BACKEND DOCUMENTATION

Team: TechBrewers

WRITTEN BY: ANUPAM SHARMA Edited by: Dhruba Das

Business Logic (Business Layer) (Booking.java)

Package: com.mycompany.buisness

Description

The Booking class represents a booking for a meeting room. It encapsulates information about the booking, including the booking ID, meeting room ID, date, start time, end time, and the ID of the user who made the booking.

Constructors

- *Booking()* → Default constructor with no parameters.
- *Booking(String bookingId, String meetingRoomId, Date date, int startTime, int endTime, String bookedById)* → Constructor to create a Booking object with specified attributes.

Attributes

- *private String bookingId* → Represents the unique booking ID associated with the booking.
- *private String meetingRoomId* → Represents the ID of the meeting room for which the booking is made.
- *private Date date* → Represents the date of the booking.
- *private int startTime* → Represents the start time of the booking (e.g., in hours or minutes).
- *private int endTime* → Represents the end time of the booking (e.g., in hours or minutes).
- *private String bookedById* → Represents the unique ID of the user who made the booking.

Methods

- *public String getBookingId()* → Returns the booking ID.
- *public void setBookingId(String bookingId)*→Sets the booking ID.
- *public String getMeetingRoomId()*→Returns the meeting room ID.
- *public void setMeetingRoomId(String meetingRoomId)*→Sets the meeting room ID.
- *public Date getDate()*→Returns the date of the booking.
- *public void setDate(Date date)*→Sets the date of the booking.
- *public int getStartTime()*→Returns the start time of the booking.
- *public void setStartTime(int startTime)*→Sets the start time of the booking.

- *public int getEndTime()*→Returns the end time of the booking.
- *public void setEndTime(int endTime)*→Sets the end time of the booking.
- *public String getBookedById()*→Returns the ID of the user who made the booking.
- *public void setBookedById(String bookedById)*→Sets the ID of the user who made the booking.

Usage

This class can be used to create and manage booking objects within a booking management system. Users can create instances of the Booking class to represent bookings for meeting rooms, and the class provides methods to get and set attributes associated with each booking.

Meeting Class Documentation (Meeting.java)

Package: com.mycompany.buisness

Description

The Meeting class represents a meeting event within a business application. It contains information about the meeting, including the meeting ID, title, organizer, meeting date, start time, end time, type, and a list of attendees.

Attributes

- private String meetingId →Represents the unique identifier for the meeting.
- *private String title*→ Represents the title or name of the meeting.
- *private String organizedBy* → Represents the user ID of the organizer who scheduled the meeting.
- *private Date meetingDate*→Represents the date on which the meeting is scheduled.
- *private int startTime*→ Represents the start time of the meeting (e.g., in hours or minutes).
- *private int endTime*→Represents the end time of the meeting (e.g., in hours or minutes).
- *private String type*→Represents the type or category of the meeting (e.g., "Team Meeting," "Client Meeting").
- *private List<String> attendees*→Represents a list of user IDs of attendees who are invited to the meeting.

Methods

- *public String getMeetingId()*→Returns the meeting ID.
- *public void setMeetingId(String meetingId)* → Sets the meeting ID.
- *public String getTitle()*→Returns the title of the meeting.
- *public void setTitle(String title)*→Sets the title of the meeting.
- *public String getOrganizedBy()*→Returns the user ID of the organizer.
- *public void setOrganizedBy(String organizedBy)*→Sets the user ID of the organizer.
- *public Date getMeetingDate()*→Returns the date of the meeting.
- *public void setMeetingDate(Date meetingDate)*→Sets the date of the meeting.
- *public int getStartTime()*→Returns the start time of the meeting.
- *public void setStartTime(int startTime*)→Sets the start time of the meeting.
- public int getEndTime()→Returns the end time of the meeting.
- *public void setEndTime(int endTime)*→Sets the end time of the meeting.

