**integrate Swagger UI with Spring Boot 3.4**

### Step 1: Create a Spring Boot Project

1. Use existing producer project as mybanking.

### Step 2: Add Dependencies for Swagger (Springdoc OpenAPI)

In your pom.xml, include the following dependency:

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>

<version>2.1.0</version>

</dependency>

For Gradle, use:

implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.1.0'

### Step 3: Create a REST Controller

No change in any layer code

### Step 4: Start Spring Boot and Access Swagger UI

1. Run the Spring Boot application:

mvn spring-boot:run

Or use your IDE to run the main() method.

1. Open **Swagger UI** in your browser:
   * Default URL: http://localhost:10000/swagger-ui.html

### Step 5: Customize Swagger Documentation

Create new configuration file in package com.mybanking.config.

#### Add OpenAPIConfig:

**package** com.mybanking.config;

**import** io.swagger.v3.oas.models.OpenAPI;

**import** io.swagger.v3.oas.models.info.Contact;

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

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

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

**import** org.springframework.context.annotation.Profile;

@Configuration

@Profile("dev")

**public** **class** OpenAPIConfig {

@Bean

**public** OpenAPI customOpenAPI() {

**return** **new** OpenAPI()

.info(**new** Info()

.title("Employee Management API")

.version("1.0.0")

.description("API for managing employees in the UPCI Family")

.contact(**new** Contact()

.name("Rajesh Upadhyay")

.email("upadhyay.rajesh@rediffmail.com")

.url("https://rkcptraining.com")));

}

}

### Step 6: Add API Descriptions

### No change in our any layer code

### Step 7: Explore Swagger UI

1. Access Swagger UI: http://localhost:8080/swagger-ui.html.
2. Test the endpoints directly from the UI.
   * **GET /employees**: Retrieves the book list.
   * **POST /employees**: Adds a new book (requires JSON input).

### Step 8: Advanced Configuration (Optional)

#### 1. ****Change Base URL****:

To serve Swagger UI under a custom path, configure in application.properties:

springdoc.swagger-ui.path=/api-docs

Access at http://localhost:10000/api-docs.

#### 2. ****Enable/Disable Swagger in Production****:

To disable Swagger in production, use a Spring profile:

@Configuration

@Profile("dev")

public class OpenAPIConfig {

// Configuration here

}

**How to integrate Swagger Editor and Swagger Codegen into project.**

**Step 1: Swagger Editor for OpenAPI Specification**

**Option 1: Online Swagger Editor**

1. Visit Swagger Editor https://editor.swagger.io/.
2. Write or import your OpenAPI specification:
   * Example YAML for a "Bookstore API":

openapi: 3.0.0

info:

title: Employee Management API

version: 1.0.0

servers:

- url: http://localhost:10000/api/v1

paths:

/employees:

get:

summary: Get all employees

responses:

'200':

description: List of employees

content:

application/json:

schema:

type: array

items:

$ref: '#/components/schemas/Employee'

post:

summary: Add a new Employee

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/Employee'

responses:

'201':

description: Employee added

components:

schemas:

Employee:

type: object

properties:

name:

type: string

password:

type: string

email:

type: string

address:

type: string

1. Validate and Save:
   * Fix any errors in the editor.
   * Export the YAML or JSON file for your project.

**Option 2: Install Swagger Editor Locally**

1. Use Docker to run Swagger Editor:

docker run -d -p 8081:8080 swaggerapi/swagger-editor

1. Access the editor at http://localhost:8081.

**Step 2: Swagger Codegen for Spring Boot**

**Install Swagger Codegen**

1. Download the Swagger Codegen CLI:
   * Visit [Swagger Codegen Releases](https://github.com/swagger-api/swagger-codegen/releases).
   * Download the latest JAR file (swagger-codegen-cli.jar).
2. Move the JAR file to a convenient location (e.g., /tools/swagger-codegen-cli.jar).

**Generate Code**

1. Use the Swagger Codegen CLI to generate a Spring Boot server:

java -jar /tools/swagger-codegen-cli.jar generate \

-i employee-api.yaml \

-l spring \

-o ./employee-api-server

* + -i: Path to your OpenAPI spec file (employee-api.yaml).
  + -l: Language/framework (spring for Spring Boot).
  + -o: Output directory.

1. The output includes:
   * A Spring Boot project with pre-generated controllers, models, and APIs.

Example directory structure:

bookstore-api-server

├── src/main/java/com/example/api

│ ├── controller

│ ├── model

│ ├── ApiUtil.java

│ └── Application.java

├── src/main/resources/application.yml

├── pom.xml

└── README.md

1. Open the generated project in your IDE (e.g., IntelliJ or Eclipse).

**Step 3: Update and Customize Generated Code**

1. **Models**:
   * Customize generated model classes (e.g., Employee.java).
2. **Configuration**:
   * Update application.yml or application.properties for your application-specific settings.

**Step 4: Run the Generated Spring Boot Application**

1. Navigate to the generated project directory.
2. Run the application:
3. Access the API:
   * **GET /employees**
   * **POST /employees**:

**Step 5: Generate Client SDK (Optional)**

If you need to create a client SDK for your API:

1. Use Swagger Codegen to generate a client:

java -jar /tools/swagger-codegen-cli.jar generate \

-i employee-api.yaml \

-l java \

-o ./employee-api-client

1. The output directory (employee-api-client) contains:
   * Pre-generated client classes for API interactions.
   * Example usage:

ApiClient apiClient = new ApiClient();

apiClient.setBasePath("http://localhost:10000/api/v1/");

EmployeeApi api = new EmployeeApi(apiClient);

List<Employee> employees= api.getEmployees();

System.out.println(employees);