



Spring WebFlux

Workshop

Icindekiler

1. Reaktif Programlama
2. Spring WebFlux
3. WebFlux ile basit bir proje implementasyonu
4. Sonuc

Reaktif Progamlama



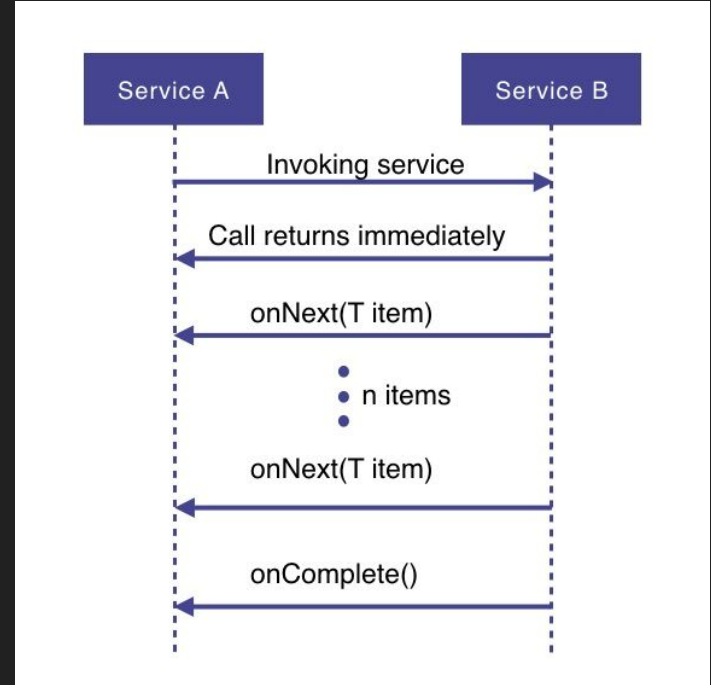
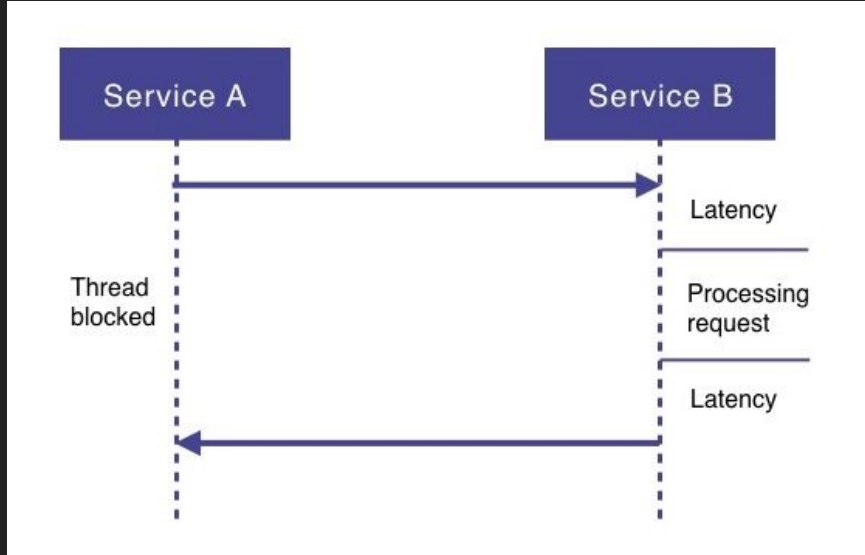
Reaktif Programlama

- Asenkron ve non-blocking
- Event temelli
- Data Streamler üzerinde BackPressure

Asekron veri akisina(data stream) dayali, gercek zamanli olay(event) bazli bir yazilim paradigmasidir

Islemler senkron biciminde birbirlerinin tamamlanmasini beklemezler.

