Applying Key Methods in the Single Class
(Part 3)

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 methods in the Single class & how they are applied in the case studies
  - Case study ex1
  - Case study ex2
  - Case study ex3

```java
Random random = new Random();

Single<BigFraction> m1 = makeBigFraction(random);
Single<BigFraction> m2 = makeBigFraction(random);

return m1.zipWith(m2, BigFraction::add).doOnSuccess(mixedFractionPrinter).then();

return Single.just(BigFractionUtils.makeBigFraction(...).multiply(sBigReducedFrac)).subscribeOn(Schedulers.parallel());
```

Applying Key Methods in the Single Class in ex3
Applying Key Methods in the Single Class in ex3

- ex3 shows how to apply RxJava features asynchronously to perform various Single operations
  - e.g., subscribeOn(), doOnSuccess(), ignoreElement(), just(), zipWith(), & Schedulers.computation()

```java
Random random = new Random();

Single<BigFraction> m1 = makeBigFraction(random);
Single<BigFraction> m2 = makeBigFraction(random);

return m1.zipWith(m2, BigFraction::add).doOnSuccess(mixedFractionPrinter).ignoreElement();

return Single.just(BigFractionUtils.makeBigFraction(...).multiply(sReducedFrac)).doOnSuccess(fractionPrinter).subscribeOn(Schedulers.computation());
```

Applying Key Methods in the Single Class in ex3

• The just() method
  • Create a new Single that emits the specified item

static <T> Single<T> just(T data)

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Single.html#just
Applying Key Methods in the Single Class in ex3

• The just() method
  • Create a new Single that emits the specified item
    • This value is captured at instantiation time & is the value returned for all subscribers
      • i.e., it’s “eager”

static <T> Single<T> just(T data)
Applying Key Methods in the Single Class in ex3

- The just() method
  - Create a new Single that emits the specified item
    - This value is captured at instantiation time & is the value returned for all subscribers
    - In contrast, Single.fromCallable() invokes the callable param at the time of subscription & separately for each subscriber
      - i.e., it’s “lazy”

```java
static <T> Single<T> fromCallable
(Callable<? extends T> supplier)
```
Applying Key Methods in the Single Class in ex3

- The `just()` method
  - Create a new Single that emits the specified item
  - This factory method adapts non-reactive input sources into the reactive model
Applying Key Methods in the Single Class in ex3

- The `just()` method
  - Create a new Single that emits the specified item
  - This factory method adapts non-reactive input sources into the reactive model
  - Project Reactor’s `Mono.just()` works the same way

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#just](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#just)
Applying Key Methods in the Single Class in ex3

- The `zipWith()` method
- Joins two results into a single result after they both emit

```java
<T2, O> Single<O>
zipWith(Single<? extends T2> other,
       BiFunction<? super T, ? super T2, ? extends O>
          combinator)
```

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Single.html#zipWith](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Single.html#zipWith)
Applying Key Methods in the Single Class in ex3

- The zipWith() method
- Joins two results into a single result after they both emit
- Combine the result from this & other Single into another object via a given combinator bifunction

```
<T2, O> Single<O>
zipWith(Single<? extends T2> other,
       BiFunction<? super T,
                  ? super T2,
                  ? extends O
               combinator)
```

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/functions/BiFunction.html](reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/functions/BiFunction.html)
• The `zipWith()` method
• Joins two results into a single result after they both emit
• Combine the result from this & other Single into another object via a given combinator bifunction

```
zipWith( (✓, ✓) ➔ ✓
```

`zipWith()` can transform the type of elements it processes
Applying Key Methods in the Single Class in ex3

- The `zipWith()` method
  - Joins two results into a single result after they both emit
- Project Reactor’s Mono `.zipWith()` works the same

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Single.html#zipWith](http://projectreactor.io/docs/core/release/api/reactor/core/publisher/Single.html#zipWith)
Applying Key Methods in the Single Class in ex3

- The `zipWith()` method
  - Joins two results into a single result after they both emit
  - Project Reactor’s Mono `zipWith()` works the same
- Similar to the Java `CompletableFuture.thenCombine()` method

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#thenCombine
Applying Key Methods in the Single Class in ex3

• The Schedulers.computation() method
  • Hosts a fixed pool of single-threaded Executor Service-based workers that is suitable for parallel work

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/schedulers/Schedulers.html#computation
Applying Key Methods in the Single Class in ex3

- The Schedulers.computation() method
- Hosts a fixed pool of single-threaded Executor Service-based workers that is suitable for parallel work
  - Optimized for fast running non-blocking operations
  - i.e., computation-intensive not I/O-intensive!

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/schedulers/Schedulers.html
Applying Key Methods in the Single Class in ex3

- The Schedulers.computation() method
- Hosts a fixed pool of single-threaded Executor Service-based workers that is suitable for parallel work
  - Optimized for fast running non-blocking operations
- Implemented via daemon threads that won’t prevent the app from exiting even if its work isn’t done

See [www.baeldung.com/java-daemon-thread](http://www.baeldung.com/java-daemon-thread)
Applying Key Methods in the Single Class in ex3

- The `Schedulers.computation()` method
  - Hosts a fixed pool of single-threaded Executor Service-based workers that is suitable for parallel work
- Project Reactor’s `Schedulers.parallel()` method is similar

See [projectreactor.io/docs/core/release/api/reactor/core/scheduler/Schedulers.html#parallel](https://projectreactor.io/docs/core/release/api/reactor/core/scheduler/Schedulers.html#parallel)
Applying Key Methods in the Single Class in ex3

```java
import ...

/**
 * This class shows how to apply Project Reactor features
 * asynchronously and concurrently reduce, multiply, and display
 * BigFractions via various Mono operations, including fromCallable(),
 * subscribeOn(), zipWith(), doOnSuccess(), ambArray(),
 * ignoreElement(), and the parallel thread pool.
 */

generic class SingleEx {
    /**
     * Test asynchronous BigFraction multiplication and addition using
     * zipWith().
     */
    public static Completable testFractionCombine() {
        StringBuffer sb =
            new StringBuffer("\" Calling testFractionCombine()\\n\"");

        // A random number generator.
        Random random = new Random();

        // Create a random BigFraction and reduce/multiply it
        // asynchronously.
        Single<BigFraction> m1 = makeBigFraction(random, sb);

        // Create another random BigFraction and reduce/multiply it
    }
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/Single/ex3
End of Applying Key Methods in the Single Class (Part 3)