Exercise 6\_1 Crud operations

Steps and screenshots

991667681

JAVA Enterprise

Balwinder Kaur

Some few same steps in every exercise and summary at the end  
  
A screenshot of a computer

Description automatically generated  
  
  
A screenshot of a computer

Description automatically generated

Declaring the local variables as mentioned in the exercise and adding Lombok annotations to it.  
A screenshot of a computer

Description automatically generated  
  
  
  
A screenshot of a computer

Description automatically generated

Creating appointment table with id as primary key and auto incremented  
A screenshot of a computer

Description automatically generated with medium confidence  
  
Data.sql file with some examples  
A screenshot of a computer

Description automatically generated with medium confidence  
  
  
  
A screenshot of a computer

Description automatically generated

in this class everything is same as previous but adding new methods delte and edit that will be interlinked with database class and index.html explained in summary  
A screenshot of a computer

Description automatically generated with medium confidence  
  
A screenshot of a computer code

Description automatically generated with low confidence

same adding auto dependency jdbc and than adding method edit and delete with printing the lined deleted and updated on console just to verify if the data is inserted and printed or not  
A screenshot of a computer

Description automatically generated with medium confidence  
  
  
  
A screenshot of a computer

Description automatically generated  
  
  
  
A screenshot of a computer

Description automatically generated with medium confidence  
  
  
  
  
  
A screenshot of a computer

Description automatically generated  
  
  
  
  
A screenshot of a computer

Description automatically generated  
  
  
  
  
package ca.sheridancollege.kohliman;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Ex61FullCrudApplication {

public static void main(String[] args) {

SpringApplication.run(Ex61FullCrudApplication.class, args);

}

}  
  
  
package ca.sheridancollege.kohliman.beans;

import java.time.LocalDate;

import java.time.LocalTime;

import org.springframework.format.annotation.DateTimeFormat;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

public class Appointment {

private Long id;

private String firstName;

private String email;

@DateTimeFormat(pattern = "yyyy-MM-dd")

private LocalDate appointmentDate;

@DateTimeFormat(pattern = "HH:mm")

private LocalTime appointmentTime;

}  
  
  
package ca.sheridancollege.kohliman.controllers;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import ca.sheridancollege.kohliman.beans.Appointment;

import ca.sheridancollege.kohliman.database.DatabaseAccess;

@Controller

public class AppointmentController {

@Autowired

private DatabaseAccess da;

public void setDa(DatabaseAccess da) {

this.da = da;

}

@GetMapping("/")

public String index(Model model) {

model.addAttribute("appointment", new Appointment());

model.addAttribute("studentList", da.getStudentList());

return "index";

}

@PostMapping("/insertappointment")

public String insertappointment(Model model, @ModelAttribute("appointment") Appointment app) {

da.insertappointment(app);

model.addAttribute("appointment", new Appointment());

model.addAttribute("studentList", da.getStudentList());

return "index";

}

@GetMapping("/deleteappById/{id}")

public String deleteStudentById(Model model, @PathVariable Long id) {

da.deleteappbyid(id);

model.addAttribute("appointment", new Appointment());

model.addAttribute("studentList", da.getappListById(id));

return "index";

}

@GetMapping("/editappById/{id}")

public String editappById(Model model, @PathVariable Long id) {

Appointment app = da.getappListById(id).get(0);

model.addAttribute("appointment", app);

da.deleteappbyid(id);

model.addAttribute("studentList", da.getStudentList());

// da.editappbyid(id);

// model.addAttribute("studentList", da.getappListById(id));

return "index";

}

}  
  
  
package ca.sheridancollege.kohliman.database;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.BeanPropertyRowMapper;

import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;

import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;

import org.springframework.stereotype.Repository;

import ca.sheridancollege.kohliman.beans.Appointment;

@Repository

public class DatabaseAccess {

@Autowired

protected NamedParameterJdbcTemplate jdbc;

public void insertappointment(Appointment app) {

MapSqlParameterSource namedParameters = new MapSqlParameterSource();

String query = "INSERT INTO appointment(firstName,email,appointmentDate,appointmentTime) VALUES(:firstName, :email, :appointmentDate, :appointmentTime)";

namedParameters.addValue("firstName", app.getFirstName());

namedParameters.addValue("email", app.getEmail());

namedParameters.addValue("appointmentDate", app.getAppointmentDate());

namedParameters.addValue("appointmentTime", app.getAppointmentTime());

int affectedrows = jdbc.update(query, namedParameters);

if (affectedrows > 0) {

System.out.println("Inserted student into the database.");

}

}

public void deleteappbyid(Long id) {

MapSqlParameterSource namedParameters =new MapSqlParameterSource();

String query ="DELETE FROM appointment where id=:id";

namedParameters.addValue("id", id);

if (jdbc.update(query, namedParameters) > 0)

System.out.println("Deleted appointment " + id + " from database");

}

public void editappbyid(Long id) {

MapSqlParameterSource namedParameters = new MapSqlParameterSource();

String query = "UPDATE appointment SET id = :id WHERE id = :id";

namedParameters.addValue("id", id);

if (jdbc.update(query, namedParameters) > 0) {

System.out.println("Updated appointment " + id + " in the database");

}

}

public List<Appointment> getStudentList()

{ MapSqlParameterSource namedParameters =new MapSqlParameterSource();

String query = "SELECT \* FROM appointment";

return jdbc.query(query, namedParameters, new BeanPropertyRowMapper<Appointment>(Appointment.class));

}

public List<Appointment> getappListById(Long id) {

MapSqlParameterSource namedParameters = new

MapSqlParameterSource(); String query = "SELECT \* FROM appointment WHERE id = :id"; namedParameters.addValue("id", id);

return jdbc.query(query, namedParameters, new

BeanPropertyRowMapper<Appointment>(Appointment.class));

}

}  
  
  
Summary

Summary  
Everything is same as explained in the previous exercise but the different here is that we added a new method in the Controller and as well as in Database class method that is the delete appointment by id and edit appintment by id that is get mapping from the index page and calling the method from the controller class and in that method we are calling the method from the database class that is deleting the student the string query and adding that in the namedparamater via id and if row affected that will print onto the console and than we refreshing the appointment list and calling the student list again via adding the attribute to the model class , calling all the student list from the database class and at the end refreshing the index page via returning it as a string.