LN03 - API Validation, API Docs

Created time	@October 21, 2024 5:07 PM
Type	Documentation
Author	Nguyễn Khang Hy Minh Huy callmejippo
Projects	Advanced Web App
Status	Proposed

▼ 1/ API Validation

> 💡 `class-validator` và `class-transformer` là hai thư viện mạnh mẽ thường được sử dụng cùng nhau trong NestJS để thực hiện validation và transformation dữ liệu. Chúng giúp đảm bảo tính toàn vẹn và định dạng dữ liệu trước khi nó được sử dụng trong ứng dụng.

1.1 `class-validator`

- Thư viện này cung cấp các decorator để xác thực dữ liệu của các lớp. Có thể sử dụng các decorator này để xác định các quy tắc validation cho các thuộc tính của lớp, chẳng hạn như required, minLength, maxLength, isEmail, isNumber, v.v.
- `class-validator` sử dụng các decorator để thêm metadata vào các thuộc tính của lớp. Khi thực hiện validation, nó sẽ đọc metadata này và kiểm tra xem dữ liệu đầu vào có đáp ứng các quy tắc đã được định nghĩa hay không
- Một số decorators thông dụng
  - **@IsNotEmpty():** Kiểm tra xem giá trị có rỗng hay không (chuỗi rỗng, mảng rỗng, object rỗng, null, undefined).
  - **@IsString()/@IsNumber()/@IsBoolean()/@IsDate()/@IsArray()/@IsObject():** Kiểm tra kiểu dữ liệu.
  - **@Min(n)/@Max(n):** Kiểm tra giá trị số có lớn hơn hoặc bằng/nhỏ hơn hoặc bằng n.
  - **@MinLength(n)/@MaxLength(n):** Kiểm tra độ dài của chuỗi.
  - **@IsEmail():** Kiểm tra xem có phải là địa chỉ email hợp lệ.
  - **@IsPhoneNumber(region?: string):** Kiểm tra số điện thoại (cần cài thêm libphonenumber-js).
  - **@IsUrl(options?: IsUrlOptions):** Kiểm tra URL.
  - **@IsUUID(version?: "3" | "4" | "5" | "all"):** Kiểm tra UUID.
  - **@Matches(regex: RegExp, options?: MatchesOptions):** Kiểm tra xem chuỗi có khớp với regex.

- **@IsOptional():** Cho phép thuộc tính có thể null hoặc undefined.
- **@IsEnum(entity):** Kiểm tra xem giá trị có thuộc enum.
- **@ValidateNested({ each: true }):** Validate các object lồng nhau. each: true dùng để validate từng phần tử trong mảng.

1.2 `class-transformer`

- Thư viện này giúp bạn chuyển đổi dữ liệu từ một dạng sang một dạng khác. Nó đặc biệt hữu ích khi làm việc với các plain JavaScript object và muốn chuyển đổi chúng thành các instance của lớp TypeScript.
- `class-transformer` sử dụng các decorator như @Expose, @Transform, @Type để định nghĩa cách chuyển đổi dữ liệu.
- Một số decorators thông dụng
  - **@Expose():** Đánh dấu thuộc tính sẽ được expose khi serialize object.
  - **@Exclude():** Ngược lại với @Expose(), đánh dấu thuộc tính sẽ bị loại bỏ khi serialize.
  - **@Transform(transformFn):** Áp dụng một hàm transform cho giá trị.
  - **@Type(typeFn):** Xác định kiểu dữ liệu cho thuộc tính, thường dùng cho các object lồng nhau. Ví dụ: @Type(() ⇒ AddressDto)
  - **@PlainToClass(cls, plain) (static method):** Chuyển đổi plain object thành class instance.
  - **@ClassToPlain(object) (static method):** Chuyển đổi class instance thành plain object.

> 💡 Hai thư viện này hoạt động rất tốt cùng nhau. class-transformer chuyển đổi dữ liệu plain object thành instance của lớp, sau đó class-validator sẽ validate instance này.

1.3 Cài đặt

Sử dụng npm hoặc yarn để cài đặt class-validator và class-transformer:

```
npm install class-validator class-transformer
```

hoặc

```
yarn add class-validator class-transformer
```

▼ 2/ API Docs

OpenAPI (trước đây gọi là Swagger) là một tiêu chuẩn để mô tả API RESTful. Nó cung cấp một cách tiêu chuẩn và dễ đọc máy để định nghĩa các endpoint, tham số, yêu cầu, phản hồi và các thông tin khác của API.

- Tài liệu API tự động: Swagger tạo tài liệu API dựa trên code, giúp tiết kiệm thời gian và đảm bảo tài liệu luôn được cập nhật.
- Giao diện người dùng trực quan: Swagger UI cung cấp một giao diện web cho phép người dùng dễ dàng khám phá, thử nghiệm và tương tác với API.

