

VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY



**Môn:** Phát triển Ứng dụng Web Nâng cao

Lớp 20KTPM1

20127237 - NGUYỄN TẤN LỰC

20127507 - BÙI TRẦN HUÂN

20127224 - DƯƠNG ĐẶNG THÀNH LÂM

20127470 - THÂN MINH ĐỨC

19127629- NGUYỄN TÂN VIỆT

20127491- LÊ ĐỨC HANH

# Seminar 02

Giảng viên: Ngô Ngọc Đăng Khoa

Thành phố Hồ Chí Minh – 2023

---

# Validation

## Cài đặt dependency:

Thêm dòng implementation 'org.springframework.boot:spring-boot-starter-validation' trong build.gradle mục dependencies

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'mysql:mysql-connector-java:8.0.27'  
    implementation 'org.springframework.boot:spring-boot-starter-validation'  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.hibernate.validator:hibernate-validator:7.0.1.Final'  
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.0.2'  
}
```

Tham khảo: <https://hibernate.org/validator/>

## Tạo Class Exception để bắt lỗi trả về cho người dùng

1. Tạo thư mục exception
2. Cài đặt class GlobalExceptionHandler để bắt lỗi ném ra từ Validation

```
@RestControllerAdvice  
public class GlobalExceptionHandler {  
    @ExceptionHandler(MethodArgumentNotValidException.class)  
    @ResponseStatus(HttpStatus.BAD_REQUEST)  
    public ResponseEntity<String>  
    handleValidationException(MethodArgumentNotValidException e) {  
        StringBuilder errorMessage = new StringBuilder();  
        e.getBindingResult().getFieldErrors().forEach(fieldError -> {  
            errorMessage.append(fieldError.getDefaultMessage()).append(" ");  
        });  
        return ResponseEntity.badRequest().body(errorMessage.toString());  
    }  
  
    @ExceptionHandler(Exception.class)  
    public ResponseEntity<String> handleUnwantedException(Exception e) {  
        String errorMessage = "An unexpected error occurred: " + e.getMessage();  
        return  
        ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(errorMessage);  
    }  
}
```

@RestControllerAdvice để thông báo cho Spring biết đây là Class bắt lỗi từ Controller (Lỗi sẽ được bắt từ Controller ném ra các Exception

vd: `MethodArgumentNotValidException, Exception, ...`) tùy vào cài đặt do Dev ném ra, với Validation sẽ ném ra **BindException** nếu điều kiện đầu vào bị sai

`MethodArgumentNotValidException` là class kế thừa từ `BindException`

Ta phải cài đặt mặc định `handleUnwantedException` để tự động bắt các Exception khác, việc cài đặt class `GlobalExceptionHandler` sẽ tổng hợp được tất cả các Exception bắn ra từ chương trình, thay vì ta phải dùng try-catch để bắt cho từng Exception.

## Cài đặt điều kiện Validation cho các field cần thiết

```
public class Film {

    static final List<String> ERatings = Arrays.asList("NC-17", "R", "G",
    "PG-13", "PG");

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "film_id", nullable = false)
    private Short id;

    @Size(max = 255)
    @NotNull(message = "Title is not null")
    @Column(name = "title", nullable = false)
    private String title;

    @Lob
    @Column(name = "description")
    private String description;

    @Column(name = "release_year")
    private Integer releaseYear;

    @NotNull(message = "Duration is not null")
    @Column(name = "rental_duration", nullable = false)
    private Byte rentalDuration;

    @NotNull(message = "rentalRate is not null")
    @Column(name = "rental_rate", nullable = false, precision = 4, scale = 2)
    private BigDecimal rentalRate;

    @Column(name = "length")
    private Short length;

    @NotNull(message = "replacementCost is not Null")
    @Column(name = "replacement_cost", nullable = false, precision = 5, scale = 2)
    private BigDecimal replacementCost;

    @Lob
    @Column(name = "rating")
    private String rating;
```

Thêm các Annotation cùng message để nhận được nội dung lỗi để in ra trong `GlobalExceptionHandler`. Các Annotation phổ biến:

`@NotNull`, `@NotEmpty`, `@Email`, `@Min`, `@Max` ... tham khảo thêm tại

<https://reflectoring.io/bean-validation-with-spring-boot/#:~:text=Some%20of%20the%20most%20common.have%20at%20least%20one%20character>).

# Thêm Annotation @Valid vào trong Controller

```
@PutMapping(path =("/{id}")  
public ResponseEntity<String> updateFilmById(@PathVariable Short  
id, @RequestBody @Valid Film film) {  
    if (!Film.ERatings.contains(film.getRating())) {  
        return new ResponseEntity<>("Ratings value is not correct",  
            HttpStatus.BAD_REQUEST);  
    }  
    return filmService.updateFilmById(id, film);  
}
```

Khi thêm @Valid trước RequestBody film, Validation sẽ được thực hiện và bắt lỗi khi nhận RequestBody, (các điều kiện được đặt trong Class Film), nếu fail sẽ bắn ra lỗi BindException cho GlobalExceptionHandler nhận và trả về BAD\_REQUEST cho người dùng (được cài đặt tại class GlobalExceptionHandler), chỉ khi Pass được Validation thì hàm trong Controller mới được thực hiện!

## Cài đặt các điều kiện đặc biệt khác

Validation chỉ validate được các điều kiện cơ bản, nếu muốn cài đặt thêm các điều kiện khác, tham khảo thêm tại <https://www.baeldung.com/spring-boot-bean-validation>

Ngoài ra, có thể validate bằng tay tại Service hoặc Controller (Lời khuyên là nên giảm tối đa code tại Controller để tiện trong việc sửa lỗi và phát triển, nên cài trong Service)

Vd:

```
public ResponseEntity<String> createFilm(Film film) {  
    String rating=film.getRating();  
    if (!Film.ERatings.contains(rating)) {  
        return new ResponseEntity<>("Rating value is invalid",  
            HttpStatus.BAD_REQUEST);  
    }  
  
    Optional<Language> languageOptional =  
        languageRepository.findById(film.getLanguage().getId());  
  
    if (languageOptional.isEmpty()) {  
        return new ResponseEntity<>("Language is not exist", HttpStatus.NOT_FOUND);  
    }  
  
    film.setLanguage(languageOptional.get());  
  
    filmRepository.save(film);  
    return new ResponseEntity<>("Saved Success", HttpStatus.CREATED);  
}
```

# 1. Cài đặt dependency Document APIs:

Cài đặt dependency springdoc để tự động khởi tạo doc cho dự án

Chi tiết dependency có thể xem tại đây:

<https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui/2.1.0>

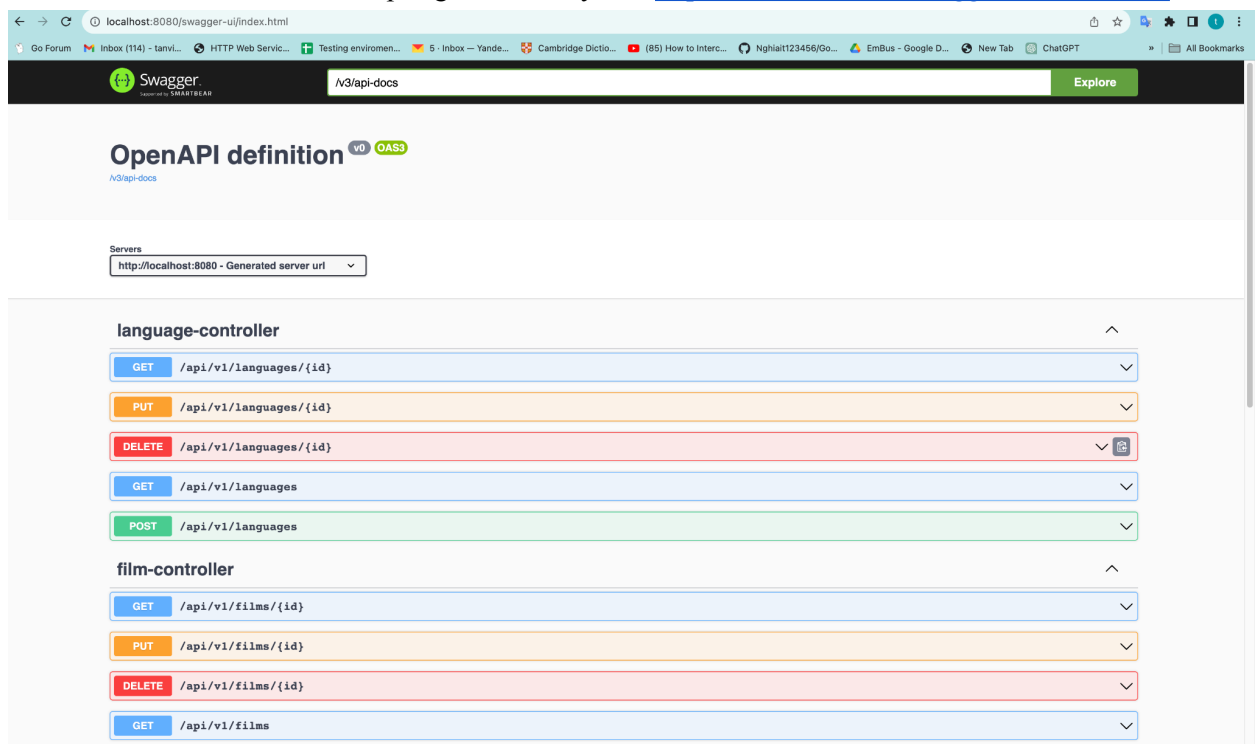
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui -->
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
  <version>2.1.0</version>
</dependency>

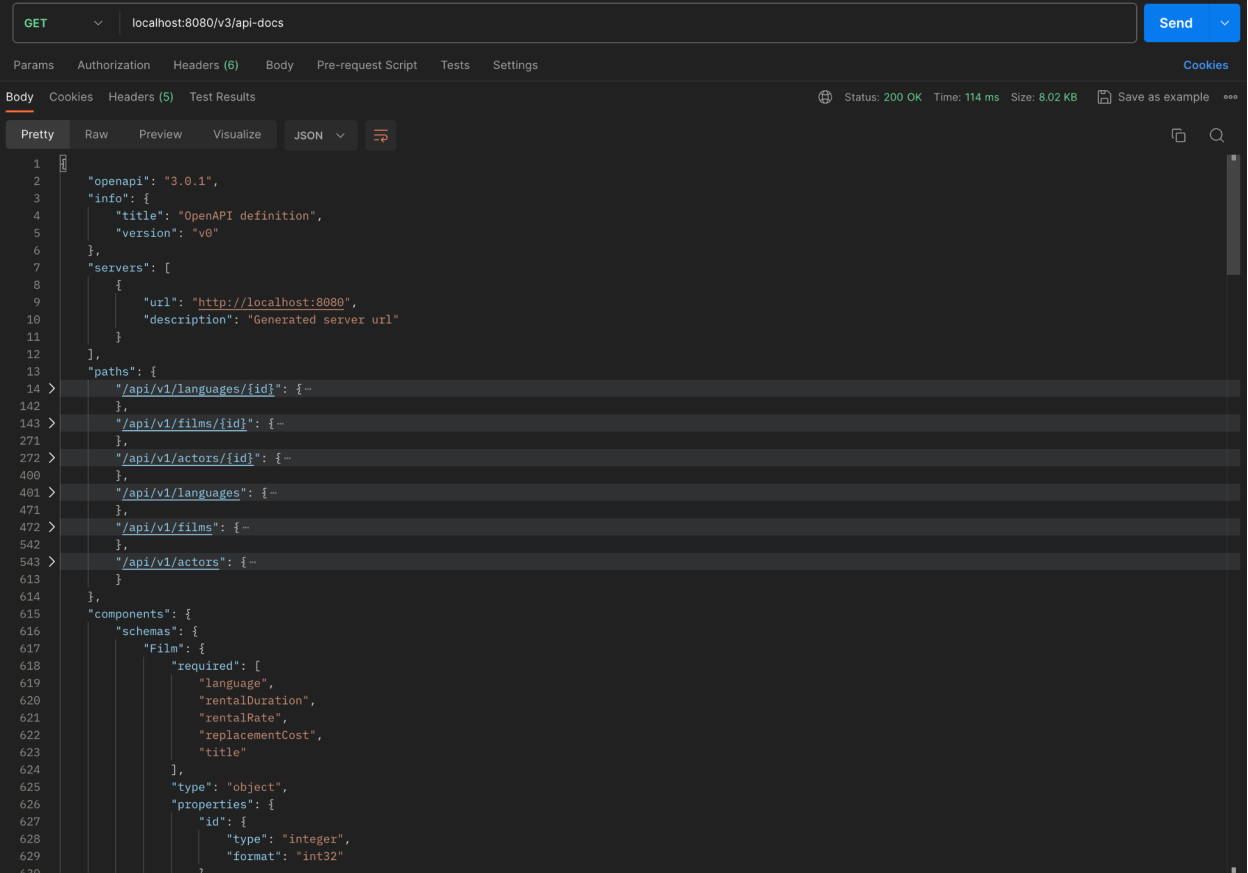
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter</artifactId>
</dependency>

<dependency>
  <groupId>org.springframework.boot</groupId>
```

