**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе №3

по дисциплине «**Технологии программирования**»

Автор: Кошкин Михаил Сергеевич

Факультет: ИТиП

Группа: М32051



Санкт-Петербург 2022

Описание:

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).

В рамках лабораторной к проекту должен быть добавлен CI/CD, запускающий тесты проекта kotiki-java.

Исходный код:

Добавлен файл kotiki-java/.github/workflows/maven.yml:

# This workflow will build a Java project with Maven, and cache/restore any dependencies to improve the workflow execution time

# For more information see: https://help.github.com/actions/language-and-framework-guides/building-and-testing-java-with-maven

name: Java CI with Maven

on:

push:

branches: [ master ]

pull\_request:

branches: [ master ]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Set up JDK 11

uses: actions/setup-java@v3

with:

java-version: '11'

distribution: 'temurin'

cache: maven

- name: Build with Maven

run: mvn -B test --file kotiki-java/pom.xml

Добавлен файл kotiki-java/.gitignore:

# Project related fils

db\_files/\*

# Maven build directory

target

# Compiled class file

\*.class

# Log file

\*.log

# BlueJ files

\*.ctxt

# Mobile Tools for Java (J2ME)

.mtj.tmp/

\*.idea

\*.iml

# Package Files #

\*.jar

\*.war

\*.nar

\*.ear

\*.zip

\*.tar.gz

\*.rar

# virtual machine crash logs, see http://www.java.com/en/download/help/error\_hotspot.xml

hs\_err\_pid\*

Добавлен файл kotiki-java/api/pom.xml:

<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>kotiki-java</artifactId>

<groupId>kotiki-java</groupId>

<version>1.0-SNAPSHOT</version>

</parent>

<modelVersion>4.0.0</modelVersion>

<artifactId>api</artifactId>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>kotiki-java</groupId>

<artifactId>dal</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.4.0</version>

</dependency>

</dependencies>

</project>

Добавлен файл kotiki-java/api/src/main/java/app/Application.java:

package app;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

import org.springframework.context.annotation.ComponentScan;

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

@SpringBootApplication

@EntityScan("models")

@EnableJpaRepositories("dao")

@ComponentScan({"controllers", "services", "utils"})

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

Добавлен файл kotiki-java/api/src/main/java/app/BrowserLauncher.java:

package app;

import org.springframework.boot.context.event.ApplicationReadyEvent;

import org.springframework.context.event.EventListener;

import org.springframework.stereotype.Component;

import java.awt.\*;

import java.io.IOException;

import java.net.URI;

import java.net.URISyntaxException;

@Component

public class BrowserLauncher{

@EventListener(ApplicationReadyEvent.class)

public void launchBrowser() throws URISyntaxException, IOException {

System.setProperty("java.awt.headless", "false");

Desktop.getDesktop().browse(new URI("http://localhost:9090/api/cat/get/all"));

}

}

Добавлен файл kotiki-java/api/src/main/java/controllers/CatController.java:

package controllers;

import dto.CatDto;

import enums.Color;

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

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

import org.springframework.http.HttpStatus;

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

import org.springframework.web.server.ResponseStatusException;

import services.AbstractCatService;

import utils.CatConverter;

import java.util.List;

import java.util.stream.Collectors;

@RestController

@RequestMapping("/api/cat")

public class CatController {

private final AbstractCatService service;

private final CatConverter converter;

@Autowired

public CatController(AbstractCatService service, CatConverter converter) {

this.service = service;

this.converter = converter;

}

@GetMapping("/get/all")

public List<CatDto> getAll() {

try {

return service

.getAll()

.stream()

.map(converter::toDto)

.collect(Collectors.toList());

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

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

public CatDto get(@PathVariable int id) {

try {

return converter.toDto(service.get(id));

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.NOT\_FOUND);

}

}

@GetMapping("/get/breed/{breed}")

public List<CatDto> getByBreed(@PathVariable String breed) {

try {

return service

.getByBreed(breed)

.stream()

.map(converter::toDto)

.collect(Collectors.toList());

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

@GetMapping("/get/color/{color}")

public List<CatDto> getByColor(@PathVariable String color) {

try {

return service

.getByColor(Color.valueOf(color.toUpperCase()))

.stream()

