|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |
|  | Институт информационных технологий (ИТ) |
|  | Кафедра инструментального и прикладного программного обеспечения (ИиППО) |

|  |  |  |  |
| --- | --- | --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ №17 - 22** | | | |
| **по дисциплине** | | | |
| **«Шаблоны программных платформ языка Java**  **Вариант 21** | | | |
| Выполнил студент группы ИКБО-20-19 | | Николаев-Аксенов И. С. | |
|  | |  | |
| Принял  *Ассистент* | | Батанов А. О. | |
| Практические работы выполнены | «\_\_\_»\_\_\_\_\_\_\_2021 г. | | (подпись студента) | |
| «Зачтено» | «\_\_\_»\_\_\_\_\_\_\_2021 г. | | (подпись руководителя) | |
|  |  | |  | |

Москва 2021

**Содержание**

[Практическая работа №17 3](#_Toc69053610)

[Практическая работа №18 14](#_Toc69053611)

[Практическая работа №19 24](#_Toc69053612)

[Практическая работа №20 34](#_Toc69053613)

[Практическая работа №21 44](#_Toc69053614)

[Практическая работа №22 56](#_Toc69053615)

[Вывод 69](#_Toc69053616)

[Список использованных источников 69](#_Toc69053617)

Практическая работа №17

***Цель работы***

Тема: Знакомство с Criteria API в Hibernate.

Постановка задачи: Добавить возможность фильтрации по всем полям всех классов с использованием Criteria API в Hibernate для программы из предыдущего задания. Добавить эндпоинты для каждой фильтрации.

***Листинг программы***

*Application.java (в следующих работах тоже присутствует, но не изменяется)*

package app.Application;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

*User.java*

package app.Application.model;

import com.sun.istack.NotNull;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.\*;

@Entity

@Table(name = "users")

public class User implements Serializable {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "last\_name")

@NotNull

private String lastName;

@Column(name = "first\_name")

@NotNull

private String firstName;

@Column(name = "middle\_name")

@NotNull

private String middleName;

@Column(name = "birth\_date")

@NotNull

private String birthDate;

@OneToMany(mappedBy = "user")

private List<Post> posts = new ArrayList<>();

public User() {

}

public User(String lastName, String firstName, String middleName, String birthDate) {

this.lastName = lastName;

this.firstName = firstName;

this.middleName = middleName;

this.birthDate = birthDate;

}

public UUID getId() {

return id;

}

public String getLastName() {

return lastName;

}

public String getFirstName() {

return firstName;

}

public String getMiddleName() {

return middleName;

}

public String getBirthDate() {

return birthDate;

}

@Override

public String toString() {

return "Пользователь #" + id + " " + lastName + " " + firstName + " " + middleName + ", день рождения: " + birthDate;

}

}

*Post.java*

package app.Application.model;

import com.sun.istack.NotNull;

import org.hibernate.annotations.CreationTimestamp;

import org.hibernate.annotations.GenericGenerator;

import org.springframework.format.annotation.DateTimeFormat;

import javax.persistence.\*;

import java.time.LocalDateTime;

import java.util.Date;

import java.util.UUID;

@Entity

@Table(name = "posts")

public class Post {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "text")

@NotNull

private String text;

@CreationTimestamp

@Column(name = "creation\_date")

private LocalDateTime creationDate;

@ManyToOne

private User user;

public Post() {

}

public Post(String text) {

this.text = text;

}

public UUID getId() {

return id;

}

public String getText() {

return text;

}

public LocalDateTime getCreationDate() {

return creationDate;

}

public User getUser() {

return user;

}

}

*UserService.java*

package app.Application.service;

import app.Application.model.User;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

public class UserService {

@Autowired

private final SessionFactory sessionFactory;

private Session session;

private CriteriaBuilder builder;

private CriteriaQuery<User> userCriteriaQuery;

private Root<User> root;

public UserService(SessionFactory sessionFactory) {

this.sessionFactory = sessionFactory;

}

@PostConstruct

public void init() {

session = sessionFactory.openSession();

builder = session.getCriteriaBuilder();

userCriteriaQuery = builder.createQuery(User.class);

root = userCriteriaQuery.from(User.class);

