Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных средств

Лабораторная работа № 2

«JPA (Hibernate/Spring Data)»

Проверил: Выполнил:

асс. каф. ЭВМ ст. гр. 250504

И.Г. Скиба B.C. Моисеев

МИНСК 2024

Лабораторная работа 2

“Тема: JPA (Hibernate/Spring Data)”

**Цель:** познакомиться с базой данных и различными связями.

**Задача:** расширить функционал своего веб сервиса.

**Условия:**

Подключить в проект БД (PostgreSQL/MySQL/и т.д.). Реализация связи один ко многим @OneToMany. Реализация связи многие ко многим @ManyToMany. Реализовать CRUD-операции со всеми сущностями.

**Структура проекта:**

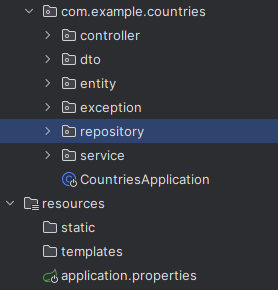
****

Рисунок 1.1 – Структура проекта

Код программы:

Класс CountriesApplication

package com.example.countries;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CountriesApplication {

public static void main(String[] args) {

SpringApplication.run(CountriesApplication.class, args);

}

}

Пакет controller класс CountryController

package com.example.countries.controller;

import com.example.countries.entity.CountryEntity;

import com.example.countries.exception.CountryAlreadyExistException;

import com.example.countries.exception.CountryNotFoundException;

import com.example.countries.model.Country;

import com.example.countries.service.CountryService;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

private static final String ERROR\_MESSAGE = "Произошла ошибка!";

private final CountryService countryService;

public CountryController(CountryService countryService) {

this.countryService = countryService;

}

@PostMapping

public ResponseEntity addCountry(@RequestBody CountryEntity country) {

try {

countryService.add(country);

return ResponseEntity.ok("Страна успешно сохранена!");

} catch (CountryAlreadyExistException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

@GetMapping

public ResponseEntity getCountryFromDb(@RequestParam(required = false) String name) {

try {

return ResponseEntity.ok(countryService.getFromDb(name));

} catch (CountryNotFoundException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

@GetMapping("/name")

public ResponseEntity<?> getCountryByName(@RequestParam String name) {

try {

List<Country> countries = countryService.getByCountryName(name);

return ResponseEntity.ok(countries);

} catch (CountryNotFoundException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

@DeleteMapping("/{id}")

public ResponseEntity deleteCountry(@PathVariable Long id){

try {

countryService.delete(id);

return ResponseEntity.ok("Страна успешно удалена!");

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

}

Пакет controller класс CityController

package com.example.countries.controller;

import com.example.countries.entity.City;

import com.example.countries.exception.CityAlreadyExistException;

import com.example.countries.exception.CityNotFoundException;

import com.example.countries.exception.CountryNotFoundException;

import com.example.countries.service.CityService;

import org.springframework.http.ResponseEntity;

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

@RestController

@RequestMapping("/cities")

public class CityController {

private final CityService cityService;

public CityController(CityService cityService) {

this.cityService = cityService;

}

private static final String ERROR\_MESSAGE = "Произошла ошибка!";

@PostMapping

public ResponseEntity<?> addCity(@RequestParam Long countryId, @RequestBody City city) {

try {

cityService.addCity(countryId, city);

return ResponseEntity.ok("Город был успешно сохранен!");

} catch (CountryNotFoundException | CityAlreadyExistException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

@GetMapping

public ResponseEntity<?> getCity(@RequestParam Long id) {

try {

return ResponseEntity.ok(cityService.getCity(id));

} catch (CityNotFoundException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

@PutMapping

public ResponseEntity<?> updateCity(@RequestParam Long id, @RequestBody City updatedCity) {

try {

cityService.updateCity(id, updatedCity);

return ResponseEntity.ok("Город был успешно изсенен!");

} catch (CityNotFoundException e) {

return ResponseEntity.notFound().build();

