Service

Package: com.backend.server.service

Description

The CurrentUser class implements the Spring Security UserDetails interface, which represents the user details required for authentication and authorization. It is used to provide user information to Spring Security during the authentication process.

Constructors

public CurrentUser(Optional<User> user)

Description: Constructs a CurrentUser object with the user information provided as an Optional<User> parameter.

Parameters:

user (Type: Optional<User>) - An optional User object containing the user information.

Methods

getAuthorities()

Description: Retrieves the authorities granted to the user.

Returns:

(Type: Collection<? extends GrantedAuthority>) A collection of authorities representing the user's roles or permissions. In this implementation, it returns a singleton collection with a SimpleGrantedAuthority with the role "USER".

getPassword()

Description: Retrieves the password of the user.

Returns:

(Type: String) The password of the user.

getUsername()

Description: Retrieves the username (mobile number) of the user.

Returns:

(Type: String) The mobile number of the user.

isAccountNonExpired()

Description: Indicates whether the user's account is not expired (always returns true).

Returns:

(Type: boolean) true if the user's account is not expired, otherwise false.

isAccountNonLocked()

Description: Indicates whether the user's account is not locked (always returns true).

Returns:

(Type: boolean) true if the user's account is not locked, otherwise false.

isCredentialsNonExpired()

Description: Indicates whether the user's credentials (password) are not expired (always returns true).

Returns:

(Type: boolean) true if the user's credentials are not expired, otherwise false.

isEnabled()

Description: Indicates whether the user's account is enabled (always returns true).

Returns:

(Type: boolean) true if the user's account is enabled, otherwise false.

Usage

The CurrentUser class is used as part of the authentication process in Spring Security. It provides the necessary user details, such as username (mobile number) and authorities (roles), to validate and authenticate the user during login.

When a user attempts to log in, Spring Security will use the loadUserByUsername method of the UserDetailsService to retrieve the user's details, which typically include the user's password and authorities. In this case, the CurrentUser class acts as a custom implementation of the UserDetails interface, providing the necessary user information based on the User object passed to its constructor.

Please ensure that the User class is correctly implemented and used to represent user data, and the appropriate security configurations are in place to handle authentication and authorization in your Spring Security setup. Additionally, consider enhancing the CurrentUser class to include additional user-specific details and permissions if required by your application's security requirements.

Package: com.backend.server.service

Description

The CurrentUserDetails class is a Spring service that implements the UserDetailsService interface. It is responsible for loading user details from the database based on the provided username (mobile number) during the authentication process in Spring Security.

Fields

userRepository (Type: UserRepository)

Description: An instance of the UserRepository interface, which is used to interact with the database to retrieve user data.

Methods

loadUserByUsername(String username)

Description: Loads user details from the database based on the provided username (mobile number). It is called by Spring Security during the authentication process when a user attempts to log in.

Parameters:

username (Type: String) - The mobile number (username) of the user attempting to log in.

Returns:

(Type: UserDetails) A UserDetails object representing the user's details, or throws a UsernameNotFoundException if the user is not found.

Annotations

@Service: Indicates that this class is a Spring service and should be automatically detected during component scanning.

Autowired Dependencies

UserRepository userRepository: An instance of the UserRepository interface, which is autowired by Spring to interact with the database and retrieve user data.

Usage

The CurrentUserDetails class is used as part of the authentication process in Spring Security. When a user attempts to log in, Spring Security will call the loadUserByUsername method to retrieve the user's details based on the provided username (mobile number).

In the loadUserByUsername method, the user details are retrieved from the database using the UserRepository by querying for the user with the matching mobile number. If the user is found, a CurrentUser object is created, wrapping the retrieved User object to implement the UserDetails interface. This CurrentUser object contains the necessary user information and authorities (roles) required for authentication and authorization.

If the user is not found in the database, the method throws a UsernameNotFoundException, indicating that the user with the provided username does not exist.

Please ensure that the UserRepository is correctly implemented to interact with the database and retrieve user data based on the mobile number (username). Additionally, ensure that the appropriate security configurations are in place to use the CurrentUserDetails class as the UserDetailsService in your Spring Security setup.

Package: com.backend.server.service

Description

The MovieService interface defines the contract for the service layer to handle movie-related operations in the backend server. It declares several methods to perform CRUD operations on movie data.

Methods

getAllMovies()

Description: Retrieves a list of all movies available in the system.

Returns:

(Type: List<Movies>) A list of all movies available in the system.

addAllMovies(List<Movies> movies)

Description: Saves multiple movies in the system.

