**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Лабораторная работа № 3

*Spring*

**Выполнил студент группы M32051**

Писарева Юлия Игоревна

****

**Подпись:**

**Проверил:**

Чикишев Константин Максимович

Санкт-Петербург

2022

**Текст задания**

К созданному в прошлой лабораторной сервису добавляется Spring.

Сервис должен предоставлять http интерфейс (REST API) для получения информации о конкретных котиках и владельцах и для получения фильтрованной информации (например, получить всех рыжих котиков)

Внимание: недопустимо отдавать через HTTP интерфейс сущности JPA. Рекомендуется создать отдельные оберточные классы.

Сервисы и dao должны превратиться в Spring Bean’ы с использованием Dependency Injection (Autowired). Dao при этом наследуют JpaRepository и имеет шаблонные Spring Data JPA методы: https://www.baeldung.com/spring-data-repositories#repositories

При сдаче лабораторной нужно будет показать работоспособность endpoint’ов через http запросы (рекомендуется Postman).

**Решение с комментариями**

package com.kotiki.core.entities;

import com.kotiki.core.models.Color;

import javax.persistence.\*;

import java.io.Serializable;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

@Entity

@Table(name = "cats")

public class Cat implements Serializable {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@Column(name = "birth\_date")

private LocalDateTime birthDate;

private String breed;

private Color color;

@ManyToOne

@JoinColumn(name = "owner\_id")

private Owner owner;

@ManyToMany

@JoinTable(

name = "cat\_friends",

joinColumns = { @JoinColumn(name = "left\_id") },

inverseJoinColumns = { @JoinColumn(name = "right\_id") }

)

private List<Cat> friends;

public Cat(String name, LocalDateTime birthDate, String breed, Color color, Owner owner) {

this.name = name;

this.birthDate = birthDate;

this.breed = breed;

this.color = color;

this.owner = owner;

this.friends = new ArrayList<>();

}

public Cat(String name, LocalDateTime birthDate, String breed, Color color) {

this(name, birthDate, breed, color, null);

}

protected Cat() { }

public Long getId() { return id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public LocalDateTime getBirthDate() { return birthDate; }

public String getBreed() { return breed; }

public Color getColor() { return color; }

public void setColor(Color color) { this.color = color; }

public Owner getOwner() { return owner; }

public void setOwner(Owner owner) { this.owner = owner; }

public List<Cat> getFriends() { return Collections.unmodifiableList(friends); }

public void addFriend(Cat... cats) { Collections.addAll(friends, cats); }

public void removeFriend(Cat... cats) { friends.removeAll(Arrays.asList(cats)); }

}

package com.kotiki.core.entities;

import javax.persistence.\*;

import java.io.Serializable;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

@Entity

@Table(name = "owners")

public class Owner implements Serializable {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@Column(name = "birth\_date")

private LocalDateTime birthDate;

@OneToMany(mappedBy = "owner")

private List<Cat> cats;

public Owner(String name, LocalDateTime birthDate) {

this.name = name;

this.birthDate = birthDate;

this.cats = new ArrayList<>();

}

protected Owner() { }

public Long getId() { return id; }

public String getName() { return name; }

public LocalDateTime getBirthDate() { return birthDate; }

public List<Cat> getCats() { return Collections.unmodifiableList(cats); }

public void addCat(Cat... cats) { Collections.addAll(this.cats, cats); }

public void removeCat(Cat... cats) { this.cats.removeAll(Arrays.asList(cats)); }

}

package com.kotiki.core.models;

public enum Color {

WHITE,

BLACK,

ORANGE,

GRAY,

BLUE,

}

package com.kotiki.core.services;

import com.kotiki.core.entities.Cat;

import com.kotiki.core.models.Color;

public class CatService {

public void renameCat(Cat cat, String name) { cat.setName(name); }

public void friendCats(Cat left, Cat right) {

left.addFriend(right);

right.addFriend(left);

}

public void unfriendCats(Cat left, Cat right) {

left.removeFriend(right);

right.removeFriend(left);

}

public void paintCat(Cat cat, Color color) { cat.setColor(color); }

}

package com.kotiki.core.services;

import com.kotiki.core.entities.Cat;

import com.kotiki.core.entities.Owner;

public class OwnerService {

public void addCat(Owner owner, Cat cat) {

if (cat.getOwner() != null) {

removeCat(cat.getOwner(), cat);

