Bookshelf Management Subsystem

Books-Core:

TagDto

> (books-core/src/main/java/com/sismics/books/core/dao/jpa/dto/TagDto.java):
>
> 1. Represents tag information crucial for efficient book organization within the Bookshelf Management Subsystem.
> 2. Attributes (`id`, `name`, `color`) ensure each tag is uniquely identified, has a recognizable name, and a distinct color for user-friendly book categorization.
> 3. It interacts with the database for efficient storage and retrieval using methods like `getId()`, `setId()`, `getName()`, `setName()`, `getColor()`, `setColor()`.
> 4. Methods facilitate easy access and modification of tag information, ensuring reliable functionality within the Bookshelf Management Subsystem
> 5. Enhances user experience by simplifying navigation and recognition of tags, providing an intuitive way to organize and categorize books within the bookshelf.

UserBookDto

(UserBookDto - com.sismics.books.core.dao.jpa.dto.UserBookDto):

1. Represents user book information, essential for tracking individual book details within the Bookshelf Management Subsystem.
2. Attributes (`id`, `title`, `subtitle`, `author`, `language`, `publishTimestamp`, `createTimestamp`, `readTimestamp`) capture key information such as book title, author, language, and timestamps for publication, creation, and reading.
3. Interacts with the database through methods like `getId()`, `setTitle()`, `setAuthor()`, facilitating efficient storage and retrieval of user book details.
4. Supports getter and setter methods for attributes, ensuring easy access and modification of user book information, contributing to the reliability of the Bookshelf Management Subsystem.
5. Facilitates user experience by providing detailed book information, including title, author, and timestamps, enhancing the book organization process within the bookshelf.

TagDao

(TagDao - com.sismics.books.core.dao.jpa.TagDao):

1. Manages tag data access and operations within the Bookshelf Management Subsystem, offering methods for interacting with the database and performing tag-related functionalities.
2. Includes methods like `getById(String id)` and `getByUserId(String userId)` to retrieve tag details by ID or user ID, ensuring efficient data retrieval for book organization.
3. Implements `updateTagList(String userBookId, Set<String> tagIdSet)` to manage the association between user books and tags. This method efficiently updates tag links, enhancing the Bookshelf Management Subsystem's functionality.

4. Facilitates the retrieval of tag information linked to a user book through `getByUserBookId(String userBookId)`, returning a list of `TagDto` objects containing tag details.
5. Provides functionalities for tag creation (`create(Tag tag)`), deletion (`delete(String tagId)`), and searching by name (`findByName(String userId, String name)`), contributing to a comprehensive tag management system within the Bookshelf Management Subsystem.

UserBookDao

(UserBookDao - com.sismics.books.core.dao.jpa.UserBookDao):

1. Manages user book data access and operations within the Bookshelf Management Subsystem, providing methods for creating, deleting, and searching user books based on various criteria.
2. Implements `create(UserBook userBook)` to generate a new user book entry, assigning a unique ID using UUID and storing the user book in the database.
3. Supports `delete(String id)` for logically deleting a user book by marking its deletion date, ensuring that historical data is maintained.
4. Includes methods like `getUserBook(String userBookId, String userId)` and `getUserBook(String userBookId)` to retrieve user book details by ID or book ID, respectively, considering the deletion status.
5. Enables searching and retrieval of user books based on criteria, such as title, subtitle, author, language, publish date, creation date, and read date. This is achieved through `findByCriteria(PaginatedList<UserBookDto> paginatedList, UserBookCriteria criteria, SortCriteria sortCriteria)`.
6. Utilizes efficient database queries to assemble paginated lists of `UserBookDto` objects, providing a comprehensive view of user book information for the Bookshelf Management Subsystem.

Tag

(Tag - com.sismics.books.core.model.jpa.Tag):

1. Represents a tag entity within the Bookshelf Management Subsystem, encapsulating information such as Tag ID (`id`), Tag Name (`name`), User ID (`userId`), Creation Date (`createDate`), Deletion Date (`deleteDate`), and Tag Color (`color`).
2. Utilizes JPA annotations (`@Entity`, `@Table`, `@Id`, `@Column`) for defining the entity and its attributes in the database.
3. Ensures data integrity with constraints such as non-nullable fields (`name`, `userId`, `createDate`, `color`), enforcing essential information for reliable tag management.
4. Supports getter and setter methods for accessing and modifying tag attributes, allowing seamless integration with the Bookshelf Management Subsystem.
5. Implements `toString()` for generating a concise and informative representation of the tag, particularly useful for logging and debugging purposes.

