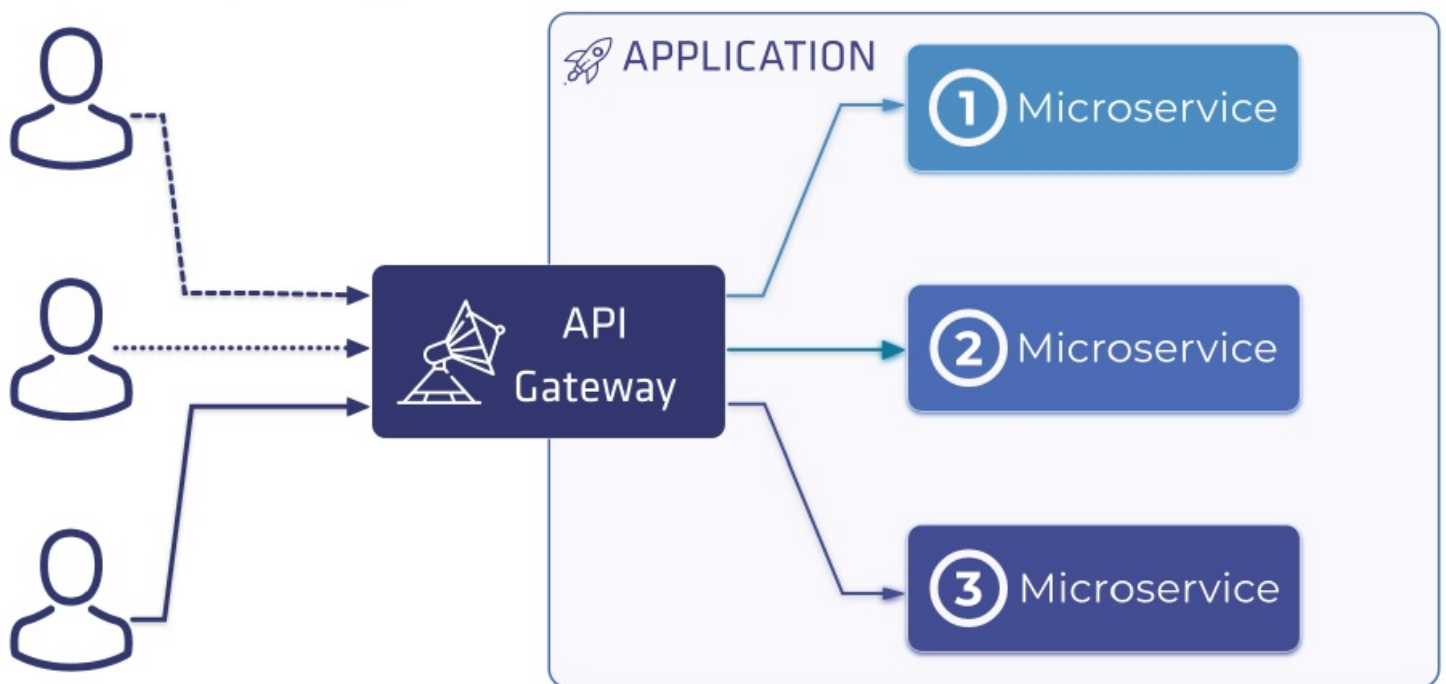


OpenAPI để tạo văn bản chú thích cho REST API

Thực tế ngày hôm nay, nhiều sản phẩm, ứng dụng Spring Boot đã chuyển sang mô hình Single Page App, Mobile App nối vào REST API. Với doanh nghiệp lớn, mô hình ứng dụng phức tạp, người ta chuyển dần sang microservice.

API Gateway



Trong mô hình này, vai trò của lập trình viên back end và front end rất rõ ràng. Lập trình viên back end tập trung vào xử lý thao tác dữ liệu, kết nối đến các dịch vụ phía sau. Còn lập trình viên front end cần những kỹ năng code HTML, CSS, JavaScript hoặc IOS, Android, Flutter hay React Native.

Để hai nhóm lập trình viên này có thể nói chuyện được với nhau, phía front end cần giải thích trình bày mock up giao diện để lập trình viên back end hình dung ra luồng ứng dụng chạy. Còn lập trình viên back end phải document rõ ràng, chi tiết API mình đang xây dựng để lập viên front end kết nối đến.