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

@DeleteMapping

public ResponseEntity<?> deleteCity(@RequestParam Long id) {

try {

cityService.deleteCity(id);

return ResponseEntity.ok("Город был успешно удален!");

} catch (CityNotFoundException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

}

Пакет controller класс LanguageController

package com.example.countries.controller;

import com.example.countries.entity.Language;

import com.example.countries.exception.CountryNotFoundException;

import com.example.countries.exception.LanguageAlreadyExistException;

import com.example.countries.exception.LanguageNotFoundException;

import com.example.countries.service.LanguageService;

import org.springframework.http.ResponseEntity;

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

@RestController

@RequestMapping("/languages")

public class LanguageController {

private final LanguageService languageService;

public LanguageController(LanguageService languageService) {

this.languageService = languageService;

}

private static final String ERROR\_MESSAGE = "Произошла ошибка!";

@PostMapping

public ResponseEntity<?> addLanguage(@RequestParam Long countryId, @RequestBody Language language) {

try {

languageService.addLanguage(countryId, language);

return ResponseEntity.ok("Язык был успешно сохранен!");

} catch (CountryNotFoundException | LanguageAlreadyExistException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

@GetMapping

public ResponseEntity<?> getLanguage(@RequestParam Long id) {

try {

return ResponseEntity.ok(languageService.getLanguage(id));

} catch (LanguageNotFoundException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

@PutMapping

public ResponseEntity<?> updateLanguage(@RequestParam Long id, @RequestBody Language updatedLanguage) {

try {

languageService.updateLanguage(id, updatedLanguage);

return ResponseEntity.ok("Язык был успешно изменен!");

} catch (LanguageNotFoundException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

@DeleteMapping

public ResponseEntity<?> deleteLanguage(@RequestParam Long countryId, @RequestParam Long languageId) {

try {

languageService.deleteLanguage(countryId, languageId);

return ResponseEntity.ok("Язык был успешно удален!");

} catch (LanguageNotFoundException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body(ERROR\_MESSAGE);

}

}

}

Пакет entity класс CountryEntity

package com.example.countries.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import com.fasterxml.jackson.databind.JsonNode;

@Entity

public class CountryEntity {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

private String name;

private String capital;

public CountryEntity(){

}

public CountryEntity(JsonNode jsonNode) {

this.name = jsonNode.get("name").get("common").asText();

JsonNode capitalNode = jsonNode.get("capital");

this.capital = (capitalNode != null && !capitalNode.isNull()) ? capitalNode.asText() : null;

}

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 String getCapital() {

return capital;

}

public void setCapital(String capital) {

this.capital = capital;

}

}

Пакет entity класс CityEntity

package com.example.countries.entity;

import jakarta.persistence.\*;

@Entity

public class City {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

private String name;

@ManyToOne

@JoinColumn(name = "countryId")

private Country country;

public City(){

// No initialization logic needed for this constructor

}

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 Country getCountry() {

return country;

}

public void setCountry(Country country) {

this.country = country;

}

}

Пакет entity класс LanguageEntity

package com.example.countries.entity;

import jakarta.persistence.\*;

import java.util.ArrayList;

import java.util.List;

@Entity

public class Language {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

private String name;

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(name = "COUNTRY\_LANGUAGE\_MAPPING",

joinColumns = @JoinColumn(name = "languageId"),

inverseJoinColumns = @JoinColumn(name = "countryId"))

private List<Country> countryList = new ArrayList<>();

public Language() {

// No initialization logic needed for this constructor

}

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 List<Country> getCountryList() {

return countryList;

}

public void setCountryList(List<Country> countryList) {

this.countryList = countryList;

}

}

Пакет exception класс CountryAlreadyExistException

package com.example.countries.exception;

public class CountryAlreadyExistException extends Exception{

public CountryAlreadyExistException(String message) {

super(message);

