User Management Subsystem:

books-core

Constants

> (books-core/src/main/java/com/sismics/books/core/constant/Constants.java)

1. This class is used for storing application-wide constants that are used across different parts of the application.
2. Defines constants for default locale, timezone, theme ID, administrator's default password, and default generic user role.
3. Provides a convenient way to access these constants throughout the application without hardcoding their values.

AuthenticationToken (Model JPA)

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

1. The AuthenticationToken class is a JPA entity representing an authentication token stored in the database
2. Used in to manage user sessions and permissions.
3. Stores essential details about authentication tokens:
    - `id`: Token ID, primary key.
    - `userId`: ID of the associated user.
    - `longLasted`: Indicates if the token represents a long-lasting session.
    - `creationDate`: Date when the token was created.
    - `lastConnectionDate`: Date of the last connection using this token.

BaseFunction (Model JPA)

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

1. JPA entity representing information about base functions.
2. Represent fundamental capabilities or permissions within an application.
3. associated with `Role` to define roles and their associated base functions.

Role (Model JPA)

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

1. JPA entity representing information about roles.
2. Captures essential data related to roles, including their identifiers, names, and creation/deletion dates.
3. Allows for the management and tracking of roles throughout their lifecycle, including creation and deletion.

RoleBaseFunction (Model JPA)

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

1. JPA entity representing associations between roles and base functions.
2. Role base functions represent the permissions or capabilities associated with specific roles within the application.
3. Facilitates the management of role-permission relationships within the application's security model.
4. Enables the assignment of specific permissions (base functions) to roles, defining the access rights of users associated with those roles.
5. Supports the association of multiple base functions with a single role, allowing for flexible and granular access control.
6. Tracks the creation and deletion dates of role base function associations, providing insight into their lifecycle and history.

User (Model JPA)

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

1. JPA entity representing individual users within the application, including their login credentials, profile settings, and status flags.
2. Stores the following information about a user:
    - `id`: Unique identifier for the user.
    - `localeId`: ID of the user's preferred locale.
    - `roleId`: ID of the role associated with the user.
    - `username`: Username used for authentication.
    - `password`: Password used for authentication (typically stored securely hashed).
    - `email`: Email address associated with the user.
    - `theme`: Theme preference for the user interface.
    - `firstConnection`: Flag indicating whether the user has completed the first connection process.
    - `createDate`: Date when the user account was created.
    - `deleteDate`: Date when the user account was deleted (if applicable).
3. Supports the storage of user-specific settings and preferences, such as language and theme preferences.
4. Enables user-related actions such as authentication, user profile management, and account lifecycle management.

UserApp (Model JPA)

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

1. JPA entity used for Managing the association between users and connected applications, including OAuth tokens and user-specific data within each application.
2. Represents the link between a user account and an external application or service, such as a social media platform or third-party integration.
3. Stores the following information about a user's connection to an application:
    - `id`: Unique identifier for the user-app association.

    - `userId`: ID of the user associated with the application.
    - `appId`: ID of the connected application.
    - `accessToken`: OAuth access token obtained during authentication with the application.
    - `username`: User's username within the connected application (if applicable).
    - `externalId`: User's ID within the connected application (if applicable).
    - `email`: User's email address associated with the connected application (if applicable).
    - `sharing`: Flag indicating whether the user has enabled sharing data with the connected application.
    - `createDate`: Date when the user-app association was created.
    - `deleteDate`: Date when the user-app association was deleted (if applicable).
4. Facilitates the integration of third-party services or applications with the user management system.
5. Supports the storage of OAuth tokens and user-specific data required for interacting with external applications.
6. Enables features such as single sign-on (SSO) and data sharing between the application and connected services.

UserContact (Model JPA)

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

1. JPA entity used for Managing the contact information associated with a user in external applications or services.
2. Represents the linkage between a user account and a contact record in a connected application, such as a social media platform or external CRM system.
3. Stores the following information about a user's contact in a connected application:
    - `id`: Unique identifier for the user contact record.
    - `userId`: ID of the user associated with the contact.
    - `appId`: ID of the connected application.
    - `externalId`: ID of the contact within the connected application.
    - `fullName`: Full name of the contact in the connected application (if available).
    - `email`: Email address of the contact in the connected application (if available).
    - `createDate`: Date when the user contact record was created.
    - `updateDate`: Date when the user contact record was last updated.
    - `deleteDate`: Date when the user contact record was deleted (if applicable).

