**SPRING MVC VÀ ĐỨA EM SINH SAU ĐẺ MUỘN**

**Spring WebFlux**

* Spring WebFlux là công cụ phát triển web mới từ Spring 5, hỗ trợ xây dựng ứng dụng web theo chuẩn Reactive
* Spring WebFlux mặc định sử dụng Project Reactor để hiện thực Reactive Streams specification. Chúng ta cũng có thể sử dụng các thư viện khác hiện thực Reactive Streams specification như RxJava, … để làm việc với Spring WebFlux
* Mặc định, Spring Webflux sử dụng Server Netty để run hệ thống reactive:



* Depedency:

A black text on a white background

Description automatically generated

* Reactive và non-blocking nhìn chung thì không làm cho ứng dụng chạy nhanh hơn mà tạo ra ứng dụng linh hoạt hơn với ít yêu cầu về luồng và bộ nhớ
* Trong Spring MVC thì các thông tin của request từ client sẽ được đối tượng HttpServletRequest và những thông tin trả về cho client sẽ do đối tượng HttpServletResponse nắm giữ. Nhưng đối với Spring WebFlux với Reactive support, những thông tin từ request sẽ chứa trong đối tượng ServerHttpRequest, còn thông tin về response sẽ do đối tượng ServerHttpResponse nắm giữ. Body của request và response thay vì là các đối tượng InputStream và OutputStream, giờ sẽ là đối tượng Flux<DataBuffer>

**Sự khác biệt quan trọng giữa Spring MVC và Spring WebFlux**

* Kiểu xử lý đồng bộ và bất đồng bộ:
  + Spring MVC:
    - Mỗi request từ client tạo ra một luồng xử lý riêng biệt trong servlet container
    - Luồng này bị chặn (blocked) cho đến khi xử lý hoàn tất cho request đó
    - Mỗi request sẽ có một luồng riêng biệt và nếu có nhiều request đồng thời, servlet container sẽ tạo ra nhiều luồng riêng biệt tương ứng
  + Spring WebFlux:
    - Sử dụng một pool thread (thread pool) có sẵn để xử lý request
    - Khi có request đến, nó sử dụng các luồng từ pool này để xử lý request đó
    - Khi một request tới có nhiều công việc bất đồng bộ, Spring WebFlux sẽ sử dụng các luồng từ pool thread để xử lý các công việc này trong cùng một request, mà không cần chặn (block) các luồng

**A computer screen shot of a code

Description automatically generated**

* Thư viện Reactive vs Servlet:
  + Spring MVC: Dựa trên Servlet API và sử dụng các thư viện Servlet truyền thống như HttpServlet, HttpServletRequest và HttpServletResponse
  + Spring WebFlux: Sử dụng mô hình lập trình Reactive và dựa trên Project Reactor, một thư viện Reactive Programming. Nó sử dụng các lớp như Mono và Flux để làm việc với dữ liệu bất đồng bộ
* Phong cách lập trình:
  + Spring MVC: Dựa vào phong cách lập trình imperative, trong đó chúng ta chỉ định từng bước xử lý yêu cầu một
  + Spring WebFlux: Dựa vào phong cách lập trình reactive, trong đó các định nghĩa các phản ứng (reactive) trước để xử lý dữ liệu khi nó có sẵn
* Cấu hình:
  + Spring MVC: Sử dụng @Controller và @RestController để xác định các controller và hỗ trợ cấu hình truyền thống dựa trên DispatcherServlet
  + Spring WebFlux: Sử dụng @Controller và @RestController giống như Spring MVC, nhưng cũng có thể sử dụng RouterFunctions để định nghĩa các endpoint mà không cần DispatcherServlet

**Mô hình giữa Spring MVC và Spring WebFlux**

A diagram of a network

Description automatically generated

* Lớp thứ nhất về cú pháp lập trình: Webflux ngoài cú pháp dựa trên các annotation trong Spring MVC còn có thể viết chương trình theo cú pháp hoàn toàn mới đó là Router Function. Router Functions chỉ áp dụng cho Webflux
* Lớp thứ hai về bản thân framework ta sử dụng: Spring MVC và Spring Webflux
* Lớp thứ ba là chuẩn giao tiếp với các web server mà ứng dụng chạy trên đó: Spring MVC tuân theo chuẩn Servlet, còn Spring Webflux thì tuân theo chuẩn mới là Reactive Streams cho giao thức HTTP. Với chuẩn Reactive Streams thì các ứng dụng Webflux sẽ được chạy trên các webserver hỗ trợ các xử lý Non-Blocking để tối ưu tài nguyên tính toán
* Lớp thứ tư là các web server hỗ trợ các chuẩn giao tiếp ở bên trên: Đối với Spring MVC thì ứng dụng sẽ cần phải được chạy trên các Servlet Container ví dụ như Tomcat, Jetty … nhưng sang đến Spring Webflux thì lại không hạn chế như vậy. Đối với các servlet container tuân theo chuẩn Servlet 3.1 có hỗ trợ Non-Blocking IO, thì chúng cũng có thể chạy cùng với các ứng dụng Webflux. Ngoài ra, đối với các ứng dụng Webflux thì các Servlet container sẽ không phải là sự lựa chọn duy nhất, mà các HTTP Server như Netty hay Undertow có thể là những lựa chọn tuyệt vời khác cho việc xử lý không đồng bộ các HTTP Request đến từ ứng dụng Webflux