Key Concurrency & Scheduler

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
  • Factory method operators
  • Transforming operators
• Concurrency & scheduler operators
  • These operators arrange to run other operators in designated threads & thread pools
  • e.g., Schedulers.parallel()
Key Scheduler Operators in the Flux Class
Key Scheduler Operators in the Flux Class

- The Schedulers.parallel() operator
- Hosts a fixed pool of single-threaded ExecutorService-based workers

See projectreactor.io/docs/core/release/api/reactor/core/scheduler/Schedulers.html#parallel
Key Scheduler Operators in the Flux Class

- The Schedulers.parallel() operator
  - Hosts a fixed pool of single-threaded ExecutorService-based workers
  - Returns a new Scheduler that is suited for parallel work
Key Scheduler Operators in the Flux Class

- The `Schedulers.parallel()` operator
- Hosts a fixed pool of single-threaded ExecutorService-based workers
- Returns a new Scheduler that is suited for parallel work
  - Size obtained by system property `reactor.schedulers.defaultPoolSize`

See [projectreactor.io/docs/core/release/api/reactor/core/scheduler/Schedulers.html#DEFAULT_POOL_SIZE](http://projectreactor.io/docs/core/release/api/reactor/core/scheduler/Schedulers.html#DEFAULT_POOL_SIZE)
Key Scheduler Operators in the Flux Class

- The Schedulers.parallel() operator
  - Hosts a fixed pool of single-threaded ExecutorService-based workers
  - Returns a new Scheduler that is suited for parallel work
    - Size obtained by system property `reactor.schedulers.defaultPoolSize`
    - Falls back to # of processors available to the runtime on init

See [docs.oracle.com/javase/8/docs/api/java/lang/Runtime.html](docs.oracle.com/javase/8/docs/api/java/lang/Runtime.html)
The Schedulers.parallel() operator

- Hosts a fixed pool of single-threaded ExecutorService-based workers
- Returns a new Scheduler that is suited for parallel work
  - Size obtained by system property reactor.schedulers.defaultPoolSize
- Optimized for computation-intensive non-blocking tasks due to its fixed-size

See projectreactor.io/docs/core/release/api/reactor/core/scheduler/Schedulers.html
Key Scheduler Operators in the Flux Class

- The Schedulers.parallel() operator
  - Hosts a fixed pool of single-threaded ExecutorService-based workers
  - Can be used to leverage the default parallel scheduler

```
Flux
  .fromIterable(bigFractions)
  .flatMap
    (bf -> Flux
      .just(bf)
      .subscribeOn
        (Schedulers.parallel())
    )
  .map(multiplyBigFracs)
...
```

*Use the “flatMap() concurrency idiom” to multiply the big integers in a background thread in the parallel thread pool*

See [Reactive/flux/ex3/src/main/java/FluxEx.java](Reactive/flux/ex3/src/main/java/FluxEx.java)
Key Scheduler Operators in the Flux Class

- The Schedulers.parallel() operator
  - Hosts a fixed pool of single-threaded ExecutorService-based workers
  - Can be used to leverage the default parallel scheduler
  - Implemented via “daemon threads”
    - i.e., won’t prevent the app from exiting even if its work isn’t done

See [www.baeldung.com/java-daemon-thread](www.baeldung.com/java-daemon-thread)
The Schedulers.parallel() operator
• Hosts a fixed pool of single-threaded ExecutorService-based workers
• Can be used to leverage the default parallel scheduler
• Implemented via “daemon threads”

RxJava’s Schedulers.computation() works in a similar way
• i.e., it’s fixed-size & intended for compute-intensive & non-blocking tasks

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/schedulers/Schedulers.html#computation](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/schedulers/Schedulers.html#computation)
End of Key Concurrency & Scheduler Operators in the Flux Class (Part 2)