7

Reactive Programming

Java Backend Developer I

Objetivos

Comprender los conceptos:

- Reactive Programming
- Reactive Stream
  - Manifesto
  - Specification
  - Implementations
- Operadores reactivos básicos
- Flowables y Backpressure

Agenda

Revisión de los siguientes conceptos:

- Reactive Programming
- Reactive Stream
  - Manifesto
  - Specification
  - Implementations
- Operadores reactivos básicos
- Flowables y Backpressure

Reactive Programming

Definición

La programación Reactiva es un concepto nuevo que actualmente va ganando muchos adeptos. Esta basado en el patrón Observer, tomando además las mejores practicas del patrón Iterator y la programación funcional.

La programación Reactiva se orienta al manejo de streams de datos asíncronos y la propagación del cambio.

Stream

Un stream es un flujo de datos y tradicionalmente los streams han estado asociados a operaciones del tipo I/O como lectura/escritura de ficheros o querys a base de datos.

Reactive Programming

Introducción

Una de las máximas en la programación reactiva es que "todo es un stream" por lo que, cualquier flujo de información será tratada como un stream. Eventos del mouse, Arrays, rangos de números, etc. Todo será un stream.

En la programación reactiva los streams están representados por "secuencias observables" o simplemente Observables, por lo que todo, absolutamente todo es un Observable, lo que lógicamente nos lleva al patrón Observer.

Reactive Programming

Patron Observer:

El patrón Observer define un productor de información, nuestro stream y que en la programación reactiva se representa por una secuencia Observable o simplemente Observable y un consumidor de la misma, que sería el Observer. En la programación reactiva el Observable es nuestro stream el cual nos debe servir para prácticamente todo: eventos del ratón, rangos de números, etc.

Reactive Programming

Patron Iterator:

Es otro de los patrones en los que se inspira la programación reactiva, el cual nos permite iterar contenedores de información como, por ejemplo, un Array, sin exponer su representación interna. Para ello se define un iterator -next-, que será el encargado de recorrer el contenedor de información, manteniendo el cursor o índice con la posición del último valor dado.

Reactive Programming

Conclusión:

La programación reactiva es un paradigma de programación que trata con flujos de datos asincrónicos (secuencias de eventos) y la propagación específica del cambio, lo que significa que implementa modificaciones en el entorno de ejecución (contexto) en un cierto orden.

Reactive Stream Manifesto

Introducción

Reactive Streams es una iniciativa para proporcionar un estándar para el procesamiento de flujo asíncrono con back pressure no bloqueante. Esto abarca esfuerzos dirigidos a entornos de tiempo de ejecución (JVM y JavaScript), así como a protocolos de red.

El manifiesto reactivo nos permite conocer dicha iniciativa de estandarización.

Reactive Stream Manifesto

Características

Responsivos: aseguran la calidad del servicio cumpliendo unos tiempos de respuesta establecidos. Además define límites en dichos tiempos de respuesta, de forma que los problemas pueden ser detectados rápidamente y tratados de forma efectiva

Resilientes: se mantienen responsivos incluso cuando se enfrentan a situaciones de error.