.map(converter::toDto)

.collect(Collectors.toList());

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

@PostMapping("/create")

@ResponseStatus(HttpStatus.OK)

public void create(@RequestBody CatDto cat) {

try {

service.save(converter.toModel(cat));

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

@PostMapping("/update")

public void update(@RequestBody CatDto cat) {

if (!service.exists(cat.getId()))

throw new ResponseStatusException(HttpStatus.NOT\_FOUND);

try {

service.save(converter.toModel(cat));

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

@PostMapping("/delete/{id}")

public void delete(@PathVariable int id) {

try {

service.delete(id);

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

}

Добавлен файл kotiki-java/api/src/main/java/controllers/OwnerController.java:

package controllers;

import dto.OwnerDto;

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

import org.springframework.http.HttpStatus;

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

import org.springframework.web.server.ResponseStatusException;

import services.AbstractOwnerService;

import utils.OwnerConverter;

import java.util.List;

import java.util.stream.Collectors;

@RestController

@RequestMapping("/api/owner")

public class OwnerController {

private final AbstractOwnerService service;

private final OwnerConverter converter;

@Autowired

public OwnerController(AbstractOwnerService service, OwnerConverter converter) {

this.service = service;

this.converter = converter;

}

@GetMapping("/get/all")

public List<OwnerDto> getAll() {

try {

return service

.getAll()

.stream()

.map(converter::toDto)

.collect(Collectors.toList());

} catch (Exception e){

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

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

public OwnerDto get(@PathVariable int id) {

try {

return converter.toDto(service.get(id));

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.NOT\_FOUND);

}

}

@PostMapping("/create")

public void create(@RequestBody OwnerDto owner) {

try {

service.save(converter.toModel(owner));

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

@PostMapping("/update")

public void update(@RequestBody OwnerDto owner) {

if (!service.exists(owner.getId()))

throw new ResponseStatusException(HttpStatus.NOT\_FOUND);

try {

service.save(converter.toModel(owner));

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

@PostMapping("/delete/{id}")

public void delete(@PathVariable int id) {

try {

service.delete(id);

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

}

Добавлен файл kotiki-java/api/src/main/java/dto/CatDto.java:

package dto;

import enums.Color;

import lombok.\*;

import java.time.LocalDate;

import java.util.List;

@Data

@Builder

public class CatDto {

private int id;

private String name;

private LocalDate birthDate;

private String breed;

private Color color;

private Integer ownerId;

private List<Integer> friendIds;

}

Добавлен файл kotiki-java/api/src/main/java/dto/OwnerDto.java:

package dto;

import lombok.\*;

import java.util.List;

@Data

@Builder

public class OwnerDto {

private int id;

private String name;

private List<Integer> catIds;

}

Добавлен файл kotiki-java/api/src/main/java/services/AbstractCatService.java:

package services;

import enums.Color;

import models.Cat;

import java.util.List;

public interface AbstractCatService {

List<Cat> getAll();

Cat get(int id);

List<Cat> getByBreed(String breed);

List<Cat> getByColor(Color color);

void save(Cat cat);

void delete(int id);

boolean exists(int id);

}

Добавлен файл kotiki-java/api/src/main/java/services/AbstractOwnerService.java:

package services;

import models.Owner;

import java.util.List;

public interface AbstractOwnerService {

List<Owner> getAll();

Owner get(int id);

void save(Owner owner);

void delete(int id);

boolean exists(int id);

}

Добавлен файл kotiki-java/api/src/main/java/services/CatService.java:

package services;

import dao.CatDao;

import enums.Color;

import models.Cat;

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

import org.springframework.stereotype.Component;

import org.springframework.stereotype.Service;

import tools.DalException;

import java.util.List;

import java.util.Optional;

@Service

public class CatService implements AbstractCatService {

private final CatDao dao;

@Autowired

public CatService(CatDao dao) {

this.dao = dao;

}

public List<Cat> getAll() {

return dao.findAll();

}

public Cat get(int id) {

Optional<Cat> result = dao.findById(id);

if (result.isEmpty())

throw new DalException("No cat with such id found");

return result.get();

}

public List<Cat> getByBreed(String breed) {

return dao.findByBreed(breed);

}

public List<Cat> getByColor(Color color) {

return dao.findByColor(color);

