

Daily task – 13/10/2025

Pom.xml

```
33④    <dependencies>
34④      <dependency>
35        <groupId>org.springdoc</groupId>
36        <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
37        <version>2.6.0</version>
38    </dependency>
39
```

Security config:

```
@Configuration
public class SecurityConfig {
    @Bean
    public OpenAPI customOpenAPI() {
        final String securitySchemeName = "bearerAuth";
        return new OpenAPI()
            .info(new Info().title("Library Management System - Book Service API")
                .description("This API manages books within the Library Management System.\n\n"
                    + "Use the `/auth/login` endpoint to generate a JWT token.\n"
                    + "Then click the **Authorize** button ( ) and paste your token as:\n"
                    + "Bearer eyJhbGciOiJIUzI1NiJ9...`"))
            .version("1.0.0").license(new License().name("Apache 2.0").url("http://springdoc.org")))
            .addSecurityItem(new SecurityRequirement().addList(securitySchemeName))
            .components(new io.swagger.v3.oas.models.Components().addSecuritySchemes(securitySchemeName,
                new SecurityScheme().name(securitySchemeName).type(SecurityScheme.Type.HTTP).scheme("bearer")
                    .bearerFormat("JWT")));
    }
}
```

Post books

The screenshot shows the Springdoc UI interface for a POST request to the '/books' endpoint. The top bar indicates the method (POST) and the URL (/books). A 'Add a new book' button is visible. The main area contains the following details:

- Description:** Creates a new book entry in the library system.
- Parameters:** No parameters.
- Request body (required):** application/json. The schema is defined as follows:

```
{
  "label": "Java",
  "title": "Java Basics",
  "author": "Oracle",
  "quantity": 10
}
```
- Book details to be added:** A text input field containing the JSON schema provided above.
- Buttons at the bottom:** 'Execute' and 'Clear'.

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8081/books' \
  -H 'Accept: */*' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9eyJzdWIiOiJhZGlpbGltbmhdc3RhQmJcSWhwZ20wIjoxNzYwOTY2MzAwfQ.sImh4QzXICpoel8Q48eFLGRUlhGidewWtrSy66sk' \
  -H 'Content-Type: application/json' \
  -d '{
    "isbn": "100",
    "title": "Java Basics",
    "author": "Oracle",
    "quantity": 10
}'
```

Request URL

<http://localhost:8081/books>

Server response

Code Details

201

Response body

```
{
  "isbn": "100",
  "title": "Java Basics",
  "author": "Oracle",
  "quantity": 10
}
```



Download

Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-type: application/json
date: Mon,13 Oct 2025 13:40:28 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
transfer-encoding: chunked
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0
```

Get books

GET

/books Get all books



Retrieves a list of all available books in the library.

Parameters

[Cancel](#)

No parameters

[Execute](#)

[Clear](#)

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8081/books' \
  -H 'Accept: */*' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9eyJzdWIiOiJhZGlpbGltbmhdc3RhQmJcSWhwZ20wIjoxNzYwOTY2MzAwfQ.sImh4QzXICpoel8Q48eFLGRUlhGidewWtrSy66sk'
```



Request URL

<http://localhost:8081/books>

Server response

Code Details

200

Response body

```
[
  {
    "isbn": "100",
    "title": "Java Basics",
    "author": "Oracle",
    "quantity": 10
  }
]
```



Download

Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-type: application/json
date: Mon,13 Oct 2025 13:41:56 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
transfer-encoding: chunked
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0
```