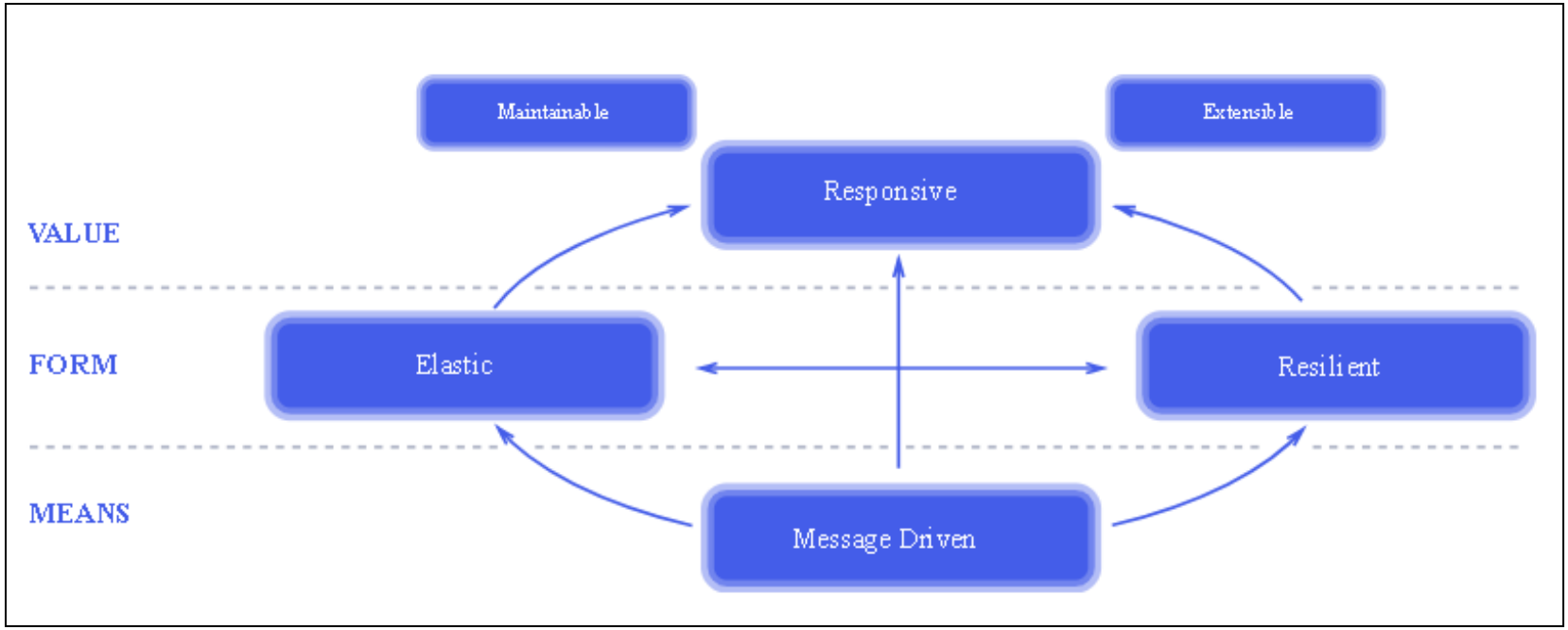
Reactive Stream Manifesto

Características

Elásticos: se mantienen responsivos incluso ante aumentos en la carga de trabajo.

Orientados a mensajes: minimizan el acoplamiento entre componentes al establecer interacciones basadas en el intercambio de mensajes de manera asíncrona. Afectando (de manera positiva) todo el sistema.

Reactive Stream Manifesto



Reactive Stream Manifesto

Conclusión

Los sistemas construidos como sistemas reactivos son más flexibles, poco acoplados y escalables . Esto los hace más fáciles de desarrollar y susceptibles de cambio. Ellos son significativamente más tolerantes a fallos y cuando falla hace aparecer cuando cumplan con elegancia en lugar de desastre. Los sistemas reactivos son altamente receptivos y brindan a los usuarios comentarios interactivos efectivos.

Reactive Stream Specification

La API para Reactive Stream consta de los siguientes componentes los cuales las implementaciones de Reactive Stream deben proporcionar:

- Publisher
- Subscriber
- Subscription
- Processor

Reactive Stream Specification

Publisher

Es un proveedor de un número potencialmente ilimitado de elementos secuenciados, que los publica de acuerdo con la demanda recibida de sus suscriptores.

Subscriber

Un suscriptor informa a un publisher que está dispuesto a aceptar un número dado de items (solicita un número dado de items), y si hay items disponibles, el publisher empuja (pushes) el número máximo de items por cobrar al suscriptor.