}

}

Пакет exception класс CountryNotFoundException

package com.example.countries.exception;

public class CountryNotFoundException extends Exception{

public CountryNotFoundException(String message) {

super(message);

}

}

Пакет dto класс CountryDTO

package com.example.countries.model;

import com.example.countries.entity.CountryEntity;

public class Country {

private String name;

private String capital;

public static Country toModel(CountryEntity entity){

Country model = new Country();

model.setName(entity.getName());

model.setCapital(entity.getCapital());

return model;

}

public Country() {

//сонар попросил комент

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCapital() {

return capital;

}

public void setCapital(String capital) {

this.capital = capital;

}

}

Пакет dto класс CityDTO

package com.example.countries.dto;

import com.example.countries.entity.City;

public class CityDTO {

private String name;

public static CityDTO toModel(City entity) {

CityDTO model = new CityDTO();

model.setName(entity.getName());

return model;

}

public CityDTO() {

// No initialization logic needed for this constructor

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

Пакет dto класс LanguageDTO

package com.example.countries.dto;

import com.example.countries.entity.Language;

public class LanguageDTO {

private String name;

public static LanguageDTO toModel(Language entity){

LanguageDTO model = new LanguageDTO();

model.setName(entity.getName());

return model;

}

public LanguageDTO() {

// No initialization logic needed for this constructor

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

Пакет repository класс CountryRepository

package com.example.countries.repository;

import com.example.countries.entity.CountryEntity;

import org.springframework.data.repository.CrudRepository;

import java.util.List;

public interface CountryRepo extends CrudRepository<CountryEntity, Long> {

List<CountryEntity> findByName(String name);

}

Пакет repository класс CityRepository

package com.example.countries.repository;

import com.example.countries.entity.City;

import org.springframework.data.repository.CrudRepository;

public interface CityRepository extends CrudRepository<City, Long> {

City findByName(String name);

}

Пакет repository класс LanguageRepository

package com.example.countries.repository;

import com.example.countries.entity.Language;

import org.springframework.data.repository.CrudRepository;

public interface LanguageRepository extends CrudRepository<Language, Long> {

Language findByName(String name);

}

Пакет service класс CountryService

package com.example.countries.service;

import com.example.countries.entity.CountryEntity;

import com.example.countries.exception.CountryAlreadyExistException;

import com.example.countries.exception.CountryNotFoundException;

import com.example.countries.model.Country;

import com.example.countries.repository.CountryRepo;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.fasterxml.jackson.databind.JsonNode;

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

import org.springframework.stereotype.Service;

import org.springframework.web.client.RestTemplate;

import java.net.URLEncoder;

import java.nio.charset.StandardCharsets;

import java.util.ArrayList;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@Service

public class CountryService {

private final CountryRepo countryRepo;

private static final Logger logger = LoggerFactory.getLogger(CountryService.class);

@Autowired

public CountryService(ObjectMapper objectMapper, CountryRepo countryRepo){

this.countryRepo = countryRepo;

}

public CountryEntity add(CountryEntity country) throws CountryAlreadyExistException {

List<CountryEntity> countries = countryRepo.findByName(country.getName());

if (!countries.isEmpty()) {

throw new CountryAlreadyExistException("Страна с таким название уже существует!");

}

return countryRepo.save(country);

}

public List<Country> getFromDb(String name) throws CountryNotFoundException {

List<CountryEntity> countryList = countryRepo.findByName(name);

if (countryList.isEmpty()) {

throw new CountryNotFoundException("Страна с таким названием не была найдена!");

}

return countryList.stream()

.map(Country::toModel)

.toList();

}

public List<Country> getByCountryName(String countryName) throws CountryNotFoundException {

String apiUrl = "https://restcountries.com/v3.1/name/" + URLEncoder.encode(countryName, StandardCharsets.UTF\_8);

RestTemplate restTemplate = new RestTemplate();

try {

String jsonString = restTemplate.getForObject(apiUrl, String.class);

logger.info("JSON-ответ от {}: {}", apiUrl, jsonString);

