**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе №2

по дисциплине «**Технологии программирования**»

Автор: Юрпалов Сергей Николаевич

Факультет: ИТиП

Группа: М32051



Санкт-Петербург 2022

Описание:

Нужно написать сервис по учету котиков и их владельцев.

Существующая информация о котиках:

Имя

Дата рождения

Порода

Цвет (один из заранее заданных вариантов)

Хозяин

Список котиков, с которыми дружит этот котик (из представленных в базе)

Существующая информация о хозяевах:

Имя

Дата рождения

Список котиков

Сервис должен реализовывать архитектуру controller-service-dao.

Вся информация хранится в БД PostgreSQL. Для связи с БД должен использоваться Hibernate.

Проект должен собираться с помощью Maven или Gradle (на выбор студента). Слой доступа к данным и сервисный слой должны являться двумя разными модулями Maven/Gradle. При этом проект должен полностью собираться одной командой.

При тестировании рекомендуется использовать Mockito, чтобы избежать подключения к реальным базам данных. Фреймворк для тестирования рекомендуется Junit 5.

В данной лабораторной нельзя использовать Spring или подобные ему фреймворки.

Исходный код:

Удален файл kotiki-java/Main.java

Добавлен файл kotiki-java/database-access/pom.xml:

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

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<parent>

<artifactId>java-kotiki</artifactId>

<groupId>org.wilfordaf</groupId>

<version>1.0-SNAPSHOT</version>

</parent>

<modelVersion>4.0.0</modelVersion>

<artifactId>database-access</artifactId>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.3.3</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.6.5.Final</version>

</dependency>

</dependencies>

</project>

Добавлен файл kotiki-java/database-access/src/main/java/dao/CatDao.java:

package dao;

import dao.interfaces.AbstractDao;

import models.Cat;

import org.hibernate.Session;

import org.hibernate.Transaction;

import tools.CatsException;

import utils.HibernateSessionFactoryUtil;

import java.util.List;

public class CatDao implements AbstractDao<Cat> {

public Cat findById(int id) {

return HibernateSessionFactoryUtil.getSessionFactory().openSession().get(Cat.class, id);

}

public void save(Cat cat) {

Session session = HibernateSessionFactoryUtil.getSessionFactory().openSession();

Transaction transaction = session.beginTransaction();

session.save(cat);

transaction.commit();

session.close();

}

public void update(Cat cat) {

Session session = HibernateSessionFactoryUtil.getSessionFactory().openSession();

Transaction transaction = session.beginTransaction();

session.merge(cat);

transaction.commit();

session.close();

}

public void delete(int id) throws CatsException {

Session session = HibernateSessionFactoryUtil.getSessionFactory().openSession();

Transaction transaction = session.beginTransaction();

Cat cat = session.get(Cat.class, id);

for (Cat friend : cat.getFriends()) {

friend.removeFriend(cat);

}

session.delete(cat);

transaction.commit();

session.close();

}

public List<Cat> findAll() {

return HibernateSessionFactoryUtil

.getSessionFactory()

.openSession()

.createQuery("FROM Cat", Cat.class)

.list();

}

}

Добавлен файл kotiki-java/database-access/src/main/java/dao/OwnerDao.java:

package dao;

import dao.interfaces.AbstractDao;

import models.Owner;

import org.hibernate.Session;

import org.hibernate.Transaction;

import utils.HibernateSessionFactoryUtil;

import java.util.List;

public class OwnerDao implements AbstractDao<Owner> {

public Owner findById(int id) {

return HibernateSessionFactoryUtil.openSession().get(Owner.class, id);

}

public void save(Owner owner) {

Session session = HibernateSessionFactoryUtil.openSession();

Transaction transaction = session.beginTransaction();

session.save(owner);

transaction.commit();

session.close();

}

public void update(Owner owner) {

Session session = HibernateSessionFactoryUtil.openSession();

Transaction transaction = session.beginTransaction();

session.merge(owner);

transaction.commit();

session.close();

}

public void delete(int id) {

Session session = HibernateSessionFactoryUtil.openSession();

Transaction transaction = session.beginTransaction();

Owner owner = session.get(Owner.class, id);

session.delete(owner);

transaction.commit();

session.close();

}

public List<Owner> findAll() {

return HibernateSessionFactoryUtil

.getSessionFactory()

.openSession()

.createQuery("FROM Owner", Owner.class)

.list();

}

}

Добавлен файл kotiki-java/database-access/src/main/java/dao/interfaces/AbstractDao.java:

package dao.interfaces;

import tools.CatsException;

import java.util.List;

public interface AbstractDao<T> {

T findById(int id);

void save(T t);

void update(T t);

void delete(int id) throws CatsException;

