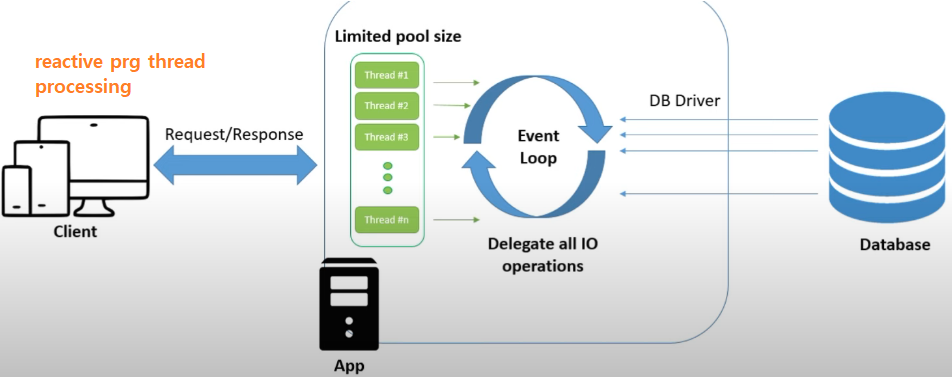
Reactive doesn’t means fast, this will not see much change in basic crud operations, u should use this for streaming based applications

Reactive Systems rely on asynchronous message passing to establish a boundary between components.

Reactive Programming focuses on non-blocking, asynchronous execution - a key characteristic of Reactive Systems

• Reactive Programming focuses on processing streams of data.

How this works



Let’s say if thread 1 needs to hit db and get response in old synchronous & blocking behaviour, that thread will hit db and waits for the response if db gives resp after 10 mins then that thread will wait for 10mins which is a waste idle time period, then once db gave response then that same thread will process that request

Async & Non-blocking model

Instead of these threads waiting, in async model, the thread-1 will hit db and just leave away, whenever db sends response if thread-10 is available then that thread will take

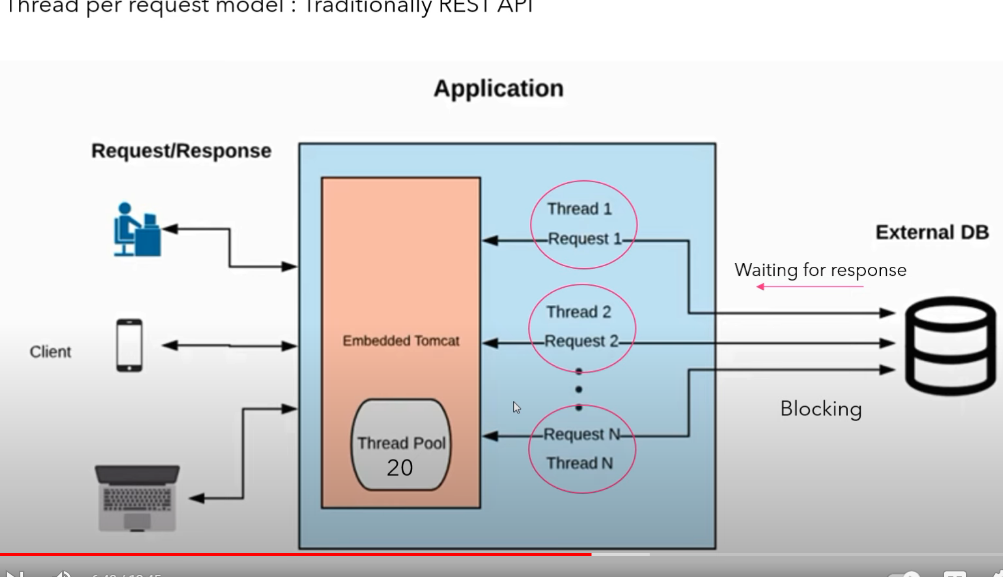
Response from db and process it (fired by 1 thread response processed by another thread)

* Here same thread may not process the full flow and
* Those threads will not be idle

When to use spring reactive

1. If u have more traffic to ur appln then use this (High concurrent users )

Ex:- if u are maintaining a thread pool of 20 (means at at time 20 people can hit )but if at a time if 60 people came as no thread is available in thread pool to take the incoming request then new request will not be served,



1. If ur app is making more 3rd /External party service calls (in those scenarios limited threads in thread pool will waste their time for 3rd party service responses due to synchronous-blocking behavior, in reactive the thread won’t wait )
2. In streaming & asynchronous applications u can use this ex:- live streaming like crickbuzz
3. Abstraction over async processing
4. Abstract whether or not ur program is sync or async

Advantages

With this approach threads in thread pool will be u continuously busy, those threads will not be wasting time in waiting response for 3rd party rest service calls or db calls

• Performance benefit is more clearly visible when a system under load (when 100 concurrent are hitting an app of thread pool 10then those 10 threads will be effectively utilised instead of those threads waiting for blocking calls)

Features

* Data streams
* Async & non-blocking: in low load (among 10 threads in pool only 1 is utilized) we can’t see much performance changes, but in heavy load we can see the change in response
* Backpressure: if producer is sending msgs at higher speed which can’t be processed by consumer, then as per throttling , it is the ability of subscriber to say to sender slow down as I need to consume the data, this mostly we won’t do framework will implement this
* Failures as messages

Async

=======

• Events are captured asynchronously. And we will register the functions to be executed when something happened, like publisher & subscriber pattern publisher will notify all subscribers when there is a update, lly when there is a change publisher will call those register functions

• A function is defined to execute when an event is emitted.

• Another function is defined if an error is emitted.

• Another function is defined when complete is emitted.

• This can be a difficult paradigm to adjust to when first getting started!

Failures as messages

1. Means exceptions are not thrown
2. Would break processing of a stream
3. Exceptions are processed by handler function, means we should have registered the function to be executed when error scenario occurred