Trong mô hình Continuous Development - Continuous Integration, thì phần back end luôn phải cung cấp văn bản cập nhật nhất. Swagger, Open API là các kỹ thuật để tạo ra văn bản mô tả API rất tiện lợi.

← → ↻ 🏠 ⓘ localhost:8080/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config/ ☆ (n) m 🌟 👤 ⋮

Swagger
Supported by SMARTBEAR

/v3/api-docs **Explore**

BOOK REST API Document 0.0.1-SNAPSHOT OAS3

[/v3/api-docs](#)

Books REST API

[Terms of service](#)

[Trình Minh Cường - Website](#)

[Send email to Trình Minh Cường](#)

[Apache 2.0](#)

http://localhost:8080/swagger-ui.html

Servers

http://localhost:8080 - Generated server url ▼

book-api



GET /api/books/{bookId} Get a book by id

PUT /api/books/{bookId} Update a book by id

DELETE /api/books/{bookId}

PATCH /api/books/{bookId}

GET /api/books/ Get all books

POST /api/books/ Create a new book

Thực hành từng bước

1. Bổ xung dependency vào pom

Bổ xung dependency vào [pom.xml](#):

```
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-ui</artifactId>
  <version>1.6.3</version>
</dependency>
```

2. Chạy ứng dụng

Truy cập vào địa chỉ <http://localhost:8080/swagger-ui.html>

3. Bổ xung mô tả cho từng phương thức

Để có một mô tả chi tiết như hình dưới đây

The screenshot shows the Swagger UI interface for an API named 'book-api'. The selected endpoint is a GET request to '/api/books/{bookId}' with the description 'Get a book by id'. Under the 'Parameters' tab, there is a single path parameter named 'bookId' which is required and has a description 'id of book to be searched'. Its type is 'integer(\$int64)'. A text input field next to it contains the value '1', and a red arrow points to it with the Vietnamese text 'Điền tham số vào đây' (Enter parameter here). Below the parameters, there is a blue 'Execute' button. The 'Responses' section shows a 200 status code with the description 'OK'. It includes a 'Media type' dropdown set to '*/*' and a checkbox for 'Controls Accept header'. An 'Example Value' tab is selected, showing a JSON object: { 'id': 15, 'title': 'Để mền phiếu lưu ký', 'author': 'Tô Hoài' }. A red arrow points to this JSON example with the Vietnamese text 'Dữ liệu mẫu trả về' (Sample data returned).

- @Operation(summary = "Get a book by id") : Mô tả tên phương thức, sẽ được OpenAPI trích xuất
- @Parameter(description = "id of book to be searched") : Mô tả tên tham số sẽ được OpenAPI trích xuất

```
@GetMapping("/{bookId}")
@Operation(summary = "Get a book by id")
public ResponseEntity<Book> findBookById(
    @Parameter(description = "id of book to be searched")
    @PathVariable long bookId) {
    Optional<Book> optionalBook = bookService.findById(bookId);
    if (optionalBook.isPresent()) {
        return ResponseEntity.ok(optionalBook.get()); // return 200, with json body
    } else {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(null); // return 404, with i
    }
}
```

4. Làm sao để có mô tả ràng buộc và dữ liệu mẫu?

Nếu OpenAPI chỉ trích xuất kiểu dữ liệu cho từng trường thì chưa đủ. Để front end developer dễ hình dung hơn, cần phải mô tả ràng buộc (bộc phải có, độ dài trường...) và dữ liệu mẫu.



Xem mô tả trong file [BookPOJO.java](#)

Hãy sử dụng các annotation:

- @NotBlank : không được rỗng
- @Size(min = 5, max = 400, message = "Tên sách từ 4 đến 400 ký tự") : số lượng ký tự min và max. Tham số message dùng để báo lỗi khi dữ liệu phạm quy.

