**Repositories**Package: com.backend.server.repository

Description

The DataTheaterRepository class is an interface that extends MongoRepository and provides data access methods to interact with the MongoDB database for the DataTheater entity. It enables retrieval and manipulation of DataTheater objects using MongoDB queries.

Methods

List<Movies> findByStatus(String status)

Description:

This method retrieves a list of Movies objects based on the provided status. It performs a query on the MongoDB database and returns a list of Movies objects that match the given status.

Parameters:

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

Returns:

(Type: List<Movies>) A list of Movies objects that match the given status. If no matches are found, an empty list is returned.

Note: The DataTheaterRepository interface implicitly inherits all the methods from the MongoRepository<DataTheater, Integer> interface, which includes various CRUD (Create, Read, Update, Delete) operations for the DataTheater entity.

Annotations

@Repository: Indicates that this interface is a Spring Data repository, allowing it to be automatically discovered and used as a bean in the Spring application context.

Package: com.backend.server.repository

Description

The MovieRepository interface is a Spring Data repository that extends MongoRepository and provides data access methods to interact with the MongoDB database for the Movies entity. It enables retrieval and manipulation of Movies objects using MongoDB queries.

Methods

List<Movies> findByStatus(String status)

Description:

This method retrieves a list of Movies objects based on the provided status. It performs a query on the MongoDB database and returns a list of Movies objects that match the given status.

Parameters:

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

Returns:

(Type: List<Movies>) A list of Movies objects that match the given status. If no matches are found, an empty list is returned.

Note: The MovieRepository interface implicitly inherits all the methods from the MongoRepository<Movies, Integer> interface, which includes various CRUD (Create, Read, Update, Delete) operations for the Movies entity.

Annotations

@Repository: Indicates that this interface is a Spring Data repository, allowing it to be automatically discovered and used as a bean in the Spring application context.

Package: com.backend.server.repository

Description

The ShowTimeRepository interface is a Spring Data repository that extends MongoRepository and provides data access methods to interact with the MongoDB database for the ShowTime entity. It enables retrieval and manipulation of ShowTime objects using MongoDB queries.

Methods

The ShowTimeRepository interface inherits various CRUD (Create, Read, Update, Delete) operations from the MongoRepository<ShowTime, Integer> interface. These methods include:

save(S entity): Saves a given ShowTime object in the database.

findById(ID id): Retrieves a ShowTime object by its unique ID.

findAll(): Retrieves all ShowTime objects from the database.

deleteById(ID id): Deletes a ShowTime object by its unique ID.

...and other CRUD methods.

Note: The ShowTimeRepository interface does not contain any custom query method beyond the ones provided by MongoRepository. If you need to define custom queries, you can add them to this interface following the naming conventions of Spring Data.

Annotations

@Repository: Indicates that this interface is a Spring Data repository, allowing it to be automatically discovered and used as a bean in the Spring application context.

Package: com.backend.server.repository

Description

The TheaterRepository interface is a Spring Data repository that extends MongoRepository and provides data access methods to interact with the MongoDB database for the Movies entity. It enables retrieval and manipulation of Movies objects using MongoDB queries in the context of theaters.

Methods

List<Movies> findByStatus(String status)

Description:

This method retrieves a list of Movies objects based on the provided status. It performs a query on the MongoDB database and returns a list of Movies objects that match the given status.

Parameters:

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

Returns:

(Type: List<Movies>) A list of Movies objects that match the given status. If no matches are found, an empty list is returned.

Note: The TheaterRepository interface implicitly inherits all the methods from the MongoRepository<Movies, Integer> interface, which includes various CRUD (Create, Read, Update, Delete) operations for the Movies entity.

Annotations

@Repository: Indicates that this interface is a Spring Data repository, allowing it to be automatically discovered and used as a bean in the Spring application context.

Package: com.backend.server.repository

Description

The TicketsRepository interface is a Spring Data repository that extends MongoRepository and provides data access methods to interact with the MongoDB database for the Movies entity. It enables retrieval and manipulation of Movies objects using MongoDB queries in the context of tickets.

Methods

List<Movies> findByStatus(String status)

Description:

This method retrieves a list of Movies objects based on the provided status. It performs a query on the MongoDB database and returns a list of Movies objects that match the given status.

Parameters:

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

Returns:

(Type: List<Movies>) A list of Movies objects that match the given status. If no matches are found, an empty list is returned.

Note: The TicketsRepository interface implicitly inherits all the methods from the MongoRepository<Movies, Integer> interface, which includes various CRUD (Create, Read, Update, Delete) operations for the Movies entity.

Annotations

@Repository: Indicates that this interface is a Spring Data repository, allowing it to be automatically discovered and used as a bean in the Spring application context.

Package: com.backend.server.repository

Description

The TicketTableRepository interface is a Spring Data repository that extends MongoRepository and provides data access methods to interact with the MongoDB database for the Movies entity. It enables retrieval and manipulation of Movies objects using MongoDB queries in the context of ticket tables.

Methods

List<Movies> findByStatus(String status)

Description:

This method retrieves a list of Movies objects based on the provided status. It performs a query on the MongoDB database and returns a list of Movies objects that match the given status.

Parameters:

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

Returns:

(Type: List<Movies>) A list of Movies objects that match the given status. If no matches are found, an empty list is returned.

Note: The TicketTableRepository interface implicitly inherits all the methods from the MongoRepository<Movies, Integer> interface, which includes various CRUD (Create, Read, Update, Delete) operations for the Movies entity.

Annotations

@Repository: Indicates that this interface is a Spring Data repository, allowing it to be automatically discovered and used as a bean in the Spring application context.

Package: com.backend.server.repository

Description

The UserRepository interface is a Spring Data repository that extends MongoRepository and provides data access methods to interact with the MongoDB database for the User entity. It enables retrieval and manipulation of User objects using MongoDB queries.

Methods

Note: The UserRepository interface inherits various CRUD (Create, Read, Update, Delete) operations from the MongoRepository<User, String> interface. These methods include:

save(S entity): Saves a given User object in the database.

findById(ID id): Retrieves a User object by its unique ID.

findAll(): Retrieves all User objects from the database.

deleteById(ID id): Deletes a User object by its unique ID.

...and other CRUD methods.

Annotations

@Repository: Indicates that this interface is a Spring Data repository, allowing it to be automatically discovered and used as a bean in the Spring application context.