Book-Author App Documentation

Overview

A Spring Boot 3.3 application that manages Authors and Books with CRUD (Create, Read, Update) and a repository join to show book-title/author-name pairs. The stack uses Spring MVC, Spring Data JPA, JSP views, and an in-memory H2 database seeded with 10 authors and 10 books on startup.

Entity Design

- **Author** (`Author.java`): `id`, `name`, `biography`, one-to-many `books`.
- **Book** (`Book.java`): `id`, `title`, `genre`, many-to-one `author` via `author_id`.
- Relationship: one Author to many Books; Books reference their Author.

Implementation Layers

- **Repository**: `AuthorRepository`, `BookRepository` (with `findBooksWithAuthors()` JPQL inner join).
- **Service**: `AuthorService`, `BookService` wrap repository access.
- **Controller**: `AuthorController`, `BookController` handle list/create/edit flows and bind data to JSPs.
- **Views**: JSPs under `src/main/webapp/WEB-INF/views` (`authors.jsp`, `books.jsp`, `author-form.jsp`, `book-form.jsp`) plus `css/styles.css`.
- **Bootstrap/Seed**: `BookAuthorAppApplication` seeds 10 authors and 10 books if DB is empty.

Key Endpoints

- `GET /authors` – list authors
- `GET /authors/new` – show author form
- `POST /authors` or `POST /authors/new` – create author
- `GET /authors/edit/{id}` – edit author form
- `POST /authors/edit/{id}` – update author
- `GET /books` – list books and joined data
- `GET /books/new` – show book form
- `POST /books` or `POST /books/new` – create book
- `GET /books/edit/{id}` – edit book form
- `POST /books/edit/{id}` – update book
- H2 console: `http://localhost:8080/h2-console` (JDBC URL `jdbc:h2:mem:testdb`, user `sa`, password `password`)

Build & Run

- Prereqs: JDK 17+ (code avoids Lombok), Maven (embedded copy in `.maven/` works).

- Build/tests: `./.maven/apache-maven-3.9.6/bin/mvn clean test`
- Run: `./.maven/apache-maven-3.9.6/bin/mvn spring-boot:run`
- Access app at `http://localhost:8080/authors` or `/books`.

Testing

- Repository tests validate CRUD and join query.
- Service tests use Mockito to confirm basic service behavior.
- Run: `mvn test` (or path to embedded maven above).

CSS/UX

- Basic styling in `src/main/webapp/css/styles.css` for tables, forms, and links.

Challenges Faced & Solutions

- **405 on edit/new routes**: added explicit POST mappings `/edit/{id}` and `/new` plus form actions to matching endpoints.
- **Local JDK 25 & Lombok**: removed Lombok dependency and added explicit getters/setters to ensure compilation on newer JDK.
- **Maven availability**: included local Maven bootstrap under `.maven/` and ignored it in `.gitignore`.

Screenshots

1. Authors list page
2. Books list with joined table
3. Author creation form
4. Book creation form

Embed with Markdown once captured, for example: `![Authors List](screenshots/authors-list.png)`

File Map (high level)

- `src/main/java/com/example/bookauthorapp/` – main code (app, controllers, services, repositories, models)
- `src/main/resources/application.properties` – datasource and view config
- `src/main/webapp/WEB-INF/views/` – JSP views
- `src/main/webapp/css/styles.css` – styling
- `src/test/java/com/example/bookauthorapp/` – tests
- `.gitignore` – ignores build output and local maven

How to Extend

- Add Delete operations in controllers, services, and views.

- Switch to a persistent DB (e.g., MySQL/PostgreSQL) by updating `application.properties` and dependencies.
- Add validation annotations and form error display for better UX.