Date : 30/06/2021
Spring Boot 9AM
Mr. RAGHU

Spring Boot Reactive Programming
(Spring Web Flux and Reactive MongoDB)

Blocking v/s non-Blocking Process:-

*) Blocking process:-

Server Execution
=> When client makes any request, server allocates one
   thread (which is doing nothing/Free Thread)
=> Such Thread is also called as Request Processing thread.
  (This thread will process request).
=> Now, if it needs any DB data, then it will make Network call
   (JDBC-SQL, ORM-JPA..etc)
   also called as NIO - Network-Input-Output

=> Consider This DB call make take 10 sec around (example)
   2 sec to goto DB , 5 sec to process SQL, 3 sec to give
   data back to App.

**** Mean while(10 sec) Thread should be in WAITING/SILENT MODE.
     Even Such thread will not be allocated to other request
     until response given.
     ie Mean while (in 10 sec) if any request has to come to
     server, thread be in silent, never process it.
     In this case server allocates another thread.

  A Thread in WAITING/SILENT MODE for NIO call is known as
  BLOCKING THREAD.

*** Resource Utilization Issue.
    ie if all thread are busy, then BLOCKING(SILENT/WAITING) THREAD
    can be used, but here it is not happend.
  If we use them properly then no.of request processing count
  get increased.

=> This is default nature of Webserver, it is server design problem.
  It is not code problem (Servlet, Spring WEB MVC, Spring Boot REST).

Request#1 --- thread#1
Request#2 --- thread#2
Request#3 --- thread#3
Request#4 --- thread#1 (if it gets free, else another thread)
..etc

*) No.of thread created by default is 200. we can modify them
        using key:
                server.jetty.threads.max=300
                server.tomcat.threads.max=300

```
-----------------------------------------------------------------
        Non-Blocking Process : Reactive Process

One Event Loop Design is added to Webservers, that holds details
of all threads (ALLOCATED(1)/NOT(0)).

=> This time thread will not be waiting for NIO call.
  If a Thread needs NIO call then Event Loop makes Thread as FREE
  and parallel NIO call.

=> once Response/Data given By DB, then either Same Thread (or)
  Another Thread(which one is Free) is allocated for sending Response
  ie Keeping all threads busy

*) This time normal servers like JBoss, Tomcat..etc are not used
   (Not supported for Reactive Programming).
   New Server is provided Netty designed using non-blocking
   with latest API Servlet 3.1 +

*) new APIs given
   Spring Web Flux (It is replacement of Spring Web)
   Spring Data Reactive (as of now only NoSQL Dbs).

spring-boot-starter-webflux
spring-boot-starter-data-mongodb-reactive

pom.xml
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-mongodb-
reactive</artifactId>
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-webflux</artifactId>
</dependency>

-----------------------------------------------
Spring  Boot Current Version : 2.x (2.3)
Spring Current Version is : 5.x (5.2)
-----------------------------------------------
=> This time we get data in Mono<T> or Flux<T>
  From event loop(DB) and converted into Response Format
  by server.

*) Return only data from Event Loop, this time it is not Response
    /ResponseEntity.

  Mono<T>  ---> 0 ... 1 object
  Flux<T>  ---> 0 / n objects

Later these is converted by Server response by allocated Thread
and send to client device.

ex:
Mono<String> m =  Mono.just("Hello");
Mono m = Mono.empty();
```

```
Flux<String> f =  Flux.just("Hello","HI","WELCOME");
Flux f = Flux.empty();

These are all APIs are given By Project Reactor
 (reactor.core.publisher)
-----------------------------------------------------
```