Key Factory Method Operators in the Observable 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 Observable operators
• Concurrency & scheduler operators
• Factory method operators
  • These operators create Observable streams in various ways
  • e.g., create(), interval(), & rangeLong()

See en.wikipedia.org/wiki/Factory_method_pattern
Key Factory Method
Operators in the Observable Class
Key Factory Method Operators in the Observable Class

- The `create()` operator
- Create an Observable that can emit multiple elements synchronously or asynchronously

```java
static <T> Observable<T> create(
    ObservableOnSubscribe<T> source
)
```

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#create
Key Factory Method Operators in the Observable Class

- The create() operator
- Create an Observable that can emit multiple elements synchronously or asynchronously
- The ObservableOnSubscribe() subscribe() method receives an ObservableEmitter instance

```java
static <T> Observable<T> create(ObservableOnSubscribe<T> source)
```

A functional interface that has a subscribe() method that receives an instance of an ObservableEmitter instance that allows pushing events in a cancellation-safe manner.

Key Factory Method Operators in the Observable Class

- The `create()` operator
  - Create an Observable that can emit multiple elements synchronously or asynchronously

- The `ObservableOnSubscribe()` `subscribe()` method receives an `ObservableEmitter` instance
  - `ObservableEmitter` can emit events via `onNext()`, `onError()`, & `onComplete()`

```java
class ObservableEmitter<T> {
    // Method definitions...
}

public interface ObservableEmitter<T> extends Emitter<T> {
    // Implementation details...
}
```

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/ObservableEmitter.html](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/ObservableEmitter.html)
Key Factory Method Operators in the Observable Class

- The `create()` operator
  - Create an Observable that can emit multiple elements synchronously or asynchronously
  - The `ObservableOnSubscribe()` subscribe() method receives an `ObservableEmitter` instance
    - `ObservableEmitter` can emit events via `onNext()`, `onError()`, & `onComplete()`
  - Supports more dynamic use cases than the `Observable just()` & `fromIterable()` operators

```java
static <T> Observable<T> create
( ObservableOnSubscribe<T> source )
```

See earlier lesson on “Key Factory Method Operators in the Observable Class (Part 1)”
Key Factory Method Operators in the Observable Class

- The create() operator
  - Create an Observable that can emit multiple elements synchronously or asynchronously
    - The ObservableOnSubscribe() subscribe() method receives an ObservableEmitter instance
    - Returns an Observable that emits all elements generated by the ObservableEmitter

```java
static <T> Observable<T> create
    (ObservableOnSubscribe<T> source)
```
Key Factory Method Operators in the Observable Class

- The create() operator
  - Create an Observable that can emit multiple elements synchronously or asynchronously
  - Elements can be emitted from one or more threads

```java
Observable.create(ObservableEx::emitInterval)
```

See Reactive/Observable/ex2/src/main/java/ObservableEx.java
Key Factory Method Operators in the Observable Class

- The `create()` operator
  - Create an Observable that can emit multiple elements synchronously or asynchronously
  - Elements can be emitted from one or more threads

```cpp
Observable
    .create(ObservableEx::emitInterval)
```

See upcoming discussion of the `Observable.interval()` operator
Key Factory Method Operators in the Observable Class

- The create() operator
  - Create an Observable that can emit multiple elements synchronously or asynchronously
  - Elements can be emitted from one or more threads
- Project Reactor’s Flux.create() operator works in a similar way

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#create](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#create)
Key Factory Method Operators in the Observable Class

- The `create()` operator
  - Create an Observable that can emit multiple elements synchronously or asynchronously
  - Elements can be emitted from one or more threads
- Project Reactor’s Flux.create() operator works in a similar way

However, Project Reactor’s Flux.create() operator is more complex..
Key Factory Method Operators in the Observable Class

- The create() operator
  - Create an Observable that can emit multiple elements synchronously or asynchronously
  - Elements can be emitted from one or more threads
  - Project Reactor’s Flux.create() operator works in a similar way
  - Also similar to Stream.generate() in Java Streams

```java
Generate a stream of random, large, & unreduced big fractions
```

```java
Stream.generate(() -> BigFractionUtils.makeBigFraction(new Random(), false))
```

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#generate](docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#generate)
Key Factory Method Operators in the Observable Class

- The interval() operator
- Create a Observable that emits long values starting with zero (0)

```java
static Observable<Long> interval(long period, TimeUnit unit)
```

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#interval](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#interval)
Key Factory Method Operators in the Observable Class

- The interval() operator
  - Create a Observable that emits long values starting with zero (0)
  - The value is incremented at the specified time intervals on the global timer

```java
static Observable<Long> interval
    (long period, TimeUnit unit)
```
Key Factory Method Operators in the Observable Class

- The interval() operator
- Create a Observable that emits long values starting with zero (0)
  - The value is incremented at the specified time intervals on the global timer
  - Returns a new Observable emitting increasing #’s at regular intervals

```java
static Observable<Long>
interval
  (long period, TimeUnit unit)
```
Key Factory Method Operators in the Observable Class

- The interval() operator
  - Create a Observable that emits long values starting with zero (0)
  - Emits values on the Schedulers .computation() Scheduler

```java
@NonNull
public static @NonNull Scheduler computation()
```

Returns a default, shared Scheduler instance intended for computational work.