}

@PreDestroy

public void unSession() {

session.close();

}

public void addUser(User user) {

session.beginTransaction();

session.saveOrUpdate(user);

session.getTransaction().commit();

}

public List<User> getUsers() {

return session.createQuery("select u from User u", User.class).list();

}

public User getUser(UUID id) {

return session.createQuery("select u from User u where u.id = p.id = '" + id + "'", User.class).getSingleResult();

}

public void deleteUser(UUID id) {

session.beginTransaction();

User t = session.load(User.class, id);

session.delete(t);

session.getTransaction().commit();

}

public List<User> getByFirstName() {

userCriteriaQuery.select(root).orderBy(builder.asc(root.get("firstName")));

Query<User> query = session.createQuery(userCriteriaQuery);

return query.getResultList();

}

public List<User> getByLastName() {

userCriteriaQuery.select(root).orderBy(builder.asc(root.get("lastName")));

Query<User> query = session.createQuery(userCriteriaQuery);

return query.getResultList();

}

}

*PostService.java*

package app.Application.service;

import app.Application.model.Post;

import app.Application.model.User;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

public class PostService {

@Autowired

private final SessionFactory sessionFactory;

private Session session;

private CriteriaBuilder builder;

private CriteriaQuery<Post> criteriaQuery;

private Root<Post> root;

public PostService(SessionFactory sessionFactory) {

this.sessionFactory = sessionFactory;

}

@PostConstruct

public void init() {

session = sessionFactory.openSession();

builder = session.getCriteriaBuilder();

criteriaQuery = builder.createQuery(Post.class);

root = criteriaQuery.from(Post.class);

}

@PreDestroy

public void unSession() {

session.close();

}

public void addPost(Post post) {

session.beginTransaction();

session.saveOrUpdate(post);

session.getTransaction().commit();

}

public List<Post> getPosts() {

return session.createQuery("select p from Post p", Post.class).list();

}

public User getUser(UUID id) {

return session.createQuery("from Post where id = :id", Post.class).setParameter("id",id).getSingleResult().getUser();

}

public void deletePosts(Post post) {

session.beginTransaction();

List<Post> query = session.createQuery("select p from Post p where p.id = '" + post.getId() + "'", Post.class).list();

for (Post p : query) {

session.delete(p);

}

session.getTransaction().commit();

}

public void deletePost(UUID id) {

session.beginTransaction();

Post t = session.load(Post.class, id);

session.delete(t);

session.getTransaction().commit();

}

public List<Post> getByText() {

criteriaQuery.select(root).orderBy(builder.asc(root.get("text")));

Query<Post> query = session.createQuery(criteriaQuery);

return query.getResultList();

}

public List<Post> getByCreationDate() {

criteriaQuery.select(root).orderBy(builder.asc(root.get("creationDate")));

Query<Post> query = session.createQuery(criteriaQuery);

return query.getResultList();

}

}

*UserController.java*

package app.Application.controller;

import app.Application.model.Post;

import app.Application.model.User;

import app.Application.service.UserService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class UserController {

@Autowired

private UserService userService;

@PostMapping("/users")

public void addUser(@RequestBody User user) {

userService.addUser(user);

}

@GetMapping("/users")

public List<User> getUsers() {

return userService.getUsers();

}

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

public User getUser(@PathVariable UUID id) {

return userService.getUser(id);

}

@DeleteMapping("/users/{id}")

public void deleteUser(@PathVariable UUID id) {

userService.deleteUser(id);

}

@GetMapping("/getByFirstName")

public List<User> getByFirstName() {

return userService.getByFirstName();

}

@GetMapping("/getByLastName")

public List<User> getByLastName() {

return userService.getByLastName();

}

}

*PostController.java*

package app.Application.controller;

import app.Application.model.Post;

import app.Application.model.User;

import app.Application.service.PostService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class PostController {

@Autowired

private PostService postService;

@PostMapping("/post")

public void addPost(@RequestBody Post post) {

postService.addPost(post);

}

@GetMapping("/posts")

public List<Post> getPosts() {

return postService.getPosts();