Sau khi cài đặt dependency thành công, khởi chạy lại chương trình, mặc định springdoc sẽ tự động sinh ra swagger tại endpoint <http://server:port/context-path/swagger-ui.html> (server là domain của chương trình, port là cổng đầu vào, context-path tùy vào chương trình của bạn có cấu hình hay không), mặc định cho dự án cơ bản của springboot sẽ chạy trên <http://localhost:8080/swagger-ui/index.html>



Ngoài ra, OpenAPI sẽ sinh ra tài liệu cơ bản có sẵn tại url <http://localhost:8080/v3/api-docs> dưới định dạng json:



```
1  {
2    "openapi": "3.0.1",
3    "info": {
4      "title": "OpenAPI definition",
5      "version": "v0"
6    },
7    "servers": [
8      {
9        "url": "http://localhost:8080",
10       "description": "Generated server url"
11      }
12    ],
13    "paths": {
14      "/api/v1/languages/{id}": {
15        "get": {
16          "summary": "Get languages by id",
17          "description": "Get languages by id",
18          "parameters": [
19            {
20              "name": "id",
21              "in": "path",
22              "required": true,
23              "schema": {
24                "type": "string"
25              }
26            }
27          ],
28          "responses": {
29            "200": {
30              "description": "Successful response",
31              "content": {
32                "application/json": {
33                  "schema": {
34                    "type": "array",
35                    "items": {
36                      "type": "string"
37                    }
38                  }
39                }
40              }
41            }
42          }
43        }
44      },
45      "/api/v1/films/{id}": {
46        "get": {
47          "summary": "Get films by id",
48          "description": "Get films by id",
49          "parameters": [
50            {
51              "name": "id",
52              "in": "path",
53              "required": true,
54              "schema": {
55                "type": "string"
56              }
57            }
58          ],
59          "responses": {
60            "200": {
61              "description": "Successful response",
62              "content": {
63                "application/json": {
64                  "schema": {
65                    "type": "object",
66                    "properties": {
67                      "title": {
68                        "type": "string"
69                      },
70                      "rentalRate": {
71                        "type": "integer",
72                        "format": "int32"
73                      },
74                      "replacementCost": {
75                        "type": "integer",
76                        "format": "int32"
77                      },
78                      "language": {
79                        "type": "string"
80                      },
81                      "rentalDuration": {
82                        "type": "integer",
83                        "format": "int32"
84                      }
85                    }
86                  }
87                }
88              }
89            }
90          }
91        }
92      },
93      "/api/v1/actors/{id}": {
94        "get": {
95          "summary": "Get actors by id",
96          "description": "Get actors by id",
97          "parameters": [
98            {
99              "name": "id",
100             "in": "path",
101             "required": true,
102             "schema": {
103               "type": "string"
104             }
105            }
106          ],
107          "responses": {
108            "200": {
109              "description": "Successful response",
110              "content": {
111                "application/json": {
112                  "schema": {
113                    "type": "object",
114                    "properties": {
115                      "title": {
116                        "type": "string"
117                      },
118                      "rentalRate": {
119                        "type": "integer",
120                        "format": "int32"
121                      },
122                      "replacementCost": {
123                        "type": "integer",
124                        "format": "int32"
125                      },
126                      "language": {
127                        "type": "string"
128                      },
129                      "rentalDuration": {
130                        "type": "integer",
131                        "format": "int32"
132                      }
133                    }
134                  }
135                }
136              }
137            }
138          }
139        }
140      }
141    },
142    "components": {
143      "schemas": {
144        "Film": {
145          "required": [
146            "language",
147            "rentalDuration",
148            "rentalRate",
149            "replacementCost",
150            "title"
151          ],
152          "type": "object",
153          "properties": {
154            "id": {
155              "type": "integer",
156              "format": "int32"
157            },
158            "title": {
159              "type": "string"
160            },
161            "rentalRate": {
162              "type": "integer",
163              "format": "int32"
164            },
165            "replacementCost": {
166              "type": "integer",
167              "format": "int32"
168            },
169            "language": {
170              "type": "string"
171            },
172            "rentalDuration": {
173              "type": "integer",
174              "format": "int32"
175            }
176          }
177        }
178      }
179    }
180  }
```

## 2.Mô tả swagger:

Springdoc tự động tìm các `@RestController` và các api có ở trong source code, kèm theo các mẫu dữ liệu input, output và các mã lỗi, kiểu dữ liệu trả về ở mức cơ bản từ source code.

Ví dụ mẫu bên dưới là file FilmController.java mô tả controller như sau:

```

@RestController
@RequestMapping(path = "/api/v1/films")
public class FilmController {
    6 usages
    private final FilmService filmService;

    tranhuanhcmus
    public FilmController(FilmService filmService) { this.filmService = filmService; }

    tranhuanhcmus
    @GetMapping(path = "")
    public ResponseEntity<List<FilmDTO>> getAllFilm() { return filmService.getAll(); }

    tranhuanhcmus
    @PostMapping(path = "")
    public ResponseEntity<String> createFilm(@RequestBody @Valid Film film) { return filmService.createFilm(film); }

    tranhuanhcmus *
    @GetMapping(path =("/{id}")

    public ResponseEntity<FilmDTO> getFilmById( @PathVariable Short id) { return filmService.getFilmById(id); }

    tranhuanhcmus *
    @DeleteMapping(path =("/{id}")
    public ResponseEntity<String> deleteFilmById( @PathVariable Short id) { return filmService.deleteFilmById(id); }

    Vietnguyenjr +1 *
    @PutMapping(path =("/{id}")
    public ResponseEntity<String> updateFilmById(
        @PathVariable Short id,
        @RequestBody @Valid UpdateFilmDTO film){...}
}

```

Từ thông tin controller trên, swagger sẽ tự động sinh ra các api tương ứng, bao gồm các thông tin như phương thức (GET, PUT, DELETE, POST,...), các endpoint cụ thể

film-controller		^
GET	/api/v1/films/{id}	▼
PUT	/api/v1/films/{id}	▼
DELETE	/api/v1/films/{id}	▼
GET	/api/v1/films	▼
POST	/api/v1/films	▼

Nhấp vào từng mục, ta có thể xem chi tiết từng API:



POST

/api/v1/films

Parameters

Try it out

No parameters

Request body required

application/json

Example Value | Schema

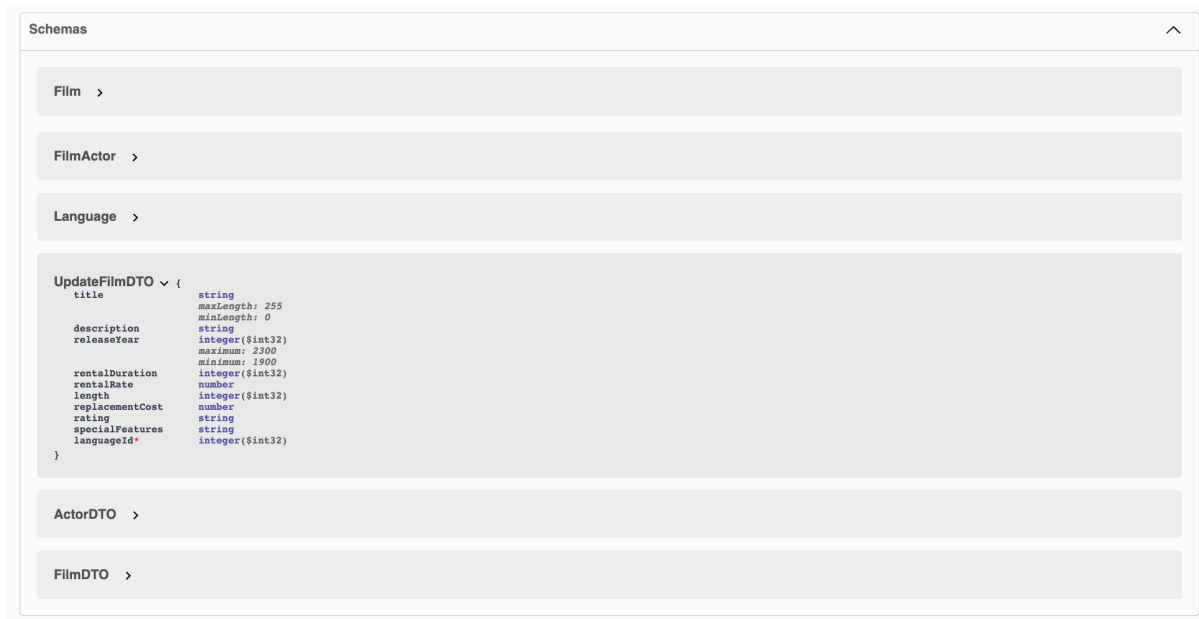
```
{
  "id": 0,
  "title": "string",
  "description": "string",
  "releaseYear": 0,
  "rentalDuration": "string",
  "rentalRate": 0,
  "length": 0,
  "replacementCost": 0,
  "rating": "string",
  "specialFeatures": "string",
  "language": {
    "id": "string",
    "name": "string",
    "films": [
      "string"
    ]
  },
  "filmActors": [
    {
      "film": "string",
      "actor": "string"
    }
  ],
  "updateData": {
```

Responses

Code	Description	Links
200	OK	No links
	<div>Media type</div> <div>*/*</div> <div>Controls Accept header.</div> <div>Example Value   Schema</div> <div>string</div>	
400	Bad Request	No links
	<div>Media type</div> <div>*/*</div> <div>Example Value   Schema</div> <div>string</div>	

Swagger đã tự động sinh ra các thông tin cơ bản của API như Request body, kiểu dữ liệu của Request body, Response, Response Code

Ngoài ra swagger còn mô tả các schemas (có thể nói là các object được sử dụng trong request body hoặc response được dùng trong các API)

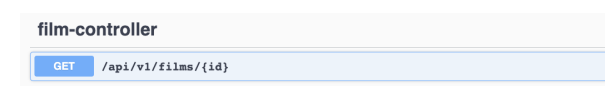
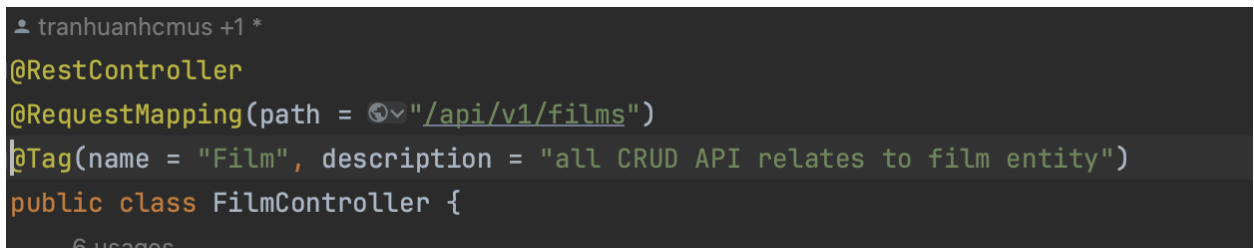


### 3. Thêm thông tin, mô tả dữ liệu swagger:

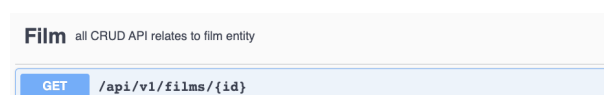
Ta có thể mô tả, thêm thông tin về các API dựa trên các annotation được cung cấp trong package `io.swagger.v3.oas.annotations` (có sẵn khi cài đặt dependency `springdoc`):

#### @Tag

Thêm `@Tag` để chỉnh sửa mô tả cho controller, với name là tên controller, description để mô tả controller như sau



Trước khi thêm `@Tag`



Sau khi thêm `@Tag`

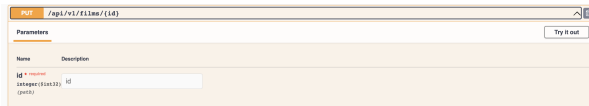
#### @Operation

Thêm `@Operation` để chỉnh sửa mô tả cho 1 API cụ thể, với summary là tên API, description để mô tả API như sau