Parameters:

movies (Type: List<Movies>) - A list of Movies objects to be added to the system.

Returns:

(Type: List<Movies>) A list of movies that have been successfully added to the system.

addMovie(Movies movie)

Description: Saves a single movie in the system.

Parameters:

movie (Type: Movies) - The Movies object to be added to the system.

Returns:

(Type: Movies) The saved Movies object with its generated ID.

findByStatus(String status)

Description: Retrieves a list of movies based on the provided status.

Parameters:

status (Type: String) - The status to filter the movies.

Returns:

(Type: List<Movies>) A list of movies that match the given status.

updateMovie(Movies movie)

Description: Updates a movie in the system.

Parameters:

movie (Type: Movies) - The Movies object to be updated in the system.

Returns:

(Type: Movies) The updated Movies object.

findByGener(String gener)

Description: Retrieves a list of movies based on the provided genre.

Parameters:

gener (Type: String) - The genre to filter the movies.

Returns:

(Type: List<Movies>) A list of movies that match the given genre.

deleteMovie(Movies movie)

Description: Deletes a movie from the system.

Parameters:

movie (Type: Movies) - The Movies object to be deleted from the system.

Returns: None

Usage

The MovieService interface serves as a blueprint for the service layer to handle movie-related operations in the backend server. Implementations of this interface will provide the actual business logic to interact with the database and perform CRUD operations on movie data.

To use this service, create a concrete implementation of the MovieService interface and provide the necessary logic to interact with the database or perform other business-specific operations related to movies.

The MovieService interface is helpful in achieving separation of concerns and maintaining a clear distinction between the controller layer and the data access layer. By defining the movie-related methods in this interface, you can ensure that different implementations can be used seamlessly by the controllers without affecting the overall functionality of the application.

Ensure that the concrete implementation of MovieService is correctly injected into the controllers or other components that require movie-related functionality. Also, consider adding necessary error handling and validation logic based on the specific requirements of your application.

Package: com.backend.server.service

Description

The MovieServiceImpl class is a Spring service that implements the MovieService interface. It provides the actual implementation for handling movie-related operations in the backend server. This class interacts with the MovieRepository to perform CRUD operations on movie data.

Fields

movieRepository (Type: MovieRepository)

Description: An instance of the MovieRepository interface, which is used to interact with the database to perform movie-related operations.

Methods

getAllMovies()

Description: Retrieves a list of all movies available in the system.

Returns:

(Type: List<Movies>) A list of all movies available in the system.

addAllMovies(List<Movies> movies)

Description: Saves multiple movies in the system.

Parameters:

movies (Type: List<Movies>) - A list of Movies objects to be added to the system.

Returns:

(Type: List<Movies>) A list of movies that have been successfully added to the system.

addMovie(Movies movie)

Description: Saves a single movie in the system.

Parameters:

movie (Type: Movies) - The Movies object to be added to the system.

Returns:

(Type: Movies) The saved Movies object with its generated ID.

updateMovie(Movies movie)

Description: Updates a movie in the system.

Parameters:

movie (Type: Movies) - The Movies object to be updated in the system.

Returns:

(Type: Movies) The updated Movies object.

findByStatus(String status)

Description: Retrieves a list of movies based on the provided status.

Parameters:

status (Type: String) - The status to filter the movies.

Returns:

(Type: List<Movies>) A list of movies that match the given status.

findByGener(String gener)

Description: Retrieves a list of movies based on the provided genre.

Parameters:

gener (Type: String) - The genre to filter the movies.

Returns:

(Type: List<Movies>) A list of movies that match the given genre.

deleteMovie(Movies movie)

Description: Deletes a movie from the system.

Parameters:

movie (Type: Movies) - The Movies object to be deleted from the system.

Returns: None

Annotations

@Service: Indicates that this class is a Spring service and should be automatically detected during component scanning.

Autowired Dependencies

MovieRepository movieRepository: An instance of the MovieRepository interface, which is autowired by Spring to interact with the database and perform movie-related operations.

Usage

The MovieServiceImpl class serves as the implementation of the MovieService interface, providing the actual business logic to interact with the database and perform CRUD operations on movie data.

When the methods of the MovieService interface are called by the controller or other components, the MovieServiceImpl class will perform the corresponding database operations using the MovieRepository. For example, getAllMovies() will retrieve all movies from the database, addMovie() will save a new movie, and updateMovie() will update an existing movie.