- *public String getType()*→Returns the type of the meeting.
- *public void setType(String type)*→Sets the type of the meeting.
- *public List<String> getAttendees()*→*Returns* the list of user IDs of attendees.
- *public void setAttendees(List<String> attendees)*→Sets the list of user IDs of attendees.

Usage

The Meeting class can be used to represent individual meetings within a business application. Users can create instances of the Meeting class to store and manage meeting details, including the organizer, attendees, date, and time. It is a valuable component for building scheduling and collaboration features in business software.

MeetingManager Class Documentation

Package: com.mycompany.buisness

Description

The MeetingManager class is a component of a business application designed to manage meetings. It provides methods for retrieving, creating, updating, and deleting meetings. This class encapsulates the business logic related to meetings and handles exceptions that may occur during meeting management operations.

Constructor

- *public MeetingManager(MeetingDao meetingDao)*

Constructor that initializes the MeetingManager with a MeetingDao object, which is responsible for data access operations related to meetings.

Methods

- *public Meeting getMeetingById(String meetingId*) throws BusinessException →Retrieves a meeting by its unique meeting ID.

Throws a BusinessException if the meeting is not found or if there is an error during data retrieval.

- *public List<Meeting> getAllMeetings() throws BusinessException* →Retrieves a list of all meetings.

Throws a BusinessException if there is an error during data retrieval.

- *public Meeting createMeeting(Meeting meeting) throws BusinessException* →Creates a new meeting with the provided meeting details.

Validates the meeting input to ensure it meets required criteria.

Generates a unique meeting ID for the new meeting.

Throws a BusinessException if the meeting data is invalid or if there is an error during the creation process.

- *public Meeting updateMeeting(Meeting meeting)* throws BusinessException

Updates an existing meeting with new details.

Validates the meeting input to ensure it meets required criteria.

Throws a BusinessException if the meeting data is invalid or if there is an error during the update process.

- *public void deleteMeeting(String meetingId) throws BusinessException*

Deletes a meeting based on its meeting ID.

Throws a BusinessException if there is an error during the deletion process.

(Additional Business Methods)

The class may contain other business methods related to meeting management.

Helper Methods

- *private void validateMeeting(Meeting meeting) throws BusinessException*→Validates the input data for a meeting to ensure it meets necessary criteria (e.g., non-null fields, valid date, and time ranges).
- *private String generateUniqueMeetingId()*→Generates a unique meeting ID, typically using UUIDs. This method can be customized as needed for generating unique IDs.

Usage

The MeetingManager class is intended to be used within a larger business application to manage meetings. It provides a convenient interface for interacting with meetings, including creating, updating, and deleting them. It also handles data access exceptions and enforces meeting data validation.

MeetingRoom Class Documentation

Package: com.mycompany.buisness

Description

The MeetingRoom class represents a meeting room within a business environment. It contains detailed information about the meeting room, including its unique room ID, name, seating capacity, hourly cost, and various amenities and facilities available in the room.

Constructors

- *public MeetingRoom(String name, int seatingCapacity, boolean hasProjector, boolean hasWiFi, boolean hasConferenceCallFacility, boolean hasWhiteboard, boolean hasWaterDispenser, boolean hasTV, boolean hasCoffeeMachine)*

Constructor for creating a new meeting room with specified attributes.

Initializes the name, seating capacity, and availability of amenities.

- *public MeetingRoom(String roomId, String name, int seatingCapacity)*

Constructor for creating a meeting room with a unique room ID, name, and seating capacity.

Useful for situations where the room ID is pre-assigned or retrieved from a data source.