@Data

```
public class BookPOJO {
    @NotBlank
    @Size(min = 5, max = 400, message = "Tên sách từ 4 đến 400 ký tự")
    @Schema(description = "Tên sách", example = "Đế mèn phiêu lưu ký", required = true)
    private String title;

    @NotBlank
    @Size(min = 5, max = 200)
    @Schema(description = "Tác giả", example = "Tô Hoài", required = true)
    private String author;
}
```

```

BookPOJO {
  description:
    book to be created

  title*
    string
    maxLength: 400
    minLength: 5
    example: Để mền phiêu lưu ký

  author*
    string
    maxLength: 200
    minLength: 5
    example: Tô Hoài

  Tác giả
}

```

Các ràng buộc



Thêm ví dụ Validation Annotations. Tham khảo bài này [Java Bean Validation Basics](#)

```

public class User {

    @NotNull(message = "Name cannot be null")
    private String name;

    @AssertTrue
    private boolean working;

    @Size(min = 10, max = 200, message
        = "About Me must be between 10 and 200 characters")
    private String aboutMe;

    @Min(value = 18, message = "Age should not be less than 18")
    @Max(value = 150, message = "Age should not be greater than 150")
    private int age;

    @Email(message = "Email should be valid")
    private String email;
}

```

Thêm ví dụ về annotation `@Schema` ở file [Contact.java](#)

```

public class Contact implements Serializable {
    @Schema(description = "Unique ID of Contact.", example = "1", required = true)
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

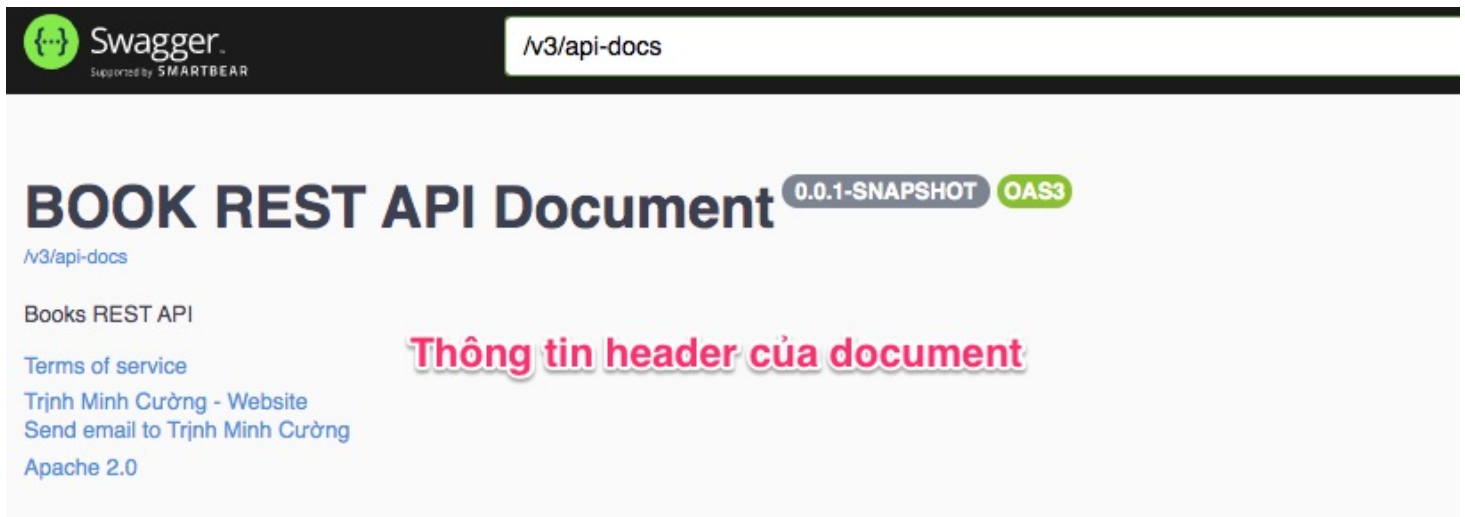
    @Schema(description = "Full name of contact.", example = "Trịnh Minh Cường", required = true)
    @NotBlank
    @Size(max = 100)
    private String name;

    @Schema(description = "Mobile contact", example = "0902209011", required = false)
    @Pattern(regexp = "^\\+?[0-9]{10,11}$", message = "Mobile")
    @Size(min=10, max = 11)
    private String phone;

    @Schema(description = "Email of contact", example = "cuong@techmaster.vn", required = true)
    @Email(message = "Email Address")
    @Size(max = 100)
    private String email;
}

```

5. Bổ xung thông tin tổng quan cho OpenAPI



The screenshot shows a Swagger UI interface. At the top, there's a Swagger logo and a search bar containing '/v3/api-docs'. Below this, the main heading is 'BOOK REST API Document' with two version tags: '0.0.1-SNAPSHOT' and 'OAS3'. Under the heading, there are links for 'Books REST API', 'Terms of service', 'Trịnh Minh Cường - Website', 'Send email to Trịnh Minh Cường', and 'Apache 2.0'. A red text overlay with a white border reads 'Thông tin header của document'.