Asenkron

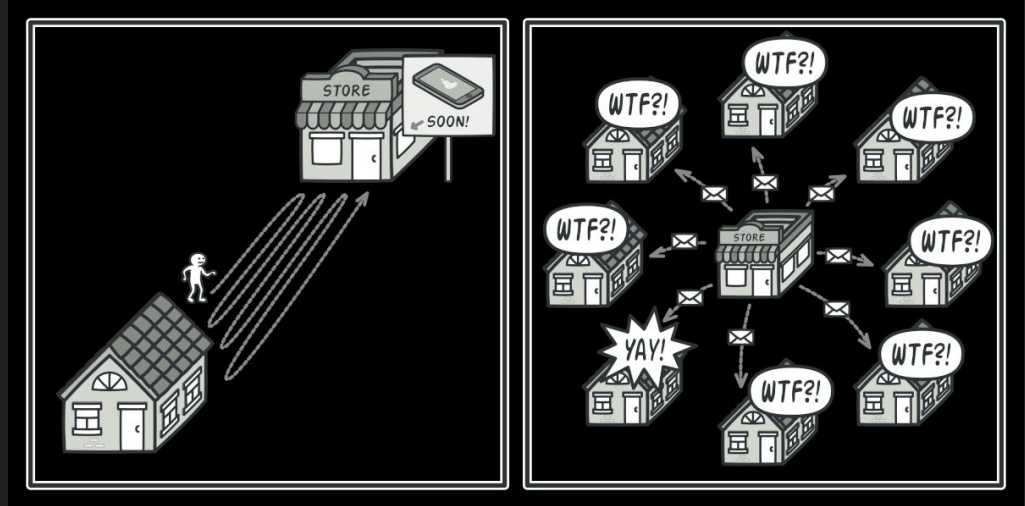


Data Stream

Sirayla gerceklesen olay/event dizisi

Deger, hata, tamamlandi gibi sinyaller doner

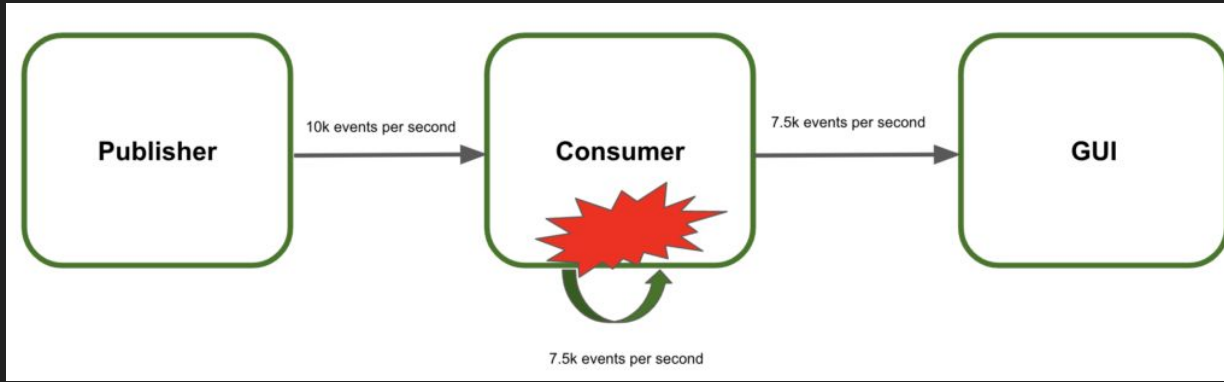
Reaktif programlama bu akislari/streamleri dinleyerek (observe ederek) gerceklesir



Backpressure

Data'nin nasıl regule edileceğini belirler

Reaktif Programlamanın non-blocking doğası gereği data stream tek seferde gönderilmez, elimizde handle edebileceğimizden büyük bir veri varsa backpressure desteği ile kullanıcının consume edebileceği şekilde datayı ayarlarız.



