Reactive Programing using webFlux:

intended for only dev and architect.

asper the discussion on cashback and MOMO balance with @Taniya Biswas if the TPS of current system does not match required TPS then we have to do for this approach.

owner @indranil banerjee

Objective:

Create a configuration **Service Layer**(`manifestation`), and two other API **COMMECTONE**, **CONNECTTWO**

- Create FLUX RANGE (SAME KIND OF OPERATION), FLUX JUST for different kind of operation.
- create communication, Through different module. Manifestation to CONNECTONE, CONNECTTWO
- show the non-blocking events.

ATTENDENCE:

Indranil Banerjee [MTN Nigeria]
Ankit TARVE [MTN Cote d'Ivoire]
P Chandrakanth
Vijay Chandramohan [MTN Nigeria]
Bipul Kumar [MTN Nigeria]
Mayank Kansara [MTN Nigeria]
Rupesh
Shatabdi Chakraborty [MTN Nigeria]
UZOCHUKWU NWOSU [MTN Nigeria]
Moona
Oyegoke Ogundere [MTN Nigeria]
Siddiq Saliha [MTN Nigeria]
Adarsh Singh [MTN Nigeria]
Prashanth Baddam [MTN Nigeria]
Omolaja Abubakar [MTN Nigeria]
Abhishek MONDAL [MTN Cote d'Ivoire]
Michael Sanni [MTN Nigeria]

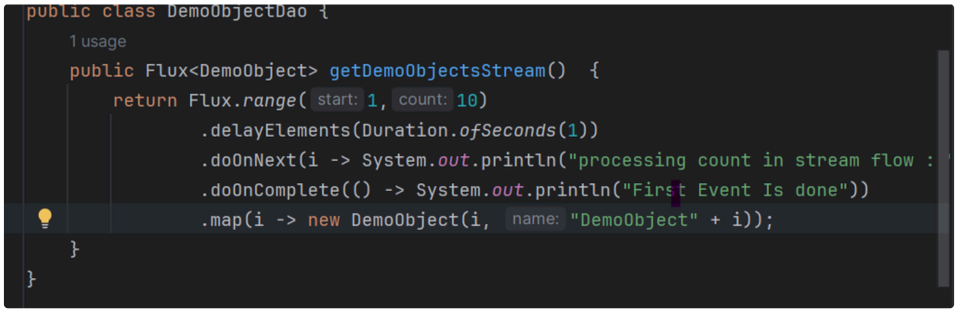
PROBLEM STATEMENT:

- As part of the discussion for TPS increase this will give the increased TPS, approach one we have already covered approach two we will discussed here.

SOLUTION APPROCH:

Connect one service:

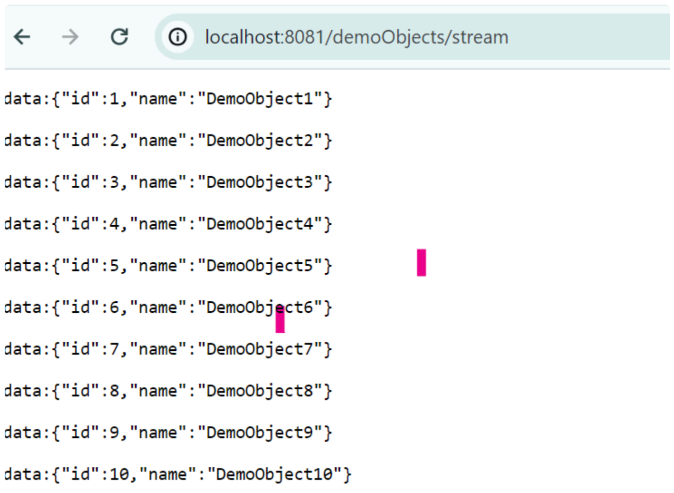
connect one service should give user a experience on non -blocking queue.



```java
public class DemoObjectDao {
    1 usage
    public Flux<DemoObject> getDemoObjectsStream()  {
        return Flux.range( start: 1, count: 10)
                .delayElements(Duration.ofSeconds(1))
                .doOnNext(i -> System.out.println("processing count in stream flow : 
                .doOnComplete(() -> System.out.println("First Event Is done"))
                .map(i -> new DemoObject(i,  name: "DemoObject" + i));
    }
}
```

delayed one second same kind of event.

output response.



