package net.javaguides.usermanagement.dao;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import net.javaguides.usermanagement.model.User;

/\*\*

\* AbstractDAO.java This DAO class provides CRUD database operations for the

\* table users in the database.

\*

\* @author Ramesh Fadatare

\*

\*/

public class UserDAO {

private String jdbcURL = "jdbc:mysql://localhost:3306/demo?useSSL=false";

private String jdbcUsername = "root";

private String jdbcPassword = "root";

private static final String INSERT\_USERS\_SQL = "INSERT INTO users" + " (name, email, country) VALUES " +

" (?, ?, ?);";

private static final String SELECT\_USER\_BY\_ID = "select id,name,email,country from users where id =?";

private static final String SELECT\_ALL\_USERS = "select \* from users";

private static final String DELETE\_USERS\_SQL = "delete from users where id = ?;";

private static final String UPDATE\_USERS\_SQL = "update users set name = ?,email= ?, country =? where id = ?;";

public UserDAO() {}

protected Connection getConnection() {

Connection connection = null;

try {

Class.forName("com.mysql.jdbc.Driver");

connection = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return connection;

}

public void insertUser(User user) throws SQLException {

System.out.println(INSERT\_USERS\_SQL);

// try-with-resource statement will auto close the connection.

try (Connection connection = getConnection(); PreparedStatement preparedStatement = connection.prepareStatement(INSERT\_USERS\_SQL)) {

preparedStatement.setString(1, user.getName());

preparedStatement.setString(2, user.getEmail());

preparedStatement.setString(3, user.getCountry());

System.out.println(preparedStatement);

preparedStatement.executeUpdate();

} catch (SQLException e) {

printSQLException(e);

}

}

public User selectUser(int id) {

User user = null;

// Step 1: Establishing a Connection

try (Connection connection = getConnection();

// Step 2:Create a statement using connection object

PreparedStatement preparedStatement = connection.prepareStatement(SELECT\_USER\_BY\_ID);) {

preparedStatement.setInt(1, id);

System.out.println(preparedStatement);

// Step 3: Execute the query or update query

ResultSet rs = preparedStatement.executeQuery();

// Step 4: Process the ResultSet object.

while (rs.next()) {

String name = rs.getString("name");

String email = rs.getString("email");

String country = rs.getString("country");

user = new User(id, name, email, country);

}

} catch (SQLException e) {

printSQLException(e);

}

return user;

}

public List < User > selectAllUsers() {

// using try-with-resources to avoid closing resources (boiler plate code)

List < User > users = new ArrayList < > ();

// Step 1: Establishing a Connection

try (Connection connection = getConnection();

// Step 2:Create a statement using connection object

PreparedStatement preparedStatement = connection.prepareStatement(SELECT\_ALL\_USERS);) {

System.out.println(preparedStatement);

// Step 3: Execute the query or update query

ResultSet rs = preparedStatement.executeQuery();

// Step 4: Process the ResultSet object.

while (rs.next()) {

int id = rs.getInt("id");

String name = rs.getString("name");

String email = rs.getString("email");

String country = rs.getString("country");

users.add(new User(id, name, email, country));

}

} catch (SQLException e) {

printSQLException(e);

}

return users;

}

public boolean deleteUser(int id) throws SQLException {

boolean rowDeleted;

try (Connection connection = getConnection(); PreparedStatement statement = connection.prepareStatement(DELETE\_USERS\_SQL);) {

statement.setInt(1, id);

rowDeleted = statement.executeUpdate() > 0;

}

return rowDeleted;

}

public boolean updateUser(User user) throws SQLException {

boolean rowUpdated;

try (Connection connection = getConnection(); PreparedStatement statement = connection.prepareStatement(UPDATE\_USERS\_SQL);) {

statement.setString(1, user.getName());

statement.setString(2, user.getEmail());

statement.setString(3, user.getCountry());

statement.setInt(4, user.getId());

rowUpdated = statement.executeUpdate() > 0;

}

return rowUpdated;

}

private void printSQLException(SQLException ex) {

for (Throwable e: ex) {

if (e instanceof SQLException) {

e.printStackTrace(System.err);

System.err.println("SQLState: " + ((SQLException) e).getSQLState());

System.err.println("Error Code: " + ((SQLException) e).getErrorCode());

System.err.println("Message: " + e.getMessage());

Throwable t = ex.getCause();

while (t != null) {

System.out.println("Cause: " + t);

t = t.getCause();

}

}

}

}

}