}

public void save(Cat cat) {

dao.save(cat);

}

public void delete(int id) {

dao.deleteById(id);

}

public boolean exists(int id) {

return dao.existsById(id);

}

}

Добавлен файл kotiki-java/api/src/main/java/services/OwnerService.java:

package services;

import dao.OwnerDao;

import models.Cat;

import models.Owner;

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

import org.springframework.stereotype.Service;

import tools.DalException;

import java.util.List;

import java.util.Optional;

@Service

public class OwnerService implements AbstractOwnerService {

private final OwnerDao dao;

@Autowired

public OwnerService(OwnerDao dao) {

this.dao = dao;

}

public List<Owner> getAll() {

return dao.findAll();

}

public Owner get(int id) {

Optional<Owner> result = dao.findById(id);

if (result.isEmpty())

throw new DalException("No owner with such id found");

return result.get();

}

public void save(Owner owner) {

dao.save(owner);

}

public void delete(int id) {

dao.deleteById(id);

}

public boolean exists(int id) {

return dao.existsById(id);

}

}

Добавлен файл kotiki-java/api/src/main/java/tools/ControllerException.java:

package tools;

public class ControllerException extends RuntimeException {

public ControllerException() {

super();

}

public ControllerException(String message) {

super(message);

}

}

Добавлен файл kotiki-java/api/src/main/java/utils/CatConverter.java:

package utils;

import dto.CatDto;

import models.Cat;

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

import org.springframework.stereotype.Component;

import services.CatService;

import services.OwnerService;

import java.util.stream.Collectors;

@Component

public class CatConverter {

private final CatService catService;

private final OwnerService ownerService;

@Autowired

public CatConverter(CatService catService, OwnerService ownerService) {

this.catService = catService;

this.ownerService = ownerService;

}

public CatDto toDto(Cat cat) {

return CatDto.builder()

.id(cat.getId())

.name(cat.getName())

.birthDate(cat.getBirthDate())

.breed(cat.getBreed())

.color(cat.getColor())

.ownerId(cat.getOwner() == null ? null : cat.getOwner().getId())

.friendIds(cat.getFriends()

.stream()

.map(Cat::getId)

.collect(Collectors.toList()))

.build();

}

public Cat toModel(CatDto cat) {

Cat.CatBuilder catBuilder = Cat.builder()

.id(cat.getId())

.name(cat.getName())

.birthDate(cat.getBirthDate())

.breed(cat.getBreed())

.color(cat.getColor());

if (cat.getOwnerId() != null)

catBuilder.owner(ownerService.get(cat.getOwnerId()));

if (cat.getFriendIds() != null)

catBuilder.friends(cat.getFriendIds()

.stream()

.map(catService::get)

.collect(Collectors.toList()));

return catBuilder.build();

}

}

Добавлен файл kotiki-java/api/src/main/java/utils/OwnerConverter.java:

package utils;

import dto.OwnerDto;

import models.Cat;

import models.Owner;

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

import org.springframework.stereotype.Component;

import services.CatService;

import java.util.stream.Collectors;

@Component

public class OwnerConverter {

private final CatService catService;

@Autowired

public OwnerConverter(CatService catService) {

this.catService = catService;

}

public OwnerDto toDto(Owner owner) {

return OwnerDto.builder()

.id(owner.getId())

.name(owner.getName())

.catIds(owner.getCats()

.stream()

.map(Cat::getId)

.collect(Collectors.toList()))

.build();

}

public Owner toModel(OwnerDto owner) {

Owner.OwnerBuilder ownerBuilder = Owner.builder()

.id(owner.getId())

.name(owner.getName());

if (owner.getCatIds() != null)

ownerBuilder.cats(owner.getCatIds()

.stream()

.map(catService::get)

.collect(Collectors.toList()));

return ownerBuilder.build();

}

}

Добавлен файл kotiki-java/api/src/main/resources/application.properties:

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

spring.datasource.username=postgres

spring.datasource.password=koshkin

spring.jpa.show-sql=true

## Hibernate Properties

# The SQL dialect makes Hibernate generate better SQL for the chosen database

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect

spring.jpa.properties.hibernate.event.merge.entity\_copy\_observer=allow

# Hibernate ddl auto (create, create-drop, validate, update)

spring.jpa.hibernate.ddl-auto = update

server.port=9090

Добавлен файл kotiki-java/api/src/test/java/services/CatServiceTest.java:

package services;

import dao.CatDao;

import models.Cat;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import java.util.List;

import java.util.Optional;