}

owner.addCat(cat);

cat.setOwner(owner);

}

public void removeCat(Owner owner, Cat cat) {

owner.removeCat(cat);

cat.setOwner(null);

}

}

package com.kotiki.dataAccess.daos;

import com.kotiki.core.entities.Cat;

import org.springframework.data.jpa.repository.JpaRepository;

public interface CatDao extends JpaRepository<Cat, Long> {

}

package com.kotiki.dataAccess.daos;

import com.kotiki.core.entities.Owner;

import org.springframework.data.jpa.repository.JpaRepository;

public interface OwnerDao extends JpaRepository<Owner, Long> {

}

package com.kotiki.dataAccess.tools;

public class KotikiExeption extends RuntimeException {

private Integer errorCode;

public KotikiExeption(String message) { super(message); }

public KotikiExeption(String message, Throwable cause) { super(message, cause); }

public Integer getErrorCode() { return errorCode; }

}

package com.kotiki.dataAccess.tools;

import com.kotiki.core.entities.Cat;

import com.kotiki.core.entities.Owner;

import org.hibernate.SessionFactory;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.service.ServiceRegistry;

import java.util.HashMap;

import java.util.Map;

public class SessionFactoryBuilder {

public SessionFactory build(String url, String username, String password, String dialect, String driverClass) {

Map<String, Object> settings = new HashMap<>();

settings.put("connection.driver\_class", driverClass);

settings.put("dialect", "org.hibernate.dialect." + dialect);

settings.put("hibernate.connection.url", url);

settings.put("hibernate.connection.username", username);

settings.put("hibernate.connection.password", password);

settings.put("hibernate.current\_session\_context\_class", "thread");

settings.put("hibernate.show\_sql", "true");

settings.put("hibernate.format\_sql", "true");

ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()

.applySettings(settings).build();

MetadataSources metadataSources = new MetadataSources(serviceRegistry);

metadataSources.addAnnotatedClass(Owner.class);

metadataSources.addAnnotatedClass(Cat.class);

Metadata metadata = metadataSources.buildMetadata();

return metadata.getSessionFactoryBuilder().build();

}

}

package com.kotiki.infrastructure.services;

import com.kotiki.dataAccess.daos.CatDao;

import com.kotiki.core.entities.Cat;

import com.kotiki.core.models.Color;

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

import org.springframework.stereotype.Service;

import com.kotiki.core.services.CatService;

import java.util.List;

@Service

public class InfrastructureCatService {

private final CatService catService;

private final CatDao catDao;

@Autowired

public InfrastructureCatService(CatService catService, CatDao catDao) {

this.catService = catService;

this.catDao = catDao;

}

public List<Cat> getAll() { return catDao.findAll(); }

public Cat getById(Long id) { return catDao.getById(id); }

public void renameCat(Cat cat, String name) {

catService.renameCat(cat, name);

catDao.save(cat);

}

public void friendCats(Cat left, Cat right) {

catService.friendCats(left, right);

catDao.save(left);

catDao.save(right);

}

public void unfriendCats(Cat left, Cat right) {

catService.unfriendCats(left, right);

catDao.save(left);

catDao.save(right);

}

public void paintCat(Cat cat, Color color) {

catService.paintCat(cat, color);

catDao.save(cat);

}

public Cat addToDatabase(Cat cat) { return catDao.save(cat); }

}

package com.kotiki.infrastructure.services;

import com.kotiki.dataAccess.daos.CatDao;

import com.kotiki.dataAccess.daos.OwnerDao;

import com.kotiki.core.entities.Cat;

import com.kotiki.core.entities.Owner;

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

import org.springframework.stereotype.Service;

import com.kotiki.core.services.OwnerService;

import java.util.List;

@Service

public class InfrastructureOwnerService {

private final OwnerService ownerService;

private final CatDao catDao;

private final OwnerDao ownerDao;

@Autowired

public InfrastructureOwnerService(OwnerService ownerService, CatDao catDao, OwnerDao ownerDao) {

this.ownerService = ownerService;

this.catDao = catDao;

this.ownerDao = ownerDao;

}

public List<Owner> getAll() { return ownerDao.findAll(); }

public Owner getById(Long id) { return ownerDao.getById(id); }

public void addCat(Owner owner, Cat cat) {

ownerService.addCat(owner, cat);

ownerDao.save(owner);

catDao.save(cat);