List<T> findAll();

}

Добавлен файл kotiki-java/database-access/src/main/java/models/Cat.java:

package models;

import tools.CatsException;

import javax.persistence.\*;

import java.time.LocalDate;

import java.util.\*;

@Entity

@Table(name = "cats", schema = "public", catalog = "postgres")

public class Cat {

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Id

@Column(name = "id")

private int id;

@Basic

@Column(name = "name")

private String name;

@Basic

@Column(name = "birth\_date")

private LocalDate birthDate;

@Basic

@Column(name = "breed")

private String breed;

@Enumerated(EnumType.STRING)

@Column(name = "color")

private CatColors color;

@ManyToOne(fetch = FetchType.LAZY)

@JoinTable(

name = "owner\_cats",

joinColumns = @JoinColumn(name = "cat\_id"),

inverseJoinColumns = @JoinColumn(name = "owner\_id")

)

private Owner owner;

@ManyToMany(cascade = {CascadeType.PERSIST, CascadeType.MERGE, CascadeType.REFRESH})

@JoinTable(name="cat\_friends",

joinColumns=@JoinColumn(name="cat\_id"),

inverseJoinColumns=@JoinColumn(name="friend\_id")

)

private Set<Cat> friends = new HashSet<>();

@ManyToMany(mappedBy = "friends", cascade = {CascadeType.PERSIST, CascadeType.MERGE, CascadeType.REFRESH})

private Set<Cat> friendOf = new HashSet<>();

public Cat() {}

public Cat(String name, LocalDate birthDate, String breed, CatColors color) {

this.name = name;

this.birthDate = birthDate;

this.breed = breed;

this.color = color;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public LocalDate getBirthDate() {

return birthDate;

}

public void setBirthDate(LocalDate birthDate) {

this.birthDate = birthDate;

}

public String getBreed() {

return breed;

}

public void setBreed(String breed) {

this.breed = breed;

}

public CatColors getColor() {

return color;

}

public void setColor(CatColors color) {

this.color = color;

}

public Set<Cat> getFriends() {

return friends;

}

public void setFriends(Set<Cat> friends) {

this.friends = friends;

}

public Set<Cat> getFriendOf() {

return friendOf;

}

public void setFriendOf(Set<Cat> friendOf) {

this.friendOf = friendOf;

}

public Owner getOwner() {

return owner;

}

public void setOwner(Owner owner) {

this.owner = owner;

}

public void addFriend(Cat cat) throws CatsException {

if (equals(cat)) {

throw new CatsException("Cat cannot make friends with himself");

}

friendOf.add(cat);

friends.add(cat);

}

public void removeFriend(Cat cat) throws CatsException {

if (!getFriends().contains(cat)) {

throw new CatsException("Cat does not have such friend");

}

friendOf.remove(cat);

friends.remove(cat);

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Cat cat = (Cat) o;

return id == cat.id && Objects.equals(name, cat.name) && Objects.equals(birthDate, cat.birthDate) && Objects.equals(breed, cat.breed) && Objects.equals(color, cat.color);

}

@Override

public int hashCode() {

return Objects.hash(id, name, birthDate, breed, color);

}

}

Добавлен файл kotiki-java/database-access/src/main/java/models/CatColors.java:

package models;

public enum CatColors {

WHITE,

BLACK,

RED,

GREY,

BROWN

}

Добавлен файл kotiki-java/database-access/src/main/java/models/Owner.java:

package models;

import javax.persistence.\*;

import java.time.LocalDate;

import java.util.HashSet;

import java.util.Objects;

import java.util.Set;

@Entity

@Table(name = "owners", schema = "public", catalog = "postgres")

public class Owner {

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Id

@Column(name = "id")

private int id;

@Basic

@Column(name = "name")

private String name;

@Basic

@Column(name = "birth\_date")

private LocalDate birthDate;

@OneToMany(mappedBy = "owner", cascade = CascadeType.ALL, fetch = FetchType.LAZY)

private Set<Cat> cats = new HashSet<>();

public Owner() {}

public Owner(String name, LocalDate birthDate) {

this.name = name;

this.birthDate = birthDate;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public LocalDate getBirthDate() {

return birthDate;

}

public void setBirthDate(LocalDate birthDate) {

this.birthDate = birthDate;

}

public Set<Cat> getCats() {

return cats;

}

public void setCats(Set<Cat> cats) {

this.cats = cats;

}

public void addCat(Cat cat) {

cat.setOwner(this);

cats.add(cat);

}

public void removeCat(Cat cat) {

cats.remove(cat);

