Key Error Handling Operators
in the Flux Class

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
- Error handling operators
  - These operators handle errors that occur in a Flux chain
    - e.g., onErrorContinue(), onErrorResume(), & onErrorStop()
Key Error Handling
Operators in the Flux Class
Key Error Handling Operators in the Flux Class

- The `onErrorContinue()` operator
- Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element

```java
Flux<T> onErrorContinue(
    BiConsumer<Throwable, Object> errorConsumer)
```

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#onErrorContinue](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#onErrorContinue)
Key Error Handling Operators in the Flux Class

- The `onErrorContinue()` operator
- Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  - The param is a `BiConsumer` that is fed with errors matching the predicate & the value that triggered the error

```
Flux<T> onErrorContinue
    (BiConsumer<Throwable, Object> errorConsumer)
```

**Interface BiConsumer<T,U>**

Type Parameters:
- T - the type of the first argument to the operation
- U - the type of the second argument to the operation

Functional Interface:
This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

See [docs.oracle.com/javase/8/docs/api/java/util/function/BiConsumer.html](docs.oracle.com/javase/8/docs/api/java/util/function/BiConsumer.html)
Key Error Handling Operators in the Flux Class

- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  - The param is a BiConsumer that is fed with errors matching the predicate & the value that triggered the error
  - The type of the error is a subclass of Throwable

```java
Flux<T> onErrorContinue
    (BiConsumer<Throwable, Object> errorConsumer)
```

public class Throwable
    extends Object
    implements Serializable

The Throwable class is the superclass of all errors and exceptions in the Java language. Only objects that are instances of this class (or one of its subclasses) are thrown by the Java Virtual Machine or can be thrown by the Java throw statement. Similarly, only this class or one of its subclasses can be the argument type in a catch clause. For the purposes of compile-time checking of exceptions, Throwable and any subclass of Throwable that is not also a subclass of either RuntimeException or Error are regarded as checked exceptions.

See [docs.oracle.com/javase/8/docs/api/java/lang/Throwable.html](docs.oracle.com/javase/8/docs/api/java/lang/Throwable.html)
Key Error Handling Operators in the Flux Class

• The `onErrorContinue()` operator
  • Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
    • The param is a `BiConsumer` that is fed with errors matching the predicate & the value that triggered the error
  • Returns a Flux that attempts to continue processing when errors (exceptions) occur

```
Flux<T> onErrorContinue
    (BiConsumer<Throwable, Object> errorConsumer)
```
Key Error Handling Operators in the Flux Class

- The `onErrorContinue()` operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  - This operator “swallows” the exception so it won’t propagate up the call chain/stack further

See [en.wikipedia.org/wiki/Error_hiding](en.wikipedia.org/wiki/Error_hiding)
The onErrorContinue() operator

- Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
- This operator “swallows” the exception so it won’t propagate up the call chain/stack further

```java
return Flux
    .fromIterable(denominators)
    .map(denominator -> BigFraction
        .valueOf(Math.abs(sRANDOM.nextInt()), denominator))
    .onErrorContinue(logErrorAndContinue)
...```

See Reactive/flux/ex3/src/main/java/FluxEx.java
Key Error Handling Operators in the Flux Class

- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  - This operator “swallows” the exception so it won’t propagate up the call chain/stack further
  - It also affects the behavior of onErrorResume() operators.

Flux

```
Flux.range(1, 5)
    .doOnNext(i -> log("i = " + i))
    .map(i -> i == 2 ? i / 0 : i)
    .map(i -> i * 2)
    .onErrorResume(err -> {
        log("resuming");
        return Flux.empty();
    })
    .onErrorContinue((err, i) -> {
        log("continuing={}", i);
    })
    .reduce(Math::addExact)
    .doOnNext(i -> println("sum=" + i))
    .block();
```

*onErrorResume() is ignored if onErrorContinue() appears downstream*

See [devdojo.com/ketonemaniac/reactor-onerrorcontinue-vs-onerrorresume](devdojo.com/ketonemaniac/reactor-onerrorcontinue-vs-onerrorresume)
Key Error Handling Operators in the Flux Class

- The `onErrorContinue()` operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  - This operator “swallows” the exception so it won’t propagate up the call chain/stack further
  - It also affects the behavior of `onErrorResume()` operators.
    - See upcoming discussion of `onErrorStop()` for a solution

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#onErrorStop](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#onErrorStop)
Key Error Handling Operators in the Flux Class

• The onErrorContinue() operator
  • Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  • This operator “swallows” the exception so it won’t propagate up the call chain/stack further
• RxJava’s has no direct equivalent
Key Error Handling Operators in the Flux Class

- The `onErrorResume()` operator
- Subscribe to a returned fallback publisher when any error occurs

