МИНИСТЕРСТВО ТРАНСПОРТА РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

Кафедра ЦТУТП

**Отчёт**

По лабораторной работе №7  
по дисциплине «Корпоративные информационные системы»

Тема: «Разработка REST-контроллера»

Вариант №13

Выполнил: Веселков М.Ю.

Группа: УИС-411

Преподаватель: доц. Кафедры ЦТУТП

Козьяков П. О.

­

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc180325218)

[ХОД ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc180325219)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 5](#_Toc180325220)

[КОД ПРОГРАММЫ 6](#_Toc180325221)

[ВЫВОД 10](#_Toc180325222)

# **ПОСТАНОВКА ЗАДАЧИ**

Для данных по магазинам, выборка из БД которых реализована в лабораторной работе №6, реализовать контроллер на все операции Добавления, Удаления, Изменения и Выборки данных. Для документации и проверки запросов использовать Swagger.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

Лабораторная работа началась с подготовки базы данных. В PostgreSQL была создана таблица shops, включающая поля id, title и adress. Поле id настроено как автоинкрементное и является первичным ключом. Эти действия обеспечили базовую инфраструктуру для хранения данных о магазинах.

Далее в проекте Spring Boot была выполнена настройка подключения к базе данных. Для этого использовался файл application.properties, где указывались URL базы данных, имя пользователя и пароль. Это позволило приложению связаться с PostgreSQL и использовать созданную таблицу.

На следующем этапе был разработан слой модели с использованием аннотаций Lombok для упрощения создания геттеров, сеттеров и конструктора. Затем был реализован слой маппинга с помощью MyBatis, включающий аннотации и XML-конфигурацию для операций вставки, обновления, удаления и выборки данных. Это позволило связать бизнес-логику приложения с базой данных.

Слой сервиса был разработан для управления основными операциями над объектами Shop. Реализованы методы для добавления нового магазина, обновления данных, получения информации о магазине по ID и его удаления. Эти методы использовались в контроллере для обеспечения API взаимодействия.7

Завершающим этапом стала разработка REST-контроллера. Были реализованы эндпоинты для выполнения CRUD-операций над объектами Shop. Для тестирования использовались инструменты Swagger UI, что позволило проверить корректность работы API и базы данных.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package rut.miit.lab6mybatisdb.model;

import com.fasterxml.jackson.annotation.JsonProperty;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

public class Shop {

private long id;

@JsonProperty("title")

private String title;

@JsonProperty("adress")

private String adress;

@JsonProperty("phone")

private String phone;

}

package rut.miit.lab6mybatisdb.mapper;

import org.apache.ibatis.annotations.\*;

import rut.miit.lab6mybatisdb.model.Shop;

import java.util.List;

@Mapper

public interface ShopMapper {

@Insert("INSERT INTO shops (title, adress, phone) VALUES (#{title}, #{adress}, #{phone})")

void insertShop(Shop shop) ;

@Update("UPDATE shops SET title = #{title}, adress=#{adress}, phone=#{phone} WHERE id = #{id}")

void updateShop(Shop shop) ;

@Select("SELECT \* FROM lab6.shops")

@Results({

@Result(property = "title", column = "title"),

@Result(property = "adress", column = "adress"),

@Result(property = "phone", column = "phone")

})

List<Shop> selectShop();

@Delete("DELETE FROM shops WHERE id= #{id}")

void deleteShop(Long id) ;

}

package rut.miit.lab6mybatisdb.mapper;

import org.apache.ibatis.annotations.Mapper;

import org.springframework.stereotype.Repository;

import rut.miit.lab6mybatisdb.dto.ShopRequest;

import rut.miit.lab6mybatisdb.model.Shop;

import java.util.List;

@Mapper

public interface ShopDao {

int deleteByPrimaryKey(Long id);

int insert(Shop record);

int insertSelective(Shop record);

Shop selectByPrimaryKey(Long id);

int updateByPrimaryKeySelective(Shop record);

int updateByPrimaryKey(Shop record);

List<Shop> selectAll();

}

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="rut.miit.lab6mybatisdb.mapper.ShopDao">

<resultMap id="ShopResultMap" type="rut.miit.lab6mybatisdb.model.Shop">

<id column="id" jdbcType="BIGINT" property="id" />

<result column="title" jdbcType="VARCHAR" property="title" />

<result column="adress" jdbcType="VARCHAR" property="adress" />

<result column="phone" jdbcType="VARCHAR" property="phone" />

</resultMap>

<sql id="Shop\_Column\_List">

id, title, adress, phone

</sql>

<select id="selectByPrimaryKey" parameterType="java.lang.Long" resultMap="ShopResultMap">

select

<include refid="Shop\_Column\_List" />

from lab6.shops <!-- таблица называется shops -->

where id = #{id,jdbcType=BIGINT}

</select>

<insert id="insert" parameterType="rut.miit.lab6mybatisdb.model.Shop">

insert into lab6.shops (title, adress, phone) <!-- таблица называется shops -->

values (#{title,jdbcType=VARCHAR}, #{adress,jdbcType=VARCHAR}, #{phone,jdbcType=VARCHAR})