Es importante tener en cuenta que esta es una comunicación bidireccional, donde el suscriptor informa al editor cuántos items está dispuesto a manejar y el publisher envía esa cantidad de items al suscriptor.

Reactive Stream Specification

Subscription

La conexión bidireccional entre un publisher y un suscriptor se denomina suscripción. Esta suscripción vincula a un único publisher a un solo suscriptor (relación uno a uno) y puede ser unicast o multicast. Del mismo modo, un único publisher puede tener suscriptores múltiples suscritos, pero un solo suscriptor solo puede estar suscrito a un solo productor (un publisher puede tener muchos suscriptores, pero un suscriptor puede suscribirse a un máximo de un publisher).

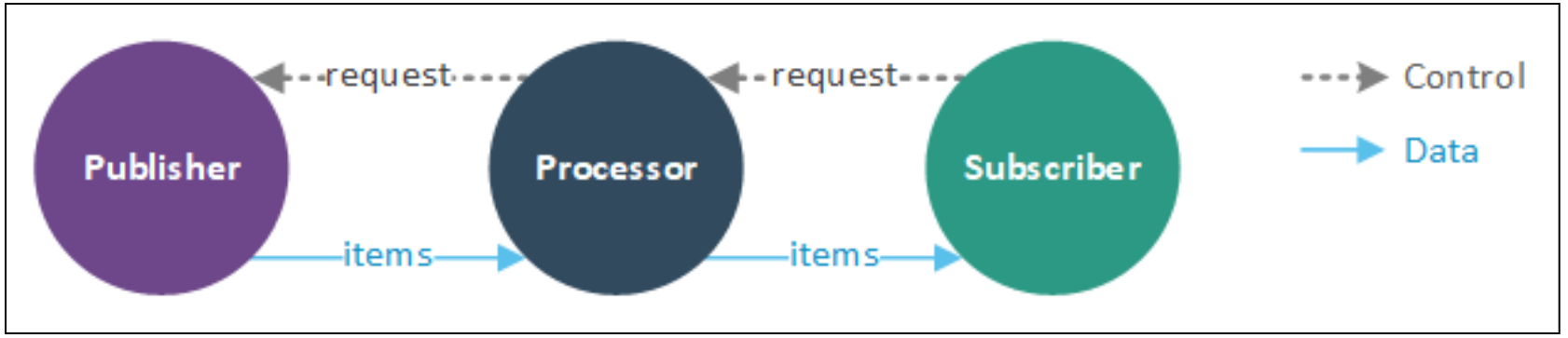
Reactive Stream Specification

Processor

Si una entidad es tanto un publisher como un suscriptor, se llama procesador. Un procesador comúnmente actúa como intermediario entre otro publisher y suscriptor (cualquiera de los cuales puede ser otro procesador), realizando alguna transformación en el flujo de datos. Por ejemplo, se puede crear un procesador que filtre elementos que coincidan con algunos criterios antes de pasarlos a su suscriptor. Una representación visual de un procesador se ilustra en la figura a continuación.