- Giảm thiểu thời gian phát triển: Swagger giúp cải thiện giao tiếp giữa các nhóm phát triển front-end và back-end, giảm thiểu thời gian tích hợp.
- Kiểm thử API dễ dàng: Swagger UI cho phép người dùng gửi các yêu cầu thử nghiệm đến API trực tiếp từ giao diện web

2.1 Cài đặt

Cài đặt:

```
npm install @nestjs/swagger swagger-ui-express
```

hoặc

```
yarn add @nestjs/swagger swagger-ui-express
```

2.2 Cấu hình cơ bản

```
import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  const config = new DocumentBuilder()
    .setTitle('My API')
    .setDescription('API description')
    .setVersion('1.0')
    .build();

  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('api/documentation', app, document);

  await app.listen(3000);
}
```

- Một số decorators thông dụng
  - **@ApiTags(...tags: string[])**: Gắn tag (nhóm) cho controller hoặc method. Hữu ích khi bạn có nhiều controller và muốn nhóm chúng lại trong Swagger UI.
  - **@ApiOperation({ summary, description })**: Mô tả chức năng của method. summary là mô tả ngắn gọn, description là mô tả chi tiết hơn.
  - **@ApiResponse({ status, description, type, isArray })**: Mô tả phản hồi của API. status là mã trạng thái HTTP, description là mô tả phản hồi, type là kiểu dữ liệu của phản hồi (thường là DTO, Model), isArray cho biết phản hồi có phải là một mảng hay không.
  - **@ApiParam({ name, description, type, required, enum, examples })**: Mô tả tham số của method (trong route).
  - **@ApiQuery({ name, description, type, required, enum, examples })**: Mô tả query parameter.

- **@ApiBody({ description, type })**: Mô tả body của request.
- **@ApiProperty({ description, type, isArray, required, example, examples, enum, default, format })**: Dùng trong DTO, Models để mô tả thuộc tính.

3/ Sakila Demo

3.1 Validate and document example

> 💡 Ở đây, mình sẽ demo cho API Get film by ID

```
import { ApiProperty, PickType } from '@nestjs/swagger';
import { FilmRating } from '../../domain/enums/film-rating';
import {
  IsInt,
  IsString,
  IsOptional,
  IsDate,
  IsArray,
  IsEnum,
  IsNumber,
} from 'class-validator';
import { Transform } from 'class-transformer';

export class FilmDto {
  @ApiProperty({ example: '1' })
  @IsString()
  film_id!: string;

  // Other properties
}

export class GetFilmParamDto extends PickType(FilmDto, ['film_id']) {}
```