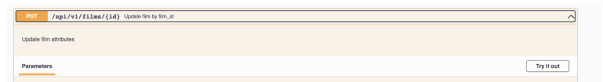
```

Vietnguyenjr +1 *
@Operation(summary = "Update film by film_id", description = "Update film attributes", tags = { "Film" } )
@PutMapping(path = "/{id}")
public ResponseEntity<String> updateFilmById(
    @PathVariable Short id,
    @RequestBody @Valid UpdateFilmDTO film){...}

```



Trước khi thêm @Operation



Sau khi thêm @Operation

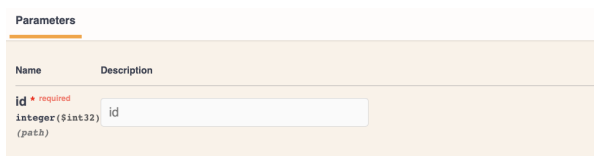
## @Parameter

Thêm @Parameter để thêm mô tả cho các tham số truyền vào khi gọi API, thêm description để mô tả như sau

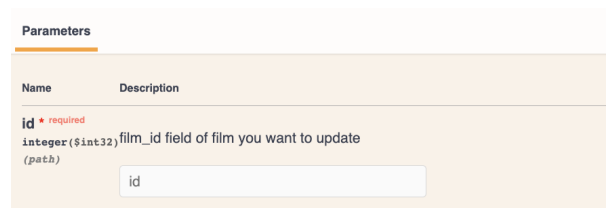
```

@Operation(summary = "Update film by film_id", description = "Update film attributes", tags = { "Film" } )
@PutMapping(path = "/{id}")
public ResponseEntity<String> updateFilmById(
    @Parameter(description = "film_id field of film you want to update")
    @PathVariable Short id,
    @RequestBody @Valid UpdateFilmDTO film){
    return filmService.updateFilmById(id,film);
}

```



Trước khi thêm @Parameter



Sau khi thêm @Parameter

## @Parameters

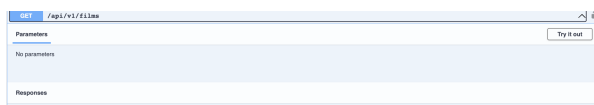
Thêm @Parameters để thêm các tham số truyền vào khi gọi API trên swagger, trong đó:

- name: Tên params
- example: giá trị mặc định
- description: mô tả cho tham số
- in: chỉ ra vị trí của tham số (trong header, query)
  - ParameterIn.HEADER: trong header
  - ParameterIn.QUERY: trong query
  - ParameterIn.COOKIE: trong cookie
  - ParameterIn.PATH: trong đường dẫn

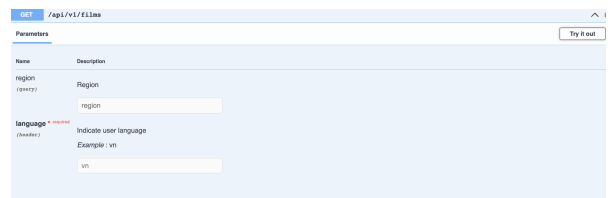
```

@GetMapping(path = "")
@Parameters({
    @Parameter(name = "region", description = "Region", in = ParameterIn.QUERY, required = false),
    @Parameter(
        name = "language",
        example = "vn",
        description = "Indicate user language",
        in = ParameterIn.HEADER,
        required = true
    ),
})
public ResponseEntity<List<FilmDTO>> getAllFilm() {
    return filmService.getAll();
}

```



Trước khi thêm @Parameters



Sau khi thêm @Parameters

## @ApiResponse

Thêm @ApiResponse để chỉnh sửa mô tả cho response, với value chứa các thông tin:

- ApiResponse đại diện 1 response API trả về, trong đó:
  - responseCode: mã status code trả về (200, 201, 400, 404,...)
  - header: dữ liệu header trả về, trong đó
    - name: tên key trả về
    - description: mô tả header
    - schema: Kiểu đối tượng trả về
  - description: Mô tả responses
  - Content: nội dung của responses, là @Content, trong đó:
    - mediaType: Kiểu dữ liệu trả về
    - schema: Kiểu đối tượng trả về

```

@Operation(summary = "Update film by film_id", description = "Update film attributes", tags = {"Film"})
@ApiResponses(value = {
    @ApiResponse(
        responseCode = "200",
        headers = {
            @Header(
                name = "X-Rate-Limit",
                description = "calls per hour allowed by the user",
                schema = @Schema(type = "integer",
                    format = "int32")
            ),
        },
        description = "Update film successfully",
        content = {
            @Content(
                mediaType = "application/json",
                schema = @Schema(type = "string", format = "int32", example = "Update success")
            )
        }
    ),
    @ApiResponse(
        responseCode = "404",
        description = "Film or language is not exist",
        content = {
            @Content(
                mediaType = "string",
                schema = @Schema(type = "string", format = "int32", example = "Film is not exist")
            ),
        }
    ),
})
@PutMapping(path = "{id}")
public ResponseEntity<String> updateFilmById(
    @Parameter(description = "film_id field of film you want to update")
    @PathVariable Short id,
    @RequestBody @Valid UpdateFilmDTO film) {

