Jihan Sleem

BSc in Computer and Communication Engineer

Library Management System API

Maids.cc-Software Engineer

Development Steps:

Entity Creation:

Start by creating entities for Book, Patron, and Borrowing Record with the specified attributes.
Annotate entities with appropriate annotations like @Entity, define relationships, and use annotations like @OneToMany for associations.

API Endpoint Implementation:

Implement RESTful endpoints for book management, patron management, and borrowing as described in the requirements.
Annotate your controllers with @RestController and use mapping annotations like @GetMapping, @PostMapping, @PutMapping, @DeleteMapping.
Handle request parameters, path variables, and request bodies appropriately.

Data Storage Configuration:

Configure a database (using Microsoft SQLSERVER) using Spring Data JPA.
Define repositories for each entity and leverage Spring Data JPA features for CRUD operations.

Validation and Error Handling:

Implement input validation using annotations like @Valid and handle validation errors gracefully.
Return appropriate HTTP status codes and error messages.

Security (Optional):

If implementing security, explain the choice between basic authentication.
Annotate your security configurations with @EnableWebSecurity and implement necessary filters.

Aspects (Optional):

If using AOP for logging, show examples of aspects that log method calls, exceptions, and performance metrics.
Annotate aspect classes with @Aspect and define pointcuts for the targeted operations.

Caching (Optional):

If utilizing Spring's caching mechanisms, explain the setup for caching frequently accessed data.
Annotate methods with @Cacheable or other relevant caching annotations.

Transaction Management:

Implement declarative transaction management using @Transactional to ensure data integrity. Annotate methods requiring transactions with @Transactional.

Testing:

Write unit tests using testing frameworks like Postman by using the needed URL.

Example Usage with Postman:
GET /api/books:
http://localhost:3366/api/books/1

GET /api/books/{id}
POST /api/books
PUT /api/books/{id}
DELETE /api/books/{id}
GET /api/patrons
GET /api/patrons/{id}
POST /api/patrons
PUT /api/patrons/{id}
DELETE /api/patrons/{id}
POST /api/borrow/{bookId}/patron/{patronId}
PUT /api/return/{bookId}/patron/{patronId}

Database:

Using Microsoft SQLSERVER we create a database called **LibraryManagementSystem** and coonect it to out API in STS4 by ***application.properties*** file.

NOTE tht just in my laptop I'm facing an error with processes and ports for that I faced a prolem by conneting it to my project but all the API is correct.

```
Use LibraryManagementSystem;
Go

-- Create Book table
CREATE TABLE Book (
    id INT IDENTITY(1,1) PRIMARY KEY,
    title NVARCHAR(255) NOT NULL,
    author NVARCHAR(255) NOT NULL,
    isbn NVARCHAR(20) NOT NULL,
    publication_year INT NOT NULL
);

-- Create Patron table
CREATE TABLE Patron (
    id INT PRIMARY KEY IDENTITY,
    name NVARCHAR(MAX) NOT NULL,
    contactInformation NVARCHAR(MAX) NOT NULL
);
```

```
-- Create BorrowingRecord table
CREATE TABLE BorrowingRecord (
    id INT PRIMARY KEY IDENTITY,
    book_id INT FOREIGN KEY REFERENCES Book(id) NOT NULL,
    patron_id INT FOREIGN KEY REFERENCES Patron(id) NOT NULL,
    borrowingDate DATE NOT NULL,
    returnDate DATE,
);


-- Insert fake data into Book table
INSERT INTO Book (title, author, isbn, publication_year)
VALUES
    ('The Great Gatsby', 'F. Scott Fitzgerald', '9780140283292', 1925),
    ('To Kill a Mockingbird', 'Harper Lee', '9780061120084', 1960),
    ('1984', 'George Orwell', '9780451524935', 1949),
    ('Pride and Prejudice', 'Jane Austen', '9780141439518', 1813),
    ('The Catcher in the Rye', 'J.D. Salinger', '9780241950425', 1951),
    ('The Hobbit', 'J.R.R. Tolkien', '9780261102217', 1937);


       -- Insert fake data into Patron table
INSERT INTO Patron (name, contactInformation)
VALUES
    ('Alice Johnson', 'alice@example.com'),
    ('Bob Smith', 'bob@example.com'),
    ('Charlie Brown', 'charlie@example.com'),
    ('David Davis', 'david@example.com'),
    ('Eva White', 'eva@example.com'),
    ('Frank Johnson', 'frank@example.com');


       -- Insert fake data into BorrowingRecord table
INSERT INTO BorrowingRecord (book_id, patron_id, borrowingDate, returnDate)
VALUES
    (1, 1, '2022-01-01', '2022-01-15'),
    (2, 2, '2022-02-01', '2023-02-20'),
    (3, 3, '2022-03-01', NULL),
    (4, 4, '2022-04-01', '2024-01-20'),
    (5, 5, '2022-05-01', NULL),
    (6, 6, '2022-06-01', '2024-01-05');
```

Book’s Table:

DESKTOP-DUEDHN4\...ystem - dbo.Book

	id	title	author	isbn	publication_year
▶	1	The Great Gatsby	F. Scott Fitzgera...	9780140283292	1925
	2	To Kill a Mockin...	Harper Lee	9780061120084	1960
	3	1984	George Orwell	9780451524935	1949
	4	Pride and Preju...	Jane Austen	9780141439518	1813
	5	The Catcher in t...	J.D. Salinger	9780241950425	1951
	6	The Hobbit	J.R.R. Tolkien	9780261102217	1937

Patron's Table:

DESKTOP-DUEDHN4\...tem - dbo.Patron

id	name	contactInform...
1	Alice Johnson	alice@example....
2	Bob Smith	bob@example....
3	Charlie Brown	charlie@examp...
4	David Davis	david@exampl...
5	Eva White	eva@example.c...
6	Frank Johnson	frank@example...

BorrowingRecord's Table:

DESKTOP-DUEDHN4....BorrowingRecord

id	book_id	patron_id	borrowingDate	returnDate
1	1	1	2022-01-01	2022-01-15
2	2	2	2022-02-01	2023-02-20
3	3	3	2022-03-01	*NULL*
4	4	4	2022-04-01	2024-01-20
5	5	5	2022-05-01	*NULL*
6	6	6	2022-06-01	2024-01-05