scheduler` instance for work requiring strongly-sequential execution on the same background thread.

Uses:

- event loop
- support Schedulers.from(Executor) and from(ExecutorService) with delayed scheduling
- support benchmarks that pipeline data from some thread to another thread and avoid core-bashing of computation's round-robin nature

Unhandled errors will be delivered to the scheduler Thread's `Thread.UncaughtExceptionHandler`.

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/schedulers/Schedulers.html#single

End of Key Concurrency & Scheduler Operators in the Mono Class (Part 1)