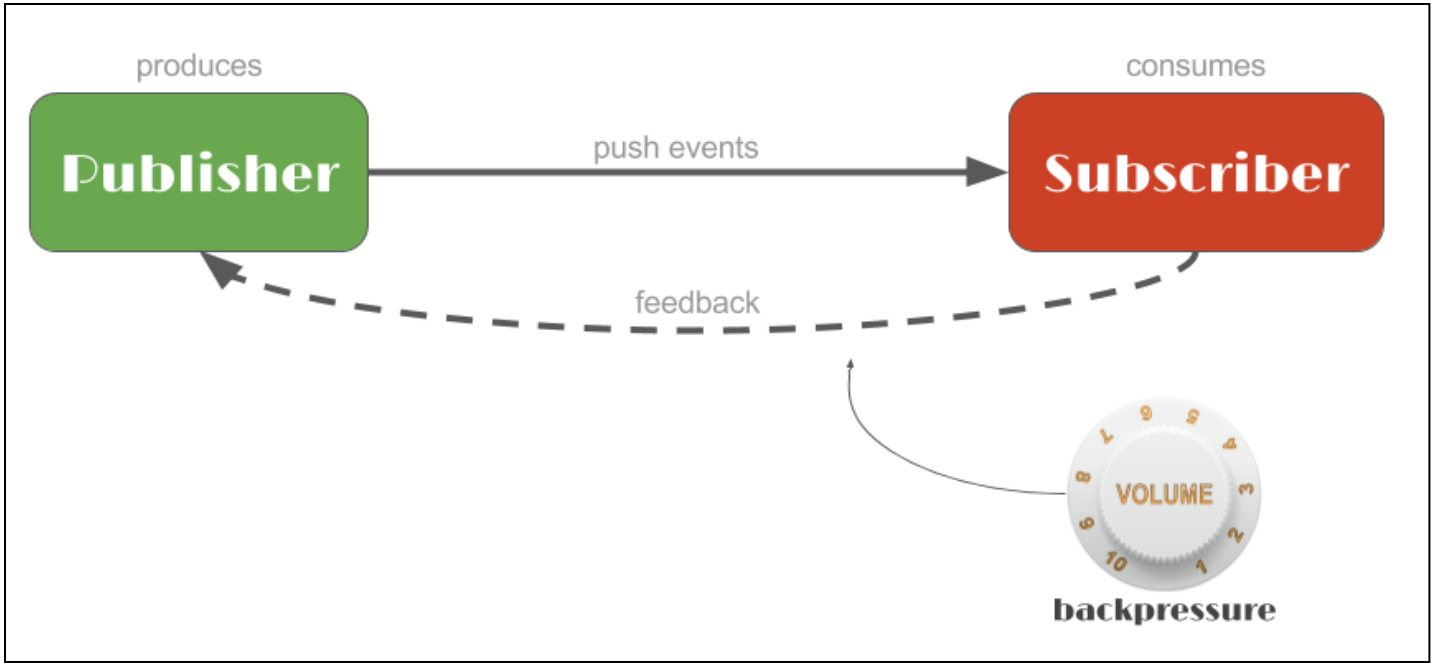
Reactive Stream Specification

Diagrama



Con una comprensión fundamental de cómo operan los flujos reactivos, podemos transformar estos conceptos en el ámbito de Java codificándolos en interfaces.

Reactive Stream Specification



Reactive Stream Implementations

Tecnologías

En JAVA

ReactiveX es una API que facilita el manejo de flujos de datos y eventos, a partir de una combinación de **el patrón Observer**, el patrón **Iterator**, y características de la **Programación Funcional**.

El manejo de **datos en tiempo real** es una tarea común en el desarrollo de aplicaciones.

Por lo tanto, tener una manera eficiente y limpia de lidiar con esta tarea es muy importante.

ReactiveX (mediante el uso de **Observables** y operadores) nos ofrece una API flexible para **crear y actuar sobre los flujos de datos**.

Además, simplifica la programación asíncrona, como la creación de hilos y los problemas de concurrencia.

Reactive Stream Implementations

Tecnologias

En JAVA

Es así que las dos librerías recomendadas por la comunidad son:

RxJava esta librería, y su versión 1.x fueron las pioneras en el desarrollo reactivo Java. Se encuentran completamente integradas en Frameworks como Spring MVC, Spring Cloud y Netflix OSS.

Project Reactor Fue concebida con la implicación del equipo responsable de RxJava 2, por lo que comparten gran parte de la base arquitectónica. Su principal ventaja es que al ser parte de Pivotal ha sido la elegida como fundación del futuro Spring 5 WebFlux Framework. Este API introduce los tipos Flux y Mono como implementaciones de Publisher, los cuales generan series de 0…N y 0…1 elementos respectivamente.