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

import static org.mockito.Mockito.\*;

public class CatServiceTest {

private CatService service;

private CatDao dao;

@BeforeEach

public void setup() {

dao = mock(CatDao.class);

service = new CatService(dao);

}

@Test

public void testGetAll() {

var cat1 = Cat.builder().id(1).name("Cat").build();

var cat2 = Cat.builder().id(2).name("Also A Cat").build();

var cats = List.of(cat1, cat2);

when(dao.findAll()).thenReturn(cats);

List<Cat> result = service.getAll();

verify(dao).findAll();

assertEquals(2, result.size());

assertEquals(cat1, result.get(0));

assertEquals(cat2, result.get(1));

}

@Test

public void testGet() {

var cat = Cat.builder().id(1).name("Name").build();

when(dao.findById(1)).thenReturn(Optional.of(cat));

Cat result = service.get(1);

verify(dao).findById(anyInt());

assertEquals(cat, result);

}

@Test

public void testSave() {

var cat = Cat.builder()

.id(1)

.name("Name")

.friends(List.of(new Cat("Another Cat")))

.build();

service.save(cat);

verify(dao, times(1)).save(any());

}

@Test

public void testDelete() {

service.delete(1);

verify(dao, times(1)).deleteById(anyInt());

}

}

Добавлен файл kotiki-java/api/src/test/java/services/OwnerServiceTest.java:

package services;

import dao.OwnerDao;

import models.Cat;

import models.Owner;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import java.time.LocalDate;

import java.util.List;

import java.util.Optional;

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

import static org.mockito.Mockito.\*;

public class OwnerServiceTest {

private OwnerService service;

private OwnerDao dao;

@BeforeEach

public void setup() {

dao = mock(OwnerDao.class);

service = new OwnerService(dao);

}

@Test

public void testGetAll() {

var owner1 = Owner.builder().id(1).name("Name1").build();

var owner2 = Owner.builder().id(2).name("Name2").build();

var owners = List.of(owner1, owner2);

when(dao.findAll()).thenReturn(owners);

List<Owner> result = service.getAll();

verify(dao).findAll();

assertEquals(2, result.size());

assertEquals(owner1, result.get(0));

assertEquals(owner2, result.get(1));

}

@Test

public void testGet() {

var owner = Owner.builder().id(1).name("Name").build();

when(dao.findById(1)).thenReturn(Optional.of(owner));

Owner result = service.get(1);

verify(dao).findById(anyInt());

assertEquals(owner, result);

}

@Test

public void testSave() {

var owner = Owner.builder()

.id(1)

.name("SomeName")

.birthDate(LocalDate.now())

.cats(List.of(new Cat("Cat")))

.build();

service.save(owner);

verify(dao, times(1)).save(any());

}

@Test

public void testDelete() {

service.delete(1);

verify(dao, times(1)).deleteById(anyInt());

}

}

Добавлен файл kotiki-java/dal/pom.xml:

<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>kotiki-java</artifactId>

<groupId>kotiki-java</groupId>

<version>1.0-SNAPSHOT</version>

</parent>

<modelVersion>4.0.0</modelVersion>

<artifactId>dal</artifactId>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.3.3</version>

</dependency>

</dependencies>

</project>

Добавлен файл kotiki-java/dal/src/main/java/Main.java:

public class Main {

public static void main(String[] args) {

}

}

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

package dao;

import enums.Color;

import models.Cat;

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

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

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface CatDao extends JpaRepository<Cat, Integer> {

List<Cat> findByBreed(@Param("breed") String breed);

List<Cat> findByColor(@Param("color") Color color);

}

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

package dao;

import models.Owner;

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

import org.springframework.stereotype.Repository;