</insert>

<update id="updateByPrimaryKey" parameterType="rut.miit.lab6mybatisdb.model.Shop">

update lab6.shops <!-- таблица называется shops -->

set title = #{title,jdbcType=VARCHAR},

adress = #{adress,jdbcType=VARCHAR},

phone = #{phone,jdbcType=VARCHAR}

where id = #{id,jdbcType=BIGINT}

</update>

<delete id="deleteByPrimaryKey" parameterType="java.lang.Long">

delete from lab6.shops where id = #{id,jdbcType=BIGINT} <!-- таблица называется shops -->

</delete>

<select id="selectAll" resultMap="ShopResultMap">

select <include refid="Shop\_Column\_List" /> from lab6.shops

</select>

</mapper>

package rut.miit.lab6mybatisdb.services;

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

import org.springframework.stereotype.Service;

import rut.miit.lab6mybatisdb.dto.ShopRequest;

import rut.miit.lab6mybatisdb.mapper.ShopDao;

import rut.miit.lab6mybatisdb.model.Shop;

import java.util.List;

@Service

public class ShopService {

@Autowired

private ShopDao shopDao;

public Shop getShopById(Long id) {

return shopDao.selectByPrimaryKey(id);

}

public int addShop(Shop shop) {

return shopDao.insert(shop);

}

public int updateShop(Shop shop) {

return shopDao.updateByPrimaryKey(shop);

}

public int deleteShop(Long id) {

return shopDao.deleteByPrimaryKey(id);

}

public List<Shop> getAllShops() {

return shopDao.selectAll();

}

}

package rut.miit.lab6mybatisdb.controllers;

import io.swagger.v3.oas.annotations.Operation;

import lombok.extern.slf4j.Slf4j;

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

import org.springframework.http.MediaType;

import org.springframework.web.bind.annotation.\*;

import rut.miit.lab6mybatisdb.dto.ResponseWrapper;

import rut.miit.lab6mybatisdb.dto.ShopRequest;

import rut.miit.lab6mybatisdb.model.Shop;

import rut.miit.lab6mybatisdb.services.ShopService;

import java.util.List;

@RestController

@RequestMapping("/shop/op")

@Slf4j

public class ShopController {

@Autowired

private ShopService shopService;

@Operation(description = "Версия")

@GetMapping(value = "/ver", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseWrapper<String> shopVersion() {

ResponseWrapper<String> response = new ResponseWrapper<>();

response.setSuccess(true);

response.setData("Version 1.0.0");

return response;

}

@Operation(description = "Получить информацию о магазине по ID")

@PostMapping(value = "/id", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseWrapper<Shop> shopById(@RequestBody ShopRequest request) {

ResponseWrapper<Shop> response = new ResponseWrapper<>();

response.setSuccess(true);

response.setData(shopService.getShopById(request.getShopId()));

return response;

}

@Operation(description = "Получить список всех магазинов")

@GetMapping(value = "/list", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseWrapper<List<Shop>> getAllShops() {

ResponseWrapper<List<Shop>> response = new ResponseWrapper<>();

response.setSuccess(true);

response.setData(shopService.getAllShops());

return response;

}

@Operation(description = "Добавить новый магазин")

@PostMapping(value = "/add", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseWrapper<Integer> addShop(@RequestBody Shop shop) {

ResponseWrapper<Integer> response = new ResponseWrapper<>();

response.setSuccess(true);

response.setData(shopService.addShop(shop));

return response;

}

@Operation(description = "Обновить данные магазина")

@PutMapping(value = "/update", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseWrapper<Integer> updateShop(@RequestBody Shop shop) {

ResponseWrapper<Integer> response = new ResponseWrapper<>();

response.setSuccess(true);

response.setData(shopService.updateShop(shop));

return response;

}

@Operation(description = "Удалить магазин по ID")

@DeleteMapping(value = "/delete/{id}", produces = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseWrapper<Integer> deleteShop(@PathVariable Long id) {

ResponseWrapper<Integer> response = new ResponseWrapper<>();

response.setSuccess(true);

response.setData(shopService.deleteShop(id));

return response;

}

}

spring.application.name=Lab6MybatisDb

server.port=8080

spring.datasource.url=jdbc:postgresql://localhost:5432/postgres

spring.datasource.username=postgres

spring.datasource.password=postgres

spring.datasource.driver-class-name=org.postgresql.Driver

logging.level.org.apache.ibatis=DEBUG

logging.level.rut.miit.lab6mybatisdb.mapper=DEBUG

mybatis.mapper-locations=classpath\*:src/main/resources/rut/miit/lab6mybatisdb/mapper/\*.xml

emd.version=1.0.0

# **ВЫВОД**

В результате выполнения лабораторной работы было создано приложение, которое успешно взаимодействует с базой данных PostgreSQL через MyBatis. Реализованные функции позволяют добавлять, обновлять, удалять и получать данные о магазинах. Приложение показало стабильную работу и корректное выполнение SQL-запросов, что подтверждается выводами в логах и результатами тестирования API.