Reactive Stream Implementations

Tecnologias

Akka Streams (version 2.6.8)

Akka es un conjunto de herramientas de código abierto y tiempo de ejecución que simplifica la construcción de aplicaciones concurrentes y distribuidas en la JVM.

- Para mas información ver Activator template y la documentación.

Ratpack (version 1.8.0)

En general, la transmisión en Ratpack se basa en el estándar emergente API Reactive Streams.

Reactive Streams es una iniciativa para proporcionar un estándar para el procesamiento de flujo asíncrono con backpressure sin bloqueo en la JVM.

- Para mas información ver capitulo de “Streams” del manual.

Reactive Stream Implementations

Tecnologias

Slick (version 3.0.0)

Basado en Scala, los elementos se transmiten directamente desde el conjunto de resultados a través de un Reactive Streams Publisher, que puede ser procesado y consumido por ejemplo por Akka.

- Para mas información ver "Streaming" sección del manual.

Vert.x 3 (version 3.9.2)

- Eclipse Vert.x es impulsado por eventos y sin bloqueo. Esto significa que su aplicación puede manejar una gran cantidad de concurrencia utilizando una pequeña cantidad de hilos del núcleo. Vert.x permite que su aplicación escale con un hardware mínimo.
- Para mas información revisar la página de vertx.io

Operadores Reactivos Básicos

Introducción

Cada implementación específica de lenguaje de ReactiveX implementa un conjunto de operadores. Aunque hay mucha superposición entre las implementaciones, también hay algunos operadores que solo se implementan en ciertas implementaciones.

Además, cada implementación tiende a nombrar a sus operadores para que se parezcan a los de métodos similares que ya son familiares desde otros contextos en ese lenguaje.

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Crear observables

Operadores que originan nuevos Observables.

Create cree un Observable desde cero llamando a los métodos de observación mediante programación

Defer no crea el Observable hasta que el observador se suscriba, y cree un Observable nuevo para cada observador

Empty / Never / Throw crea observables que tienen un comportamiento muy preciso y limitado

From convierte algún otro objeto o estructura de datos en un observable

Interval crea un Observable que emite una secuencia de enteros separados por un intervalo de tiempo particular

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Crear observables

Operadores que originan nuevos Observables.

Just convierta un objeto o un conjunto de objetos en un Observable que emita ese o esos objetos

Range crea un Observable que emite un rango de enteros secuenciales

Repeat crea un Observable que emite un elemento particular o una secuencia de elementos repetidamente

Start crea un Observable que emite el valor de retorno de una función

Timer crea un Observable que emite un solo elemento después de un retraso determinado

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Transformando observables

Operadores que transforman elementos emitidos por un Observable.

Buffer reúne periódicamente elementos de un Observable en paquetes y emite estos paquetes en lugar de emitir los elementos uno a la vez

FlatMap transforma los elementos emitidos por un observable en observables, luego aplana las emisiones de esos en un solo observable

GroupBy divida un Observable en un conjunto de Observables que emitan un grupo diferente de elementos del Observable original, organizados por clave

Map transforma los elementos emitidos por un Observable aplicando una función a cada elemento

Escanear: aplique una función a cada elemento emitido por un

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Transformando observables

Operadores que transforman elementos emitidos por un Observable.

Scan aplique una función a cada elemento emitido por un Observable, secuencialmente, y emita cada valor sucesivo

Window subdividir periódicamente elementos de un observable en ventanas observables y emitir estas ventanas en lugar de emitir los elementos de uno en uno

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Filtering Observables

Operadores que emiten elementos de forma selectiva desde una fuente Observable.

Debounce solo emite un elemento de un Observable si ha transcurrido un período de tiempo determinado sin que emita otro elemento

Distinct suprime los elementos duplicados emitidos por un observable

ElementAt emite solo el elemento n emitido por un observable

Filter emite solo aquellos elementos de un Observable que pasan una prueba de predicado.