Attributes

- *private String roomId*→Represents the unique identifier for the meeting room.
- *private String name*→Represents the name or identifier of the meeting room.
- *private int seatingCapacity*→Indicates the maximum number of individuals that can be accommodated in the meeting room.
- *private int perHourCost*→Represents the hourly cost associated with booking the meeting room. (Getter and setter methods are available)
- *private boolean hasProjector*→Indicates whether the meeting room is equipped with a projector. (Getter and setter methods are available)
- *private boolean hasWiFi :* Indicates whether the meeting room has WiFi connectivity. (Getter and setter methods are available)
- *private boolean hasConferenceCallFacility*→Indicates whether the meeting room has conference call facilities. (Getter and setter methods are available)

- *private boolean hasWhiteboard*→Indicates whether the meeting room has a whiteboard. (Getter and setter methods are available)
- *private boolean hasWaterDispenser*→Indicates whether the meeting room is equipped with a water dispenser. (Getter and setter methods are available)
- *private boolean hasTV*→Indicates whether the meeting room has a television. (Getter and setter methods are available)
- *private boolean hasCoffeeMachine*→Indicates whether the meeting room is equipped with a coffee machine. (Getter and setter methods are available)

Methods

- *public String getRoomId()*→Returns the unique room ID of the meeting room.
- *public void setRoomId(String roomId)*->Sets the unique room ID of the meeting room.
- *public String getName()*→Returns the name or identifier of the meeting room.
- *public void setName(String name*)→Sets the name or identifier of the meeting room.
- *public int getSeatingCapacity()*→Returns the seating capacity of the meeting room.
- *public void setSeatingCapacity(int seatingCapacity)*→Sets the seating capacity of the meeting room.

(Additional getter and setter methods for other attributes)

There are getter and setter methods for the remaining attributes of the meeting room, including cost and amenities.

Usage

The MeetingRoom class is intended to represent individual meeting rooms within a business environment. Users can create instances of the MeetingRoom class to store and manage detailed information about meeting room attributes, making it useful for meeting room booking and management systems.

MeetingRoomManager Class Documentation

Package: com.mycompany.buisness

Description

The MeetingRoomManager class is responsible for managing meeting rooms within a business environment. It provides methods for retrieving, creating, updating, and deleting meeting rooms. This class encapsulates the business logic related to meeting room management and handles exceptions that may occur during operations.

Constructors

- *public MeetingRoomManager(MeetingRoomDao meetingRoomDao)*→Constructor that initializes the MeetingRoomManager with a MeetingRoomDao object, responsible for data access operations related to meeting rooms.
- *public MeetingRoomManager()*→Default constructor.

Methods

- *public MeetingRoom getMeetingRoomByName(String roomName) throws BusinessException*→Retrieves a meeting room by its name.

Throws a BusinessException if the meeting room is not found or if there is an error during data retrieval.

- *public List<MeetingRoom> getAllMeetingRooms() throws BusinessException*→Retrieves a list of all meeting rooms.

Throws a BusinessException if there is an error during data retrieval.

- *public MeetingRoom createMeetingRoom(MeetingRoom meetingRoom) throws BusinessException*→Creates a new meeting room with the provided attributes.

Validates the meeting room input to ensure it meets the required criteria.

Generates a unique room ID for the new meeting room.

Throws a BusinessException if the meeting room data is invalid or if there is an error during the creation process.

- *public MeetingRoom updateMeetingRoom(MeetingRoom meetingRoom) throws BusinessException*

Updates an existing meeting room with new details.

Validates the meeting room input to ensure it meets the required criteria.

Throws a BusinessException if the meeting room data is invalid or if there is an error during the update process.

- *public void deleteMeetingRoom(String roomId) throws BusinessException*

Deletes a meeting room based on its room ID.

Throws a BusinessException if there is an error during the deletion process.

(Additional Business Methods)

The class may contain other business methods related to meeting room management.

Helper Methods

- *private void validateMeetingRoom(MeetingRoom meetingRoom) throws BusinessException*→Validates the input data for a meeting room to ensure it meets necessary criteria (e.g., non-null fields, positive seating capacity).
- *private String generateUniqueRoomId()*→Generates a unique room ID, typically using UUIDs. This method can be customized as needed for generating unique IDs.