LogCriteria

(LogCriteria - com.sismics.util.log4j.LogCriteria):

1. Represents the criteria used for searching logs within the logging system.

2. Contains attributes such as `level` (logging level like DEBUG, WARN), `tag` (logger name or tag), and `message` (logged message).
3. Utilizes the Apache Commons Lang library (`StringUtils`) to ensure consistent case handling for the `level`, `tag`, and `message` attributes.
4. Provides getter and setter methods for each attribute, allowing external components to retrieve and modify the search criteria.
5. Enhances flexibility by allowing case-insensitive comparison of log criteria, improving the accuracy and comprehensiveness of log searches within the logging system.

LogEntry

(LogEntry - com.sismics.util.log4j.LogEntry):

1. Represents a log entry within the logging system, capturing essential information about a logged event.
2. Includes attributes such as `timestamp` (time when the log entry was recorded), `level` (logging level like DEBUG, WARN), `tag` (logger name or tag), and `message` (logged message).
3. Provides a constructor that allows initializing the log entry with specific values for timestamp, level, tag, and message.
4. Offers getter methods for each attribute, enabling external components to retrieve information about the log entry.
5. Facilitates the analysis and processing of log data by encapsulating relevant details of each logged event in a structured format.

BaseResource

(BaseResource - com.sismics.books.rest.resource.BaseResource):

1. Serves as the base class for REST resources in the application.
2. Includes annotations to inject the HTTP request (`@Context`), allowing access to request-related information.
3. Contains a query parameter `appKey` representing the application key.
4. Maintains a `principal` attribute representing the authenticated user principal.
5. Provides a method `authenticate()` to check if the user is authenticated and not anonymous.
6. Implements a method `checkBaseFunction(BaseFunction baseFunction)` to verify if the user has a specific base function, throwing a `ForbiddenClientException` if the check fails.
7. Defines a method `hasBaseFunction(BaseFunction baseFunction)` to determine if the user has a specific base function, returning true if the user has the required base function.
8. Ensures that the class is designed for extensibility and serves as a foundational component for other REST resources in the application.

TagResource

(TagResource - com.sismics.books.rest.resource.TagResource):

1. Represents a REST resource for handling operations related to tags.
2. Includes methods for retrieving, creating, updating, and deleting tags.
3. Uses JAX-RS annotations such as `@Path`, `@GET`, `@PUT`, `@POST`, `@DELETE`, and `@Produces` to define the resource's URI, HTTP methods, and media type.

4. Extends `BaseResource`, indicating that it serves as a subclass of the base REST resource.
5. The `list()` method retrieves the list of tags for the authenticated user.
6. The `add()` method creates a new tag with the specified name and color.
7. The `update()` method updates an existing tag with the specified ID, allowing changes to the name and color.
8. The `delete()` method deletes an existing tag with the specified ID.
9. Utilizes the `TagDao` class to interact with the database for tag-related operations.
10. Includes input validation using `ValidationUtil` for parameters such as name and color.
11. Throws `ForbiddenClientException` for unauthorized access and `ClientException` for various client-related errors.
12. Returns JSON responses for successful operations.
13. Demonstrates good practices for handling RESTful resources and their CRUD operations.
14. Provides a RESTful API for managing tags in the application.

UserBookCriteria

(UserBookCriteria - com.sismics.books.core.dao.jpa.criteria.UserBookCriteria):

1. Represents criteria for searching user books.
2. Contains attributes such as `userId`, `search`, `read`, and `tagIdList` to define search parameters.
3. Provides getter and setter methods for each attribute.
4. The `userId` attribute represents the user ID for which the search is performed.
5. The `search` attribute holds the search query for filtering user books based on titles, subtitles, or authors.
6. The `read` attribute indicates the read state of the user books.
7. The `tagIdList` attribute is a list of tag IDs used to filter user books based on associated tags.
8. Encapsulates search criteria for querying user books in the application.
9. Designed to be used in conjunction with the `UserBookDao` class for searching user books based on specified criteria.