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Owner owner = (Owner) o;

return id == owner.id && Objects.equals(name, owner.name) && Objects.equals(birthDate, owner.birthDate);

}

@Override

public int hashCode() {

return Objects.hash(id, name, birthDate);

}

}

Добавлен файл kotiki-java/database-access/src/main/java/tools/CatsException.java:

package tools;

public class CatsException extends Exception {

public CatsException(String message) {

super(message);

}

}

Добавлен файл kotiki-java/database-access/src/main/java/utils/HibernateSessionFactoryUtil.java:

package utils;

import models.Cat;

import models.Owner;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Configuration;

public class HibernateSessionFactoryUtil {

private static SessionFactory sessionFactory;

private HibernateSessionFactoryUtil() {}

public static SessionFactory getSessionFactory() {

if (sessionFactory == null) {

try {

Configuration configuration = new Configuration().configure();

configuration.addAnnotatedClass(Owner.class);

configuration.addAnnotatedClass(Cat.class);

StandardServiceRegistryBuilder builder = new StandardServiceRegistryBuilder().applySettings(configuration.getProperties());

sessionFactory = configuration.buildSessionFactory(builder.build());

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

return sessionFactory;

}

public static Session openSession() {

return getSessionFactory().openSession();

}

}

Добавлен файл kotiki-java/database-access/src/main/resources/hibernate.cfg.xml:

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">org.postgresql.Driver</property>

<property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/postgres</property>

<property name="hibernate.connection.username">postgres</property>

<property name="hibernate.connection.password">123456</property>

<property name="show\_sql">true</property>

</session-factory>

</hibernate-configuration>

Добавлен файл kotiki-java/pom.xml:

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

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.wilfordaf</groupId>

<artifactId>java-kotiki</artifactId>

<packaging>pom</packaging>

<version>1.0-SNAPSHOT</version>

<modules>

<module>database-access</module>

</modules>

<dependencies>

<dependency>

<groupId>org.wilfordaf</groupId>

<artifactId>database-access</artifactId>

<version>1.0-SNAPSHOT</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>5.6.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

<version>5.6.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

</project>

Добавлен файл kotiki-java/src/main/java/controllers/CatsController.java:

package controllers;

import models.Cat;

import models.Owner;

import services.CatsService;

import tools.CatsException;

import java.util.List;

public class CatsController {

private final CatsService catsService = new CatsService();

public Owner findOwnerById(int id) {

return catsService.findOwnerById(id);

}

public void saveOwner(Owner owner) {

catsService.saveOwner(owner);

}

public void deleteOwner(int id) throws CatsException {

catsService.deleteOwner(id);

}

public void updateOwner(Owner owner) {

catsService.updateOwner(owner);

}

public List<Owner> findAllOwners() {

return catsService.findAllOwners();

}

public void addCatToOwner(Owner owner, Cat cat) {

catsService.addCatToOwner(owner, cat);

}

public void removeCatFromOwner(Owner owner, Cat cat) {

catsService.removeCatFromOwner(owner, cat);

}

public Cat findCatById(int id) {

return catsService.findCatById(id);

}

public void saveCat(Cat cat) {

catsService.saveCat(cat);

}

public void deleteCat(int id) throws CatsException {

catsService.deleteCat(id);

}

public void updateCat(Cat cat) {

catsService.updateCat(cat);

}

public List<Cat> findAllCats() {

return catsService.findAllCats();

}

public void makeCatsFriends(Cat cat, Cat other) throws CatsException {

catsService.makeCatsFriends(cat, other);

}

public void unfriendCats(Cat cat, Cat other) throws CatsException {

catsService.unfriendCats(cat, other);

}

}

Добавлен файл kotiki-java/src/main/java/services/CatsService.java:

package services;

import dao.CatDao;

import dao.OwnerDao;

import dao.interfaces.AbstractDao;

import models.Cat;

import models.Owner;

import tools.CatsException;

import java.util.List;

public class CatsService {

private final AbstractDao<Owner> ownerDao = new OwnerDao();

private final AbstractDao<Cat> catDao = new CatDao();

public Owner findOwnerById(int id) {

return ownerDao.findById(id);

}

public void saveOwner(Owner owner) {

ownerDao.save(owner);

}

public void deleteOwner(int id) throws CatsException {

ownerDao.delete(id);

}

public void updateOwner(Owner owner) {

ownerDao.update(owner);

}

public List<Owner> findAllOwners() {

return ownerDao.findAll();

}

public void addCatToOwner(Owner owner, Cat cat) {

owner.addCat(cat);

}

public void removeCatFromOwner(Owner owner, Cat cat) {

owner.removeCat(cat);

}

public Cat findCatById(int id) {

return catDao.findById(id);

