**Giới thiệu**

**Spring Boot** hỗ trợ chúng ta tạo ra RESTful API một cách nhanh chóng và tiện lợi, giúp sản phẩm được vận hành nhanh nhất có thể.

Tuy nhiên, Việc deploy nhanh chóng một services không đồng nghĩa với việc nó có thể sử dụng được. Thông thường, tất cả các API sau khi được đưa lên sẽ phải đi kèm với document mô tả, để bất kì ai sử dụng đến thì có thể tra cứu.

Thật không may là việc làm document chưa bao giờ là dễ dàng cả :(( từ lí do này, Swagger ra đời để giúp chúng ta mô tả tài liệu dự án một cách nhanh chóng bằng annotation.

Trong bài có đề cập các kiến thức:

1. [Spring Boot](https://loda.me/spring-boot-0-series-lam-chu-spring-boot-zero-to-hero-loda1558963914472)
2. [jpa](https://loda.me/spring-boot-11-huong-dan-spring-boot-jpa-my-sql-loda1558687596060)
3. [lombok](https://loda.me/general-huong-dan-su-dung-lombok-giup-code-java-nhanh-hon-69-loda1552789752787)

**cài đặt**

Trong bài này, tôi sẽ hướng dẫn các bạn sử dụng Swagger2 và tuân theo các quy tắc cảu **Swagger Specification 2.0** nhé.

Tại thời điểm viết bài này, hiện phiên bản mới nhất là 3, tuy nhiên, nó sẽ là **OpenAPI 3.0**.

*pom.xml*

<?xml version="1.0" encoding="UTF-8"?>

<projectxmlns="http://maven.apache.org/POM/4.0.0"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xsi:schemaLocation="http://maven.apache.org/POM/4.0.0https://maven.apache.org/xsd/maven-4.0.0.xsd"><modelVersion>4.0.0</modelVersion><parent><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-parent</artifactId><version>2.0.5.RELEASE</version><relativePath/> <!-- lookup parent from repository -->

</parent><groupId>me.loda.spring</groupId><artifactId>example-independent-maven-spring-project</artifactId><version>0.0.1-SNAPSHOT</version><name>example-independent-maven-spring-project</name><description>Demo project for Spring Boot</description><properties><java.version>1.8</java.version></properties><dependencies><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-web</artifactId></dependency><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-devtools</artifactId><scope>runtime</scope><optional>true</optional></dependency><dependency><groupId>org.projectlombok</groupId><artifactId>lombok</artifactId><optional>true</optional></dependency><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-test</artifactId><scope>test</scope></dependency>

<dependency><groupId>io.springfox</groupId><artifactId>springfox-swagger2</artifactId><version>2.9.2</version></dependency><dependency><groupId>io.springfox</groupId><artifactId>springfox-swagger-ui</artifactId><version>2.9.2</version></dependency><!--spring jpa-->

<dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-data-jpa</artifactId></dependency><!--in memory database-->

<dependency><groupId>com.h2database</groupId><artifactId>h2</artifactId><scope>runtime</scope></dependency></dependencies><build><plugins><plugin><groupId>org.springframework.boot</groupId><artifactId>spring-boot-maven-plugin</artifactId></plugin></plugins></build></project>

springfox là một thư viện java implementation của Swagger Specification.

springfox-swagger2 chứa core của swagger, giúp chúng ta khai báo document cho api.

springfox-swagger-ui giúp chúng ta biểu diễn tài liệu dưới dạng web view, dễ nhìn và test.