}

@DeleteMapping("/post/{id}")

public void deletePost(@PathVariable UUID id) {

postService.deletePost(id);

}

@GetMapping("/getByText")

public List<Post> getByText() {

return postService.getByText();

}

@GetMapping("/getByCreationDate")

public List<Post> getByCreationDate() {

return postService.getByCreationDate();

}

@GetMapping(value = "/post/{id}/user")

public @ResponseBody

User getUser(@PathVariable("id") UUID id) {

return postService.getUser(id);

}

}

*Config.java*

package app.Application.config;

import com.zaxxer.hikari.HikariConfig;

import com.zaxxer.hikari.HikariDataSource;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import org.springframework.transaction.PlatformTransactionManager;

import javax.sql.DataSource;

import java.util.Properties;

@Configuration

public class Config {

@Bean

public HikariDataSource dataSource(){

HikariConfig config = new HikariConfig();

config.setJdbcUrl("jdbc:postgresql://localhost:5432/pr17db");

config.setUsername("postgres");

config.setPassword("secret");

config.setDriverClassName("org.postgresql.Driver");

return new HikariDataSource(config);

}

@Bean

public LocalSessionFactoryBean sessionFactory(DataSource dataSource){

LocalSessionFactoryBean factoryBean = new LocalSessionFactoryBean();

factoryBean.setDataSource(dataSource);

factoryBean.setPackagesToScan("app.Application");

Properties properties = new Properties();

properties.setProperty("hibernate.dialect", "org.hibernate.dialect.PostgreSQLDialect");

factoryBean.setHibernateProperties(properties);

return factoryBean;

}

@Bean

public PlatformTransactionManager platformTransactionManager(LocalSessionFactoryBean factoryBean){

HibernateTransactionManager transactionManager = new HibernateTransactionManager();

transactionManager.setSessionFactory(factoryBean.getObject());

return transactionManager;