@Repository

public interface OwnerDao extends JpaRepository<Owner, Integer> {

}

Добавлен файл kotiki-java/dal/src/main/java/enums/Color.java:

package enums;

public enum Color {

BLACK,

BLUE,

CHOCOLATE,

CINNAMON,

CREAM,

FAWN,

GRAY,

LILAC,

ORANGE,

RED,

WHITE

}

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

package models;

import enums.Color;

import lombok.\*;

import org.hibernate.Hibernate;

import org.hibernate.annotations.CascadeType;

import org.hibernate.annotations.Cascade;

import tools.DalException;

import javax.persistence.\*;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

import java.util.stream.Collectors;

import java.util.stream.Stream;

@Entity

@Table(name = "cats")

@NoArgsConstructor

@RequiredArgsConstructor

@Getter

@Setter

@ToString

public class Cat {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(nullable = false)

private int id;

@Column(nullable = false)

@NonNull

private String name;

@Column(name = "birth\_date")

private LocalDate birthDate;

private String breed;

@Enumerated(EnumType.STRING)

private Color color;

@ManyToOne

@Cascade({

CascadeType.PERSIST,

CascadeType.MERGE,

CascadeType.SAVE\_UPDATE

})

@JoinColumn(name = "owner\_id")

@ToString.Exclude

private Owner owner;

@ManyToMany

@Cascade({

CascadeType.PERSIST,

CascadeType.MERGE,

CascadeType.SAVE\_UPDATE

})

@JoinTable(name = "cats\_cats",

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

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

)

@ToString.Exclude

private List<Cat> friends = new ArrayList<>();

@ManyToMany

@JoinTable(name = "cats\_cats",

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

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

)

@ToString.Exclude

@Getter(AccessLevel.NONE)

@Setter(AccessLevel.NONE)

private List<Cat> friendsOf = new ArrayList<>();

@Builder

public Cat(int id,

@NonNull String name,

LocalDate birthDate,

String breed,

Color color,

Owner owner,

List<Cat> friends) {

this.id = id;

this.name = name;

this.birthDate = birthDate;

this.breed = breed;

this.color = color;

this.owner = owner;

setFriends(friends);

}

public List<Cat> getFriends() {

return Stream.concat(

friends.stream(),

friendsOf.stream())

.collect(Collectors.toUnmodifiableList());

}

public void setFriends(List<Cat> friends) {

if (friends != null)

this.friends = friends.stream().distinct().collect(Collectors.toList());

if (friendsOf != null)

friendsOf.clear();

}

public void addFriend(Cat friend) {

if (friend == null)

throw new NullPointerException();

if (friends.contains(friend) ||

friendsOf.contains(friend))

throw new DalException("Entity already a friend of a cat");

friends.add(friend);

}

public void deleteFriend(int friendId) {

if (friends.removeIf(f -> f.id == friendId) ||

friendsOf.removeIf(f -> f.id == friendId))

return;

throw new DalException("Entity was not deleted");

}

@Override

public boolean equals(Object o) {

if (this == o)

return true;

if (o == null || Hibernate.getClass(this) != Hibernate.getClass(o))

return false;

var cat = (Cat)o;

return Objects.equals(id, cat.id);

}

@Override

public int hashCode() {

return getClass().hashCode();

}

}

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

package models;

import lombok.\*;

import org.hibernate.Hibernate;

import org.hibernate.annotations.CascadeType;

import org.hibernate.annotations.Cascade;

import javax.persistence.\*;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

@Entity

@Table(name = "owners")

@NoArgsConstructor

@RequiredArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@ToString

@Builder

public class Owner {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(nullable = false)

private int id;

@Column(nullable = false)

@NonNull

private String name;

@Column(name = "birth\_date")

private LocalDate birthDate;

@OneToMany()

@Cascade({

CascadeType.PERSIST,

CascadeType.MERGE,

CascadeType.SAVE\_UPDATE

})

@JoinColumn(name = "owner\_id")

@ToString.Exclude

private List<Cat> cats = new ArrayList<>();

@Override

public boolean equals(Object o) {

if (this == o)

return true;

if (o == null || Hibernate.getClass(this) != Hibernate.getClass(o))

return false;

var owner = (Owner)o;

return Objects.equals(id, owner.id);

}

@Override

public int hashCode() {

return getClass().hashCode();

}

}

Добавлен файл kotiki-java/dal/src/main/java/tools/DalException.java:

package tools;

public class DalException extends RuntimeException {

public DalException() {

super();