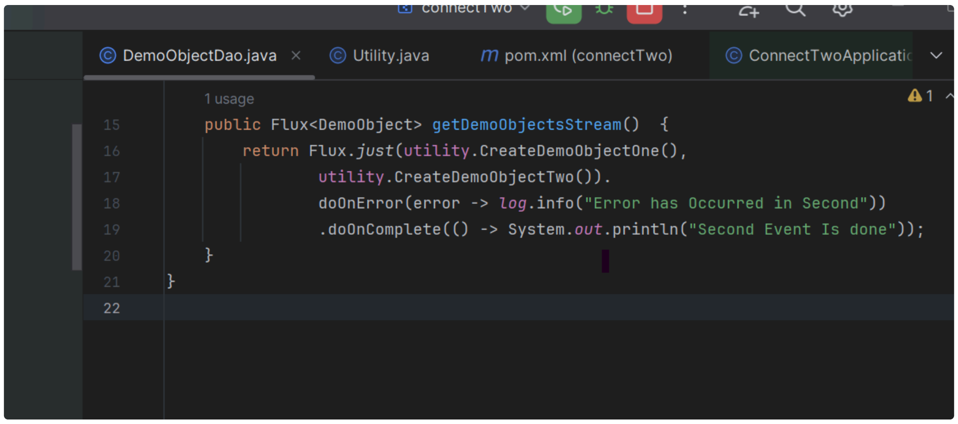
localhost:8081/demoObjects/stream

```
data:{"id":1,"name":"DemoObject1"}

data:{"id":2,"name":"DemoObject2"}

data:{"id":3,"name":"DemoObject3"}

data:{"id":4,"name":"DemoObject4"}

data:{"id":5,"name":"DemoObject5"}

data:{"id":6,"name":"DemoObject6"}

data:{"id":7,"name":"DemoObject7"}

data:{"id":8,"name":"DemoObject8"}

data:{"id":9,"name":"DemoObject9"}

data:{"id":10,"name":"DemoObject10"}
```

output events for non-blocking queues.

Connect two service:

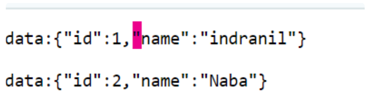
connect one service should give user experience on non -blocking queue.



```
    1 usage
    public Flux<DemoObject> getDemoObjectsStream()  {
        return Flux.just(utility.CreateDemoObjectOne(),
                utility.CreateDemoObjectTwo()).
                doOnError(error -> log.info("Error has Occurred in Second"))
                .doOnComplete(() -> System.out.println("Second Event Is done"));
    }
}
```

connecting through different kind of services.

output format.

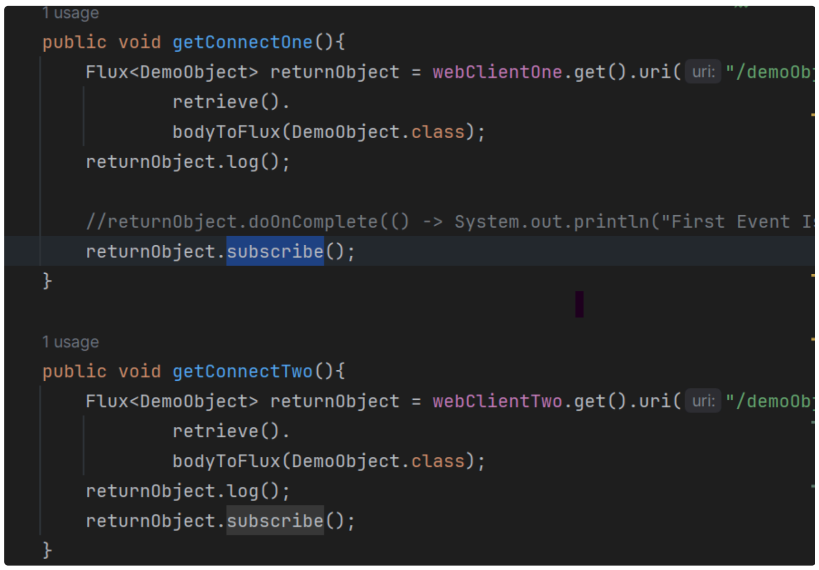


```
data:{"id":1,"name":"indranil"}

data:{"id":2,"name":"Naba"}
```

output of different kind of event.

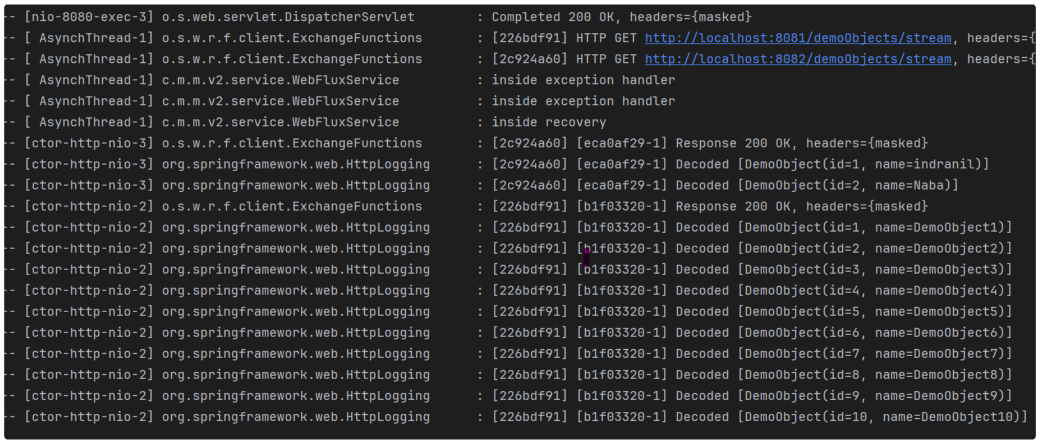
manifestation:

Please see how this two API are getting connected.



```
1 usage
public void getConnectOne(){
    Flux<DemoObject> returnObject = webClientOne.get().uri( uri: "/demoOb
            retrieve().
            bodyToFlux(DemoObject.class);
    returnObject.log();

    //returnObject.doOnComplete(() -> System.out.println("First Event I
    returnObject.subscribe();
}

1 usage
public void getConnectTwo(){
    Flux<DemoObject> returnObject = webClientTwo.get().uri( uri: "/demoOb
            retrieve().
            bodyToFlux(DemoObject.class);
    returnObject.log();
    returnObject.subscribe();
}
```

output response.



```
-- [nio-8080-exec-3] o.s.web.servlet.DispatcherServlet        : Completed 200 OK, headers={masked}
-- [ AsynchThread-1] o.s.w.r.f.client.ExchangeFunctions       : [226bdf91] HTTP GET http://localhost:8081/demoObjects/stream, headers={
-- [ AsynchThread-1] o.s.w.r.f.client.ExchangeFunctions       : [2c924a60] HTTP GET http://localhost:8082/demoObjects/stream, headers={
-- [ AsynchThread-1] c.m.m.v2.service.WebFluxService          : inside exception handler
-- [ AsynchThread-1] c.m.m.v2.service.WebFluxService          : inside exception handler
-- [ AsynchThread-1] c.m.m.v2.service.WebFluxService          : inside recovery
-- [ctor-http-nio-3] o.s.w.r.f.client.ExchangeFunctions       : [2c924a60] [eca0af29-1] Response 200 OK, headers={masked}
-- [ctor-http-nio-3] org.springframework.web.HttpLogging      : [2c924a60] [eca0af29-1] Decoded [DemoObject(id=1, name=indranil)]
-- [ctor-http-nio-3] org.springframework.web.HttpLogging      : [2c924a60] [eca0af29-1] Decoded [DemoObject(id=2, name=Naba)]
-- [ctor-http-nio-2] o.s.w.r.f.client.ExchangeFunctions       : [226bdf91] [b1f03320-1] Response 200 OK, headers={masked}
-- [ctor-http-nio-2] org.springframework.web.HttpLogging      : [226bdf91] [b1f03320-1] Decoded [DemoObject(id=1, name=DemoObject1)]
-- [ctor-http-nio-2] org.springframework.web.HttpLogging      : [226bdf91] [b1f03320-1] Decoded [DemoObject(id=2, name=DemoObject2)]
-- [ctor-http-nio-2] org.springframework.web.HttpLogging      : [226bdf91] [b1f03320-1] Decoded [DemoObject(id=3, name=DemoObject3)]
-- [ctor-http-nio-2] org.springframework.web.HttpLogging      : [226bdf91] [b1f03320-1] Decoded [DemoObject(id=4, name=DemoObject4)]
-- [ctor-http-nio-2] org.springframework.web.HttpLogging      : [226bdf91] [b1f03320-1] Decoded [DemoObject(id=5, name=DemoObject5)]
-- [ctor-http-nio-2] org.springframework.web.HttpLogging      : [226bdf91] [b1f03320-1] Decoded [DemoObject(id=6, name=DemoObject6)]
-- [ctor-http-nio-2] org.springframework.web.HttpLogging      : [226bdf91] [b1f03320-1] Decoded [DemoObject(id=7, name=DemoObject7)]
-- [ctor-http-nio-2] org.springframework.web.HttpLogging      : [226bdf91] [b1f03320-1] Decoded [DemoObject(id=8, name=DemoObject8)]
-- [ctor-http-nio-2] org.springframework.web.HttpLogging      : [226bdf91] [b1f03320-1] Decoded [DemoObject(id=9, name=DemoObject9)]
-- [ctor-http-nio-2] org.springframework.web.HttpLogging      : [226bdf91] [b1f03320-1] Decoded [DemoObject(id=10, name=DemoObject10)]
```

see all the are coming.

Action Item:

- [x] demo given to developer done by @indranil banerjee .
- [ ] dev to implement.

Source Code:

manifestation.zip
27 Mar 2024, 02:57 PM

connectTwo.zip
27 Mar 2024, 02:57 PM

connectOne.zip
27 Mar 2024, 02:57 PM