ObjectMapper mapper = new ObjectMapper();

JsonNode jsonNode = mapper.readTree(jsonString);

if (jsonNode.isArray() && jsonNode.size() > 0) {

List<Country> countries = new ArrayList<>();

for (JsonNode countryNode : jsonNode) {

CountryEntity countryEntity = new CountryEntity(countryNode);

countryEntity.setCapital(countryNode.get("capital").get(0).asText());

countries.add(Country.toModel(countryEntity));

}

return countries;

} else {

logger.warn("Получен пустой массив данных от запроса: {}", apiUrl);

throw new CountryNotFoundException("Страна с таким названием не была найдена!");

}

} catch (Exception e) {

logger.error("Ошибка при выполнении запроса", e);

throw new CountryNotFoundException("Произошла ошибка при выполнении запроса");

}

}

public Long delete(Long id){

countryRepo.deleteById(id);

return id;

}

}

Пакет service класс CityService

package com.example.countries.service;

import com.example.countries.dto.CityDTO;

import com.example.countries.entity.City;

import com.example.countries.entity.Country;

import com.example.countries.exception.CityAlreadyExistException;

import com.example.countries.exception.CityNotFoundException;

import com.example.countries.exception.CountryNotFoundException;

import com.example.countries.repository.CityRepository;

import com.example.countries.repository.CountryRepository;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import javax.swing.\*;

@Service

public class CityService {

private final CityRepository cityRepository;

private final CountryRepository countryRepository;

private static final String CITY\_NOT\_FOUND\_STRING = "Город не найден!";

@Autowired

public CityService(CityRepository cityRepository, CountryRepository countryRepository) {

this.cityRepository = cityRepository;

this.countryRepository = countryRepository;

}

public void addCity(Long id, City city) throws CountryNotFoundException, CityAlreadyExistException {

Country country = countryRepository.findById(id).orElse(null);

if (country != null) {

city.setCountry(country);

if (cityRepository.findByName(city.getName()) != null) {

throw new CityAlreadyExistException("Такой город уже существует!");

}

cityRepository.save(city);

} else {

throw new CountryNotFoundException("Не удалось добавить город. Город не найден!");

}

}

public CityDTO getCity(Long id) throws CityNotFoundException {

City cityEntity = cityRepository.findById(id).orElse(null);

if (cityEntity != null) {

return CityDTO.toModel(cityEntity);

} else {

throw new CityNotFoundException(CITY\_NOT\_FOUND\_STRING);

}

}

public void updateCity(Long id, City city) throws CityNotFoundException {

City cityEntity = cityRepository.findById(id).orElse(null);

if (cityEntity != null) {

cityEntity.setName(city.getName());

cityRepository.save(cityEntity);

} else {

throw new CityNotFoundException(CITY\_NOT\_FOUND\_STRING);

}

}

@Transactional

public void deleteCity(Long id) throws CityNotFoundException {

City cityEntity = cityRepository.findById(id).orElse(null);

if (cityEntity != null) {

cityEntity.getCountry().getCityList().remove(cityEntity);

countryRepository.save(cityEntity.getCountry());

cityRepository.deleteById(id);

} else {

throw new CityNotFoundException(CITY\_NOT\_FOUND\_STRING);

}

}

}

Пакет service класс LanguageService

package com.example.countries.service;

import com.example.countries.dto.LanguageDTO;

import com.example.countries.entity.Country;

import com.example.countries.entity.Language;

import com.example.countries.exception.CountryNotFoundException;

import com.example.countries.exception.LanguageAlreadyExistException;

import com.example.countries.exception.LanguageNotFoundException;

import com.example.countries.repository.CountryRepository;

import com.example.countries.repository.LanguageRepository;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

@Service

public class LanguageService {

private final LanguageRepository languageRepository;

private final CountryRepository countryRepository;

private static final String LANGUAGE\_NOT\_FOUND\_STRING = "Язык не найден!";

