Reactor – How to Combine Publishers (Flux/Mono)

In this tutorial, JavaSampleApproach introduces ways to combine two or more Reactor Publishers (Flux/Mono).

Contents [hide]

I. Ways to combine Publishers

0. Initialization

0.1 Reactor installation in Maven

– First, import the BOM by adding the following to **pom.xml**:

```
<dependencyManagement>
 <dependencies>
 <dependency>
 <groupId>io.projectreactor</groupId>
 <artifactId>reactor-bom</artifactId>
 <version>Aluminium-SR1</version>
 <type>pom</type>
 <scope>import</scope>
 </dependency>
 </dependencies>
</dependencyManagement>
```

– Next, add dependency:

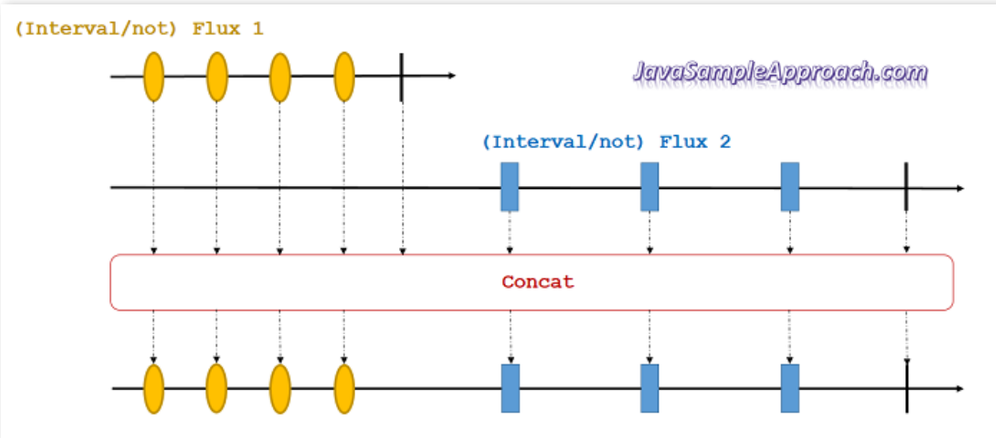
```
<dependencies>
 <dependency>
 <groupId>io.projectreactor</groupId>
 <artifactId>reactor-core</artifactId>
 </dependency>
</dependencies>
```

0.2 Declare & Initialize Publishers

```
Mono<String> mono1 = Mono.just("grokonez.com");
Mono<String> mono2 = Mono.just("|Java Technology");
Mono<String> mono3 = Mono.just("|Spring Framework");

Flux<String> flux1 = Flux.just("{1}", "{2}", "{3}", "{4}");
Flux<String> flux2 = Flux.just("|A|", "|B|", "|C|");
```

```
// FLux emits item each 500ms
Flux<String> intervalFlux1 = Flux
 .interval(Duration.ofMillis(500))
 .zipWith(flux1, (i, string) -> string);

// FLux emits item each 700ms
Flux<String> intervalFlux2 = Flux
 .interval(Duration.ofMillis(700))
 .zipWith(flux2, (i, string) -> string);
```

1. Concat methods



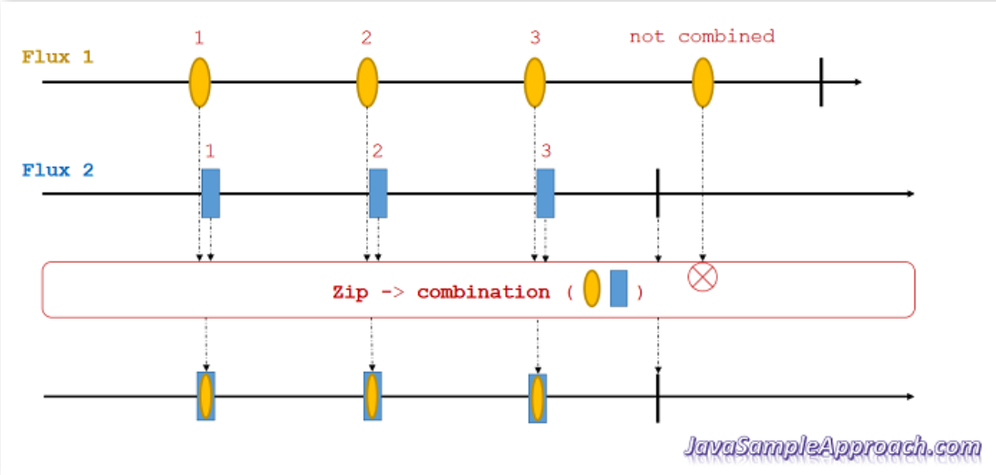
1.1 Flux#concat

```
Flux.concat(mono1, mono3, mono2).subscribe(System.out::print);
// grokonez.com|Spring Framework|Java Technology

Flux.concat(flux2, flux1).subscribe(System.out::print);
// |A||B||C|{1}{2}{3}{4}

Flux.concat(intervalFlux2, flux1).subscribe(System.out::print);
Thread.sleep(3000);
// |A||B||C|{1}{2}{3}{4}
// each of |A|,|B|,|C| emits each 700ms, then {1},{2},{3},{4} emit immediately

Flux.concat(intervalFlux2, intervalFlux1).subscribe(System.out::print);
Thread.sleep(5000);
// |A||B||C|{1}{2}{3}{4}
// each of |A|,|B|,|C| emits each 700ms, then each of {1},{2},{3},{4} emits each 500ms
```

