Key Error Handling Operators in the Mono 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 Mono operators
  - Concurrency & scheduler operators
  - Blocking operators
  - Error handling operators
    - These operators gracefully handle errors that occur in a Mono chain
      - e.g., onErrorResume()

Key Error Handling Operators in the Mono Class

Key Error Handling Operators in the Mono Class

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

```
Mono<T> onErrorResume
   (Function<? super Throwable,
             ? extends Mono
                    <? extends T>>
    fallback)
```

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

Key Error Handling Operators in the Mono Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
    - The param is a Function used to choose the fallback, depending on the error

```
Mono<T> onErrorResume
  (Function<? super Throwable,
            ? extends Mono
                  <? extends T>>
   fallback)
```

Interface Function<T,R>

Type Parameters:

T - the type of the input to the function

R - the type of the result of the function

All Known Subinterfaces:

UnaryOperator<T>

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/Function.html

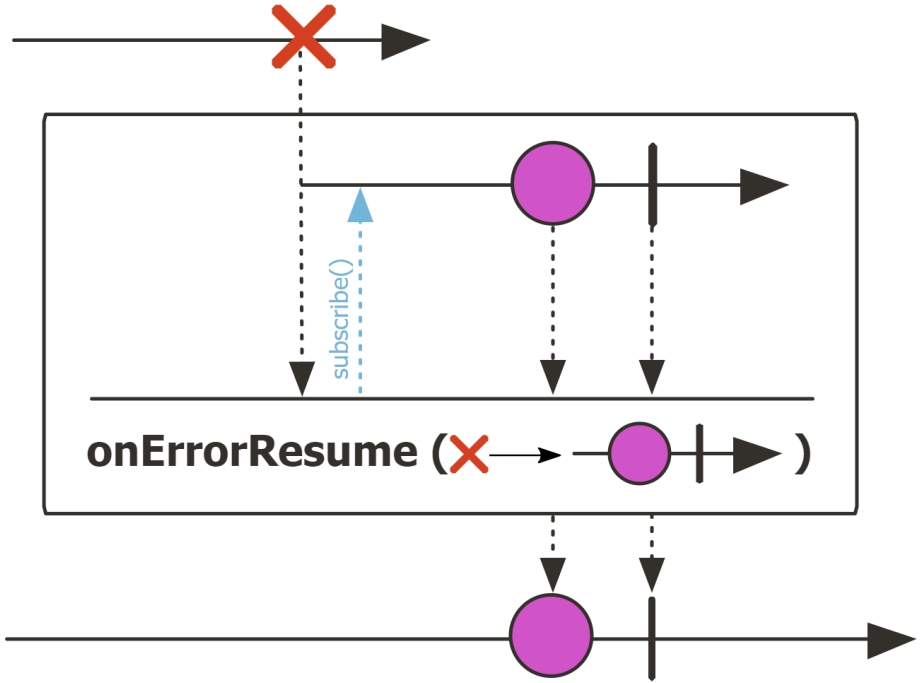
Key Error Handling Operators in the Mono Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
    - The param is a Function used to choose the fallback, depending on the error
    - Returns a Mono falling back upon source on an error

```
Mono<T> onErrorResume
   (Function<? super Throwable,
              ? extends Mono
                     <? extends T>>
    fallback)
```

Key Error Handling Operators in the Mono 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