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Filtering Observables

Operadores que emiten elementos de forma selectiva desde una fuente Observable.

First emitir solo el primer elemento, o el primer elemento que cumpla una condición, de un observable

IgnoreElements no emite ningún elemento de un Observable, sino que refleja su notificación de finalización

Last emitir solo el último elemento emitido por un observable

Sample emite el elemento más reciente emitido por un observable dentro de intervalos de tiempo periódicos

Skip suprime los primeros n elementos emitidos por un observable

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Filtering Observables

Operadores que emiten elementos de forma selectiva desde una fuente Observable.

SkipLast suprime los últimos n elementos emitidos por un observable

Take emitir solo los primeros n elementos emitidos por un observable

TakeLast emite solo los últimos n elementos emitidos por un observable

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Combining Observables

Operadores que trabajan con múltiples Observables de origen para crear un solo Observable

And/Then/When combine conjuntos de elementos emitidos por dos o más Observables por medio de intermediarios de Patrón y Plan

CombineLatest cuando un elemento es emitido por cualquiera de los dos Observables, combine el último elemento emitido por cada Observable a través de una función específica y emita elementos según los resultados de esta función

Join combinar elementos emitidos por dos Observables siempre que se emita un elemento de un Observable durante una ventana de tiempo definida de acuerdo con un elemento emitido por el otro Observable.

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Combining Observables

Operadores que trabajan con múltiples Observables de origen para crear un solo Observable

Merge combine múltiples Observables en uno fusionando sus emisiones

StartWith emite una secuencia específica de elementos antes de comenzar a emitir los elementos desde la fuente Observable

Switch convierta un Observable que emite Observables en un solo Observable que emite los elementos emitidos por el Observable emitido más recientemente.

Zip combine las emisiones de múltiples Observables a través de una función específica y emite elementos únicos para cada combinación en función de los resultados de esta función.

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Error Handling Operators

Operadores que ayudan a recuperarse de las notificaciones de error de un observable

Catch recuperarse de una notificación onError al continuar la secuencia sin error

Retry si un Observable de origen envía una notificación onError, vuelva a suscribirse con la esperanza de que se complete sin error

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Observable Utility Operators

Una caja de herramientas de operadores útiles para trabajar con Observables

Delay desplaza las emisiones de un observable hacia adelante en el tiempo en una cantidad particular

Do registre una acción para realizar una variedad de eventos de ciclo de vida observables

Materialize/Dematerialize representa los elementos emitidos y las notificaciones enviadas como elementos emitidos, o revierta este proceso

ObserveOn especifique el planificador en el que un observador observará este Observable

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Observable Utility Operators

Una caja de herramientas de operadores útiles para trabajar con Observables

Serialize obligar a un Observable a hacer llamadas en serie y a portarse bien

Suscribe operar sobre las emisiones y notificaciones de un observable

SubscribeOn especifique el planificador que un Observable debe usar cuando está suscrito

TimeInterval convierte un observable que emite elementos en uno que emite indicaciones del tiempo transcurrido entre esas emisiones

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Observable Utility Operators

Una caja de herramientas de operadores útiles para trabajar con Observables

TimeOut refleje la fuente Observable, pero emita una notificación de error si transcurre un período de tiempo determinado sin ningún elemento emitido

TimeStamp adjunte una marca de tiempo a cada elemento emitido por un observable

Using cree un recurso desechable que tenga la misma vida útil que el Observable

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Conditional and Boolean Operators

Operadores que evalúan uno o más Observables o elementos emitidos por Observables

All determine si todos los elementos emitidos por un observable cumplen con algunos criterios

Amb dado dos o más Observables de origen, emite todos los elementos desde solo el primero de estos Observables para emitir un elemento

Contains determina si un Observable emite un artículo en particular o no

DefaultIfEmpty emite elementos del Observable de origen, o un elemento predeterminado si el Observable de origen no emite nada