}

public void removeCat(Owner owner, Cat cat) {

ownerService.removeCat(owner, cat);

ownerDao.save(owner);

catDao.save(cat);

}

public Owner addToDatabase(Owner owner) { return ownerDao.save(owner); }

}

package com.kotiki.presentation.controllers;

import com.kotiki.presentation.dtos.CatDto;

import com.kotiki.core.models.Color;

import com.kotiki.presentation.dtos.OwnerDto;

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

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

import com.kotiki.infrastructure.services.InfrastructureCatService;

import java.util.List;

@RestController

@RequestMapping("cats")

public class CatController {

@Autowired

private InfrastructureCatService catService;

@GetMapping("all")

public List<CatDto> getAllCats() { return catService.getAll().stream().map(CatDto::new).toList(); }

@GetMapping("{id}")

public CatDto getCatWithId(@PathVariable Long id) { return new CatDto(catService.getById(id)); }

@GetMapping("color")

public List<CatDto> getCatsWithColor(@RequestParam String color) {

var colorEnum = Color.valueOf(color);

return catService.getAll().stream().filter(c -> c.getColor() == colorEnum).map(CatDto::new).toList();

}

@GetMapping("{id}/friends")

public List<CatDto> getCatsFriends(@PathVariable Long id) {

var cat = catService.getById(id);

return cat.getFriends().stream().map(CatDto::new).toList();

}

@GetMapping("{id}/owner")

public OwnerDto getOwner(@PathVariable Long id) {

var cat = catService.getById(id);

return new OwnerDto(cat.getOwner());

}

}

package com.kotiki.presentation.controllers;

import com.kotiki.presentation.dtos.CatDto;

import com.kotiki.presentation.dtos.OwnerDto;

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

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

import com.kotiki.infrastructure.services.InfrastructureOwnerService;

import java.util.List;

@RestController

@RequestMapping("owners")

public class OwnerController {

@Autowired

private InfrastructureOwnerService ownerService;

@GetMapping("all")

public List<OwnerDto> getAllOwners() { return ownerService.getAll().stream().map(OwnerDto::new).toList(); }

@GetMapping("{id}")

public OwnerDto getOwnerWithId(@PathVariable Long id) { return new OwnerDto(ownerService.getById(id)); }

@GetMapping("{id}/cats")

public List<CatDto> getCats(@PathVariable Long id) {

var owner = ownerService.getById(id);

return owner.getCats().stream().map(CatDto::new).toList();

}

}

package com.kotiki.presentation.dtos;

import com.kotiki.core.entities.Cat;

import com.kotiki.core.entities.Owner;

import java.time.LocalDateTime;

import java.util.List;

public class CatDto {

private Long id;

private String name;

private LocalDateTime birthDate;

private String breed;

private String color;

private Long ownerId;

private List<Long> friendIds;

public CatDto() { }

public CatDto(Cat cat) {

id = cat.getId();

name = cat.getName();

birthDate = cat.getBirthDate();

breed = cat.getBreed();

color = cat.getColor().toString();

Owner owner = cat.getOwner();

ownerId = owner == null ? -1 : owner.getId();

friendIds = cat.getFriends().stream().map(Cat::getId).toList();

}

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public LocalDateTime getBirthDate() { return birthDate; }

public void setBirthDate(LocalDateTime birthData) { this.birthDate = birthData; }

public String getBreed() { return breed; }

public void setBreed(String breed) { this.breed = breed; }

public String getColor() { return color; }

public void setColor(String color) { this.color = color; }

public Long getOwnerId() { return ownerId; }

public void setOwnerId(Long ownerId) { this.ownerId = ownerId; }

public List<Long> getFriendIds() { return friendIds; }

public void setFriendIds(List<Long> friendIds) { this.friendIds = friendIds; }

}

package com.kotiki.presentation.dtos;

import com.kotiki.core.entities.Cat;

import com.kotiki.core.entities.Owner;

import java.time.LocalDateTime;

import java.util.List;

public class OwnerDto {

private Long id;

private String name;

private LocalDateTime birthDate;

private List<Long> catsIds;

public OwnerDto() { }

public OwnerDto(Owner owner) {

id = owner.getId();

name = owner.getName();

birthDate = owner.getBirthDate();

catsIds = owner.getCats().stream().map(Cat::getId).toList();

