KONKURENTNÉ PROGRAMOVANIE

Cvičenie 12 : Reaktívne programovanie 3

Zadanie 1 a 2

- Využite endpoint https://swapi.dev/api/people
  - Zadanie 1: vytvorte prúd prvých 20 ľudí v zozname
  - Zadanie 2: vytvorte prúd všetkých ľudí

Cold and Hot publishers

- Cold publisher
  - Pre každý subscribe generuje hodnoty nanovo
    - Mono.defer(...), Flux.create(...), Flux.generate(...)
- Hot publisher
  - Nečaká na subscribe-y a proste generuje dáta
  - Typicky sleduje nejaký nezávislý zdroj a generuje prvky reprezentujúce jeho udalosti
    - používateľ, socket, senzor
  - implementácie
    - implementácie rozhrania Sinks.Many, Sinks.One
    - Mono.just(...), Flux.just(...), Flux.cache(...),...

Sinks.one()

- Sinks: Máme k dispozícii začiatok prúdu do ktorého môžeme posielať dáta, kedy chceme
- Sinks.one(): Pre jednu hodnotu:
  - `One<Long> oneSink = Sinks.one();`
    - Môžeme posielať: oneSink.tryEmitValue(value);
      - Hneď potom sa zatvorí prúd (metóda vráti OK)
      - Ak pošleme viac hodnôt vráti FAIL_TERMINATED
    - Hodnota odíde každému subscriberovi, aj takému, čo príde neskôr
  - `Mono<Long> mono = oneSink.asMono();`
    - Subscriberi sa môžu priebežne pripájať cez mono.subscribe(...)

Sinks.many()

- Pre Flux máme veľa možností
- Sinks.many()
  - .unicast() – iba jeden subscriber je povolený, ďalším hodí výnimku
    - .onBackpressureBuffer()
      - Ak subscriber zatiaľ nechce hodnotu, zapamätá sa v buffri prúdu
    - .onBackpressureError()
      - Ak subscriber nechce hodnotu a aj tak ju pošleme, nepodarí sa to
        - hodnota sa zahodí
        - už sa nedostane k subscriberovi
  - .multicast() – subscriberi dostávajú hodnoty zaslané až po ich pripojení
    - .directAllOrNothing(),
      - Ak aspoň jeden subscriber nechce hodnotu subscriberi ešte nie sú, hodnota sa nepošle
    - .directBestEffort(),
      - Ak žiaden subscriber nechce hodnotu, alebo subscriberi ešte nie sú, hodnota sa zahodí, inak sa pošle iba tým, čo chcú spracovať
    - .onBackpressureBuffer()
  - .replay() – subscriberi dostávajú aj hodnoty zaslané pred ich pripojením
    - all(), latest(), latestOrDefault(v), limit(počet), limit(duration),...

Sinks.many()

- `Many<Long> unicastSink = Sinks.many().unicast().onBackpressureBuffer();`
  - *`unicastSink.tryEmitNext(value)`*
  - *`unicastSink.tryEmitError(throwable)`*
  - *`unicastSink.tryEmitComplete()`*
  - *`currentSubscriberCount()`*
- `Flux<Long> flux = unicastBuffer.asFlux();`

Odovzdanie spracovania inému vláknu

- Defaultne beží celý prúd v hlavnom vlákne
  - Operátor volá listener ďalšieho operátora v poradí
- Niektoré operátory môžu spustiť listener ďalšieho operátora vo vlákne
  - V tomto vlákne sa znova ide jednovláknovo
    - Operátor volá listener ďalšieho operátora v poradí

Pozdržanie prvkov

- delayElements(delay)
  - Posiela ďalší prvok najskôr po uplynutí času
  - Listener ďalšieho operátora sa volá v samostatnom vlákne
  - Ďalší prvok odchádza až po spracovaní predchádzajúceho prvku zvyškom operátorov a záchytnou funkciou
  - Žiadne paralelné spracovanie

ParallelFlux

- Prúd môžeme rozdeliť na paralelné prúdy
- ParallelFlux Flux.parallel(počet)
  - Delíme prúd na určený počet paralelných ciest
  - Prichádzajúce dáta idú do prúdov cez round-robin
  - Ešte sa neriešia vlákna, len rozhadzujú dáta
- ParallelFlux Flux.parallel()
  - Počet prúdov sa rovná počtu jadier(vlákien) procesora

runOn()

- ParallelFlux.runOn(Scheduler)
  - Spracovanie paralelných prúdov vláknami zo Schedulera
    - Vlákno dostane pridelenú 1 cestu a postará sa o 1 prvok cesty až po jej koniec – spúšťa listenery operátorov cesty pre prichádzajúci prvok dát
    - Vlákno môže byť pridelené k inej ceste
- Analógia
  - Obchod má 5 pokladní (ciest), kde prichádzajú zákazníci (dáta)
  - Obchod má 3 pokladníkov (vlákien schedulera)
    - Pokladník vybaví 1 zákazníka (nablokuje, zabalí nákup, vyprevadí k dverám a rozlúči sa) a ide k inej pokladni

Schedulery

- Schedulers.immediate()
  - Použije iba hlavné vlákno
- Schedulers.single()
  - Použije 1 worker vlákno
- Schedulers.parallel()
  - Použije maximálne toľko vlákien, koľko je jadier procesora – určené pre neblokované operácie
- Schedulers.boundedElastic()
  - Použije maximálne 10x toľko vlákien, koľko je jadier procesora – určené pre blokované operácie
  - Ak je operácia blokovaná, vlákno môže zatiaľ vybaviť inú úlohu
- Schedulers.new...()
  - Na nastavovanie rôznych počtov workerov, mien pre vlákna, buffre,...

ParallelFlux.sequential()

- Spája cesty paralelných prúdov do jedného prúdu
- Keď pritečie prvok dát cez nejakú cestu pošle ho do výstupného prúdu s jednou cestou
  - Výstupom je obyčajný Flux
- O prvok vo výstupnom prúde sa stále stará to isté vlákno, ktoré sa oňho staralo počas paralelnej cesty
- Keď príde ďalší prvok z inej cesty, jeho vlákno čaká pokiaľ sa skončí práca vo výstupnom prúde s prechádzajúcim prvkom

publishOn a subscribeOn

- Flux.publishOn(Scheduler)
  - Vlákno čo sa staralo o prvok prúdu doteraz, odovzdáva štafetu vláknu z daného Schedulera
  - Použitie, keď máme rýchlych producentov a nasledujú pomalé operátory vo zvyšku prúdu – chceme uvoľniť aktuálne vlákno producentovi dát
- Flux.subscribeOn(Scheduler)
  - Hlavné vlákno nerobí subscribe, ale nechá to na vlákno z daného Schedulera
  - Použitie, keď máme pomalého producenta dát a chceme hlavné vlákno čím skôr pustiť