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Conditional and Boolean Operators

Operadores que evalúan uno o más Observables o elementos emitidos por Observables

SequenceEqual determina si dos Observables emiten la misma secuencia de elementos

SkipUntil descarta los elementos emitidos por un observable hasta que un segundo observable emita un elemento

SkipWhile descarta los elementos emitidos por un observable hasta que una condición específica se vuelva falsa

TakeUntil descarta los elementos emitidos por un Observable después de que un segundo Observable emite un elemento o termina

TakeWhile descarta los elementos emitidos por un observable después de que una condición específica se vuelva falsa

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Mathematical and Aggregate Operators

Operadores que operan en toda la secuencia de elementos emitidos por un observable

Average calcula el promedio de números emitidos por un observable y emite este promedio

Concat emite las emisiones de dos o más observables sin intercalarlas

Count cuenta el número de elementos emitidos por la fuente Observable y emite solo este valor

Max determina y emite el elemento de valor máximo emitido por un observable

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Mathematical and Aggregate Operators

Operadores que operan en toda la secuencia de elementos emitidos por un observable

Min determina y emite el elemento de valor mínimo emitido por un observable

Reduce aplica una función a cada elemento emitido por un observable, secuencialmente y emite el valor final

Sum calcule la suma de números emitidos por un observable y emita esta suma

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Backpressure Operators

Estrategias para hacer frente a los Observables que producen artículos más rápidamente de lo que sus observadores los consumen.

Por Categoria: Operators to Convert Observables

To convertir un observable en otro objeto o estructura de datos

Operadores Reactivos Básicos

Operadores de ReactiveX

Por Categoria: Connectable Observable Operators

Observables especializados que tienen una dinámica de suscripción controlada con mayor precisión

Connect indique a un observable conectable que comience a emitir elementos a sus suscriptores

Publish convierte un observable ordinario en un observable conectable

RefCount haga que un observable conectable se comporte como un observable ordinario

Replay asegúrese de que todos los observadores vean la misma secuencia de elementos emitidos, incluso si se suscriben después de que el Observable haya comenzado a emitir elementos

Flowables y Backpressure

Flowables

El funcionamiento de los Flowables es muy similar al de los Observables, pero con una diferencia importante: los Flowables sólo envían tantos ítems como solicite el Observer. Si tienes un Observable que emite más ítems de los que su Observer asignado puede consumir, quizás quieras considerar cambiarlo por un Flowable.

Backpressure

En el mundo del software, la "contrapresión" es una analogía tomada de la dinámica de fluidos, como en el escape de automóviles y las tuberías de la casa.

Flowables y Backpressure

Backpressure

Cuando un componente está luchando por mantenerse al día, el sistema en su conjunto debe responder de manera sensata. Es inaceptable que el componente bajo estrés falle de forma catastrófica o deje caer mensajes de forma incontrolada. Dado que no puede hacer frente y no puede fallar, debe comunicar el hecho de que está bajo tensión para los componentes aguas arriba y así reducir la carga. Esta contrapresión es un mecanismo de retroalimentación importante que permite a los sistemas responder con gracia a la carga en lugar de colapsar debajo de ella.

Flowables y Backpressure

Backpressure

Cuando tiene un observable que emite artículos tan rápido que el consumidor no puede mantenerse al día con el flujo que conduce a la existencia de artículos emitidos pero no consumidos.