```java
Flux<T> onErrorResume(
    Function<? super Throwable, ? extends Publisher <? extends T>> fallback)
```
Key Error Handling Operators in the Flux Class

• The `onErrorResume()` operator
  • Subscribe to a returned fallback publisher when any error occurs
  • The param is a Function that chooses the fallback, depending on the type of the error

```java
Flux<T> onErrorResume
    (Function<? super Throwable, ? extends Publisher <? extends T>> fallback)
```

See docs.oracle.com/javase/8/docs/api/java/util/function/Function.html
Key Error Handling Operators in the Flux Class

- The `onErrorResume()` operator
- Subscribe to a returned fallback publisher when any error occurs
- The param is a Function that chooses the fallback, depending on the type of the error
- The type of the error is a subclass of Throwable

```java
public class Throwable
extends Object
implements Serializable

The Throwable class is the superclass of all errors and exceptions in the Java language. Only objects that are instances of this class (or one of its subclasses) are thrown by the Java Virtual Machine or can be thrown by the Java throw statement. Similarly, only this class or one of its subclasses can be the argument type in a catch clause. For the purposes of compile-time checking of exceptions, Throwable and any subclass of Throwable that is not also a subclass of either RuntimeException or Error are regarded as checked exceptions.
```

See [docs.oracle.com/javase/8/docs/api/java/lang/Throwable.html](docs.oracle.com/javase/8/docs/api/java/lang/Throwable.html)
Key Error Handling Operators in the Flux Class

- The `onErrorResume()` operator
- Subscribe to a returned fallback publisher when any error occurs
  - The param is a Function that chooses the fallback, depending on the type of the error
  - Returns a Flux that falls back to the publisher when an `onError()` occurs

```java
Flux<T> onErrorResume
    (Function<? super Throwable, ? extends Publisher <? extends T>> fallback)
```
Key Error Handling Operators in the Flux Class

• The `onErrorResume()` operator
  • Subscribe to a returned fallback publisher when any error occurs
  • This operator “swallows” the exception so it won’t propagate up the call chain/stack further

See [en.wikipedia.org/wiki/Error_hiding](en.wikipedia.org/wiki/Error_hiding)
Key Error Handling Operators in the Flux Class

- The `onErrorResume()` operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator “swallows” the exception so it won’t propagate up the call chain/stack further

```java
return Flux
    .fromIterable(denominators)
    .map(denominator -> BigFraction.valueOf(Math.abs(sRANDOM.nextInt()), denominator))
    .onErrorResume(__ -> Flux.empty())
    .onErrorStop()
    .collectList()
...```

See `Reactive/flux/ex3/src/main/java/FluxEx.java`
The onErrorResume() operator

- Subscribe to a returned fallback publisher when any error occurs
- This operator “swallows” the exception so it won’t propagate up the call chain/stack further
- Beware when onErrorResume() is used in conjunction with onErrorContinue()

**Flux**
```
Flux.range(1,5)
  .doOnNext(i -> log("i = " + i))
  .map(i -> i == 2 ? i / 0 : i)
  .map(i -> i * 2)
  .onErrorResume(err -> {
    log("resuming");
    return Flux.empty();
  })
  .onErrorContinue((err, i) -> {
    log("continuing={}", i);
  })
  .reduce(Math::addExact)
  .doOnNext(i ->
    println("sum=" + i))
  .block();
```