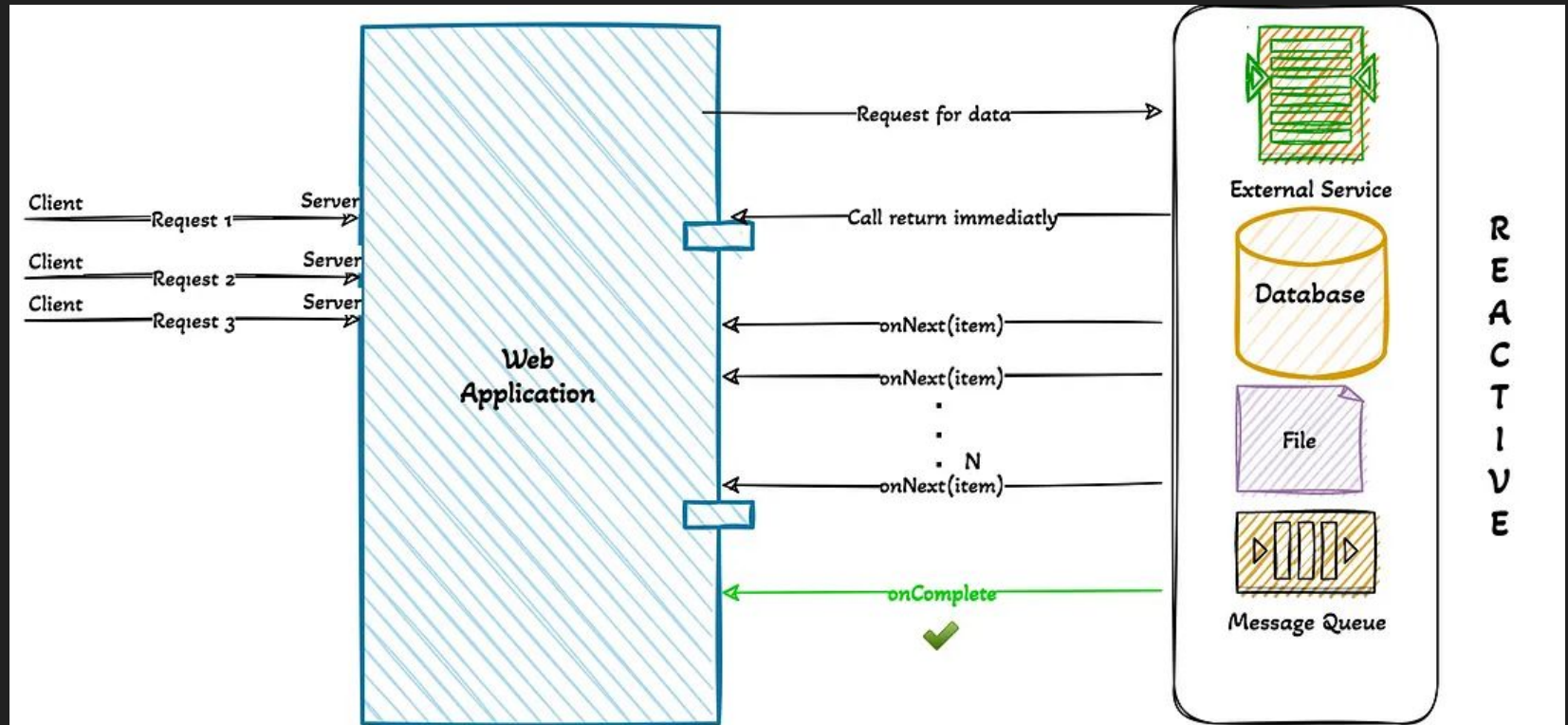
- Data stream gönderimini yavaşlat
- Ekstra data için buffer: consumer gecici olarak ekstra datayı tutar
- Ekstra datayı sil ve track etme

Imperative Programlama ile Farki Ne?

- Senkron calisiyor : Bir requeste atanan bir thread, baska bir requestin calismasi icin ilk threadin isi bitene kadar ikincisini bloklar.
- Request basina bir thread : her request icin bir thread olusturulur, bu durum yapılacak requestleri thread-pool ile sinirlendirir
- Backpressure destegi yok

Imperative	Reactive
<pre>A = 1 B = A + 1 C = B + 1 A = 10 Result: A = 10 B = 2 C = 3</pre>	<pre>A = 1 B = A + 1 C = B + 1 A = 10 Result: A = 10 B = 11 C = 12</pre>

Reaktif Programlama Nasıl Çalışıyor



Spring WebFlux



WebFlux Nedir?

- Reaktif, asenkron, non-blocking uygulamalar geliştirmek için project reactor üzerine kurulu bir Spring Frameworkudur.
- Geleneksel her requeste bir thread(default olarak Tomcat ile) yerine event bazlı reaktif bir yaklaşımı baz alır(default olarak Netty ile).
- Reactive Programlama da temel olarak, threadler önceki görevlerin tamamlanmasını beklemeden kendi işlerini tamamlayabilirler ve bloklanmadan hayatlarına devam ederler

WebFlux ile geliştirdiğimiz projeler tamamen reactive olmalıdır.

Reactive programlar doğası gereği asenkron çalıştığı için süreç içinde senkronize çalışan bir yapı olması projenin reaktiflikten uzaklaşmasına neden olur

Bu sebeple tüm stack reactive yapıda olmalıdır.

