МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
ІМЕНІ ВАСИЛЯ СТЕФАНИКА

Кафедра комп’ютерних наук та інформаційних систем

**Лабораторна робота №2**

з курсу «Платформи корпоративних інформаційних систем»

на тему:

«Використання СУБД (SQL) у Spring Boot - проекті.»

**Виконав:**

студент групи КН-41

Сенів П.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**Перевірив:**

викладач кафедри КНІС

к.т.н., доцент Горєлов В.О. «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оцінка, підпис)

м. Івано-Франківськ

2021

Мета: сформувати теоретичний базис щодо використання SQL СУБД у Spring Boot-проекті. Набути практичних навичок написання класів контролера, сервісу, репозиторію з підтримкою SQL.

Завдання: розробити проект з репозиторієм із підтримкою СУБД для реалізації каталогу домашньої бібліотеки. Робота продовжує тематику лабораторної роботи №1.

Хід роботи

1. Напишіть клас, що описує видання у каталозі домашньої бібліотеки. Поля класу:

а) id;

б) назва книги;

в) автор;

г) рік видання;

д) видавництво;

е) кількість сторінок.

Типи полів задайте самостійно.

2. Розробіть інтерфейс, котрий міститиме необхідні операції над даними. Перелік операцій:

а) додавання;

б) перегляд за id;

в) перегляд повного переліку;

г) видалення за id;

д) видалення повного переліку;

у) модифікація за id;

е) перегляд за автором;

є) перегляд за видавництвом.

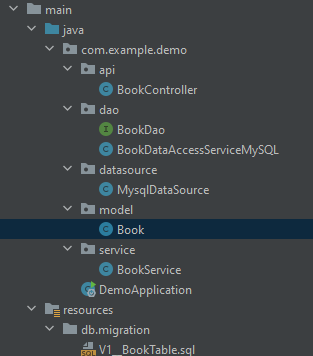
3. Розробіть репозиторій на основі інтерфейсу (п. 2). Реалізуйте його методи. Репозиторій повинен використовувати сховище на основі СУБД. 4. Побудуйте сервіс, що використовує отриманий у п. 3 репозиторій (або скористайтеся розробленим у лабораторній роботі №1).

5. Розробіть контролер для обробки запитів користувача (або скористайтеся розробленим у лабораторній роботі №1).

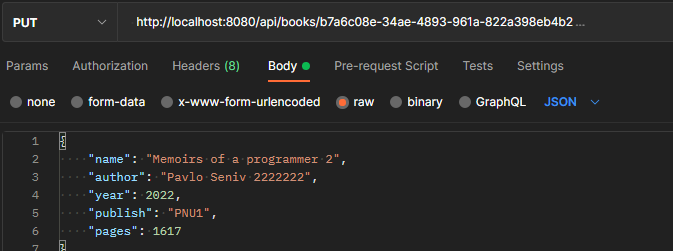
6. Продемонструйте розроблений проект у дії. Скористайтеся для надсилання запитів до сервера API-клієнтом (Postman, Insomnia тощо). Формат тіла HTTP-запитів та відповідей сервера — JSON.

7. Оформте звіт.

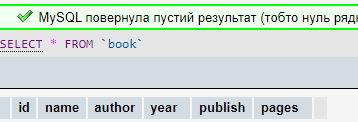
**Результати виконання роботи**



**Рисунок 1. Структура проекту.**



**Рисунок 2. Запити та приклад запиту.**



**Рисунок 3. Таблиця book.**

**Файл BookController.java**

package com.example.demo.api;  
  
import com.example.demo.model.Book;  
import com.example.demo.service.BookService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.lang.NonNull;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.validation.Valid;  
import javax.websocket.server.PathParam;  
import java.util.List;  
import java.util.UUID;  
  
@RequestMapping("api/books")  
@RestController  
public class BookController {  
 private final BookService bookService;  
  
 @Autowired  
 public BookController(BookService bookService) {  
 this.bookService = bookService;  
 }  
  
 @PostMapping  
 public void createBook(@Valid @NonNull @RequestBody Book book) {  
 bookService.createBook(book);  
 }  
  