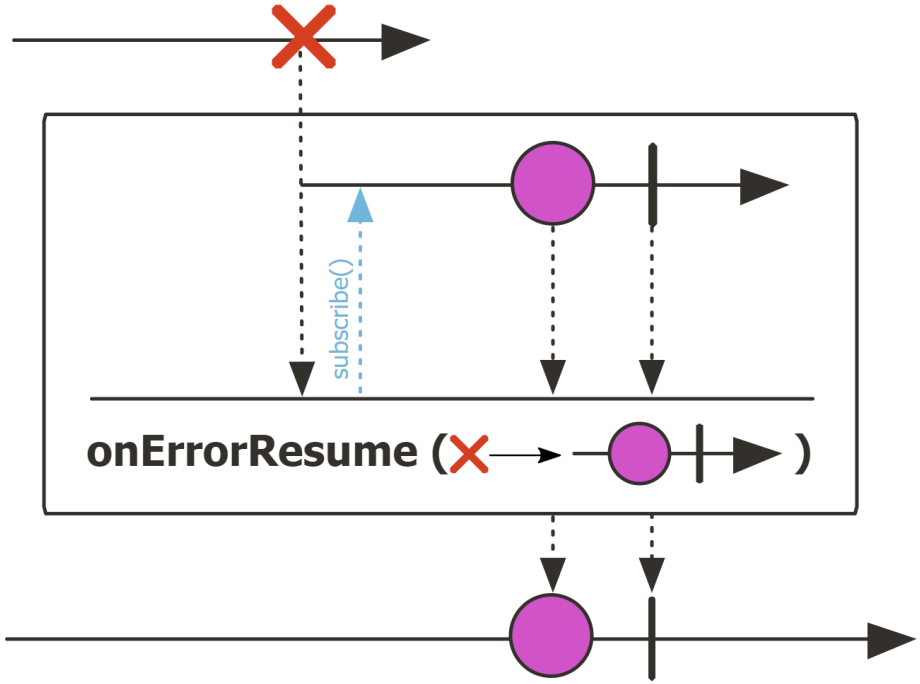


See en.wikipedia.org/wiki/Error_hiding

Key Error Handling Operators in the Mono 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

```
return Mono
   .fromCallable(...)
   .subscribeOn
      (Schedulers.single())
   .onErrorResume(errorHandler)
   .map(multiplyBigFractions);
```

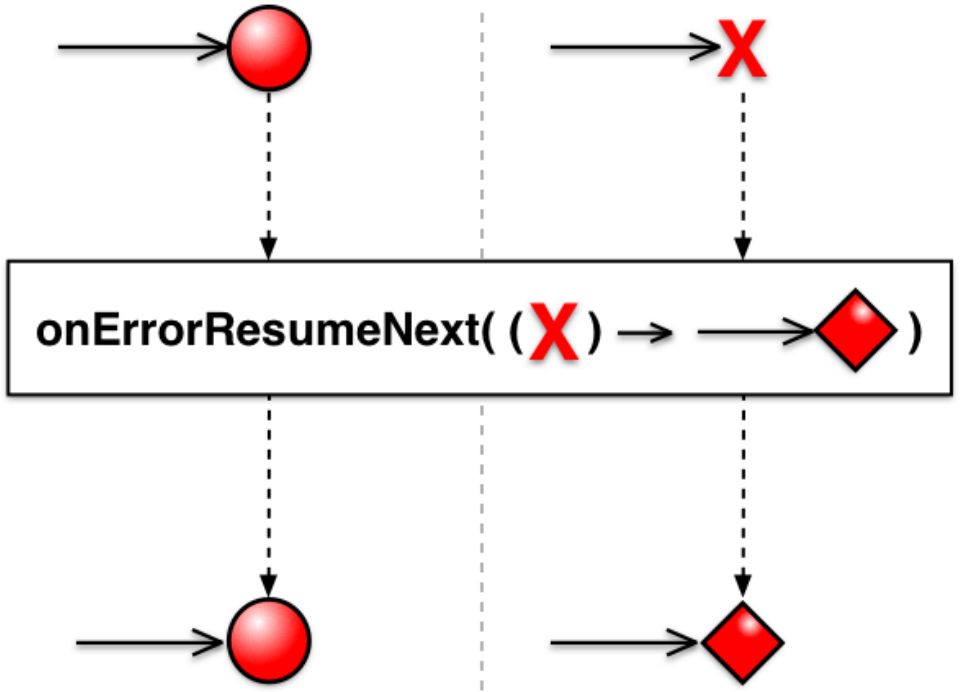


Convert Arithmetic Exception to 0

See Reactive/mono/ex3/src/main/java/MonoEx.java

Key Error Handling Operators in the Mono 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 Single.onErrorResume Next() method works the same



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

Key Error Handling Operators in the Mono 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 Single.onErrorResume Next() method works the same
  - The Java CompletableFuture exceptionally() method is similar

exceptionally

```
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

End of Key Error Handling Operators in the Mono Class