Xem file [OpenAPIConfig.java](#)

```

@Bean
public OpenAPI customOpenAPI(
    @Value("${application-description}") String appDescription,
    @Value("${application-version}") String appVersion) {

    return new OpenAPI()
        .info(new Info().title("BOOK REST API Document")
            .version(appVersion)
            .contact(new Contact().name("Trịnh Minh Cường").email("cuong@techmaster.vn").url("http://springdoc.org")))
        .description(appDescription)
        .termsOfService("http://swagger.io/terms/")
        .license(new License().name("Apache 2.0")
            .url("http://springdoc.org")));
}

```

Hai thuộc tính `application-description` và `application-version` được lấy từ `application.properties` inject qua tham số phương thức tạo Bean

```

application-description=@project.description@
application-version=@project.version@

```

Các tham số còn lại cũng dễ hiểu.

6. Chỉ định OpenAPI bỏ qua một phương thức REST API

Cũng có lúc chúng ta muốn ẩn đi secret API, hoặc API đã cũ (deprecated) hoặc API đang ở beta testing. Chúng ta không muốn OpenAPI hiển thị document những API này.

Trong annotation `@Operation` hãy thêm tham số `hidden = true` như đoạn code dưới.

```

@Operation(summary = "Beta API method, đang kiểm thử", hidden = true)
@GetMapping(value = "/top5")
public ResponseEntity<List<Book>> getTop5Books() {
    List<Book> books = bookService.findAll();
    List<Book> top5books = books.stream().limit(5).collect(Collectors.toList());
    return ResponseEntity.ok(top5books);
}

```

7. Gom nhóm cụm các API cùng chức năng

Một ứng dụng Spring Boot có thể có nhiều model, hay domain dịch vụ khác nhau, nếu liệt kê hết trong một trang thì rất rối rắm. Chúng ta nên gom những phương thức REST thuộc cùng một domain lại thì sẽ dễ hiểu, dễ đọc hơn. Trong ví dụ này, tôi có 2 models `Book.java` và `Contact.java`, tương ứng có 2 REST Controller `BookAPI.java` và `ContactAPI.java`.

Trong file `OpenAPIConfig.java`, tôi bổ xung thêm 2 bean:

```
@Bean
public GroupedOpenApi bookOpenApi() { //Phục vụ cho domain Book
    String paths[] = {"/api/books/**"}; //có thể chứa nhiều đường dẫn
    return GroupedOpenApi.builder().group("books").pathsToMatch(paths)
        .build();
}

@Bean
public GroupedOpenApi contactOpenApi() { //Phục vụ cho domain Contact
    String paths[] = {"/api/contacts/**"};
    return GroupedOpenApi.builder().group("contacts").pathsToMatch(paths)
        .build();
}
```

Giờ bạn có thể chọn xem nhóm REST API cùng một domain

The screenshot shows the Swagger UI interface in a web browser. The address bar shows 'localhost:8080/swagger-ui/index.h...'. The Swagger logo is in the top left. A dropdown menu labeled 'Select a definition' has 'books' selected. Below this, the title 'BOOK REST API Document' is displayed with version '0.0.1-SNAPSHOT' and 'OAS3' specification. A red arrow points to the 'books' dropdown with the text 'Chọn nhóm API mà bạn quan tâm'. Below the title, there are links for 'Books REST API', 'Terms of service', 'Trịnh Minh Cường - Website', 'Send email to Trịnh Minh Cường', and 'Apache 2.0'. At the bottom, the 'Servers' section shows 'http://localhost:8080 - Generated server url'. The main content area shows the 'book-api' group with two endpoints: a GET endpoint '/api/books/{bookId}' for 'Get a book by id' and a PUT endpoint '/api/books/{bookId}' for 'Update a book by id'.

