Back to Main Docs

Books API Documentation

This document provides details about the available API endpoints for managing books in the **ELibAPI** system.

Base URL:

```
/api/Books
```

Endpoints

1. Get All Books

- **Endpoint:** `GET /api/Books/get-all-Book`
- **Description:** Retrieves a list of all books.
- **Authorization:** Required (JWT Token)

Request Example:

```
GET /api/Books/get-all-Book HTTP/1.1
Host: yourapi.com
Authorization: Bearer <your-jwt-token>
```

Response Example:

```
[
    {
        "id": 1,
        "title": "Book Title 1",
        "userId": 123,
        "groupId": 456,
        "path": "/uploads/book1.pdf"
    },
    {
        "id": 2,
        "title": "Book Title 2",
        "userId": 124,
        "groupId": 457,
        "path": "/uploads/book2.pdf"
    }
]
```

Response Code:

- `200 OK` - Successfully retrieved books.

2. **Get a Specific Book**

- **Endpoint:** `GET /api/Books/get-book/{id}`
- **Description:** Retrieves a specific book by its ID.
- **Authorization:** Required (JWT Token)

Request Example:

```
GET /api/Books/get-book/1 HTTP/1.1
Host: yourapi.com
Authorization: Bearer <your-jwt-token>
```

Response Example:

```
{
    "id": 1,
    "title": "Book Title 1",
    "userId": 123,
    "groupId": 456,
    "path": "/uploads/book1.pdf"
}
```

Response Code:

- `200 OK` - Successfully retrieved the book.
- `400 Bad Request` - Book not found.

3. **Create a New Book**

- **Endpoint:** `POST /api/Books/create-book`
- **Description:** Creates a new book record in the system.
- **Authorization:** Required (JWT Token)
- **Consumes:** `multipart/form-data` (for file upload)

Request Example:

```
POST /api/Books/create-book HTTP/1.1
Host: yourapi.com
Content-Type: multipart/form-data
Authorization: Bearer <your-jwt-token>
```

```
--boundary
Content-Disposition: form-data; name="Title"
Book Title Example

--boundary
Content-Disposition: form-data; name="UserId"
123

--boundary
Content-Disposition: form-data; name="group_id"
456

--boundary
Content-Disposition: form-data; name="file"; filename="book1.pdf"
Content-Type: application/pdf
<binary content of the file>

--boundary--
```

Response Example:

```
{
    "status": "SUCCESS",
    "message": "Book created successfully.",
    "book": {
        "id": 1,
        "title": "Book Title Example",
        "userId": 123,
        "groupId": 456,
        "path": "/uploads/book1.pdf"
    }
}
```

Response Code:

- `201 Created` - Successfully created the book.
- `400 Bad Request` - Missing required fields.

4. **Update an Existing Book**

- **Endpoint:** `POST /api/Books/update-book`
- **Description:** Updates the details of an existing book.
- **Authorization:** Required (JWT Token)
- **Consumes:** `multipart/form-data` (for file upload)

Request Example:

```
POST /api/Books/update-book HTTP/1.1
Host: yourapi.com
Content-Type: multipart/form-data
Authorization: Bearer <your-jwt-token>

--boundary
Content-Disposition: form-data; name="Id"
1

--boundary
Content-Disposition: form-data; name="Title"
Updated Book Title

--boundary
Content-Disposition: form-data; name="UserId"
123

--boundary
Content-Disposition: form-data; name="group_id"
456

--boundary
Content-Disposition: form-data; name="file"; filename="updated_book.pdf"
Content-Type: application/pdf
<binary content of the file>

--boundary--
```

Response Example:

```
{
    "status": "SUCCESS",
    "message": "Book updated successfully.",
    "book": {
        "id": 1,
        "title": "Updated Book Title",
        "userId": 123,
        "groupId": 456,
        "path": "/uploads/updated_book.pdf"
    }
}
```

Response Code:

- `201 Created` - Successfully updated the book.
- `400 Bad Request` - Missing required fields or invalid book ID.

5. **Delete a Book**

- **Endpoint:** `DELETE /api/Books/{id}`
- **Description:** Deletes a book from the system by its ID.
- **Authorization:** Required (JWT Token)

Request Example:

```
DELETE /api/Books/1 HTTP/1.1
Host: yourapi.com
Authorization: Bearer <your-jwt-token>
```

Response Example:

```
{
    "message": "Successfully deleted!"
}
```

Response Code:

- `200 OK` - Successfully deleted the book.
- `404 Not Found` - Book not found.

Error Codes

- `400 Bad Request` - The request is invalid, or some required fields are missing.
- `401 Unauthorized` - Authentication failed, or no valid token provided.
- `404 Not Found` - The specified resource could not be found.
- `500 Internal Server Error` - There was an unexpected error processing the request.

Notes

- Ensure the **Authorization header** is included in all requests requiring authentication.
- The **file upload** is optional for creating and updating books, but if provided, it will be saved under `/wwwroot/Uploads/` in the application.