}

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public LocalDateTime getBirthDate() { return birthDate; }

public void setBirthDate(LocalDateTime birthData) { this.birthDate = birthData; }

public List<Long> getCatsIds() { return catsIds; }

public void setCatsIds(List<Long> catsIds) { this.catsIds = catsIds; }

}

package com.kotiki.presentation;

import com.kotiki.core.entities.Cat;

import com.kotiki.presentation.controllers.CatController;

import com.kotiki.dataAccess.daos.CatDao;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

import org.springframework.boot.jdbc.DataSourceBuilder;

import org.springframework.context.annotation.\*;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import com.kotiki.core.services.CatService;

import com.kotiki.infrastructure.services.InfrastructureCatService;

import com.kotiki.core.services.OwnerService;

import javax.sql.DataSource;

@Configuration

@ComponentScan(basePackageClasses = { InfrastructureCatService.class, CatController.class })

@EntityScan(basePackageClasses = { Cat.class })

@EnableJpaRepositories(basePackageClasses = { CatDao.class })

@SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })

public class Main {

public static void main(String[] args) {

var app = new SpringApplication(Main.class);

app.run(args);

}

@Bean

public DataSource datasource() {

return DataSourceBuilder.create()

.driverClassName("org.postgresql.Driver")

.url("jdbc:postgresql://localhost:5432/kotiki-java")

.username("postgres")

.password("1234567890")

.build();

}

@Bean

public CatService catService() { return new CatService(); }

@Bean

public OwnerService ownerService() { return new OwnerService(); }

}

import com.kotiki.dataAccess.daos.CatDao;

import com.kotiki.dataAccess.daos.OwnerDao;

import com.kotiki.core.entities.Cat;

import com.kotiki.core.entities.Owner;

import com.kotiki.core.models.Color;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.TestInstance;

import com.kotiki.core.services.CatService;

import com.kotiki.infrastructure.services.InfrastructureOwnerService;

import com.kotiki.infrastructure.services.InfrastructureCatService;

import org.mockito.\*;

import com.kotiki.core.services.OwnerService;

import java.time.LocalDateTime;

import static org.junit.jupiter.api.Assertions.assertTrue;

import static org.mockito.Matchers.any;

import static org.mockito.Mockito.\*;

@TestInstance(TestInstance.Lifecycle.PER\_CLASS)

public class KotikiTest {

private InfrastructureOwnerService infrastructureOwnerService;

private InfrastructureCatService infrastructureCatService;

private CatDao catDao;

private OwnerDao ownerDao;

@BeforeEach

public void setup() {

catDao = Mockito.mock(CatDao.class);

when(catDao.save(any(Cat.class))).thenAnswer(i -> i.getArguments()[0]);

ownerDao = Mockito.mock(OwnerDao.class);

when(ownerDao.save(any(Owner.class))).thenAnswer(i -> i.getArguments()[0]);

infrastructureOwnerService = new InfrastructureOwnerService(new OwnerService(), catDao, ownerDao);

infrastructureCatService = new InfrastructureCatService(new CatService(), catDao);

}

@Test

public void addCatToOwner\_OwnerCatsContainsCat() {

LocalDateTime date1 = LocalDateTime.of(2001, 12, 12, 11, 0, 3);

LocalDateTime date2 = LocalDateTime.of(2020, 1, 3, 4, 12, 0);

Owner owner = new Owner("Виктор", date1);

Cat cat = new Cat("Федя", date2, "Дворняга", Color.ORANGE);

infrastructureOwnerService.addCat(owner, cat);

verify(ownerDao, times(1)).save(owner);

verify(catDao, times(1)).save(cat);

assertTrue(owner.getCats().contains(cat));

}

@Test

public void makeCatsFriends\_BothCatsUpdated() {

LocalDateTime date1 = LocalDateTime.of(2019, 12, 12, 11, 0, 3);

LocalDateTime date2 = LocalDateTime.of(2020, 1, 3, 4, 12, 0);

Cat cat1 = new Cat("Виктор", date1, "Дворняга", Color.BLACK);

Cat cat2 = new Cat("Федя", date2, "Дворняга", Color.ORANGE);

infrastructureCatService.friendCats(cat1, cat2);

verify(catDao, times(1)).save(cat1);

verify(catDao, times(1)).save(cat2);

assertTrue(cat1.getFriends().contains(cat2));

}

}