Swagger. Supported by SMARTBEAR

Select a definition books

BOOK REST API Document

0.0.1-SNAPSHOT OAS3

[/v3/api-docs/books](#)

Books REST API

[Terms of service](#)

[Trịnh Minh Cường - Website](#)

[Send email to Trịnh Minh Cường](#)

[Apache 2.0](#)

Chọn nhóm API mà bạn quan tâm

Servers

http://localhost:8080 - Generated server url

book-api

GET /api/books/{bookId} Get a book by id

PUT /api/books/{bookId} Update a book by id

Bonus: HAL Explorer

HAL viết tắt của cụm từ Hypertext Application Language. Nếu chúng ta tạo ra REST API cần có một cơ chế để giúp lập trình viên phía client tìm hiểu, khám phá các phương thức REST API. Để dùng HAL Explorer hãy bổ xung vào [pom.xml](#)

```
<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-rest-hal-explorer</artifactId>
</dependency>
```

Sau đó truy cập vào địa chỉ này <http://localhost:8080/explorer>

The screenshot shows the HAL Explorer web application running in a browser. The browser address bar shows `localhost:8080/explorer/index.html#uri=http://localhost:8080`. The application has a dark header with the title "HAL Explorer" and navigation links for "Theme", "Layout", and "About". Below the header, there is a search bar with the text "http://localhost:8080" and a "Go!" button. The main content area is divided into two columns. The left column contains a "Links" section with a table of relations. The right column contains three sections: "Response Status", "Response Headers", and "Response Body".

Links

Relation	Name	Title	HTTP	Doc
contacts			<	
books			<	
profile			<	

Response Status

200 (OK)

Response Headers

connection	keep-alive
content-type	application/hal+json
date	Tue, 09 Mar 2021 04:35:41 GMT
keep-alive	timeout=60
transfer-encoding	chunked
vary	Origin, Access-Control-Request-Method, Access-Control-Request-Headers

Response Body

```
{
  "_links": {
    "contacts": {
      "href": "http://localhost:8080/contacts{?page,size}",
      "templated": true
    },
    "books": {
      "href": "http://localhost:8080/books{?page,size}",
      "templated": true
    },
    "profile": {
      "href": "http://localhost:8080/profile"
    }
  }
}
```

Lấy danh sách Books

Edit Headers

http://localhost:8080/books

Go!

JSON Properties

Lấy danh sách books

```
{  "page": {    "size": 20,    "totalElements": 6,    "totalPages": 1,    "number": 0  }}
```

Links

Relation	Name	Title	HTTP	Doc
self			<	
profile			<	

Embedded Resources

books

books [0]

books [1]

books [2]

books [3]

books [4]

Kết quả trả về

Response Status

200 (OK)

Response Headers

connection	keep-alive
content-type	application/hal+json
date	Tue, 09 Mar 2021 04:39:45 GMT
keep-alive	timeout=60
transfer-encoding	chunked
vary	Origin, Access-Control-Request-Method, Access-Control-Request-Headers

Response Body

```
{  "_embedded": {    "books": [      {        "title": "Ulysses",        "author": "James Joyce",        "_links": {          "self": {            "href": "http://localhost:8080/books/1"          },          "book": {            "href": "http://localhost:8080/books/1"          }        }      }    ]  },}
```

Có thể thao tác : Thêm, Sửa, Xoá bản ghi qua giao diện của HAL Explorer

HAL ExplorerThemeLayoutAbout

Edit Headershttp://localhost:8080/Go!

2 Columns3 Columns

JSON Properties

```
{  "page": {    "size": 20,    "totalElements": 6,    "totalPages": 1,    "number": 0  }}
```

Links

2. Sẽ thấy các nút thao tác dữ liệu

Relation	Name	Title	HTTP	Doc
self			<div><+>>✖</div>	
profile			<div><+>>✖</div>	

Embedded Resources

books

books [0]

books [1]

books [2]

books [3]

books [4]

books [5]

Response Status

200 (OK)

Response Headers

connection	keep-alive
content-type	application/hal+json
date	Tue, 09 Mar 2021 04:43:25 GMT
keep-alive	timeout=60
transfer-encoding	chunked
vary	Origin, Access-Control-Request-Method, Access-Control-Request-Headers

Response Body

```
{  "_embedded": {    "books": [      {        "title": "Ulysses",        "author": "James Joyce",        "_links": {          "self": {            "href": "http://localhost:8080/books/1"          },          "book": {            "href": "http://localhost:8080/books/1"          }        }      }    ]  },  ...}
```

Thêm mới một bản ghi

HTTP Request Input

Thêm mới một bản ghi



URI

http://localhost:8080/books

HTTP Method

Post



Spring Profile (JSON Schema)

Author

Gabriel Garcia Marquez

Title

Trăm năm cô đơn

Body

```
{
  "author": "Gabriel Garcia Marquez",
  "title": "Trăm năm cô đơn"
}
```

Go!