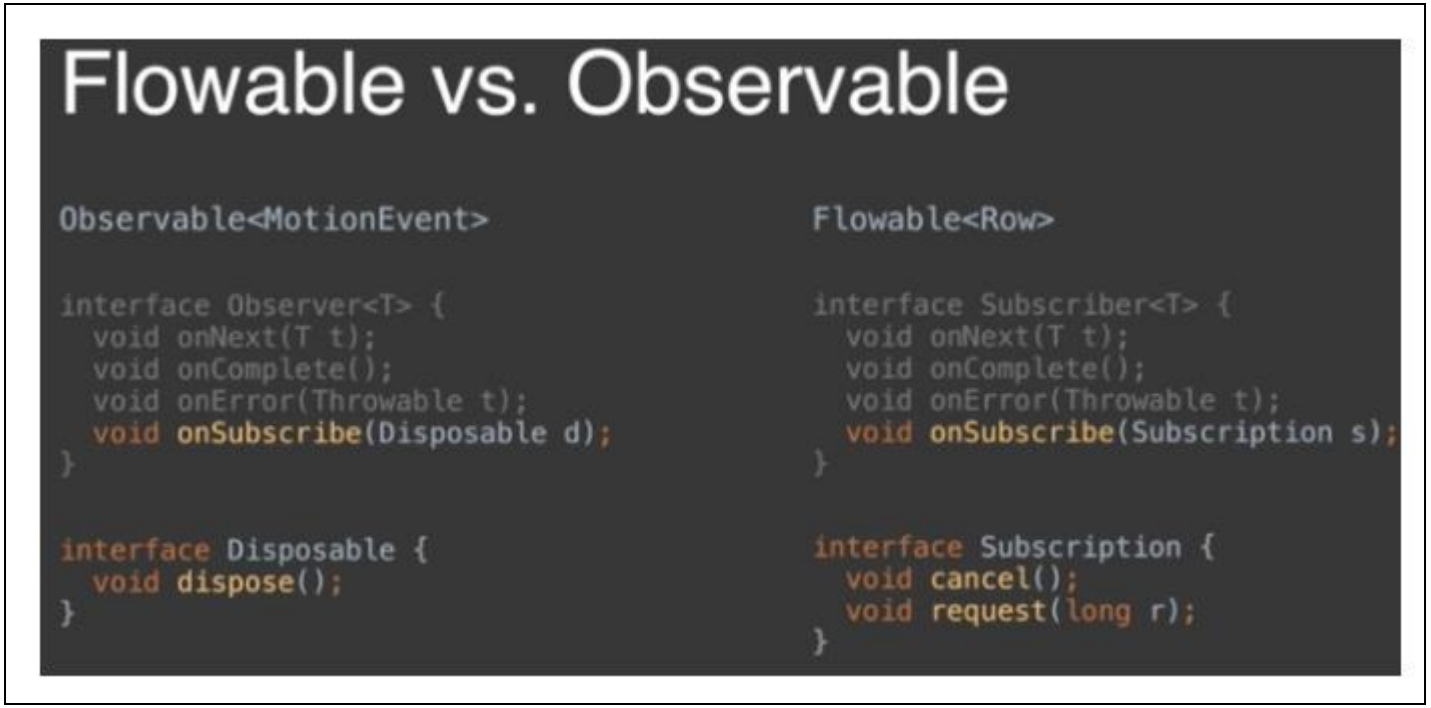
La estrategia de contrapresión se ocupa de cómo se gestionan y controlan los elementos no consumidos, que son emitidos por observables pero no consumidos por los suscriptores.

Dado que requiere recursos del sistema para manejar la contrapresión, debe elegir la estrategia de contrapresión adecuada que se adapte a sus necesidades.

Flowables y Backpressure

Diferencia

Flowable tiene backpressure debido al método request de suscripción, mientras que Observable, no tiene backpressure. Fluible también tiene stream reactiva.



Lecturas adicionales

Para obtener información adicional, puede consultar los siguientes enlaces:

https://reactivemanifesto.org/

http://reactivex.io/intro.html

http://introtorx.com/

https://projectreactor.io/

https://ratpack.io/

http://scala-slick.org/doc/3.0.0/

Resumen

En este capítulo, usted aprendió los siguientes conceptos:

- Reactive Programming
- Reactive Stream
  - Manifesto
  - Specification
  - Implementations
- Operadores reactivos básicos
- Flowables y Backpressure