Key Transforming 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
  - Factory method operators
  - Transforming operators
    - Transform the values and/or types emitted by a Mono
      - e.g., map()

Key Transforming Operators in the Mono Class

Key Transforming Operators in the Mono Class

- The map() operator
  - Transform the item emitted by this Mono

```
<R> Mono<R>
map(Function<? super T,
                ? extends R>
        mapper)
```

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

Key Transforming Operators in the Mono Class

- The map() operator
  - Transform the item emitted by this Mono
    - Applies a synchronous function to transform the item

```
<R> Mono<R>
map(Function<? super T,
             ? extends R>
    mapper)
```

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

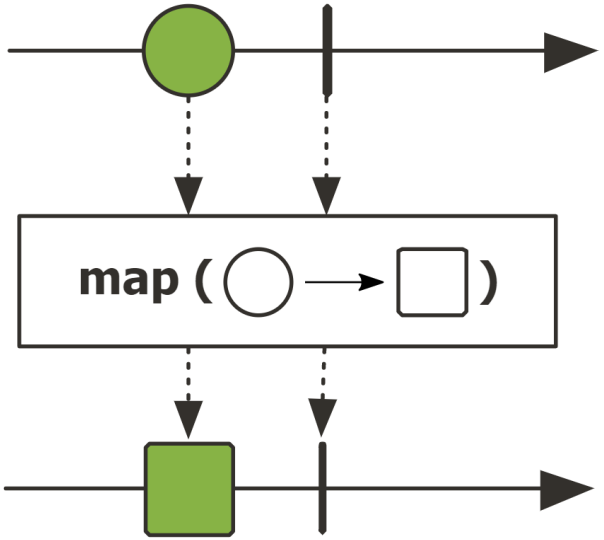
Key Transforming Operators in the Mono Class

- The map() operator
  - Transform the item emitted by this Mono
    - Applies a synchronous function to transform the item
    - Returns a new Mono that emits the results of the transformation

```
<R> Mono<R>
map(Function<? super T,
             ? extends R>
      mapper)
```

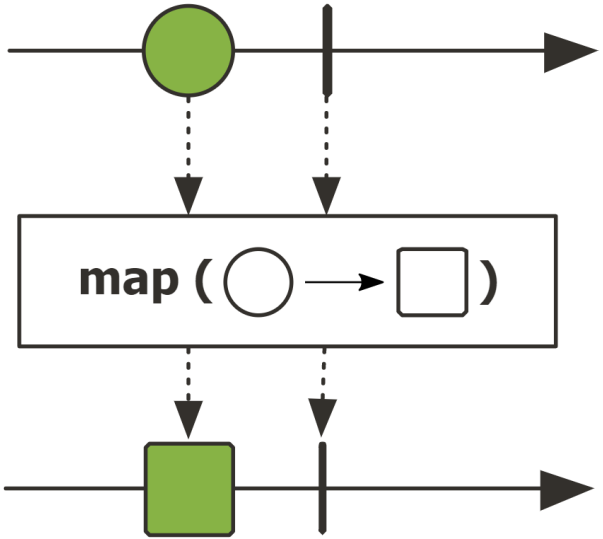
Key Transforming Operators in the Mono Class

- The map() operator
  - Transform the item emitted by this Mono
  - Can transform the value and/or type of elements it processes



Key Transforming Operators in the Mono Class

- The map() operator
  - Transform the item emitted by this Mono
  - Can transform the value and/or type of elements it processes



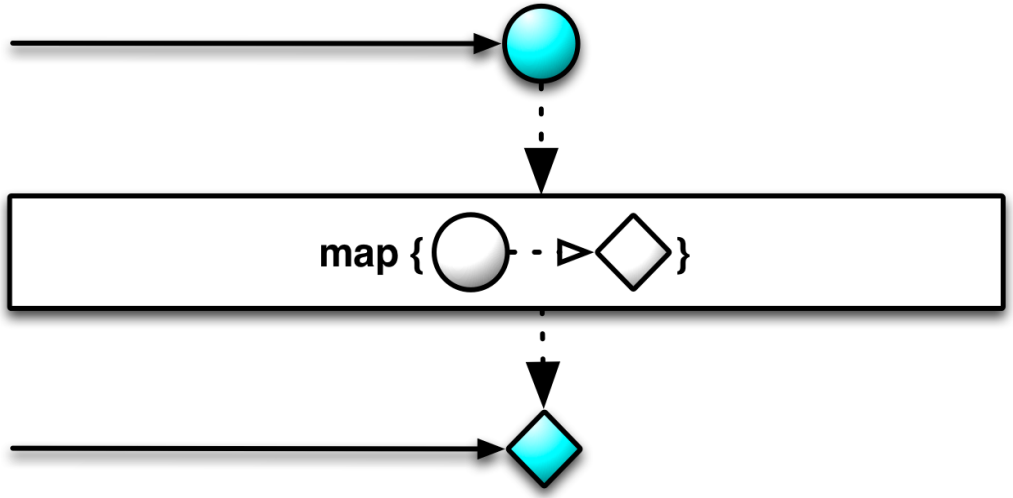

```
Mono
   .just(BigFraction
         .reduce
            (unreducedFraction))
   .map(BigFraction::toMixedString)
   ...
```

Convert a BigFraction into a Java String

See Reactive/mono/ex1/src/main/java/MonoEx.java

Key Transforming Operators in the Mono Class

- The map() operator
  - Transform the item emitted by this Mono
  - Can transform the value and/or type of elements it processes
  - RxJava's Single.map() works the same way



Convert a BigFraction into a Java String

```
Single
   .just(BigFraction
         .reduce
            (unreducedFraction))
   .map(BigFraction::toMixedString)
   ...
```

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

Key Transforming Operators in the Mono Class

- The map() operator
  - Transform the item emitted by this Mono
  - Can transform the value and/or type of elements it processes
  - RxJava's Single.map() works the same way
  - Similar to Java Completable Future thenApply() method

thenApply

```
public <U> CompletableFuture<U> thenApply(Function<? super T,? extends U> fn)
```

Description copied from interface: CompletionStage

Returns a new CompletionStage that, when this stage completes normally, is executed with this stage's result as the argument to the supplied function. See the CompletionStage documentation for rules covering exceptional completion.

Specified by:

thenApply in interface CompletionStage<T>

Type Parameters:

U - the function's return type

Parameters:

fn - the function to use to compute the value of the returned CompletionStage

Returns:

the new CompletionStage

```
CompletableFuture
   .supplyAsync(() -> BigFraction.reduce(unreducedFraction))
   .thenApply(BigFraction::toMixedString)
   ...
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#thenApply

End of Key Transforming Operators in the Mono Class (Part 1)