}

public void saveCat(Cat cat) {

catDao.save(cat);

}

public void deleteCat(int id) throws CatsException {

catDao.delete(id);

}

public void updateCat(Cat cat) {

catDao.update(cat);

}

public List<Cat> findAllCats() {

return catDao.findAll();

}

public void makeCatsFriends(Cat cat, Cat other) throws CatsException {

cat.addFriend(other);

other.addFriend(cat);

}

public void unfriendCats(Cat cat, Cat other) throws CatsException {

cat.removeFriend(other);

other.removeFriend(cat);

}

}

Добавлен файл kotiki-java/src/test/java/tests/CatsTests.java:

package tests;

import controllers.CatsController;

import models.Cat;

import models.CatColors;

import models.Owner;

import org.junit.Assert;

import org.junit.Test;

import tools.CatsException;

import java.time.LocalDate;

public class CatsTests {

private final CatsController catsController = new CatsController();

@Test

public void addOwnerWithCats\_checkAllAdded() throws CatsException {

var owner = new Owner("aaa", LocalDate.now());

var cat1 = new Cat("bbb", LocalDate.now(), "123", CatColors.WHITE);

var cat2 = new Cat("ccc", LocalDate.now(), "456", CatColors.BLACK);

catsController.makeCatsFriends(cat1, cat2);

catsController.addCatToOwner(owner, cat1);

catsController.addCatToOwner(owner, cat2);

catsController.saveOwner(owner);

Assert.assertEquals(owner, catsController.findOwnerById(owner.getId()));

Assert.assertEquals(cat1, catsController.findCatById(cat1.getId()));

Assert.assertEquals(cat2, catsController.findCatById(cat2.getId()));

Assert.assertEquals(2, catsController.findOwnerById(owner.getId()).getCats().size());

Assert.assertEquals(cat2, catsController.findCatById(cat1.getId()).getFriends().iterator().next());

catsController.deleteOwner(owner.getId());

}

@Test

public void addNewFriendToCat\_addNewCatToOwner\_makeCatFriendWithHimself\_updateSystem() throws CatsException {

var owner1 = new Owner("aaa", LocalDate.now());

var owner2 = new Owner("fk", LocalDate.now());

var cat1 = new Cat("bbb", LocalDate.now(), "123", CatColors.WHITE);

var cat2 = new Cat("ccc", LocalDate.now(), "456", CatColors.BLACK);

catsController.makeCatsFriends(cat1, cat2);

catsController.addCatToOwner(owner1, cat1);

catsController.addCatToOwner(owner1, cat2);

catsController.saveOwner(owner1);

catsController.saveOwner(owner2);

var cat3 = new Cat("101", LocalDate.now(), "101", CatColors.BROWN);

catsController.addCatToOwner(owner2, cat3);

catsController.makeCatsFriends(cat1, cat3);

Assert.assertThrows(CatsException.class, () -> {

catsController.makeCatsFriends(cat1, cat1);

});

Assert.assertEquals(0, catsController.findOwnerById(owner2.getId()).getCats().size());

Assert.assertEquals(1, catsController.findCatById(cat1.getId()).getFriends().size());

catsController.updateOwner(owner2);

Assert.assertEquals(2, catsController.findOwnerById(owner1.getId()).getCats().size());

Assert.assertEquals(1, catsController.findOwnerById(owner2.getId()).getCats().size());

Assert.assertEquals(2, catsController.findCatById(cat1.getId()).getFriends().size());

catsController.deleteOwner(owner1.getId());

catsController.deleteOwner(owner2.getId());

}

@Test

public void deleteCat\_checkOwnerCats\_checkCatFriends() throws CatsException {

var owner = new Owner("abc", LocalDate.now());

var cat1 = new Cat("bcd", LocalDate.now(), "867", CatColors.RED);

var cat2 = new Cat("cde", LocalDate.now(), "989", CatColors.BROWN);

catsController.makeCatsFriends(cat1, cat2);

catsController.addCatToOwner(owner, cat1);

catsController.addCatToOwner(owner, cat2);

catsController.saveOwner(owner);

catsController.deleteCat(cat1.getId());

Assert.assertNull(catsController.findCatById(cat1.getId()));

Assert.assertEquals(cat2, catsController.findCatById(cat2.getId()));

Assert.assertEquals(1, catsController.findOwnerById(owner.getId()).getCats().size());

Assert.assertTrue(catsController.findCatById(cat2.getId()).getFriends().isEmpty());

Assert.assertTrue(catsController.findCatById(cat2.getId()).getFriendOf().isEmpty());

catsController.deleteOwner(owner.getId());

}

}