**Tổng quan Swagger**

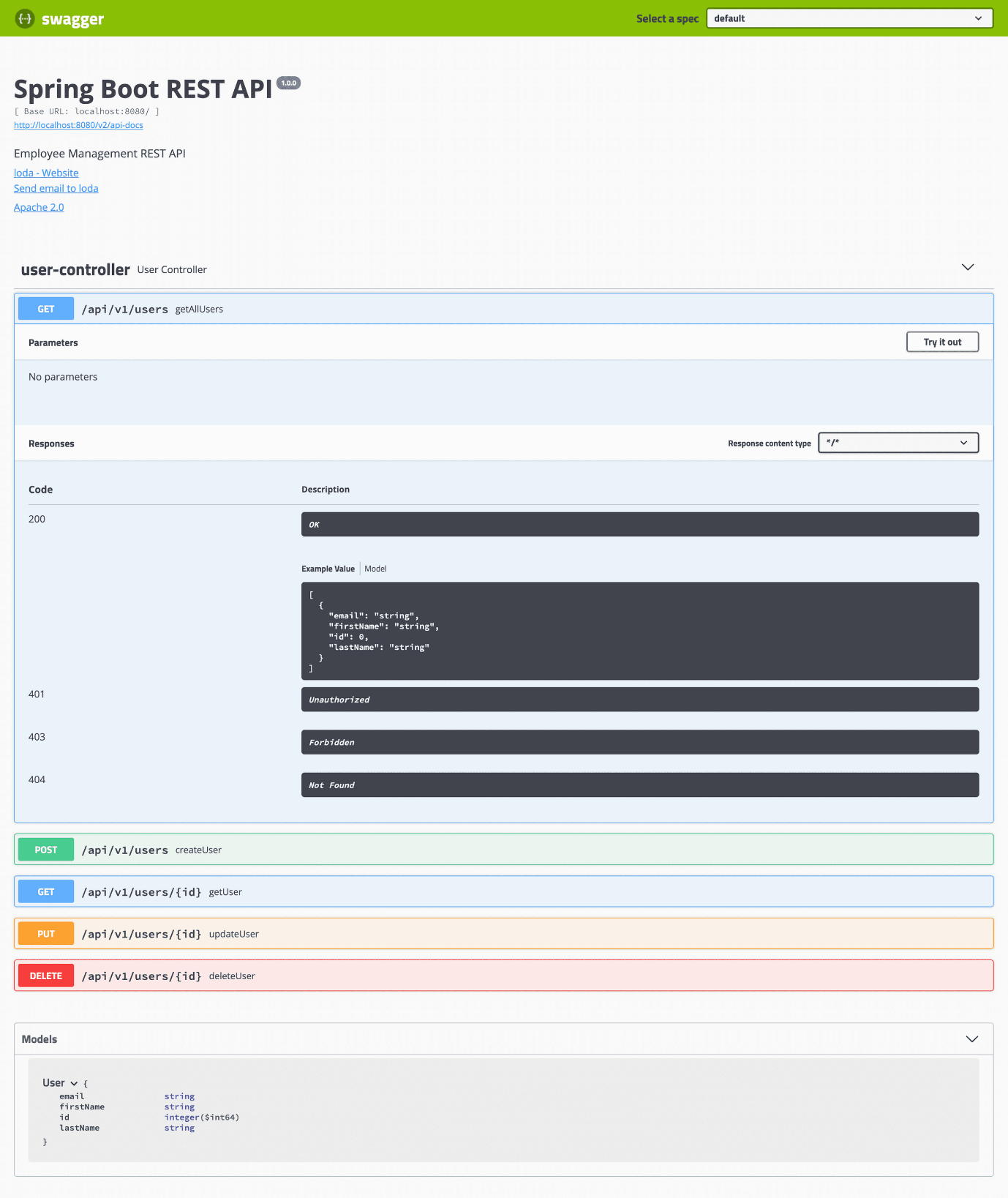
Để sử dụng cơ bản thì Swagger cung cấp một số các Annotations hữu ích sau:

| SYNTAX | DESCRIPTION |
| --- | --- |
| [@Api](https://loda.me/77453c73a2024614895d8abb4f1bd982) | Đánh dấu 1 class là nơi chứa các API |
| [@ApiModel](https://loda.me/3724071b66834c85a748ccc3123ff1a8) | Đánh dấu 1 class là Swagger Model |
| [@ApiModelProperty](https://loda.me/0191ddd037314f64950f2faeed4ba5bf) | Bổ sung các thông tin cho |
| [@ApiOperation](https://loda.me/a2de8039944a400ea0868be426be437e) | Mô tả cho một API và response của nó |
| [@ApiParam](https://loda.me/f1611159dcab45daac829d972ab88114) | Mô tả các parameter |
| [@ApiResponse](https://loda.me/0b039ef67daa4c059ba68ba9e738c083) | Mô tả status code của response |
| [@ApiResponses](https://loda.me/6c26eca3b3ba4b6987c6b8c9e69b1a28) | Mô tả danh sách các status code của response |

Có thể xem thêm tại: <https://github.com/swagger-api/swagger-core/wiki/annotations>

Chúng ta đi vào thực hành thử nhé.

Đại loại sau khi làm xong, chúng ta sẽ có 1 web view document như thế này:



**Prepare**

Tạo ra class Model

*User.java*

@Data

@Entity

@Table

@NoArgsConstructor

@AllArgsConstructor

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String firstName;

private String lastName;

private String email;

}

*UserRepository.java*

@Repository

public interface UserRepository extends JpaRepository<User, Long> {

}

Khi đã có được Model và Repository, chúng ta sẽ tạo Controller để thao tác liên quan tới User nhé.