1.2 Flux.concatWith

```
mono1.concatWith(mono2).concatWith(mono3).subscribe(System.out::print);
// grokonez.com|Java Technology|Spring Framework

flux1.concatWith(flux2).subscribe(System.out::print);
// {1}{2}{3}{4}|A||B||C|

intervalFlux1.concatWith(flux2).subscribe(System.out::print);
Thread.sleep(3000);
// {1}{2}{3}{4}|A||B||C|
// each of {1},{2},{3},{4} emits each 700ms, then |A|,|B|,|C| emit immediately

intervalFlux1.concatWith(intervalFlux2).subscribe(System.out::print);
Thread.sleep(5000);
// {1}{2}{3}{4}|A||B||C|
// each of {1},{2},{3},{4} emits each 500ms, then each of |A|,|B|,|C| emits each 700ms
```

2. Zip methods



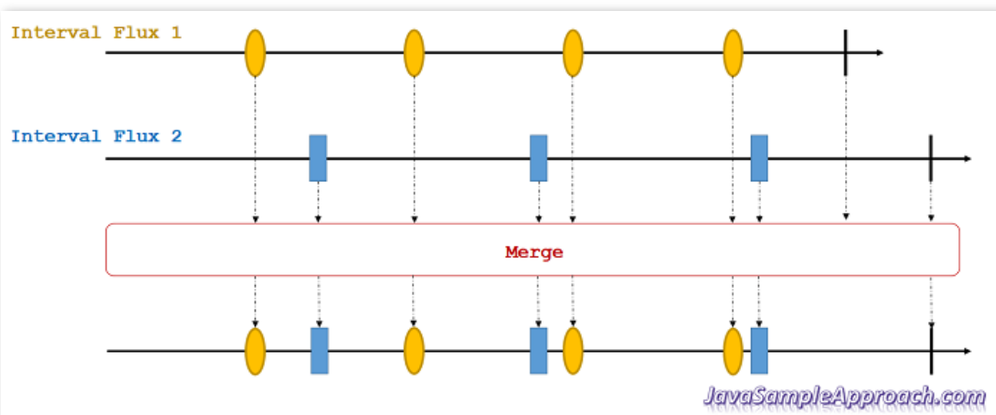
2.1 Flux#zip

```
Flux.zip(flux2, flux1,
 (itemFlux2, itemFlux1) -> "-[" + itemFlux2 + itemFlux1 + "]-")
 .subscribe(System.out::print);

// -[|A|{1}]--[|B|{2}]--[|C|{3}]-
```

2.2 Flux.zipWith

```
flux1.zipWith(flux2,
 (itemFlux1, itemFlux2) -> "-[" + itemFlux1 + itemFlux2 + "]-")
 .subscribe(System.out::print);

// -[{1}|A|]--[{2}|B|]--[{3}|C|]-
```

3. Merge methods



3.1 Flux#merge

```
Flux.merge(intervalFlux1, intervalFlux2).subscribe(System.out::print);
Thread.sleep(3000);
// {1}|A|{2}|B|{3}{4}|C|
```

3.2 Flux.mergeWith

```
intervalFlux1.mergeWith(intervalFlux2).subscribe(System.out::print);
Thread.sleep(3000);
// {1}|A|{2}|B|{3}{4}|C|
```

II. Source Code

1. Technology

- Java 8
- Maven 3.6.1
- Reactor Core 3.0.4, with the Aluminium release train.

