Book Addition & Display Subsystem

Books Core:

UserBookCriteria.java

> (../books-core/src/main/java/com/sismics/books/core/dao/jpa/criteria/UserBookCriteria.java)

> 1. Facilitates filtering and searching within user book collections based on specified criteria.
> 2. Allows filtering by `userId` to target specific user's collections.
> 3. Supports search functionality through a `search` query string.
> 4. Enables filtering by `read` state to differentiate between read and unread books.
> 5. Permits tagging based filtering using a list of `tagIdList`, enhancing book organization and retrieval.

BookDao.java

> (../books-core/src/main/java/com/sismics/books/core/dao/jpa/BookDao.java)

> 1. Handles database operations related to `Book` entities, including creation and retrieval.
> 2. Offers a method to create a new book entry, returning the new book's ID.
> 3. Provides functionality to fetch a book by its unique ID.
> 4. Includes a method to retrieve a book using its ISBN number, supporting both ISBN-10 and ISBN-13 formats.

BookImportedEvent.java

> (../books-core/src/main/java/com/sismics/books/core/event/BookImportedEvent.java)

> 1. Represents an event triggered upon a book import request.
> 2. Captures the user who initiated the book import process.
> 3. Holds reference to the temporary file used for importing book data.
> 4. Includes methods to get and set the user and the import file, facilitating event handling.

BookImportAsyncListener.java

> (../books-core/src/main/java/com/sismics/books/core/listener/async/BookImportAsyncListener.java)

> 1. Acts as an asynchronous listener for book import requests, processing the `BookImportedEvent`.
> 2. Logs the import request event and initiates the import process.
> 3. Reads book data from a CSV file specified in the import event.
> 4. Handles book data, including fetching by ISBN, creating new book entries, and user-book associations.
> 5. Manages tagging for imported books, including creating new tags and associating them with user books.
> 6. Utilizes transaction management to ensure data integrity throughout the import process.

Book.java

> (../books-core/src/main/java/com/sismics/books/core/model/jpa/Book.java)

> 1. Defines the `Book` entity with detailed attributes including ID, title, subtitle, author, and more.
> 2. Supports comprehensive book information like ISBN numbers, page count, language, and publication date.
> 3. Utilizes JPA annotations for database persistence, specifying table and column mappings.
> 4. Includes getters and setters for all attributes, facilitating easy access and modification of book data.
> 5. Designed to represent a book in the system with all necessary details for identification and categorization.

UserBook.java

> (../books-core/src/main/java/com/sismics/books/core/model/jpa/UserBook.java)

> 1. Defines the `UserBook` entity linking users to their books, with attributes for tracking reading status and ownership.
> 2. Contains unique identifiers for both the user and the book, establishing a many-to-many relationship.
> 3. Includes dates for creation, deletion (optional), and when the book was read, supporting user engagement tracking.
> 4. Implements `Serializable`, ensuring the entity can be used in serialization operations within Java.
> 5. Overrides `hashCode` and `equals` methods for entity uniqueness based on `userId` and `bookId`, crucial for database integrity and consistency.

BookDataService.java

> (../books-core/src/main/java/com/sismics/books/core/service/BookDataService.java)

> 1. Provides functionality to fetch book information from external APIs like Google Books and Open Library.
> 2. Implements asynchronous calls to APIs, respecting rate limits through `RateLimiter`.
> 3. Supports searching for books by ISBN, sanitizing input to retain only digits.
> 4. Handles JSON parsing from API responses to construct `Book` entities with detailed information.
> 5. Includes logic to download and store book thumbnails, ensuring images are in JPEG format.
> 6. Manages service configuration, including retrieval of the Google API key from system configuration.
> 7. Utilizes a single-threaded executor for processing API requests, ensuring controlled access to external services.
> 8. Offers clean-up and shutdown capabilities to properly terminate ongoing tasks and service operations.

UserBookDao.java

> (../books-core/src/main/java/com/sismics/books/core/dao/jpa/UserBookDao.java)

> 1. Facilitates user book management operations, including creation and deletion.
> 2. Provides methods for retrieving user books by various criteria such as user ID, book ID, etc.
> 3. Implements functionality for searching user books based on specified criteria.
> 4. Utilizes the `EntityManager` to interact with the database for CRUD operations.
> 5. Handles exceptions such as `NoResultException` for cases where no result is found.
> 6. Supports pagination and sorting of user book search results.
> 7. Integrates with the `ThreadLocalContext` to obtain the `EntityManager`.
> 8. Enables fetching user book details in DTO format for presentation purposes.

UserBookDto.java

> (../books-core/src/main/java/com/sismics/books/core/dto/UserBookDto.java)

> 1. Represents a data transfer object (DTO) for user book information.
> 2. Contains attributes such as ID, title, subtitle, author, language, publication date, creation date, and read date.
> 3. Utilizes annotations such as `@Id` for identifying the primary key attribute.
> 4. Provides getter and setter methods for accessing and modifying the attributes.
> 5. Stores timestamps as `Long` values for easy conversion and manipulation.
> 6. Enables the transfer of user book details between layers of the application, such as between the DAO and the REST resources.

Books Web:

BaseResource.java

> (../books-web/src/main/java/com/sismics/books/rest/resource/BaseResource.java)

> 1. Serves as the foundation for RESTful resources, providing common functionalities such as authentication and authorization checks.
> 2. Utilizes `@Context` to inject HTTP request information, enabling access to request details.
> 3. Supports querying application key directly from query parameters using `@QueryParam`.
> 4. Manages authentication status by checking for a principal object within the request, distinguishing between anonymous and authenticated users.
> 5. Implements methods to validate user permissions against specified base functions, throwing exceptions for unauthorized access attempts.
> 6. Integrates with security filters, specifically `TokenBasedSecurityFilter`, to facilitate secure access control based on tokens.

BookResource.java

> (../books-web/src/main/java/com/sismics/books/rest/resource/BookResource.java)

> 1. Facilitates book management operations via REST API, including creation, deletion, and updates of books.
> 2. Enables book searches and imports using public APIs and file uploads.
> 3. Supports manual entry and editing of book details, such as title, author, and ISBN numbers.

> 4. Provides functionality for listing user books with filters for search, read status, and tags.
> 5. Allows for setting books as read/unread and managing book tags for categorization.
> 6. Implements file handling for importing books in bulk from a CSV or similar format.
> 7. Integrates with external services for book information retrieval and cover image downloading.
> 8. Ensures user authentication and authorization for book-related actions, leveraging base class security methods.