Ensure that the MovieRepository is correctly implemented to interact with the database and perform the necessary movie-related operations. Also, consider adding additional error handling and validation logic based on the specific requirements of your application.

Finally, make sure that the MovieServiceImpl class is correctly registered as the implementation of the MovieService interface in your application's configuration to ensure that it is used as the service layer for handling movie-related operations in the backend server.

Package: com.backend.server.service

Description

The TheaterService interface defines the contract for the service layer to handle theater-related operations in the backend server. It declares several methods to perform CRUD operations and other functionalities related to users and theaters.

Methods

addUser(User user)

Description: Saves a new user in the system.

Parameters:

user (Type: User) - The User object to be added to the system.

Returns:

(Type: User) The saved User object with its generated ID.

getUser(String email)

Description: Retrieves a user based on the provided email.

Parameters:

email (Type: String) - The email address to retrieve the user.

Returns:

(Type: Optional<User>) An optional User object that matches the provided email, or an empty optional if no user is found.

updateUser(User user)

Description: Updates a user in the system.

Parameters:

user (Type: User) - The User object to be updated in the system.

Returns:

(Type: User) The updated User object.

deleteUser(String mobileNumber)

Description: Deletes a user from the system based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number of the user to be deleted.

Returns:

(Type: String) A string message indicating the status of the deletion process.

getAvailableTickets(int theaterId)

Description: Retrieves the number of available tickets for a specific theater.

Parameters:

theaterId (Type: int) - The ID of the theater to check for available tickets.

Returns:

(Type: int) The number of available tickets for the specified theater.

Usage

The TheaterService interface serves as a blueprint for the service layer to handle theater-related operations in the backend server. Implementations of this interface will provide the actual business logic to interact with the database or perform other business-specific operations related to users and theaters.

To use this service, create a concrete implementation of the TheaterService interface and provide the necessary logic to interact with the database or perform other functionalities related to users and theaters.

Ensure that the concrete implementation of TheaterService is correctly injected into the controllers or other components that require theater-related functionality. Also, consider adding necessary error handling and validation logic based on the specific requirements of your application.

By using the TheaterService interface, you can achieve separation of concerns and maintain a clear distinction between the controller layer and the data access layer. This allows for easier testing and maintenance of the application as the business logic is decoupled from the controller and can be swapped with different implementations if needed.

Package: com.backend.server.service

Description

The TheaterServiceImpl class is a Spring service that implements the UserService interface. It provides the actual implementation for handling user-related operations in the backend server. This class interacts with the UserRepository to perform CRUD operations on user data and other functionalities related to users.

Fields

userRepository (Type: UserRepository)

Description: An instance of the UserRepository interface, which is used to interact with the database to perform user-related operations.

Methods

addUser(User user)

Description: Saves a new user in the system and encrypts the user's password using BCryptPasswordEncoder before saving.

Parameters:

user (Type: User) - The User object to be added to the system.

Returns:

(Type: User) The saved User object with its generated ID.

getUser(String mobileNumber)

Description: Retrieves a user based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number to retrieve the user.

Returns:

(Type: Optional<User>) An optional User object that matches the provided mobile number, or an empty optional if no user is found.

updateUser(User user)

Description: Updates a user in the system.

Parameters:

user (Type: User) - The User object to be updated in the system.

Returns:

(Type: User) The updated User object.

deleteUser(String mobileNumber)

Description: Deletes a user from the system based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number of the user to be deleted.

Returns:

(Type: String) A string message indicating the status of the deletion process.

getAvailableTickets(int theaterId)

Description: Retrieves the number of available tickets for a specific theater based on the provided theater ID.

Parameters:

theaterId (Type: int) - The ID of the theater to check for available tickets.

Returns:

(Type: int) The number of available tickets for the specified theater.

Annotations

@Service: Indicates that this class is a Spring service and should be automatically detected during component scanning.

Autowired Dependencies

UserRepository userRepository: An instance of the UserRepository interface, which is autowired by Spring to interact with the database and perform user-related operations.

Usage

The TheaterServiceImpl class serves as the implementation of the UserService interface, providing the actual business logic to interact with the database and perform CRUD operations on user data and other functionalities related to users.

When the methods of the UserService interface are called by the controller or other components, the TheaterServiceImpl class will perform the corresponding database operations using the UserRepository. For example, addUser() will save a new user with an encrypted password, getUser() will retrieve a user based on the mobile number, and deleteUser() will delete a user based on the mobile number.