}

}

***Результат выполнения программы***

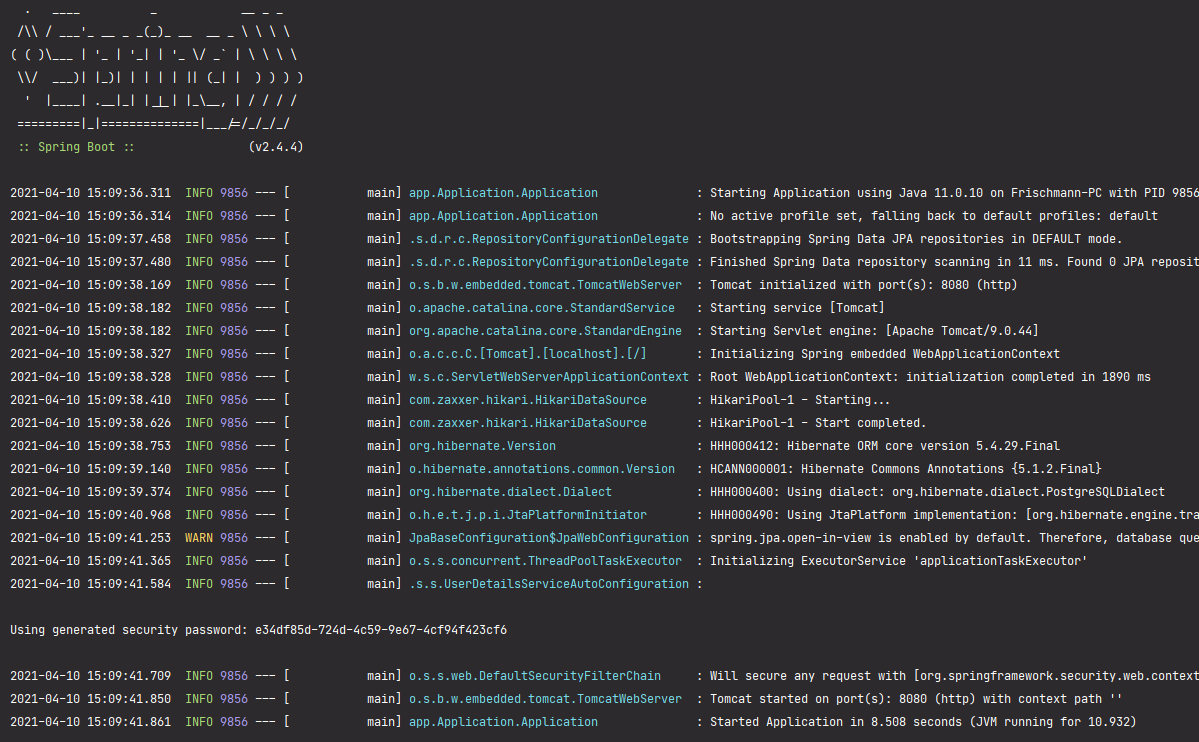


Рисунок 17.1 – Демонстрация работы программы

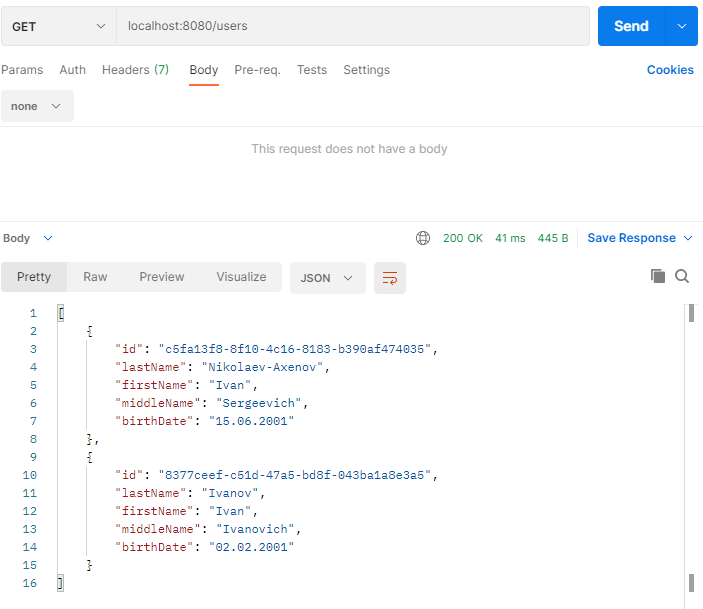


Рисунок 17.2 – Демонстрация работы программы

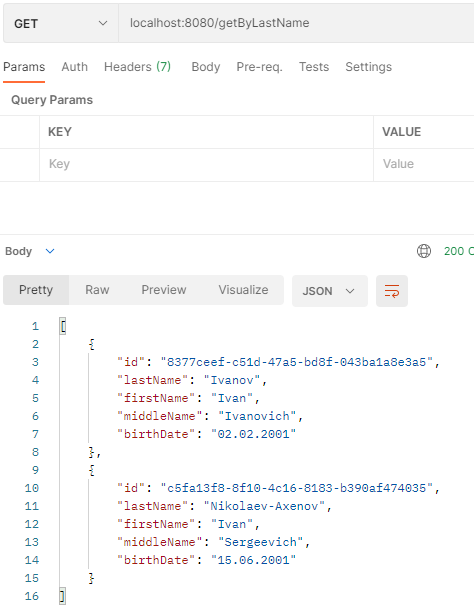


Рисунок 17.3 – Демонстрация работы программы

Практическая работа №18

***Цель работы***

Тема: Знакомство с репозиториями и сервисами, реализация в проекте. Взаимодействие с Spring Data JPA.

Постановка задачи: Переписать код предыдущего задания с использованием сервисов и отделения логики контроллера от логики сервиса и репозитория. В программе всё взаимодействие с базой данных должно быть реализовано через репозитории Spring Data Jpa.

