Reactor subscriberContext thoughts

noun

Operator : operation made to sequence,like flatMap

Subscription : represents lifecycle of a Subscriber subscribing to a publisher

Sequence : an abstract for data flow represents a Flux or a Mono

Reactor using subscriberContext carry context related entities

context is calculated at subscription time before data flows and moving from downstream to upstream(propagated from the bottom of the chain towards the top)

Correct:

String key = "message";  
Mono<String> r = Mono.*just*("Hello")  
 .flatMap( s -> Mono.*subscriberContext*()  
 .map( ctx -> s + " " + ctx.get(key)))  
 .subscriberContext(ctx -> ctx.put(key, "World"));  
  
StepVerifier.create(r)  
 .expectNext("Hello World")  
 .verifyComplete();

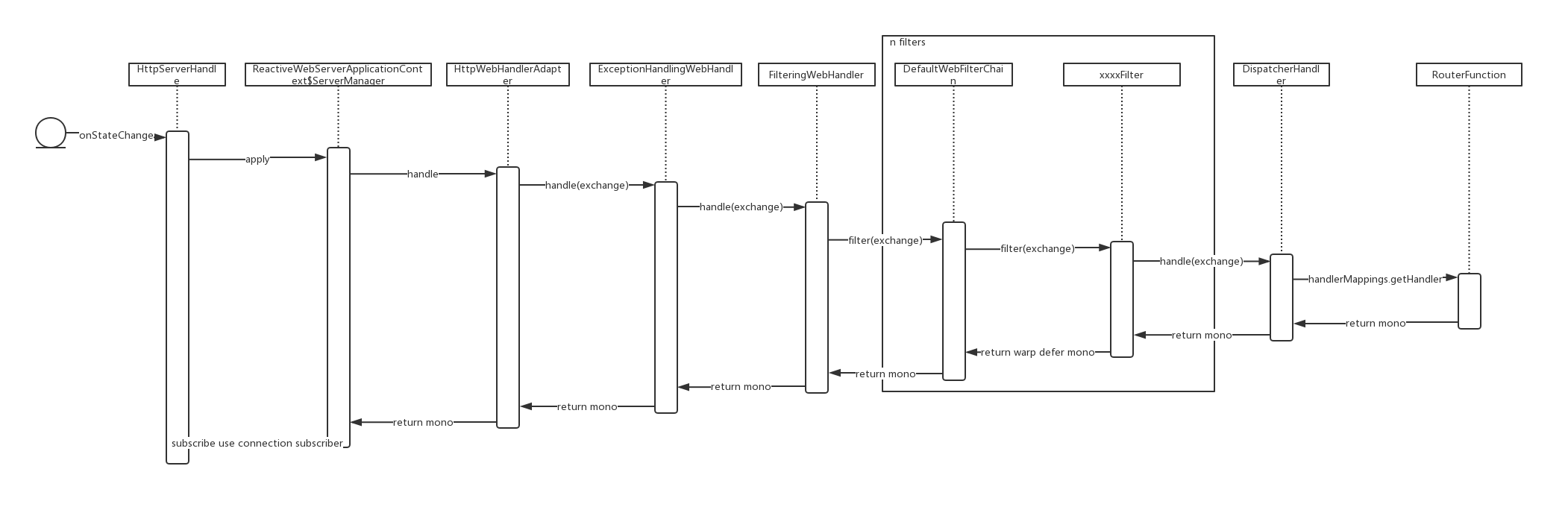
Wrong:

String key = "message";  
Mono<String> r = Mono.*just*("Hello")  
 .subscriberContext(ctx -> ctx.put(key, "World"))  
 .flatMap( s -> Mono.*subscriberContext*()  
 .map( ctx -> s + " " + ctx.get (key));  
  
StepVerifier.create(r)  
 .expectNext("Hello World")  
 .verifyComplete();//NoSuchElementException,operator from downstream cann’t see context

question : in real application world, WebFilter runs ahead of our businessHandlers,that may means subscriberContext is on top of any business Operators,why operators can still reads subscriberContext

answer : through method calling chain,WebFilter wrapped businessHandlers’s result into Mono.defer,this makes subscriberContext connect into process chain behind business Operators,this makes sure context’s visibility

Mono.*defer*((Supplier<Mono<String>>) () ->  
 Mono.*defer*((Supplier<Mono<String>>) () ->  
 Mono.*defer*((Supplier<Mono<String>>) () ->  
 Mono.*subscriberContext*().flatMap(context -> {  
 //do business code here...  
 return Mono.*just*((String)context.get("123"));  
 }))  
 .subscriberContext(Context.of("123","456"))))//simulate put context in a chain of Mono defer,this action always at downstream of business operator  
 .subscribe(s -> System.*out*.println("s = [" + s + "]"));



Since subscriberContext is tied to subscriber,so try to get same context from different sequence is not possible:

Flux.*just*(1, 2, 3).flatMap((Function<Integer, Publisher<?>>) integer ->  
 Flux.*just*("123", "456")  
 .flatMap(s1 -> Mono.*subscriberContext*()  
 .map(ctx -> integer + " " + ctx.get("key"))))//NoSuchElementException  
 .subscriberContext(reactor.util.context.Context.*of*("key", "value"));

This code also throw NoSuchElementException, for subscriberContext only tied to sequence(Flux.just(1,2,3)),no context tied to sequence(Flux.just(“123”,”456”))