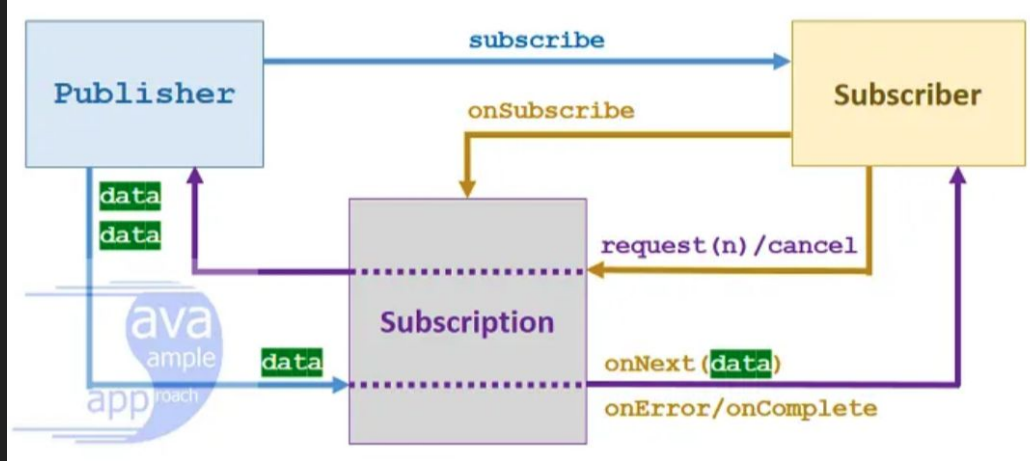
Reactive Streamler, WebFlux Yapilari

Reactive Programlamanin temelini olusturur.

Publisher - Subscriber yapisini baz alir.


- Publisher : Subscriber'dan gelen talebe gore publish islemini gerceklestirir. Birden fazla subscribera hizmet edebilir.
- Subscriber : Publisher tarafından yayınlanan eventleri yakalar. Alinan eventlerle basa cikmak icin 4 metodu vardir.
 - onSubscribe
 - onNext
 - onError
 - onComplete

- Subscription : publisher ve subscriber arasındaki ilişkiyi temsil eder. event için talepte bulunmak adına request(long n) talebi iptal etmek için cancel() metoduna sahiptir
- Processor : hem publisher hem subscriber olabilen yapıdır



WebFlux Reaktif Veri Yapilari

- Mono : 0 yada 1 tane event iceren publisherlar icin kullanilir



```
Mono<String> stringMono = Mono.just("CodeFirst");  
Mono<Integer> integerMono = Mono.just(2023);
```


- Flux : 0...N adet event iceren publisherlar icin kullanilir.



```
Flux<String> stringFlux = Flux.just("CodeFirst", "Yazilim", "Teknolojileri");  
Flux<Integer> integerFlux = Flux.just(2022, 2023);
```

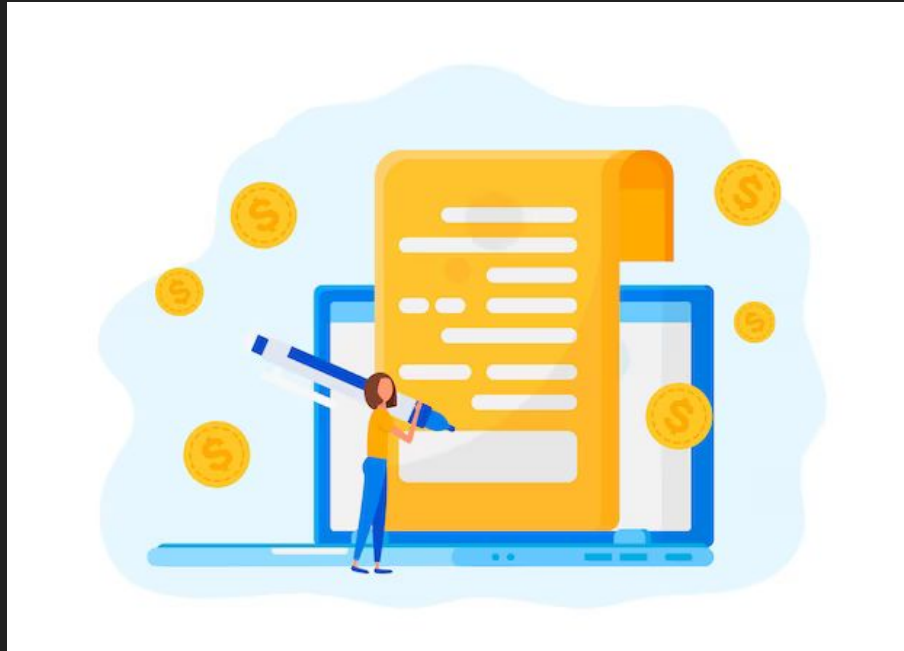
WebFlux ile Basit Bir Proje Implementasyonu



**“Talk is
cheap. Show
me the code.”**

Linus Torvalds

Sonuc



Concurensynin on planda oldugu cok sayida istegi az kaynakla yonetebilmek icin oldukca kullanisli

Metodlardan database driverlarına kadar her sey reactive yapida olmalı

Tesekkurler!

<https://github.com/lvntky/webflux-workshop>