}

public DalException(String message) {

super(message);

}

}

Добавлен файл kotiki-java/dal/src/test/java/models/CatTest.java:

package models;

import enums.Color;

import org.junit.jupiter.api.Test;

import tools.DalException;

import java.lang.reflect.Field;

import java.util.List;

import static org.junit.jupiter.api.Assertions.\*;

public class CatTest {

@Test

public void testEqualsById() {

var cat1 = Cat.builder()

.id(1)

.name("Name")

.color(Color.CREAM)

.build();

var cat2 = Cat.builder()

.id(1)

.name("Other Name")

.color(Color.BLUE)

.build();

assertEquals(cat1, cat2);

}

@Test

public void getFriendsReturnsListConcat() {

var friendCat1 = Cat.builder()

.id(1)

.name("Name1")

.build();

var friendCat2 = Cat.builder()

.id(2)

.name("Name2")

.build();

var cat = Cat.builder()

.id(3)

.name("Name")

.friends(List.of(friendCat1))

.build();

assertDoesNotThrow(() -> {

Field field = Cat.class.getDeclaredField("friendsOf");

field.setAccessible(true);

field.set(cat, List.of(friendCat2));

});

List<Cat> friendsList = cat.getFriends();

assertEquals(2, friendsList.size());

assertArrayEquals(new Cat[]{friendCat1, friendCat2},

friendsList.toArray());

}

@Test

public void getFriendsCannotModifyList\_ThrowException() {

var friendCat = Cat.builder()

.id(1)

.name("Name")

.build();

var cat = Cat.builder()

.id(2)

.name("Name")

.friends(List.of(friendCat))

.build();

List<Cat> friendsList = cat.getFriends();

assertThrows(UnsupportedOperationException.class, () ->

friendsList.add(new Cat("Some name")));

assertThrows(UnsupportedOperationException.class, () ->

friendsList.remove(friendCat));

}

@Test

public void createCatWithoutAName\_ThrowException() {

assertThrows(NullPointerException.class, () ->

Cat.builder().build());

}

@Test

public void addFriend\_CatAlreadyAFriend\_ThrowException() {

var cat = Cat.builder()

.id(1)

.name("Name")

.build();

var friendCat = Cat.builder()

.id(2)

.name("Friend")

.build();

cat.addFriend(friendCat);

assertThrows(DalException.class, () ->

cat.addFriend(friendCat));

}

@Test

public void deleteFromFriends\_CatIsNotAFriend\_ThrowException() {

var friendCat = Cat.builder()

.id(2)

.name("SussyCat")

.build();

var cat = Cat.builder()

.id(1)

.name("Name")

.friends(List.of(friendCat))

.build();

cat.deleteFriend(2);

assertThrows(DalException.class, () ->

cat.deleteFriend(2));

}

}

Добавлен файл kotiki-java/dal/src/test/java/models/OwnerTest.java:

package models;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

public class OwnerTest {

@Test

public void testEqualsById() {

var owner1 = Owner.builder()

.id(1)

.name("Name")

.build();

var owner2 = Owner.builder()

.id(1)

.name("Other Name")

.build();

assertEquals(owner1, owner2);

}

@Test

public void createOwnerWithoutAName\_ThrowException() {

assertThrows(NullPointerException.class, () ->

Owner.builder().build());

}

}

Добавлен файл kotiki-java/pom.xml:

<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>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.6.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<modelVersion>4.0.0</modelVersion>

<groupId>kotiki-java</groupId>

<artifactId>kotiki-java</artifactId>

<packaging>pom</packaging>

<version>1.0-SNAPSHOT</version>

<modules>

<module>dal</module>

<module>api</module>

</modules>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter-engine</artifactId>

<version>5.8.2</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

<version>2.6.5</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<version>2.6.5</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<version>2.6.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<version>2.6.5</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<version>10.1.0-M12</version>

</dependency>

<dependency>

<groupId>org.glassfish</groupId>

<artifactId>javax.faces</artifactId>

<version>2.4.0</version>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.22</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.0</version>

</plugin>

<plugin>

<artifactId>maven-failsafe-plugin</artifactId>

<version>2.22.0</version>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<version>2.6.5</version>

</plugin>

</plugins>

</build>

</project>