Fabricas de Flux y Mono

Métodos de fábrica de Flux

- Flux.just(T... valores)

Descripción: Crea un Flux que emite los elementos proporcionados en el orden dado y luego finaliza.

Cuándo usarlo: Ideal para prototipos rápidos o para emitir un conjunto fijo y pequeño de valores (por ejemplo, para pruebas o ejemplos simples).

- Flux.from(Publisher<? extends T> publisher)

Descripción: Envuelve cualquier Publisher existente en un Flux.

Cuándo usarlo: Si ya tienes un Publisher (por ejemplo, un Mono o una fuente Reactiva de otra librería), y quieres usar las operaciones de Flux.

- Flux.fromIterable(Iterable<? extends T> iterable)

Descripción: Crea un Flux que emite cada elemento de un Iterable y luego completa.

Cuándo usarlo: Cuando quieras transformar de manera reactiva una colección o lista en un Flux.

Flux.fromStream(Stream<? extends T> stream)

Descripción: Genera un Flux a partir de un Stream (API de Java 8+).

Cuándo usarlo: Para procesar secuencias con la API de streams de Java, pero en modo reactivo.

- Flux.range(int start, int count)

Descripción: Emite secuencialmente una serie de números enteros, iniciando en start y emitiendo count elementos.

Cuándo usarlo: Si necesitas generar rápidamente una secuencia de enteros en un rango específico (p.ej. para pruebas, demos o cálculos).

- Flux.empty()

Descripción: Crea un Flux que no emite ningún valor y finaliza inmediatamente.

Cuándo usarlo: Para representar un flujo vacío (p.ej., cuando no hay datos que proveer o en pruebas unitarias).

- Flux.error(Throwable error)

Descripción: Crea un Flux que emite inmediatamente un error y finaliza.

Cuándo usarlo: Cuando quieras forzar o simular una secuencia fallida (manejo de errores, tests, etc.).

- Flux.defer(Supplier<? extends Publisher<? extends T>> supplier)

- Descripción: Retrasa la creación del Publisher hasta que alguien se suscriba. Cada suscripción invocará el supplier para crear un nuevo Publisher.

Cuándo usarlo: Para diferir la creación de la secuencia hasta el momento de la suscripción, útil si el contenido depende de condiciones que cambian en tiempo de ejecución.

- Flux.generate(...)

Descripción: Permite generar sincrónicamente valores uno a uno (con estado mutable, si se necesita), a través de una función de tipo SynchronousSink.

Cuándo usarlo: Al construir un flujo sincrónico y controlado, donde vas emitiendo valores de manera iterativa o basados en algún estado.

- Flux.create(...)

Descripción: Similar a generate, pero permite emitir valores de manera asíncrona y de forma más flexible a través de un FluxSink.

Cuándo usarlo: Al crear flujos que necesitan emisión manual asíncrona (p.ej., integraciones con callbacks).

Flux.interval(Duration period)

Descripción: Emite un long secuencial (0, 1, 2, ...) en intervalos de tiempo regulares (asincrónicamente).

Cuándo usarlo: Para trabajos periódicos o para realizar operaciones cada cierto intervalo de tiempo.

- Flux.never()

Descripción: Crea un Flux que nunca emite ningún valor ni finaliza.

Cuándo usarlo: Para pruebas o casos extremos donde se requiere un flujo que no termine jamás (por ejemplo, para combinar con otros flujos).

- Flux.first(Publisher<? extends T>... sources)

Descripción: Emite los valores del primer Publisher (entre varios) que empiece a emitir un valor o notifique terminación.

Cuándo usarlo: Cuando quieras “competir” entre varias fuentes y te interese solo la que responda primero.

- Flux.combineLatest(...)

Descripción: Combina las emisiones más recientes de múltiples fuentes en un nuevo valor, cada vez que cualquiera de ellas emite.

Cuándo usarlo: Para agrupar datos de varios Publisher donde se requiere siempre la última emisión de cada fuente.

- Flux.using(...)

Descripción: Recurso que permite manejar la apertura y cierre de recursos de manera segura dentro de un Flux.

Cuándo usarlo: Al trabajar con recursos que requieren limpieza, como conexiones, archivos, etc., asegurándote de que se cierren correctamente al terminar el flujo o ante un error.

Métodos de fábrica de Mono

- Mono.just(T valor)

Descripción: Crea un Mono que emite un único valor y finaliza.

Cuándo usarlo: Cuando tienes un valor único y necesitas exponerlo de manera reactiva.

- Mono.empty()

Descripción: Crea un Mono que no emite ningún valor y finaliza inmediatamente.

Cuándo usarlo: Para representar la ausencia de valor (por ejemplo, una búsqueda sin resultados).

- Mono.error(Throwable error)

Descripción: Emite de manera inmediata un error y finaliza.

Cuándo usarlo: Para propagar un error de forma reactiva o simularlo en pruebas.

- Mono.defer(Supplier<? extends Mono<? extends T>> supplier)

Descripción: Retrasa la creación del Mono hasta el momento de la suscripción. Cada suscripción creará un nuevo Mono.

Cuándo usarlo: Cuando necesitas que la lógica de creación (o recuperación de datos) se ejecute cada vez que alguien se suscriba.

- Mono.fromCallable(Callable<? extends T> callable)

Descripción: Ejecuta un Callable al suscribirse y emite su resultado como valor único o un error si el Callable lanza excepción.

Cuándo usarlo: Para transformar llamadas bloqueantes (o con retorno único) en un Mono, integrándolo a la cadena reactiva.

- Mono.fromSupplier(Supplier<? extends T> supplier)

Descripción: Similar a fromCallable, pero con Supplier (sin posibilidad de lanzar checked exceptions).

Cuándo usarlo: Para producir un único valor de forma perezosa (en el momento de la suscripción), cuando no se requiere manejo de excepciones checked.

- Mono.fromRunnable(Runnable runnable)

Descripción: Ejecuta un Runnable al suscribirse y luego emite onComplete. No emite ningún valor.

Cuándo usarlo: Para integrar una acción (que no retorna nada) a la secuencia reactiva.

- Mono.create(Consumer<MonoSink<T>> callback)

Descripción: Permite crear un Mono de manera programática, pudiendo emitir un valor o error de forma manual a través de MonoSink.

Cuándo usarlo: Para adaptar llamadas callback-based (o lógicas personalizadas) a la API reactiva.

- Mono.delay(Duration duration)

Descripción: Crea un Mono<Long> que emite un solo valor (0) tras el tiempo especificado y luego finaliza.

Cuándo usarlo: Para retardar una operación o disparar un evento tras cierto tiempo.

- Mono.when(Publisher<?>... sources)

Descripción: Combina varios Publishers, completando solo cuando todos han completado (no emite valor, salvo que se use otra sobrecarga).

Cuándo usarlo: Para esperar a la terminación de operaciones reactivas múltiples en paralelo, cuando no necesitas los resultados combinados sino saber que todas terminaron.

- Mono.ignoreElements(Publisher<T> source)

Descripción: Ignora todos los elementos emitidos por source y completa normalmente (o propaga el error).

Cuándo usarlo: Cuando te interesan solamente la finalización (o error) de un Flux, sin usar sus valores.

- Mono.using(...)

Descripción: Maneja la apertura y liberación de un recurso dentro de un Mono de forma segura, similar a try-with-resources.

Cuándo usarlo: Cuando el procesamiento reactivo usa un recurso que debe cerrarse correctamente al finalizar o ante error.