Documentation

Database

The Databases-package is the same as in the system used in the labs.

CreateSchema.java

Contains helpers for creating the database schema. Each time the server starts the @{@link #createSchemaIfNotExists()} method is called.
@author Rasmus Ros, rasmus.ros@cs.lth.se

DataAccess.java

Base class for H2 database connections. Contains helper methods for common JDBC use cases. Also contains a method for setting up the database schema. To use this class, extend this class with a target data type parameter and provide a mapper for said type.
@author Rasmus Ros, rasmus.ros@cs.lth.se
@see Mapper

DataAccessException.java

Wraps SQLException and adds types @{@link ErrorType} to handled exceptions.
@author Rasmus Ros, rasmus.ros@cs.lth.se

ErrorType.java

This is used to communicate to client about usage errors without exposing underlying implementation details.
@author Rasmus Ros, rasmus.ros@cs.lth.se

Mapper.java

Helper to make working with JDBC's checked exception easier.
@param <T> Java type of mapped object.
@author Rasmus Ros, rasmus.ros@cs.lth.se

MapMapper.java

This is a convenience class which is useful for debugging SQL queries without having to write a fixed Mapper.

Ranker

Ranker.java

Used to rank rides by different criterias

rank(List<Ride> rides): Ranks the rides based on the best match in ascending order. REturns the ranked list.

List<Ride> rankByTime(List<Ride> rides): Ranks the list by departure time in ascending order. Returns the ranked list.

List<Ride> rankByDistance(List<Ride> rides): Ranks the list by their traveling distance. Returns the ranked list.

Provider

JsonExceptionMapper.java

This converts all Exceptions to HTTP responses for the REST API. It has special handling for WebApplicationException and DataAccessException.

Response toResponse(Exception exception): Converts the exceptions to http response and returns it as a Response.

AuthenticationFilter.java

Adds the Session to the current request. This is done by extracting the token in the users cookie and checking the database for the cookie.

void filter(ContainerRequestContext requestContext): see class description.

JsonProvider.java

This class converts all objects in the REST API to/from JSON using Gson.

boolean isReadable(Class<?> aClass, Type type, Annotation[] annotations, MediaType mediaType): Returns true;

public Object readFrom(Class<Object> aClass, Type type, Annotation[] annotations, MediaType mediaType, MultivaluedMap<String, String> multivaluedMap, InputStream entityStream): Reads from the Json from entityStream.

boolean isWriteable(Class<?> aClass, Type type, Annotation[] annotations, MediaType mediaType): Returns true;

void writeTo(Object o, Class<?> aClass, Type type, Annotation[] annotations, MediaType mediaType, MultivaluedMap<String, Object> multivaluedMap, OutputStream entityStream): Writes to object using entityStream.

Data

Session.java

A session is a user that has logged in. The sessionId is used to identify the user in the database.

Session(UUID sessionId, User user): sets the attributes of sessionId and user.

User getUser(): returns user

UUID getSessionId(): returns sessionId.

Principal getUserPrincipal(): returns user.

boolean isUserInRole(String role): compares role clearance of user of session to check for clearance.

boolean isSecure(): return false.

String getAuthenticationScheme(): returns authentication scheme.

Role.java

Represents the capabilities of users in the system. A user can have only one role, however there is a hierarchy in the roles, such that an admin can perform all user activities.

class Names: defines user and admins as strings.

Role(int level): sets level to level.

public int getLevel(): returns level as int.

public boolean clearanceFor(Role other): checks if security level is high enough.

User.java

Simply represents a user.

User(int id, String name, Role role, String email): sets attributes of class.

public int getId(): returns user ID.

public Role getRole(): returns user's role.

public String getName(): returns user's name.

public String getEmail(): returns user's email.

Location.java

Class to handle longitude- and latitude coordinates.

Location(float[2] coordinates, String name): sets coordinate attributes and name of the location.

Ride.java

Handles rides.

Ride(int id, Location departureLocation, Location arrivalLocation, Date departureTime, Date arrivalTime): sets relevant attributes of class.

public int getId(): returns id of the ride.

public Location getDepartureLocation(): returns departure location as an object of type Location.

public Location getArrivalLocation(): returns arrival location.

public Date getDepartureTime(): returns departure time as an object of type Date.

public Date getArrivalTime(): returns arrival time.

public int getCarSize(): returns number of seats in car.

public List<User> getTravelerList(): returns all travelers in ride.

public boolean addTraveler(User traveler): adds a traveler to the ride. Returns true if operation succeded, false if not.

public boolean removeTraveler(User traveler): removes traveler from ride. Returns true if operation succeeded, false if not.

Credentials.java

Handles user credentials, such as username, role etc.

Credentials(String username, String password, Role role): sets relevant attributes of object.

public String getUsername(): returns username.

public Role getRole(): returns role.

public boolean validPassword(): returns true if password is valid, false if not.

public static void main(String[] args): self-explanatory.

Date.java

Handles dates.

Date(String date, String time): sets attributes for date and time.
public String getDate(): returns formatted date.

public String getTime(): returns formatted time.

UserDataAccess.java

Handles operations regarding users.

UserDataAccess(String str): sets corresponding attribute in object.

public User addUser(Credentials credentials): adds user with given credentials, then returns it. Throws DataAccessException if operation failed.

public User updateUser(int userId, Credentials credentials): updates user with given ID, bases on the given credentials. Returns updated user.

public User getUser(int id): returns user with given ID.

public boolean deleteUser(int id): deletes user with given ID. Returns true if operation succeeded, false if not.

public List<User> getUsers(): returns all users.

public Session getSession(UUID id): returns session with given UUID.

public boolean removeSession(UUID id): removes session with given UUID.

public Session authenticate(Credentials credentials): authenticates credentials, returns new session based on given credentials if success. Throws DataAccessException if credentials are invalid.

public String getEmail(int userId): returns e-mail of user with given ID.

Rest

A link between the Front-End and the Back-End database.

UserResources.java

public UserResource(ContainerRequestContext containerRequestContext): Sets relevant attributes of object.

public currentUser(): returns the currently logged in user.

public login(Credentials credentials, boolean rememberMe): Tries to login the user. Returns the response from the back-end.

public logout(): Tries to log out the user. Returns the response from the back-end.

public getRoles(): Returns all the roles of the user.

public createUser(Credentials credentials): Creates a new user in the database with the given credentials. Then returns the newly created user.

public getUser(int id): Returns the user with the given id.

public putUser(int id, Credentials credentials): Updates the credentials of the user with the given id, to the given credentials. Then returns the updated user.

public deleteUser(int id): removes a user from the database.

LocationResources.java

public LocationResources(ContainerRequestContext): Sets relevant attributes of object.

public getLocation(String): Returns the Location class object of the given location.

public getAllLocations(): Returns all the locations in the system.

RideResources.java

public RideResources(ContainerRequestContext): Sets relevant attributes of object.

public postRide(int ???): Adds a ride to the database and then returns the created Ride.

public deleteRide(int id): Deletes a ride with the given id.

public getAllRides(): Returns all the rides in the server.

public searchReleventRides(int userId???, String location???): Returns the relevant rides for a given user.

public postRide(type ???): Adds a ride to the database and then returns the created Ride.

public addUserToRide(User user): Adds the given user to the private ride attribute.

public removeUserFromRide(type): Removes a user from the private ride attribute.