Locale (Model JPA)

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

1. JPA entity representing locale information within the application, including unique identifiers for different locales.
2. Stores the following information about a locale:
    - `id`: Unique identifier for the locale (e.g., "fr_FR").
3. Enables the storage and retrieval of locale-specific data and settings.
4. Facilitates internationalization and localization efforts within the application.

UserAppCreatedEvent

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

1. Represents an event indicating the creation of a user app.
2. event used for Broadcasting notifications about the creation of a user app within the system.
3. Enables loose coupling between components by allowing them to communicate through events rather than direct method calls.

UserAppDao

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

1. Used as a Data Access Object (DAO) for managing database operations related to user applications in the system.
2. Utilizes JPA Query API and native SQL queries for custom retrieval and manipulation of user app data.
3. Utilizes Data Transfer Objects (DTOs) such as UserAppDto to map database query results to Java objects for easier consumption.
4. Follows the Data Access Object (DAO) design pattern to separate the data access logic from the business logic, promoting modularity and maintainability.

RoleBaseFunctionDao

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

1. Used as a Data Access Object (DAO) for managing database operations related to base functions associated with a role.
2. Utilizes JPA Query API and native SQL queries for custom retrieval and manipulation of role-baseFunction data.

LocaleDao

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

1. Used as a Data Access Object (DAO) for managing database operations related to locales in the system.
2. Utilizes JPA Query API and native SQL queries for custom retrieval and manipulation of Locale data.

UserContactDao

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

1. Used as a Data Access Object (DAO) for managing database operations related to user contacts in the system.
2. Responsible for creating, reading, updating, and deleting user contacts in the database.
3. Additionally, it provides methods for searching user contacts based on various criteria.
4. Utilizes Data Transfer Objects (DTOs) such as UserContactDto to map database query results to Java objects for easier consumption.
5. Implements pagination for search results using PaginatedList to efficiently handle large datasets and improve performance.

6. Follows the Data Access Object (DAO) design pattern to separate the data access logic from the business logic, promoting modularity and maintainability.

UserDao

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

1. Used as a Data Access Object (DAO) for managing database operations related to users in the system.
2. Responsible for user authentication, creation, updating, deletion, and retrieval operations.
3. Provides methods for handling user passwords, including hashing and password reset.
4. Utilizes BCrypt for hashing passwords securely.
5. Implements pagination for search results using PaginatedList to efficiently handle large datasets and improve performance.
6. Follows the Data Access Object (DAO) design pattern to separate the data access logic from the business logic, promoting modularity and maintainability.

AuthenticationTokenDao

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

1. Used as a Data Access Object (DAO) for managing authentication tokens in the system.
2. Handles operations related to creating, retrieving, updating, and deleting authentication tokens.
3. Provides methods to manage the lifecycle of authentication tokens, including deletion of old tokens and updating last connection dates.
4. Follows the Data Access Object (DAO) design pattern to separate the data access logic from the business logic, promoting modularity and maintainability.

UserContactCriteria

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

1. Used as a criteria object for searching user contacts based on specific parameters.
2. Provides attributes to specify the application ID, user ID, and a full-text query for filtering contacts.
3. Primarily used for passing search criteria to methods that perform user contact searches.
4. Designed to be a lightweight container for holding search criteria, promoting encapsulation and reusability in search operations.

UserAppDto

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

1. Used as a Data Transfer Object (DTO) for transferring user application-related information between layers of the application.
2. Represents essential attributes of a user application, including identifiers, access token, username, and sharing status.
3. Designed to encapsulate user application data for ease of transfer and manipulation.
4. Supports boolean attribute sharing to indicate whether the user application shares traces, providing flexibility in representing application settings or preferences.

UserContactDto

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

1. Used as a Data Transfer Object (DTO) for transferring user contact-related information between layers of the application.
2. Represents essential attributes of a user contact, including identifiers, external contact ID, and full name.
3. Designed to encapsulate user contact data for ease of transfer and manipulation.

UserDto

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

1. Used as a Data Transfer Object (DTO) for transferring user-related information between layers of the application.
2. Represents essential attributes of a user, such as identifiers, locale, username, email, and creation timestamp.
3. Designed to encapsulate user data for ease of transfer and manipulation.

UserAppCreatedAsyncListener.java

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

> 1. Listens for `UserAppCreatedEvent` and logs the event upon occurrence.
> 2. Retrieves the `UserApp` instance from the event to identify the app and user involved.
> 3. Executes contact synchronization for the specified app, currently supporting Facebook through `FacebookService`.
> 4. Utilizes a switch case on `AppId` to handle different app integrations, indicating potential for future expansion.
> 5. Ensures the operation is wrapped in a transaction to maintain data consistency and integrity.

books-web

BaseFunction

> (books-web/src/main/java/com/sismics/books/rest/constant/BaseFunction.java)

1. Enum representing base functions within the application, defining specific functionalities that users can perform.
2. Provides a single base function:
   - `ADMIN`: Allows the user to access and utilize administrative functions.
3. Enumerates essential functionalities that users can execute within the system.
4. Used for role-based access control and permission management within the application.