> 💡 `PickType` trong NestJS, khi kết hợp với Swagger, cho phép bạn tạo DTO mới bằng cách chọn các thuộc tính từ một DTO hiện có.

```
 @ApiOperation({ summary: 'Get a film by its ID' })
  @ApiResponse({
    status: HttpStatus.OK,
    description: 'Film found',
    type: FilmModel,
  })
  @ApiResponse({
    status: HttpStatus.BAD_REQUEST,
```

```
    description: 'Film not found',
    example: { message: 'Film not found' },
  })
  @ApiParam({
    name: 'film_id',
    description: 'The ID of the film to retrieve',
    required: true,
  })
  @Get('id/:film_id')
  async get(@Param() param: GetFilmParamDto, @Res() res: Response) {
    const film = await this.getFilmUsecase.call(
      parseInt(param.film_id),
      undefined,
      ['categories', 'language', 'original_language', 'actors'],
    );

    if (!film) {
      res.status(HttpStatus.BAD_REQUEST).json({ message: 'Film not found
    }

    res.status(HttpStatus.OK).json(film.toJson());
  }
```

```
export class FilmModel {
  @ApiProperty({
    name: 'film_id',
    example: 1,
    description: 'Unique identifier for the film',
  })
  public readonly filmId: number;

  @ApiProperty({
    name: 'title',
    example: 'Inception',
    description: 'Title of the film',
  })
  public readonly title: string;

  @ApiProperty({
    name: 'description',
    example: 'A mind-bending thriller',
    required: false,
    description: 'Short description of the film',
  })
  public readonly description: string | undefined;

  @ApiProperty({
```

```
    name: 'release_year',
    example: 2010,
    required: false,
    description: 'Year the film was released',
  })
  public readonly releaseYear: number | undefined;

  @ApiProperty({
    name: 'language_id',
    example: 1,
    description: 'Language ID for the primary language of the film',
  })
  public readonly languageId: number;

  @ApiProperty({
    name: 'original_language_id',
    example: 2,
    required: false,
    description: 'ID of the original language, if applicable',
  })
  public readonly originalLanguageId: number | undefined;

  @ApiProperty({
    name: 'rental_duration',
    example: 3,
    description: 'Duration of the film rental in days',
  })
  public readonly rentalDuration: number;

  @ApiProperty({
    name: 'rental_rate',
    example: 9.99,
    description: 'Rate charged for renting the film',
  })
  public readonly rentalRate: number;

  @ApiProperty({
    name: 'length',
    example: 148,
    required: false,
    description: 'Length of the film in minutes',
  })
  public readonly length: number | undefined;

  @ApiProperty({
    name: 'replacement_cost',
    example: 20.0,
    description: 'Cost to replace the film if damaged',
```

```
  })
  public readonly replacementCost: number;

  @ApiProperty({
    name: 'rating',
    example: 'PG-13',
    enum: FilmRating,
    required: false,
    description: 'Rating of the film',
  })
  public readonly rating: FilmRating | undefined;

  @ApiProperty({
    name: 'last_update',
    example: new Date(),
    description: 'Date of the last update to the film record',
  })
  public readonly lastUpdate: Date;

  @ApiProperty({
    name: 'special_features',
    example: ['Deleted scenes', 'Behind the scenes'],
    required: false,
    description: 'Special features included with the film',
  })
  public readonly specialFeatures: string[] | undefined;

  @ApiProperty({
    name: 'fulltext',
    example: 'Inception full-text search data',
    required: false,
    description: 'Full-text search information',
  })
  public readonly fulltext: string | undefined;

  /** Relations */
  @ApiProperty({
    name: 'categories',
    type: () => [CategoryModel],
    required: false,
    description: 'Categories the film belongs to',
  })
  public readonly categories: CategoryModel[] | undefined;

  @ApiProperty({
    name: 'language',
    type: () => LanguageModel,
    required: false,
```

```
    description: 'Primary language of the film',
  })
  public readonly language: LanguageModel | undefined;

  @ApiProperty({
    name: 'original_language',
    type: () => LanguageModel,
    required: false,
    description: 'Original language of the film',
  })
  public readonly originalLanguage: LanguageModel | undefined;

  @ApiProperty({
    name: 'actors',
    type: () => [ActorModel],
    required: false,
    description: 'Actors who appeared in the film',
  })
  public readonly actors: ActorModel[] | undefined;

  constructor(
    filmId: number,
    title: string,
    description: string | undefined,
    releaseYear: number | undefined,
    languageId: number,
    originalLanguageId: number | undefined,
    rentalDuration: number,
    rentalRate: number,
    length: number | undefined,
    replacementCost: number,
    rating: FilmRating | undefined,
    lastUpdate: Date,
    specialFeatures: string[] | undefined,
    fulltext: string | undefined,
    categories: CategoryModel[] | undefined,
    language: LanguageModel | undefined,
    orginalLanguage: LanguageModel | undefined,
    actors: ActorModel[] | undefined,
  ) {
    this.filmId = filmId;
    this.title = title;
    this.description = description;
    this.releaseYear = releaseYear;
    this.languageId = languageId;
    this.originalLanguageId = originalLanguageId;
    this.rentalDuration = rentalDuration;
    this.rentalRate = rentalRate;
```

```
    this.length = length;
    this.replacementCost = replacementCost;
    this.rating = rating;
    this.lastUpdate = lastUpdate;
    this.specialFeatures = specialFeatures;
    this.fulltext = fulltext;
    this.categories = categories;
    this.language = language;
    this.originalLanguage = orginalLanguage;
    this.actors = actors;
  }
}
```

