EjbJpa Documentation

Book Entity

The Book entity stands as a cornerstone in the library management project, embodying the essence of each literary work with a meticulous design that aligns seamlessly with Java Persistence API (JPA) standards. The entity is encapsulated within a Java class, adorned with JPA annotations to facilitate effortless mapping to the corresponding database table, named books. This table holds essential attributes, including a unique identifier (id), the book's title, and a foreign key (user\_id) establishing a link to the corresponding member who possesses or borrows the book. The title column captures the literary identity of each book, while the user\_id column serves as a relational bridge between the Book and Member entities, creating a dynamic connection between books and their respective members. This thoughtful design ensures a comprehensive representation of each book's details while fostering a relational structure that enhances the system's overall coherence.

Member Entity

Complementing the Book entity, the Member entity represents the individuals who form the library community. Like its counterpart, the Member entity is meticulously crafted with JPA annotations, defining its structure for seamless database mapping. The entity corresponds to the members table, where each row captures member information, including a unique identifier (id), the member's username, and a foreign key (book\_id) establishing the relationship with the books they borrow. The username column captures the distinct identity of each member, and the book\_id column serves as a crucial link to the borrowed books. This interplay between entity attributes and database columns ensures an organized and efficient representation of member details within the system. The Member entity, thus, becomes a central figure, fostering meaningful connections with the library's literary resources.

Database Tables: books and members

books Table

The books table serves as the repository for detailed information about each book in the library. Structured with columns such as id (a unique identifier), title, and user\_id (a foreign key linking to the members table), this table provides a comprehensive catalog of the library's literary collection. The id column ensures the uniqueness of each book, allowing for precise identification and retrieval. Meanwhile, the user\_id column establishes a vital connection to the members who possess or borrow the book, creating a relational structure that enhances the dynamic interactions between books and members within the library system.

members Table

On the other side, the members table is dedicated to storing information about library members. With columns including id (a unique identifier), username, and book\_id (a foreign key linking to the books table), this table captures the personal details of each member and their borrowing activities. The id column ensures the uniqueness of each member, facilitating efficient retrieval, while the book\_id column establishes a link to the books borrowed by the member. This structured representation allows for a seamless management of the library community, enabling the tracking of member details and their interactions with the library's literary resources.

The careful design and interplay between these entities and their corresponding tables create a robust foundation for the library management system. The books and members tables, with their well-defined columns, not only capture the nuances of each entity but also establish an intricate relational structure that enriches the library's functionality, creating a coherent and efficient system for managing books and members alike.

Glassfish Client with JNDI

The Glassfish client, a vital facet of our EJB-JPA project, offers a distinct approach to accessing Enterprise JavaBeans (EJBs) through the use of Java Naming and Directory Interface (JNDI). This client application embraces the flexibility of JNDI, a powerful naming service in Java, to locate and obtain references to the EJBs hosted on the server. By employing JNDI, the Glassfish client is equipped to dynamically discover and connect to the necessary EJBs during runtime. This dynamic resolution not only enhances the client's adaptability to changes in the server environment but also enables a seamless integration with the EJB components. The Glassfish client, through its intelligent utilization of JNDI, underscores the versatility and extensibility of the EJB-JPA project, showcasing a sophisticated approach to enterprise application development.

Wildfly Client with EJB Injection

In contrast, the Wildfly client brings a different perspective to the interaction with EJBs by leveraging direct EJB injection. This client capitalizes on the inherent capabilities of the Wildfly application server, which supports the injection of EJBs directly into the client components. This approach streamlines the client's code, eliminating the need for explicit JNDI lookups and providing a more straightforward and concise integration with the EJBs. Through EJB injection, the Wildfly client benefits from the server's managed environment, gaining access to the desired EJBs as if they were local components. This direct integration simplifies the client's development process, reducing the boilerplate code associated with JNDI lookups. The Wildfly client, with its emphasis on EJB injection, exemplifies a streamlined and efficient method of incorporating EJBs into client applications, showcasing the convenience and elegance of the Wildfly application server.

At the core of our EJB-JPA project, a versatile server module emerges, serving as the unified backbone for both Glassfish and Wildfly clients. This server module encapsulates the essential components, including the EJBs, that are shared and utilized by both client applications. The creation of this centralized server module underscores a strategic architectural decision, promoting code reusability, maintainability, and a consistent business logic implementation across diverse client environments.

The server module is carefully crafted to host the session beans, persistence layer, and other essential components necessary for the library management system. By isolating these shared elements into a dedicated module, we ensure that changes or updates to the core logic are seamlessly propagated to both clients. This modular approach not only enhances the efficiency of development but also fosters a streamlined and coherent architecture that aligns with the principles of enterprise application design.

Through the server module, the Glassfish and Wildfly clients can seamlessly interact with the shared EJBs, leveraging the same underlying business logic. This unification simplifies maintenance, reduces redundancy, and establishes a robust foundation for future scalability. The server module, acting as the common thread between the two clients, exemplifies a judicious design choice that harmonizes the diverse aspects of the EJB-JPA project.

MySQL Database: The Persistent Repository

At the heart of our library management system lies the MySQL database, functioning as the persistent repository for the crucial tables – books and members. This relational database is meticulously designed to capture and organize the wealth of information associated with both books and members. The database schema reflects the entity relationships defined in the EJB-JPA project, ensuring a coherent mapping between the object-oriented model and the relational database structure.

The books and members tables, with their respective columns (id, title, user\_id, username, and book\_id), reside within the MySQL database, forming the backbone of the library management system. These tables not only store the intricate details of each book and member but also facilitate the dynamic relationships between them. The use of a robust and widely adopted database management system like MySQL ensures data integrity, reliability, and efficient querying, essential factors for the seamless functioning of the library management system.

The integration of MySQL as the backend database underscores the project's commitment to leveraging established and reliable technologies. It ensures that the persistence layer of the EJB-JPA project aligns with industry best practices, offering a stable and scalable foundation for managing the library's resources. The MySQL database, in tandem with the server module, forms a powerful duo that not only stores and retrieves data efficiently but also ensures a coherent and well-organized system for library management.

The Deployment was done from the admin console interface, adding the connection pools as well as generating the jdbc data sources.

Functionalities: are Adding a new reserved book

Getting all the books reserved by the users in the system.