Ensure that the UserRepository is correctly implemented to interact with the database and perform the necessary user-related operations. Also, consider adding additional error handling and validation logic based on the specific requirements of your application.

Finally, make sure that the TheaterServiceImpl class is correctly registered as the implementation of the UserService interface in your application's configuration to ensure that it is used as the service layer for handling user-related operations in the backend server.

Package: com.backend.server.service

Description

The TicketService interface defines the contract for the service layer to handle ticket-related operations in the backend server. It declares several methods to perform CRUD operations and other functionalities related to tickets.

Methods

addticket(ticket ticket)

Description: Saves a new ticket in the system.

Parameters:

ticket (Type: ticket) - The ticket object to be added to the system.

Returns:

(Type: ticket) The saved ticket object with its generated ID.

getticket(String email)

Description: Retrieves a ticket based on the provided email.

Parameters:

email (Type: String) - The email address to retrieve the ticket.

Returns:

(Type: Optional<ticket>) An optional ticket object that matches the provided email, or an empty optional if no ticket is found.

updateticket(ticket ticket)

Description: Updates a ticket in the system.

Parameters:

ticket (Type: ticket) - The ticket object to be updated in the system.

Returns:

(Type: ticket) The updated ticket object.

deleteticket(String mobileNumber)

Description: Deletes a ticket from the system based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number of the ticket owner to be deleted.

Returns:

(Type: String) A string message indicating the status of the deletion process.

cancelTicket(String mobileNumber)

Description: Cancels a ticket based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number of the ticket owner to be canceled.

Returns:

(Type: String) A string message indicating the status of the ticket cancellation process.

Usage

The TicketService interface serves as a blueprint for the service layer to handle ticket-related operations in the backend server. Implementations of this interface will provide the actual business logic to interact with the database or perform other business-specific operations related to tickets.

To use this service, create a concrete implementation of the TicketService interface and provide the necessary logic to interact with the database or perform other functionalities related to tickets.

Ensure that the concrete implementation of TicketService is correctly injected into the controllers or other components that require ticket-related functionality. Also, consider adding necessary error handling and validation logic based on the specific requirements of your application.

By using the TicketService interface, you can achieve separation of concerns and maintain a clear distinction between the controller layer and the data access layer. This allows for easier testing and maintenance of the application as the business logic is decoupled from the controller and can be swapped with different implementations if needed.

Package: com.backend.server.service

Description

The TicketServiceImpl class is a Spring service that implements the tickerService interface. It provides the actual implementation for handling ticker-related operations in the backend server. This class interacts with the tickerRepository and TheaterRepository to perform CRUD operations on ticker data and other functionalities related to tickers.

Fields

tickerRepository (Type: tickerRepository)

Description: An instance of the tickerRepository interface, which is used to interact with the database to perform ticker-related operations.

theterRepository (Type: TheaterRepository)

Description: An instance of the TheaterRepository interface, which is used to interact with the database to perform operations related to theaters.

Methods

addticker(ticker ticker)

Description: Saves a new ticker in the system and encrypts the ticker's password using BCryptPasswordEncoder before saving.

Parameters:

ticker (Type: ticker) - The ticker object to be added to the system.

Returns:

(Type: ticker) The saved ticker object with its generated ID.

getticker(String mobileNumber)

Description: Retrieves a ticker based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number to retrieve the ticker.

Returns:

(Type: Optional<ticker>) An optional ticker object that matches the provided mobile number, or an empty optional if no ticker is found.

updateticker(ticker ticker)

Description: Updates a ticker in the system.

Parameters:

ticker (Type: ticker) - The ticker object to be updated in the system.

Returns:

(Type: ticker) The updated ticker object.

deleteticker(String mobileNumber)

Description: Deletes a ticker from the system based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number of the ticker owner to be deleted.

Returns:

(Type: String) A string message indicating the status of the deletion process.

cancelTicket(String mobileNumber)

Description: Cancels a ticker based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number of the ticker owner to be canceled.

Returns:

(Type: String) A string message indicating the status of the ticker cancellation process.

Annotations

@Service: Indicates that this class is a Spring service and should be automatically detected during component scanning.

@Autowired: Indicates that the tickerRepository and theaterRepository instances should be autowired by Spring to interact with the database and perform ticker-related operations.

Usage

The TicketServiceImpl class serves as the implementation of the tickerService interface, providing the actual business logic to interact with the database and perform CRUD operations on ticker data and other functionalities related to tickers.

