RestController vs Controller

**1. Sự khác nhau giữa @RestController và @Controller**

* **@Controller**:
  + Được dùng để tạo các controller trong Spring MVC (thường phục vụ cho các trang web với giao diện HTML).
  + Trả về **View** (HTML hoặc JSP) sau khi xử lý dữ liệu, thay vì trả về dữ liệu trực tiếp.
  + Dữ liệu sẽ được ánh xạ vào ModelAndView hoặc sử dụng các **template engine** như Thymeleaf để tạo view.
* **@RestController**:
  + Là sự kết hợp của @Controller và @ResponseBody.
  + Trả về **JSON hoặc XML** trực tiếp cho client thay vì view (phù hợp cho các API).
  + Thường dùng trong các ứng dụng RESTful API, nơi server trả về dữ liệu thay vì giao diện.

**Trường hợp sử dụng**:

* **@Controller**: Dùng trong các ứng dụng có giao diện người dùng, hiển thị các trang HTML.
* **@RestController**: Dùng cho các API cung cấp dữ liệu JSON, XML (REST API).

**2. Routing**

Routing trong Spring Boot là quá trình định tuyến các yêu cầu HTTP tới các controller tương ứng thông qua các endpoint (URL).

* **Mapping với HTTP Method**:
  + @GetMapping: Xử lý yêu cầu GET.
  + @PostMapping: Xử lý yêu cầu POST.
  + @PutMapping: Xử lý yêu cầu PUT.
  + @DeleteMapping: Xử lý yêu cầu DELETE.

Vd:

@RestController

@RequestMapping("/api")

public class MyController {

@GetMapping("/hello")

public String sayHello() {

return "Hello, World!";

}

@PostMapping("/create")

public String createSomething(@RequestBody String data) {

return "Created: " + data;

}

}

**3. Nhận Input đầu vào từ Request**

* **Path Variable** (@PathVariable): Dùng để lấy các giá trị từ URL.

@GetMapping("/user/{id}")

public String getUserById(@PathVariable String id) {

return "User ID: " + id;

}

* **Request Parameter** (@RequestParam): Lấy các giá trị từ query string.

@GetMapping("/search")

public String search(@RequestParam String keyword) {

return "Search keyword: " + keyword;

}

* **Request Body** (@RequestBody): Nhận dữ liệu JSON/XML gửi kèm trong body của request (thường dùng trong POST, PUT).

@PostMapping("/add")

public String addUser(@RequestBody User user) {

return "Added user: " + user.getName();

}

**Request Header** (@RequestHeader): Lấy giá trị từ header của yêu cầu HTTP.

@GetMapping("/info")

public String getInfo(@RequestHeader("User-Agent") String userAgent) {

return "User Agent: " + userAgent;

}

4. Thực hành viết API với các method và các loại input đầu vào với dữ liệu tĩnh

Vd:

@RestController

@RequestMapping("/api")

public class UserController {

// GET method với Path Variable

@GetMapping("/users/{id}")

public String getUser(@PathVariable String id) {

return "User ID: " + id;

}

// POST method với Body

@PostMapping("/users")

public String addUser(@RequestBody Map<String, String> user) {

return "Added user: " + user.get("name");

}

// PUT method với Path Variable và Body

@PutMapping("/users/{id}")

public String updateUser(@PathVariable String id, @RequestBody Map<String, String> user) {

return "Updated user ID " + id + " with name: " + user.get("name");

}

// DELETE method với Path Variable

@DeleteMapping("/users/{id}")

public String deleteUser(@PathVariable String id) {

return "Deleted user ID: " + id;

}

}

**5. Validate Input trong Controller**

Spring Boot hỗ trợ validation bằng các annotation như @NotNull, @Size, @Email thông qua **Java Bean Validation** (javax.validation.constraints).

**Ví dụ**: Validating user input.

public class User {

@NotNull(message = "Name cannot be null")

@Size(min = 2, message = "Name must have at least 2 characters")

private String name;

@Email(message = "Email should be valid")

private String email;

}

Trong controller, sử dụng @Valid để kích hoạt validation.

@PostMapping("/users")

public ResponseEntity<String> addUser(@Valid @RequestBody User user) {

return ResponseEntity.ok("Added user: " + user.getName());

}

**6. Handle Exception**

* **Custom Exception Handler**: Tạo các exception handler để xử lý lỗi cụ thể bằng @ControllerAdvice và @ExceptionHandler.

**Ví dụ**:

1. **Tạo Exception Class**:

public class UserNotFoundException extends RuntimeException {

public UserNotFoundException(String message) {

super(message);

}

}

1. **Tạo Exception Handler**:

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(UserNotFoundException.class)

public ResponseEntity<String> handleUserNotFound(UserNotFoundException ex) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(ex.getMessage());

}

@ExceptionHandler(MethodArgumentNotValidException.class)

public ResponseEntity<String> handleValidationExceptions(MethodArgumentNotValidException ex) {

String message = ex.getBindingResult().getAllErrors().get(0).getDefaultMessage();

return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body("Validation Error: " + message);

}

}