 @GetMapping  
 public List<Book> getAllBook() {  
 return bookService.getAllBook();  
 }  
  
 @GetMapping(path = "/{id}")  
 public Book getBookById(@PathVariable("id") UUID id) {  
 return bookService.getBookById(id)  
 .orElse(null);  
 }  
  
 @GetMapping(path = "/author")  
 public Book getBookByAuthor(@PathParam("author") String author) {  
 return bookService.getBookByAuthor(author)  
 .orElse(null);  
 }  
  
 @GetMapping(path = "/publish")  
 public Book getByPublish(@PathParam("publish") String publish) {  
 return bookService.getBookByPublish(publish)  
 .orElse(null);  
 }  
  
 @DeleteMapping(path = "/{id}")  
 public void deleteBookById(@PathVariable("id") UUID id) {  
 bookService.deleteBook(id);  
 }  
  
 @PutMapping(path = "/{id}")  
 public void updateBook(@PathVariable("id") UUID id, @Valid @NonNull @RequestBody Book book) {  
 bookService.updateBook(id, book);  
 }  
 @DeleteMapping  
 public void deleteAll() {  
 bookService.deleteAll();  
 }  
  
}

**Файл BookService.java**

package com.example.demo.service;  
  
import com.example.demo.dao.BookDao;  
import com.example.demo.model.Book;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Qualifier;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
import java.util.UUID;  
  
@Service  
public class BookService {  
 private final BookDao bookDao;  
  
 @Autowired  
 public BookService(@Qualifier("mysqlDao") BookDao bookDao) {  
 this.bookDao = bookDao;  
 }  
  
 public int createBook(Book book) {  
 return bookDao.createBook(book);  
 }  
  
 public List<Book> getAllBook() {  
 return this.bookDao.selectAllBook();  
 }  
  
 public Optional<Book> getBookById(UUID id) {  
 return bookDao.selectBookById(id);  
 }  
  
 public Optional<Book> getBookByAuthor(String author) {  
 return bookDao.selectBookByAuthor(author);  
 }  
  
 public Optional<Book> getBookByPublish(String publish) {  
 return bookDao.selectBookByPublish(publish);  
 }  
  
 public int deleteBook(UUID id) {  
 return bookDao.deleteBookById(id);  
 }  
  
 public int updateBook(UUID id, Book newBook) {  
 return bookDao.updateBookById(id, newBook);  
 }  
  
 public int deleteAll() {  
 return bookDao.deleteAll();  
 }  
}

**Файл Book.java**

package com.example.demo.model;  
  
import com.fasterxml.jackson.annotation.JsonProperty;  
  
import javax.validation.constraints.NotBlank;  
import java.util.UUID;  
  
public class Book {  
 private final UUID id;  
 private final String name;  
 private final String author;  
 private final Integer year;  
 private final String publish;  
 private final Integer pages;  
  
 public Book(@JsonProperty("id") UUID id,  
 @JsonProperty("name") String name,  
 @JsonProperty("author") String author,  
 @JsonProperty("year") Integer year,  
 @JsonProperty("publish") String publish,  
 @JsonProperty("pages") Integer pages) {  
  
 this.id = id;  
 this.name = name;  
 this.author = author;  
 this.year = year;  
 this.publish = publish;  
 this.pages = pages;  
 }  
  
 public UUID getId() {  
 return id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getAuthor() {  
 return author;  
 }  
  
 public Integer getYear() {  
 return year;  
 }  
  
 public String getPublish() {  
 return publish;  
 }  
  
 public Integer getPages() {  
 return pages;  
 }  
}

**Висновок:** на лабораторній роботі було розроблено проект з репозиторієм із підтримкою СУБД для реалізації каталогу домашньої бібліотеки. Робота продовжує тематику лабораторної роботи №1. Я закріпив навички користування API-клієнтом Postman. Надсилав запити до сервера API-клієнтом Postman. Також із зміною даних за допомогою запитів, змінювалися дані в таблиці book у СУБД.