Usage

The MeetingRoomManager class is intended for use within a larger business application to manage meeting rooms. It provides a convenient interface for interacting with meeting rooms, including creating, updating, and deleting them. Additionally, it handles data access exceptions and enforces meeting room data validation.

User Class Documentation

Package: com.mycompany.buisness

Description

The User class represents a user entity within a business application. It encapsulates information about a user, including their unique user ID, name, email, phone number, available credits, and role within the application.

Constructors

- *public User()* →Default constructor with no parameters.
- *public User(String userId, String name, String email, String phone, int credits, String role)*→Constructor to create a User object with specified attributes.

Attributes

- *private String userId*→Represents the unique identifier associated with the user.
- *private String name*→Represents the full name of the user.
- *private String email*→Represents the email address of the user.
- *private String phone*→Represents the phone number of the user.
- *private int credits*→Indicates the number of credits available to the user.
- *private String role*→Represents the role or type of user within the application (e.g., "Admin," "Employee").

Methods

- *public String getUserId()* →Returns the user's unique user ID.
- *public void setUserId(String userId)*→Sets the user's unique user ID.
- *public String getName()* →Returns the full name of the user.
- *public void setName(String name)*→Sets the full name of the user.
- *public String getEmail()* →Returns the email address of the user.
- *public void setEmail(String email)* →Sets the email address of the user.
- *public String getPhone()* →Returns the phone number of the user.
- *public void setPhone(String phone)*→Sets the phone number of the user.
- *public int getCredits()* →Returns the number of credits available to the user.
- *public void setCredits(int credits)* →Sets the number of credits available to the user.

- *public String getRole()* →Returns the role or type of user within the application.
- *public void setRole(String role)* →Sets the role or type of user within the application.

Usage

The User class can be used to represent individual users within a business application. Users can create instances of the User class to store and manage user details, including their personal information, role, and available credits. This class serves as a fundamental entity for user management within the application.

UserManager Class Documentation

Package: com.mycompany.buisness

Description

The UserManager class is responsible for managing user entities within a business application. It provides methods for retrieving, creating, updating, and deleting users. This class encapsulates the business logic related to user management and handles exceptions that may occur during user-related operations.

Constructors

- *public UserManager(UserDao userDao*)→Constructor that initializes the UserManager with a UserDao object, responsible for data access operations related to users.
- *public UserManager()*→Default constructor.

Methods

- *public User getUserById(String userId) throws BusinessException*→Retrieves a user by their unique user ID.
- Throws a BusinessException if the user is not found or if there is an error during data retrieval.
- *public List<User> getAllUsers() throws BusinessException*→Retrieves a list of all users.
- Throws a BusinessException if there is an error during data retrieval.

- *public User createUser(User user) throws BusinessException*→Creates a new user with the provided attributes.

Validates the user input to ensure it meets the required criteria.

Generates a unique user ID for the new user.

Throws a BusinessException if the user data is invalid or if there is an error during the creation process.

- *public User updateUser(User user) throws BusinessException*→Updates an existing user with new details.

Validates the user input to ensure it meets the required criteria.

Throws a BusinessException if the user data is invalid or if there is an error during the update process.

- *public void deleteUser(String userId) throws BusinessException*→Deletes a user based on their user ID.

Throws a BusinessException if there is an error during the deletion process.

- *(Additional Business Methods*)→The class may contain other business methods related to user management.

Helper Methods

- *private void validateUser(User user) throws BusinessException*→Validates the input data for a user to ensure it meets necessary criteria (e.g., non-null fields for name, email, and phone).
- *private String generateUniqueUserId()* →Generates a unique user ID, typically using UUIDs. This method can be customized as needed for generating unique IDs.

Usage

The UserManager class is intended to be used within a larger business application to manage users. It provides a convenient interface for interacting with user data, including creating, updating, and deleting users. Additionally, it handles data access exceptions and enforces user data validation.