Spring

Web

Flux

In normal programming when we do compute intensive tasks or I/O intensive tasks, the precious treads are in waiting

Reactive manifesto outlines 4 important principals site

1. Responsive
2. Resilient
3. Message Driven
4. Elastic

Spring web flux communication patterns

1. Request, response
2. Request, Streaming response
3. Streaming request, response
4. Bidirectional streaming

## Initial Set up

**Traditional REST behavior**

curl <http://localhost:8080/traditional/products>

* It takes 10 seconds.
* Response comes when all 10 records are generated
* Even if we cancel the requests the backend traditional service is still processing the request
* On cancel since the traditional service is not stopping the external service running on 7070 is still processing. So its cascading wastage of resource

Web Flux Behavior

curl <http://localhost:8080/reactive/products>

* It takes 10 seconds.
* Response comes when all 10 records are generated. We can get streaming response on console by disabling buffer with curl -N <http://localhost:8080/reactive/products>
* When we cancel the requests the backend traditional service stops
* Backend service on 7070 also stops

We can see streaming behavior in Browser with streaming endpoint <http://localhost:8080/reactive/products/stream> which has below end MediaType defined. By default it uses Application/Json hence it waits for response

@GetMapping(value = "products/stream", produces = MediaType.TEXT\_EVENT\_STREAM\_VALUE)  
public Flux<Product> getProductsStream() {

Sec 2-10

**Error handling**

On our backend service running on 7070 we have an endpoint /demo01/products/notorious which will crash. So if we configure that endpoint with <http://localhost:8080/reactive/products> and <http://localhost:8080/traditional/products> we will see that

In traditional API it will fail with 500 with no response

In Web flux it will give partial response.

[{"id":1,"description":"product-1","price":1},{"id":2,"description":"product-2","price":2},{"id":3,"description":"product-3","price":3},{"id":4,"description":"product-4","price":4}curl: (18) transfer closed with outstanding read data remaining

This is not proper JSON. We can handle this easily

[{"id":1,"description":"product-1","price":1},{"id":2,"description":"product-2","price":2},{"id":3,"description":"product-3","price":3},{"id":4,"description":"product-4","price":4}]