BaseResource

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

1. This class serves as a abstract base class for REST resources in a web application
2. Provides common functionality for REST resources, such as handling HTTP requests, extracting query parameters, and managing user authentication.
3. Implements methods for authentication (`authenticate`()) and authorization (`checkBaseFunction`() and `hasBaseFunction`()).
4. Relies on the `HttpServletRequest` to access HTTP request details.
5. Utilizes query parameters, specifically the app_key parameter, for application-level authentication or identification.
6. Encapsulates the Principal of the authenticated user, allowing subclasses to access user-related information easily.
7. The class is designed to be extended by specific resource classes, providing them with common functionalities and enforcing security measures consistently across the application's REST endpoints.

UserResource

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

1. This class is used as a `REST resource` for user-related operations in a web application.
2. Extends the `BaseResource` class, inheriting common functionalities such as authentication and authorization checks.
3. Implements various methods to handle HTTP requests for different `user operations` such as registration, login, logout, updating user details, checking username availability, and managing user sessions.
4. Utilizes query parameters, form parameters, and path parameters to receive input data for user operations.
5. Communicates with the data access layer (`DAO`) to interact with the underlying database, performing operations such as user creation, retrieval, update, and deletion.
6. Utilizes JSON for request and response data serialization/deserialization

LocaleResource

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

1. REST resource handling locale-related operations, such as retrieving the list of all available locales.
2. Defines a resource endpoint at /locale for handling locale-related requests.
3. Provides a method list() to retrieve the list of all locales in JSON format.
4. Utilizes `LocaleDao` to fetch the list of locales from the database.
5. Returns a response with the JSON representation of the locales list.

books-web-common

IPrincipal

> (books-web-common/src/main/java/com/sismics/security/IPrincipal.java)

1. Extends the `Principal` interface, which is a standard Java interface representing the identity of a principal (e.g., user, group, service).

2. This interface defines the contract for principals in the system. Principals typically represent authenticated users or entities with certain permissions within the application.
3. Encapsulates user-related information, such as `ID`, `locale`, `timezone`, and `email`, within the principal abstraction, promoting encapsulation and separation of concerns.
4. Serves as a way to abstract the details of user authentication and authorization, allowing different implementations (e.g., `user principals`, `service principals`) to conform to a common interface.

AnonymousPrincipal

> (books-web-common/src/main/java/com/sismics/security/AnonymousPrincipal.java)

1. This class represents an anonymous principal, typically used to represent unauthenticated users or entities lacking specific identity within the application.
2. Implements the `IPrincipal` interface, providing necessary methods to adhere to the contract defined by the interface.
3. Represents a foundational component in handling anonymous access or actions within the application, serving as a placeholder for unauthenticated users or entities with limited identity information.

UserPrincipal

> (books-web-common/src/main/java/com/sismics/security/UserPrincipal.java)

1. This class represents authenticated users within the system, serving as the principal entity associated with user authentication and authorization.
2. Implements the `IPrincipal` interface, providing necessary methods to adhere to the contract defined by the interface.
3. Represents a key component in user authentication and authorization, serving as the principal entity associated with authenticated user sessions.
4. Contains a base function set to represent the set of base functions associated with the authenticated user, facilitating access control and permissions management within the system.

TokenBasedSecurityFilter

> (books-web-common/src/main/java/com/sismics/util/filter/TokenBasedSecurityFilter.java)

1. This class is a filter used for authenticating users based on an authentication token stored in a cookie.
2. It is responsible for extracting the authentication token from the request cookie, validating its authenticity, and injecting the corresponding user principal into the request attributes.
3. Provides logic to extract the authentication token from the request cookie and retrieve the corresponding server token from the database.
4. Checks the validity and expiration status of the authentication token, determining whether the user is authenticated or anonymous.
5. Injects an authenticated user principal or an anonymous user principal into the request attributes based on the authentication status.
6. Utilizes `DAO` classes (`AuthenticationTokenDao`, `UserDao`, `RoleBaseFunctionDao`) for database interactions to retrieve authentication token information and user data.

7. Serves as a critical component for user authentication and authorization within the application, implementing token-based security mechanisms.