@Autowired

public LanguageService(LanguageRepository languageRepository, CountryRepository countryRepository) {

this.languageRepository = languageRepository;

this.countryRepository = countryRepository;

}

@Transactional

public void addLanguage(Long countryId, Language language) throws CountryNotFoundException, LanguageAlreadyExistException {

Country countryEntity = countryRepository.findById(countryId).orElse(null);

if (countryEntity == null)

throw new CountryNotFoundException("Не удалось добавить язык. Язык не найден!");

if (countryEntity.getLanguageList().stream().anyMatch(existingLanguage -> existingLanguage.getName().equals(language.getName()))) {

throw new LanguageAlreadyExistException("Язык уже существует в данной стране!");

}

Language existingLanguage = languageRepository.findByName(language.getName());

if (existingLanguage != null) {

countryEntity.getLanguageList().add(existingLanguage);

} else {

languageRepository.save(language);

countryEntity.getLanguageList().add(language);

}

countryRepository.save(countryEntity);

}

public LanguageDTO getLanguage(Long id) throws LanguageNotFoundException {

Language language = languageRepository.findById(id).orElse(null);

if (language != null) {

return LanguageDTO.toModel(language);

} else {

throw new LanguageNotFoundException(LANGUAGE\_NOT\_FOUND\_STRING);

}

}

public void updateLanguage(Long id, Language language) throws LanguageNotFoundException {

Language languageEntity = languageRepository.findById(id).orElse(null);

if (languageEntity != null) {

languageEntity.setName(language.getName());

languageRepository.save(languageEntity);

} else {

throw new LanguageNotFoundException(LANGUAGE\_NOT\_FOUND\_STRING);

}

}

@Transactional

public void deleteLanguage(Long countryId, Long languageId) throws LanguageNotFoundException, CountryNotFoundException {

Country countryEntity = countryRepository.findById(countryId).orElse(null);

if (countryEntity == null)

throw new CountryNotFoundException("Блюдо не найдено");

Language languageEntity = languageRepository.findById(languageId).orElse(null);

if (languageEntity == null)

throw new LanguageNotFoundException(LANGUAGE\_NOT\_FOUND\_STRING);

countryEntity.getLanguageList().remove(languageEntity);

countryRepository.save(countryEntity);

languageEntity.getCountryList().remove(countryEntity);

languageRepository.save(languageEntity);

}

}

**Результат работы программы**

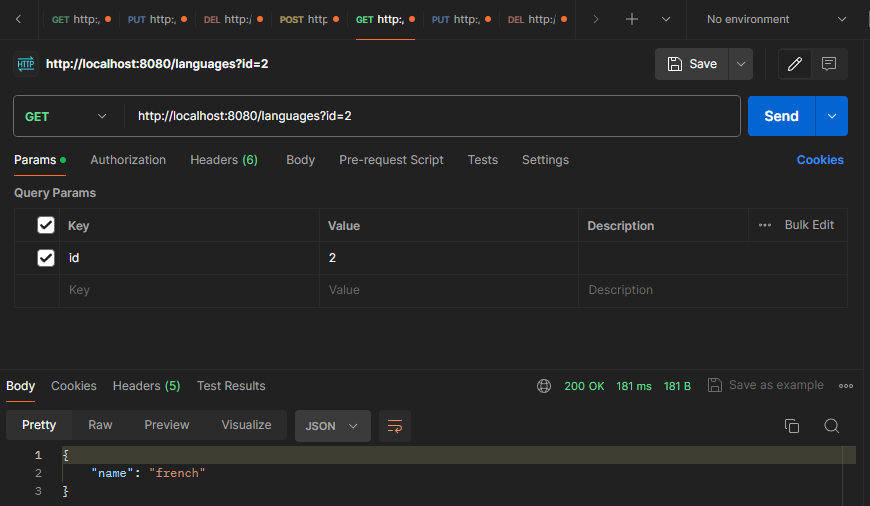


Рисунок 1.2 –результат работы программы