This can be used for event-loops, processing callbacks and other computational work.

It is not recommended to perform blocking, IO-bound work on this scheduler. Use io() instead.

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/schedulers/Schedulers.html#computation
The interval() operator

- Create a Observable that emits long values starting with zero (0)
- Emits values on the Schedulers .computation() Scheduler
- Other overloaded interval() methods can designate the Scheduler to use

Key Factory Method Operators in the Observable Class

- The interval() operator
  - Create a Observable that emits long values starting with zero (0)
  - Emits values on theSchedulers .computation() Scheduler
  - In normal conditions, the Observable never completes

... Observable
  .interval(500, MILLISECONDS)
...

Generate a stream of longs every .5 seconds in a background thread

See Reactive/Observable/ex2/src/main/java/ObservableEx.java
Key Factory Method Operators in the Observable Class

- The interval() operator
  - Create a Observable that emits long values starting with zero (0)
  - Emits values on the Schedulers .computation() Scheduler
- In normal conditions, the Observable never completes

```java
Observable
  .interval(500, MILLISECONDS)
  .take(sMAX_ITERATIONS)
```

take() only processes sMAX_ITERATIONS # of emitted values

See upcoming discussion of the Observable.take() operator
Key Factory Method Operators in the Observable Class

- The interval() operator
  - Create a Observable that emits long values starting with zero (0)
  - Emits values on theSchedulers.computation() Scheduler
  - In normal conditions, the Observable never completes

- Project Reactor’s operator Flux .interval() works the same

Use take() to only process sMAX_ITERATIONS # of emitted values from interval()

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#interval
Key Factory Method Operators in the Observable Class

- The rangeLong() operator
- Build an Observable that only emits a sequence of incrementing longs

```java
static Observable<Long> rangeLong (long start, long count)
```

See `reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#rangeLong`
Key Factory Method Operators in the Observable Class

• The rangeLong() operator
  • Build an Observable that only emits a sequence of incrementing longs
    • Emits longs between start & start + count & then completes

static Observable<Long> rangeLong (long start, long count)
Key Factory Method Operators in the Observable Class

- The rangeLong() operator
- Build an Observable that only emits a sequence of incrementing longs
  - Emits longs between start & start + count & then completes
  - start is included in the range, up to & including count

```java
static Observable<Long> rangeLong (long start, long count)
```
The `rangeLong()` operator

- Build an Observable that only emits a sequence of incrementing longs
  - Emits longs between start & start + count & then completes
  - Returns a “ranged” Observable containing count elements

```java
static Observable<Long> rangeLong (long start, long count)
```
The `rangeLong()` operator

- Build an Observable that only emits a sequence of incrementing longs
- Works much like a “reactive” for-each loop

```java
final long sMAX_ITERATIONS = 10;
...
Observable
    .rangeLong(1L, sMAX_ITERATIONS)
    ...
```

See Reactive/Observables/ex2/src/main/java/ObservableEx.java
Key Factory Method Operators in the Observable Class

- The `rangeLong()` operator
  - Build an Observable that only emits a sequence of incrementing longs
  - Works much like a “reactive” for-each loop
- Project Reactor’s `Flux.range()` operator is similar

```
final int sMAX_ITERATIONS = 10;
...
Flux
    .range(1, sMAX_ITERATIONS)
    ...
```

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#range](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#range)
Key Factory Method Operators in the Observable Class

- The `rangeLong()` operator
  - Build an Observable that only emits a sequence of incrementing longs
  - Works much like a “reactive” for-each loop
- Project Reactor’s `Flux.range()` operator is similar
  - There is no `rangeLong()` however

```java
final int sMAX_ITERATIONS = 10;
...

Flux
  .range(1, sMAX_ITERATIONS)
  .map(Integer::toUnsignedLong)
  ...
```

Convert to Long since Flux lacks a `rangeLong()` operator
Key Factory Method Operators in the Observable Class

- The rangeLong() operator
  - Build an Observable that only emits a sequence of incrementing longs
  - Works much like a “reactive” for-each loop
  - Project Reactor’s Flux.range() operator is similar
  - Similar to IntStream.rangeClosed() in Java Streams

```
IntStream.rangeClosed(1, sMAX_ITERATIONS)
... 
```

See docs.oracle.com/javase/8/docs/api/java/util/stream/IntStream.html#rangeClosed
End of Key Factory Method Operators in the Observable Class (Part 2)