When the methods of the tickerService interface are called by the controller or other components, the TicketServiceImpl class will perform the corresponding database operations using the tickerRepository and TheaterRepository. For example, addticker() will save a new ticker with an encrypted password, getticker() will retrieve a ticker based on the mobile number, and deleteticker() will delete a ticker based on the mobile number.

Ensure that the tickerRepository and TheaterRepository are correctly implemented to interact with the database and perform the necessary ticker and theater-related operations. Also, consider adding additional error handling and validation logic based on the specific requirements of your application.

Finally, make sure that the TicketServiceImpl class is correctly registered as the implementation of the tickerService interface in your application's configuration to ensure that it is used as the service layer for handling ticker-related operations in the backend server.

Package: com.backend.server.service

Description

The UserService interface defines the contract for the service layer to handle user-related operations in the backend server. It declares several methods to perform CRUD operations and other functionalities related to users.

Methods

addUser(User user)

Description: Saves a new user in the system.

Parameters:

user (Type: User) - The User object to be added to the system.

Returns:

(Type: User) The saved User object with its generated ID.

getUser(String email)

Description: Retrieves a user based on the provided email.

Parameters:

email (Type: String) - The email address to retrieve the user.

Returns:

(Type: Optional<User>) An optional User object that matches the provided email, or an empty optional if no user is found.

updateUser(User user)

Description: Updates a user in the system.

Parameters:

user (Type: User) - The User object to be updated in the system.

Returns:

(Type: User) The updated User object.

deleteUser(String mobileNumber)

Description: Deletes a user from the system based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number of the user to be deleted.

Returns:

(Type: String) A string message indicating the status of the deletion process.

Usage

The UserService interface serves as a blueprint for the service layer to handle user-related operations in the backend server. Implementations of this interface will provide the actual business logic to interact with the database or perform other business-specific operations related to users.

To use this service, create a concrete implementation of the UserService interface and provide the necessary logic to interact with the database or perform other functionalities related to users.

Ensure that the concrete implementation of UserService is correctly injected into the controllers or other components that require user-related functionality. Also, consider adding necessary error handling and validation logic based on the specific requirements of your application.

By using the UserService interface, you can achieve separation of concerns and maintain a clear distinction between the controller layer and the data access layer. This allows for easier testing and maintenance of the application as the business logic is decoupled from the controller and can be swapped with different implementations if needed.

Package: com.backend.server.service

Description

The UserServiceImpl class is a Spring service that implements the UserService interface. It provides the actual implementation for handling user-related operations in the backend server. This class interacts with the UserRepository to perform CRUD operations on user data and other functionalities related to users.

Fields

userRepository (Type: UserRepository)

Description: An instance of the UserRepository interface, which is used to interact with the database to perform user-related operations.

Methods

addUser(User user)

Description: Saves a new user in the system and encrypts the user's password using BCryptPasswordEncoder before saving.

Parameters:

user (Type: User) - The User object to be added to the system.

Returns:

(Type: User) The saved User object with its generated ID.

getUser(String mobileNumber)

Description: Retrieves a user based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number to retrieve the user.

Returns:

(Type: Optional<User>) An optional User object that matches the provided mobile number, or an empty optional if no user is found.

updateUser(User user)

Description: Updates a user in the system.

Parameters:

user (Type: User) - The User object to be updated in the system.

Returns:

(Type: User) The updated User object.

deleteUser(String mobileNumber)

Description: Deletes a user from the system based on the provided mobile number.

Parameters:

mobileNumber (Type: String) - The mobile number of the user to be deleted.

Returns:

(Type: String) A string message indicating the status of the deletion process.

Annotations

@Service: Indicates that this class is a Spring service and should be automatically detected during component scanning.

@Autowired: Indicates that the userRepository instance should be autowired by Spring to interact with the database and perform user-related operations.

Usage

The UserServiceImpl class serves as the implementation of the UserService interface, providing the actual business logic to interact with the database and perform CRUD operations on user data and other functionalities related to users.

When the methods of the UserService interface are called by the controller or other components, the UserServiceImpl class will perform the corresponding database operations using the UserRepository. For example, addUser() will save a new user with an encrypted password, getUser() will retrieve a user based on the mobile number, and deleteUser() will delete a user based on the mobile number.

Ensure that the UserRepository is correctly implemented to interact with the database and perform the necessary user-related operations. Also, consider adding additional error handling and validation logic based on the specific requirements of your application.

Finally, make sure that the UserServiceImpl class is correctly registered as the implementation of the UserService interface in your application's configuration to ensure that it is used as the service layer for handling user-related operations in the backend server.