**Spring framework**

# **2. RestController vs Controller**

# **1. Giới thiệu Spring MVC, Spring boot**

Trong Spring Framework, @Controller và @RestController đều được sử dụng để xử lý các yêu cầu HTTP, nhưng chúng có một số điểm khác biệt quan trọng trong cách chúng xử lý dữ liệu trả về và mục đích sử dụng:

### **1. @Controller**

* **Chức năng**: Đây là annotation cơ bản dùng để đánh dấu một lớp là controller trong Spring MVC. Nó chủ yếu được sử dụng để xử lý các yêu cầu web và trả về các trang JSP hoặc các tài nguyên khác.
* **Kiểu trả về**: Khi sử dụng @Controller, bạn thường sẽ trả về tên của một view (như JSP, Thymeleaf, v.v.). Để trả về dữ liệu dạng JSON hoặc XML, bạn thường sử dụng thêm @ResponseBody cho các phương thức.
* **Trường hợp sử dụng**: Thích hợp cho các ứng dụng web truyền thống nơi bạn cần trả về các trang HTML hoặc view.

### **2. @RestController**

* **Chức năng**: Đây là một dạng mở rộng của @Controller và được giới thiệu từ Spring 4.0. Nó được thiết kế để tạo các RESTful web service.
* **Kiểu trả về**: Tất cả các phương thức trong một @RestController sẽ tự động có @ResponseBody áp dụng, có nghĩa là dữ liệu trả về sẽ được chuyển đổi thành JSON hoặc XML (tùy thuộc vào cấu hình) mà không cần phải chỉ định rõ.
* **Trường hợp sử dụng**: Thích hợp cho các ứng dụng RESTful API, nơi bạn cần trả về dữ liệu (thường là JSON) thay vì các view HTML.

| Đặc điểm | @Controller | @RestController |
| --- | --- | --- |
| Kiểu trả về | View (HTML, JSP, v.v.) | Dữ liệu (JSON, XML) |
| Cần phải sử dụng @ResponseBody | Không cần (trừ khi cần) | Không cần (tự động áp dụng) |
| Trường hợp sử dụng | Ứng dụng web truyền thống | RESTful API |

# **2. Routing**

### **1. Cấu trúc cơ bản**

Trong Spring Boot, routing thường được thực hiện thông qua các annotation trong các controller. Các annotation này cho phép bạn định nghĩa các điểm cuối (endpoints) cho các yêu cầu HTTP.

### **2. Các Annotation Chính**

* **@RestController**: Được sử dụng để đánh dấu một lớp là controller cho RESTful web service. Tất cả các phương thức trong lớp sẽ tự động trả về dữ liệu (JSON hoặc XML).
* **@Controller**: Được sử dụng để đánh dấu một lớp là controller cho ứng dụng web truyền thống, thường sẽ trả về view.
* **@RequestMapping**: Được sử dụng để ánh xạ các yêu cầu HTTP đến các phương thức trong controller. Bạn có thể xác định kiểu yêu cầu (GET, POST, PUT, DELETE) và đường dẫn (URL).
* **@GetMapping**, **@PostMapping**, **@PutMapping**, **@DeleteMapping**: Là các shortcut cho @RequestMapping với phương thức HTTP tương ứng.

# **3. Tìm hiểu cách mà controller nhận input đầu vào là : params, body, path variable, header....**

### **1. Params (Tham số truy vấn)**

Tham số truy vấn là các giá trị được gửi từ client trong chuỗi truy vấn của URL. Chúng thường được sử dụng để gửi dữ liệu không bắt buộc từ client.

### **2. Body (Nội dung yêu cầu)**

Nội dung của yêu cầu thường được sử dụng khi bạn gửi dữ liệu phức tạp, chẳng hạn như JSON hoặc XML, đến server. Dữ liệu này thường được gửi trong phần body của yêu cầu POST hoặc PUT.

### **3. Path Variable (Biến đường dẫn)**

Biến đường dẫn cho phép bạn lấy các phần cụ thể từ đường dẫn URL. Chúng thường được sử dụng trong các trường hợp mà bạn cần xác định tài nguyên cụ thể.

### **4. Header (Tiêu đề yêu cầu)**

Tiêu đề của yêu cầu có thể chứa thông tin bổ sung về yêu cầu, chẳng hạn như loại nội dung, mã thông báo xác thực, v.v. Bạn có thể nhận các tiêu đề này bằng cách sử dụng annotation @RequestHeader.

# 