*onErrorResume() is ignored if onErrorContinue() appears downstream*

See [devdojo.com/ketonemaniac/reactor-onerrorcontinue-vs-onerrorresume](devdojo.com/ketonemaniac/reactor-onerrorcontinue-vs-onerrorresume)
Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator “swallows” the exception so it won’t propagate up the call chain/stack further
- Beware when onErrorResume() is used in conjunction with onErrorContinue()
- See the upcoming discussion of onErrorStop() for a solution

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#onErrorStop](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#onErrorStop)
Key Error Handling Operators in the Flux Class

- The `onErrorResume()` operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator “swallows” the exception so it won’t propagate up the call chain/stack further
- RxJava’s method `Observable.onErrorResumeNext()` works the same

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#onErrorResumeNext]
Key Error Handling Operators in the Flux Class

- The `onErrorResume()` operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator “swallows” the exception so it won’t propagate up the call chain/stack further
  - RxJava’s method `Observable.onErrorResumeNext()` works the same
- The Java `CompletableFuture` `exceptionally()` method is similar

```java
public CompletionStage<T> exceptionally(
    Function<Throwable, ? extends T> fn)
```

Returns a new `CompletionStage` that, when this stage completes exceptionally, is executed with this stage's exception as the argument to the supplied function. Otherwise, if this stage completes normally, then the returned stage also completes normally with the same value.

**Parameters:**
- `fn` - the function to use to compute the value of the returned `CompletionStage` if this `CompletionStage` completed exceptionally

**Returns:**
- the new `CompletionStage`

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#exceptionally](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#exceptionally)
Key Error Handling Operators in the Flux Class

- The `onErrorStop()` operator

- If an `onErrorContinue()` variant is used downstream, revert to the default 'STOP' mode where errors are terminal events upstream

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#onErrorStop](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#onErrorStop)
Key Error Handling Operators in the Flux Class

- The `onErrorStop()` operator
  - If an `onErrorContinue()` variant is used downstream, revert to the default 'STOP' mode where errors are terminal events upstream
  - Returns a Flux that terminates on errors, even if `onErrorContinue()` was used downstream

```
Flux<T> onErrorStop()
```
Key Error Handling Operators in the Flux Class

- The `onErrorStop()` operator
- If an `onErrorContinue()` variant is used downstream, revert to the default 'STOP' mode where errors are terminal events upstream
- It can be used for easier scoping of the `onNext()` failure strategy or to override the inherited strategy in a sub-stream

```java
return Flux
    .fromIterable(denominators)
    .map(denominator -> BigFraction
        .valueOf(..., denominator))
    .onErrorResume(__ -> Flux.empty())
    .onErrorStop()
    .collectList()
...""
Key Error Handling Operators in the Flux Class

- The `onErrorStop()` operator
  - If an `onErrorContinue()` variant is used downstream, revert to the default 'STOP' mode where errors are terminal events upstream.
  - It can be used for easier scoping of the `onNext()` failure strategy or to override the inherited strategy in a sub-stream.
  - It has no effect if `onErrorContinue()` has not been used downstream.

```java
public final Flux<T> onErrorStop()

If an `onErrorContinue(BiConsumer)` variant has been used downstream, reverts to the default 'STOP' mode where errors are terminal events upstream. It can be used for easier scoping of the `onNext` failure strategy or to override the inherited strategy in a sub-stream (for example in a flatMap). It has no effect if `onErrorContinue(BiConsumer)` has not been used downstream.

Returns:
a `Flux` that terminates on errors, even if `onErrorContinue(BiConsumer)` was used downstream.
```
Key Error Handling Operators in the Flux Class

• The `onErrorStop()` operator
  • If an `onErrorContinue()` variant is used downstream, revert to the default 'STOP' mode where errors are terminal events upstream
  • It can be used for easier scoping of the `onNext()` failure strategy or to override the inherited strategy in a sub-stream
  • It has no effect if `onErrorContinue()` has not been used downstream
• RxJava has no direct equivalent
End of Key Error Handling
Operators in the Flux Class