```

Responses		
Code	Description	Links
200	OK	No links
	Media type: <input type="text" value="application/json"/>	
	Example Value: Schema	
	string	
400	Bad Request	No links
	Media type: <input type="text" value="application/json"/>	
	Example Value: Schema	
	string	

Responses		
Code	Description	Links
200	Update film successfully	No links
	Media type: <input type="text" value="application/json"/>	
	Example Value: Schema	
	string	
	Header: X-Rate-Limit	calls per hour allowed by the user
	Type: Integer	
400	Bad Request	No links
	Media type: <input type="text" value="application/json"/>	
	Example Value: Schema	
	string	
404	Film or language is not exist	No links
	Media type: <input type="text" value="string"/>	
	Example Value: Schema	
	Film is not exist	

Trước khi thêm @ApiResponses

Sau khi thêm @ApiResponses

## @Schema

Thêm @Schema vào các class mô tả thông tin request, response, thêm ví dụ. Cụ thể hơn, trong đó:

- example: Mẫu dữ liệu mẫu cho field
- description: Mô tả field

```

@Size(max = 255)
@Schema(example = "StarWar", description = "Hollywood")
private String title;

2 usages
@Schema(example = "Action movie", description = "Description of film")
private String description;

2 usages
@Schema(example = "2017", description = "Release year of the movie")
@Min(value = 1900, message = "Release year must be greater than 1900")
@Max(value = 2300, message = "Release year must be smaller than 2300")
private Integer releaseYear;

2 usages
@Schema(example = "65", description = "Rental duration, not null")
private Short rentalDuration;

2 usages
@Schema(example = "5.71", description = "Rental rate, not null")
private BigDecimal rentalRate;

2 usages
@Schema(example = "117", description = "Duration of the movie (minutes)")
private Short length;

2 usages
@Schema(example = "21.66", description = "Cost of replacement (dollar)")
private BigDecimal replacementCost;

```

Example Value Schema

```

{
  "title": "string",
  "description": "string",
  "releaseYear": "integer",
  "rentalDuration": "integer",
  "rentalRate": "number",
  "length": "integer",
  "replacementCost": "number",
  "rating": "string",
  "specialFeatures": "string",
  "languageId": "integer"
}

```

Example Value Schema

```

{
  "title": "string",
  "description": "string",
  "releaseYear": "integer",
  "rentalDuration": "integer",
  "rentalRate": "number",
  "length": "integer",
  "replacementCost": "number",
  "rating": "string",
  "specialFeatures": "string",
  "languageId": "integer"
}

```

UpdateFilmDTO {

```

  title
    maxLength: 255
    minLength: 0
    example: StarWar
    Hollywood

  description
    string
    example: Action movie
    Description of film

  releaseYear
    integer(int32)
    maximum: 2300
    minimum: 1900
    example: 2017
    Release year of the movie

  rentalDuration
    integer(int32)
    example: 65
    Rental duration, not null

  rentalRate
    number
    example: 5.71
    Rental rate, not null

  length
    integer(int32)
    example: 117
    Duration of the movie (minutes)

  replacementCost
    number
    example: 21.66
    Cost of replacement (dollar)

  rating
    string
    example: R
    Rating

  specialFeatures
    string
    example: Trailers, Deleted Scenes
    Set of special features

  languageId*
    integer(int32)
    example: 1
    Language id
}

```

Trước khi thêm @Schema

Sau khi thêm @Schema

## 4. Cấu hình nâng cao swagger:

### Cấu hình cơ bản

Tạo 1 java class mới và thêm annotation `@Configuration` để cấu hình swagger, trong đó có 1 hàm để trả về OpenAPI có annotation `@Bean` và thêm 1 số dữ liệu như sau:

```
package com.example.accessdata.configuration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

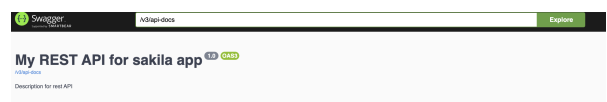
import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Info;

new *
@Configuration
public class SwaggerConfiguration {
    new *
    @Bean
    public OpenAPI openAPI() {
        return new OpenAPI()
            .info(
                new Info().title("My REST API for sakila app")
                    .description("Description for rest API")
                    .version("1.0")
            );
    }
}
```

Hàm `openAPI` sẽ trả về 1 đối tượng đã cấu hình sẵn, ta có thể thêm 1 số thông tin cơ bản như `info`