2. Code

```java
package com.javasampleapproach.reactor.mergepublishers;

import java.time.Duration;

import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

public class MainApp {

 public static void main(String[] args) throws InterruptedException {

 Mono<String> mono1 = Mono.just("grokonez.com");
 Mono<String> mono2 = Mono.just("|Java Technology");
 Mono<String> mono3 = Mono.just("|Spring Framework");

 System.out.println("=== Flux.concat(mono1, mono3, mono2) ===");
 Flux.concat(mono1, mono3, mono2).subscribe(System.out::print);

 System.out.println("\n=== combine the value of mono1 then mono2 then mono3 ===");
 mono1.concatWith(mono2).concatWith(mono3).subscribe(System.out::print);

 Flux<String> flux1 = Flux.just("{1}", "{2}", "{3}", "{4}");
 Flux<String> flux2 = Flux.just("|A|", "|B|", "|C|");

 System.out.println("\n=== Flux.zip(flux2, flux1, combination) ===");
 Flux.zip(flux2, flux1,
 (itemFlux2, itemFlux1) -> "-[" + itemFlux2 + itemFlux1 + "]-")
 .subscribe(System.out::print);

 System.out.println("\n=== flux1 values zip with flux2 values ===");
 flux1.zipWith(flux2,
 (itemFlux1, itemFlux2) -> "-[" + itemFlux1 + itemFlux2 + "]-")
 .subscribe(System.out::print);

 Flux<String> intervalFlux1 = Flux
 .interval(Duration.ofMillis(500))
 .zipWith(flux1, (i, string) -> string);

 Flux<String> intervalFlux2 = Flux
 .interval(Duration.ofMillis(700))
 .zipWith(flux2, (i, string) -> string);

 System.out.println("\n=== Flux.concat(flux2, flux1) ===");
 Flux.concat(flux2, flux1).subscribe(System.out::print);

 System.out.println("\n=== flux1 values and then flux2 values ===");
 flux1.concatWith(flux2).subscribe(System.out::print);

 System.out.println("\n=== Flux.concat(intervalFlux2, flux1) ===");
 Flux.concat(intervalFlux2, flux1).subscribe(System.out::print);
 Thread.sleep(3000);

 System.out.println("\n=== intervalFlux1 values and then flux2 values ===");
 intervalFlux1.concatWith(flux2).subscribe(System.out::print);
 Thread.sleep(3000);

 System.out.println("\n=== Flux.concat(intervalFlux2, intervalFlux1) ===");
 Flux.concat(intervalFlux2, intervalFlux1).subscribe(System.out::print);
 Thread.sleep(5000);

 System.out.println("\n=== intervalFlux1 values and then intervalFlux2 values ===");
 intervalFlux1.concatWith(intervalFlux2).subscribe(System.out::print);
 Thread.sleep(5000);

 System.out.println("\n=== Flux.merge(intervalFlux1, intervalFlux2) ===");
 Flux.merge(intervalFlux1, intervalFlux2).subscribe(System.out::print);
 Thread.sleep(3000);

 System.out.println("\n=== interleave flux1 values with flux2 values ===");
 intervalFlux1.mergeWith(intervalFlux2).subscribe(System.out::print);
 Thread.sleep(3000);

 }
}
```

3. Results

```
=== Flux.concat(mono1, mono3, mono2) ===
grokonez.com|Spring Framework|Java Technology
```

```
=== combine the value of mono1 then mono2 then mono3 ===
grokonez.com|Java Technology|Spring Framework
=== Flux.zip(flux2, flux1, combination) ===
-[|A|{1}]--[|B|{2}]--[|C|{3}]-
=== flux1 values zip with flux2 values ===
-[{1}|A|]--[{2}|B|]--[{3}|C|]-
=== Flux.concat(flux2, flux1) ===
|A||B||C|{1}{2}{3}{4}
=== flux1 values and then flux2 values ===
{1}{2}{3}{4}|A||B||C|
=== Flux.concat(intervalFlux2, flux1) ===
|A||B||C|{1}{2}{3}{4}
=== intervalFlux1 values and then flux2 values ===
{1}{2}{3}{4}|A||B||C|
=== Flux.concat(intervalFlux2, intervalFlux1) ===
|A||B||C|{1}{2}{3}{4}
=== intervalFlux1 values and then intervalFlux2 values ===
{1}{2}{3}{4}|A||B||C|
=== Flux.merge(intervalFlux1, intervalFlux2) ===
{1}|A|{2}|B|{3}{4}|C|
=== interleave flux1 values with flux2 values ===
{1}|A|{2}|B|{3}{4}|C|
```

By grokonez | June 29, 2017.
Last updated on **March 13, 2018**.

Related Posts

- Angular 6 NGXS example – Angular State Management
- Angular 6 NgRx Store example – Angular State Management
- Angular + Spring WebFlux + Spring Data Reactive Cassandra example | Full-Reactive Angular Http Client – Spring Boot RestApi Server
- Angular 4 + Spring WebFlux + Spring Data Reactive MongoDB example | Full-Reactive Angular 4 Http Client – Spring Boot RestApi Server
- Introduction to RxJS – Extensions for JavaScript Reactive Streams
- SpringData Reactive Cassandra Repositories | SpringBoot
- SpringBoot WebFlux Annotation-based RestAPIs
- Reactor – Handle Error
- Reactor – How to convert Flux into List, Map
- Reactor – How to Create Flux (Publisher) with Interval

Post Tags

reactive programming | reactor

1 thought on “Reactor – How to Combine Publishers (Flux/Mono)”

Bulat

February 12, 2019 at 9:38 am

Thank you. Fast examples in action

FOLLOW US

ABOUT US

We are passionate engineers in software development by Java Technology & Spring Framework. We believe that creating little good thing with specific orientation everyday can make great influence on the world someday.