@RestController

@RequestMapping("/api/v1")

@RequiredArgsConstructor

public class UserController {

private final UserRepository userRepository;

@GetMapping("/users")

public List<User> getAllUsers() {

return userRepository.findAll();

}

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

public User getUser(@PathVariable("id") Long id) {

return userRepository.findById(id).orElse(new User());

}

@PostMapping("/users")

public User createUser(@Valid @RequestBody User user) {

return userRepository.save(user);

}

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

public User updateUser(@PathVariable("id") Long id, @Valid @RequestBody User user) {

user.setId(id);

return userRepository.save(user);

}

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

public void deleteUser(@PathVariable("id") Long id) {

userRepository.deleteById(id);

}

}

Âu khê, tạm thời như thế đã 😂

Bây giờ vào phần chính, config Swagger cho dự án của chúng ta.

**Config Swagger**

Thật may mắn, trước khi đi vào các custom phức tạp thì Swagger hỗ trợ chúng ta tự động sinh ra tài liệu một cách mặc định mà chưa cần phải khai báo bất kì annotation nào đã giới thiệu ở trên.

Chỉ cần tạo ra đối tượng Docket của Swagger và nó sẽ quét hết các địa chỉ API mà bạn chỉ định, rồi tự động sinh ra tài liệu cơ bản cho chúng ta.

@Configuration

@EnableSwagger2

public class Swagger2Config {

@Bean

public Docket api() {

return new Docket(DocumentationType.SWAGGER\_2).select()

.apis(RequestHandlerSelectors.basePackage("me.loda.spring.swagger.controller"))

.paths(PathSelectors.regex("/.\*"))

.build()

.apiInfo(apiEndPointsInfo());

}

private ApiInfo apiEndPointsInfo() {

return new ApiInfoBuilder().title("Spring Boot REST API")

.description("Employee Management REST API")

.contact(new Contact("loda", "https://loda.me/", "loda.namnh@gmail.com"))

.license("Apache 2.0")

.licenseUrl("http://www.apache.org/licenses/LICENSE-2.0.html")

.version("1.0.0")

.build();

}

}

Các thứ cần lưu ý bao gồm:

1. Để Swagger hoạt động, bạn nhớ kích hoạt nó bằng @EnableSwagger2.
2. Bạn có thể chọn nơi chứa các API bằng RequestHandlerSelectors. Nếu muốn quét hết cả project, có thể xài RequestHandlerSelectors.any()
3. Bạn có thể chỉ định bộ lọc cho các api bằng PathSelectors. Nếu muốn quét tất cả, chọn PathSelectors.any().

@SpringBootApplication

public class App {

public static void main(String[] args) {

SpringApplication.run(App.class, args);

}

}

Bây giờ, chạy thử và vào địa chỉ <http://localhost:8080/swagger-ui.html> để xem thành quả nhé.

**Góc custom**

Bạn có thể chỉ định rõ hơn các mô tả của tài liệu bằng cách sử dụng các Annotation mà Swagger cung cấp.

*User.java*

@Data

@Entity

@Table

@NoArgsConstructor

@AllArgsConstructor

@ApiModel(value = "User model")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@ApiModelProperty(notes = "The database generated User ID")

private Long id;

private String firstName;

private String lastName;

private String email;

}

*UserController.java*

@RestController

@RequestMapping("/api/v1")

@RequiredArgsConstructor

@Api(value = "User APIs")

public class UserController {

private final UserRepository userRepository;

@ApiOperation(value = "Xem danh sách User", response = List.class)

@ApiResponses(value = {

@ApiResponse(code = 200, message = "Thành công"),

@ApiResponse(code = 401, message = "Chưa xác thực"),

@ApiResponse(code = 403, message = "Truy cập bị cấm"),

@ApiResponse(code = 404, message = "Không tìm thấy")

})

@GetMapping("/users")

public List<User> getAllUsers() {

return userRepository.findAll();

}

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

public User getUser(@PathVariable("id") Long id) {

return userRepository.findById(id).orElse(new User());

}

@PostMapping("/users")

public User createUser(

@ApiParam(value = "Đối tượng User cần tạo mới", required = true) @Valid @RequestBody User user

) {

return userRepository.save(user);

}

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

public User updateUser(@PathVariable("id") Long id, @Valid @RequestBody User user) {

user.setId(id);

return userRepository.save(user);

}

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

public void deleteUser(@PathVariable("id") Long id) {

userRepository.deleteById(id);

}

}

Tới đây bạn có thể sử dụng Swagger thoải mái rồi, tôi sẽ làm một bài khác về **OpenAPI 3.0**.