▼ 3.2 Run code

> 💡 Với điều kiện đã có database và data của sakila

B1. Extract source-code.zip

B1.1. Config file `.env` cho phù hợp với điều kiện chạy của máy. Ví dụ

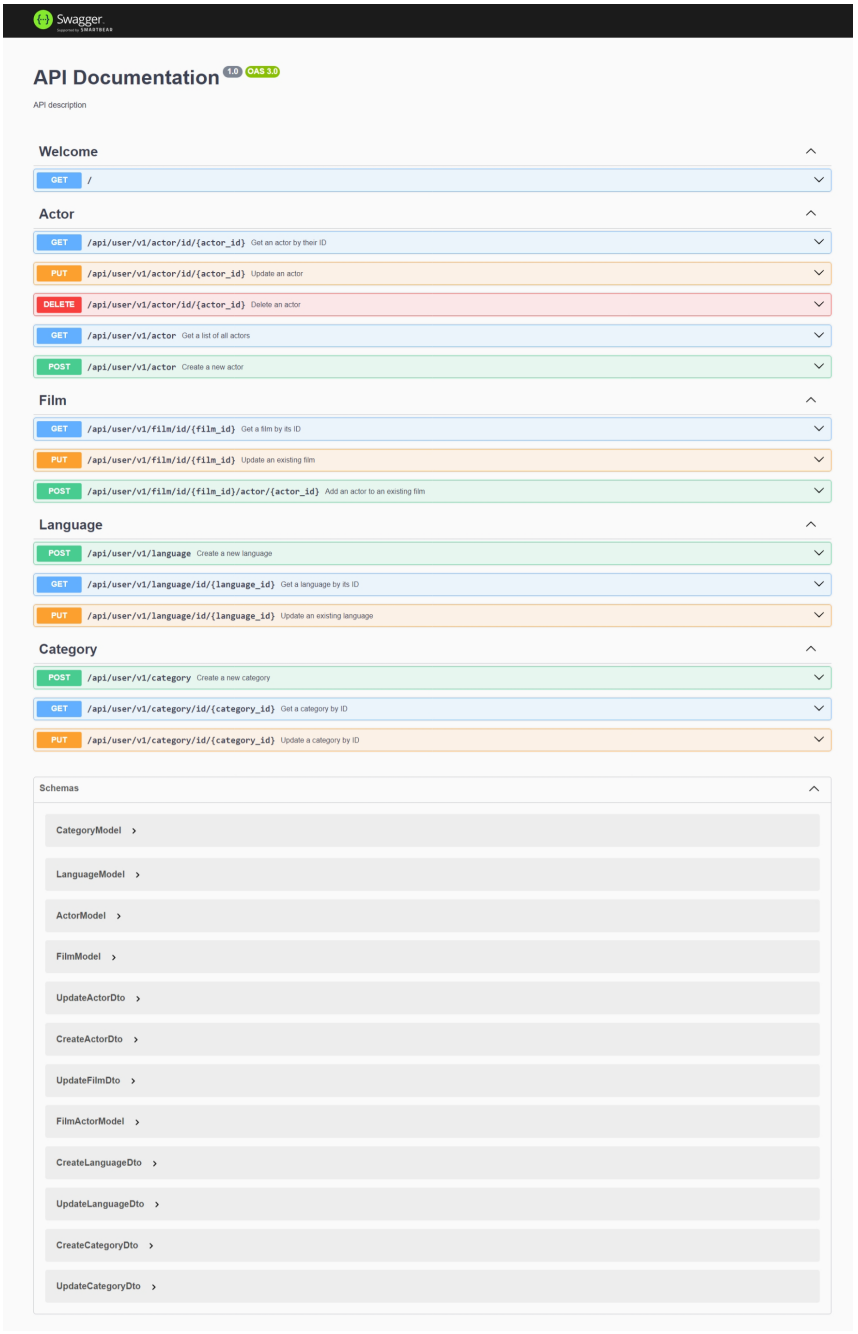
```
APP_NAME=Sakila
APP_PORT=3000
APP_DEBUG=true
APP_FALLBACK_LOCALE=en

DB_CONNECTION=pgsql
DB_TYPE=postgres
DB_HOST=localhost
DB_PORT=5432
DB_DATABASE=sakila
DB_USERNAME=postgres
DB_PASSWORD=postgres
DB_LOGGING=true
```

B2. `npm run start:dev`

B3. Lắng nghe tại cổng `APP_PORT`

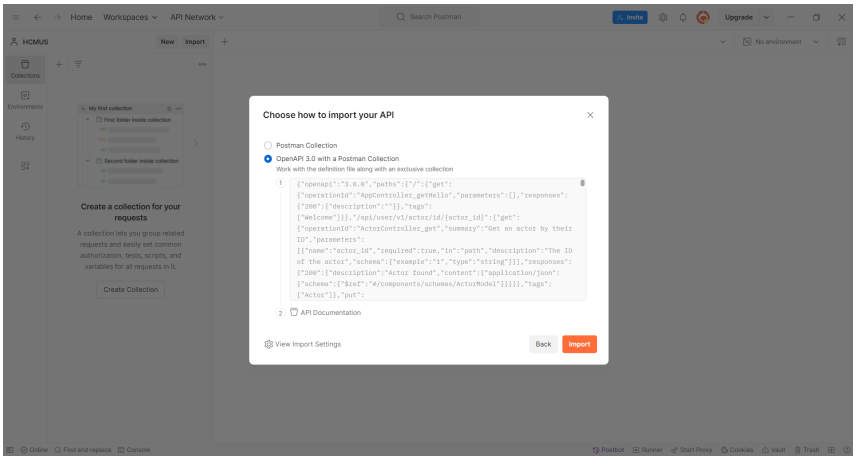
B4. Truy cập vào `/api/documentation` để xem document của API



Import OPENAPI để chạy trên Postman

B1. Truy cập `/api/documentation-json` và tải xướng file `.json`

B2. Chọn `import` và tải lên file `.json` . Chọn **OpenAPI 3.0 with a Postman Collection**



B3. Cấu hình và chạy thử trên Postman