Trước khi thêm



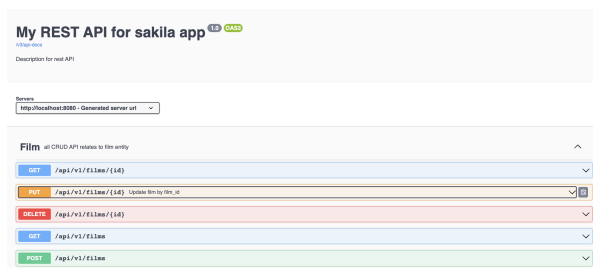
Sau khi thêm

## Cấu hình Authentication

Trong hàm cấu hình openAPI, ta cần thêm 1 SecurityScheme giống như ví dụ sau:

```
new *
@Configuration
public class SwaggerConfiguration {
    new *
    @Bean
    public OpenAPI openAPI() {
        return new OpenAPI()
            .info(
                new Info().title("My REST API for sakila app")
                    .description("Description for rest API")
                    .version("1.0")
            )
            .addSecurityItem(new SecurityRequirement().addList( name: "Bearer Authentication"))
            .components(new Components()
                .addSecuritySchemes( key: "Bearer Authentication", createAPIKeyScheme())
            );
    }

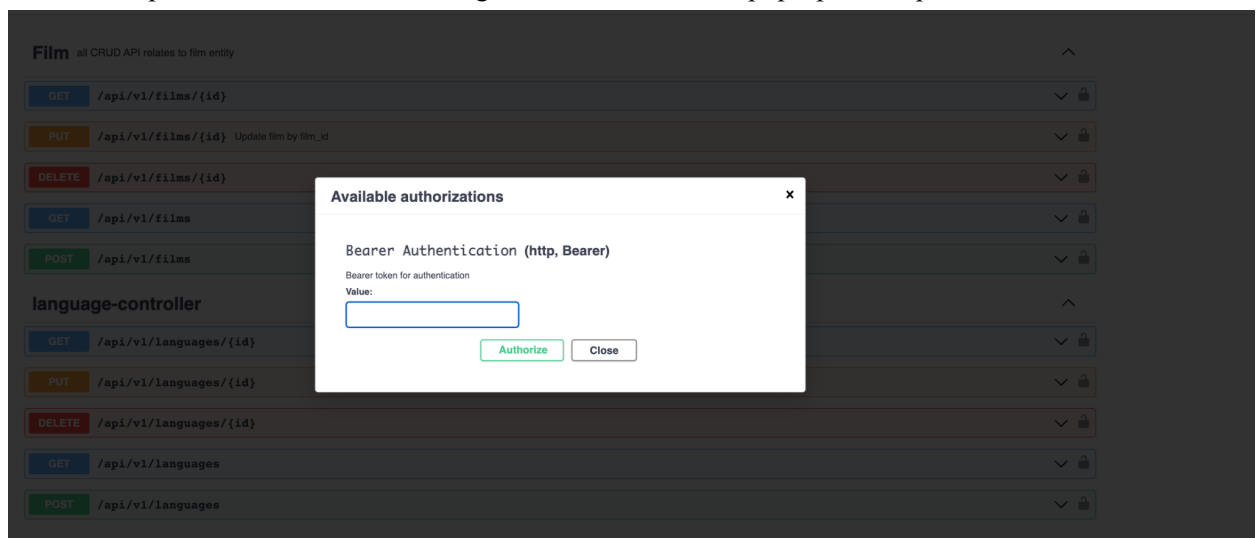
    1usage new *
    private SecurityScheme createAPIKeyScheme() {
        return new SecurityScheme().type(SecurityScheme.Type.HTTP)
            .description("Bearer token for authentication")
            .bearerFormat("JWT")
            .scheme("bearer");
    }
}
```



Trước khi thêm

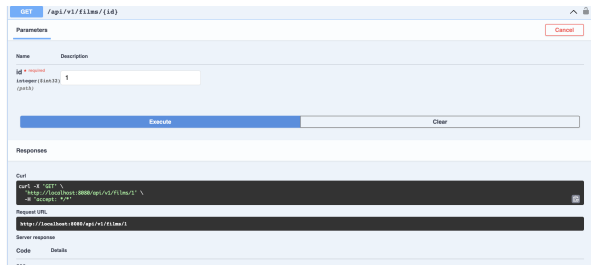
Sau khi thêm

Nhấp vào mục Authorize ở bên góc trái, sẽ xuất hiện 1 pop-up để nhập token:

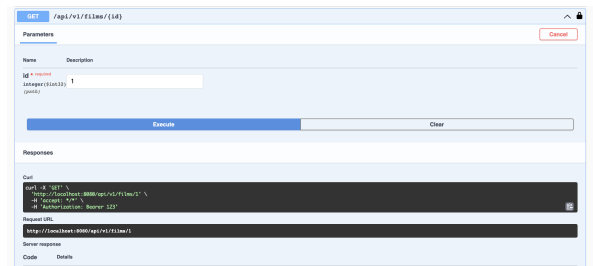




Nhập token và nhấp Authorize, swagger sẽ lưu và áp dụng cho tất cả các API. Ta có thể kiểm tra bằng cách quan sát Curl trong 1 API bất kì



Trước khi thêm token 123



Sau khi thêm token 123

Ngoài ra, springdoc còn hỗ trợ rất nhiều tính năng khác chi tiết có thể tham khảo tại website chính <https://springdoc.org/>,

Một số example có thể tìm hiểu tại <https://github.com/springdoc/springdoc-openapi-demos/tree/2.x>