***Листинг программы***

*User.java*

package app.Application.Classes;

import com.sun.istack.NotNull;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.\*;

@Entity

@Table(name = "users")

public class User implements Serializable {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "last\_name")

@NotNull

private String lastName;

@Column(name = "first\_name")

@NotNull

private String firstName;

@Column(name = "middle\_name")

@NotNull

private String middleName;

@Column(name = "birth\_date")

@NotNull

private String birthDate;

@OneToMany(mappedBy = "user")

private List<Post> posts = new ArrayList<>();

public User() {

}

public User(String lastName, String firstName, String middleName, String birthDate) {

this.lastName = lastName;

this.firstName = firstName;

this.middleName = middleName;

this.birthDate = birthDate;

}

public UUID getId() {

return id;

}

public String getLastName() {

return lastName;

}

public String getFirstName() {

return firstName;

}

public String getMiddleName() {

return middleName;

}

public String getBirthDate() {

return birthDate;

}

@Override

public String toString() {

return "Пользователь #" + id + " " + lastName + " " + firstName + " " + middleName + ", день рождения: " + birthDate;

}

}

*Post.java*

package app.Application.Classes;

import com.sun.istack.NotNull;

import org.hibernate.annotations.CreationTimestamp;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.time.LocalDateTime;

import java.util.UUID;

@Entity

@Table(name = "posts")

public class Post {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "text")

@NotNull

private String text;

@CreationTimestamp

@Column(name = "creation\_date")

private LocalDateTime creationDate;

@ManyToOne

private User user;

public Post() {

}

public Post(String text) {

this.text = text;

}

public UUID getId() {

return id;

}

public String getText() {

return text;

}

public LocalDateTime getCreationDate() {

return creationDate;

}

public User getUser() {

return user;

}

}

*UserRepository.java*

package app.Application.Interfaces;

import app.Application.Classes.User;

import com.sun.istack.NotNull;

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

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.UUID;

@Repository("UserRepository")

public interface UserRepository extends JpaRepository<User,Long> {

List<User> findAllByFirstName(String firstName);

List<User> findAllByLastName(String lastName);

@NotNull List<User> findAll();

void deleteById(UUID id);

}

*PostRepository.java*

package app.Application.Interfaces;

import app.Application.Classes.Post;

import com.sun.istack.NotNull;

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

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.UUID;

@Repository("PostRepository")

public interface PostRepository extends JpaRepository<Post,Long> {

Post findById(UUID id);

@NotNull List<Post> findAll();

void deleteById(UUID id);

}

*UserService.java*

package app.Application.Services;

import app.Application.Classes.User;

import app.Application.Interfaces.UserRepository;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

public class UserService {

@Autowired

private final UserRepository userRepository;

public UserService(UserRepository userRepository) {

this.userRepository = userRepository;

}

public void addUser(User user) {

userRepository.save(user);

}

public List<User> getUsers() {

return userRepository.findAll();

}

public void deleteUser(UUID id) {

userRepository.deleteById(id);

}

public List<User> getByFirstName(String firstName) {

return userRepository.findAllByFirstName(firstName);

}

public List<User> getByLastName(String lastName) {

return userRepository.findAllByLastName(lastName);

}

}

*PostService.java*

package app.Application.Services;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Interfaces.PostRepository;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

public class PostService {

@Autowired

private final PostRepository postRepository;

public PostService(PostRepository postRepository) {

this.postRepository = postRepository;

}

public void addPost(Post post) {

postRepository.save(post);

}

public List<Post> getPosts() {

return postRepository.findAll();

}

public void deletePost(UUID id) {

postRepository.deleteById(id);

}

public User getUserByPost(UUID id) {

return postRepository.findById(id).getUser();

}

}

*UserController.java*

package app.Application.Controllers;

import app.Application.Classes.User;

import app.Application.Services.UserService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class UserController {

@Autowired

private UserService userService;

@PostMapping("/users")

public void addUser(@RequestBody User user) {

userService.addUser(user);

}

@GetMapping("/users")

public List<User> getAll() {

return userService.getUsers();

}

@DeleteMapping("/user/{id}")

public void delete(@PathVariable UUID id) {

userService.deleteUser(id);

}

@GetMapping("/getUserByFirstName/{firstName}")

public List<User> getByFirstName(@PathVariable String firstName){

return userService.getByFirstName(firstName);

}

@GetMapping("/getUserByLastName/{lastName}")

public List<User> getByLastName(@PathVariable String lastName){

return userService.getByLastName(lastName);

}

}

*PostController.java*

package app.Application.Controllers;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Services.PostService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class PostController {

@Autowired

private PostService postService;

@PostMapping("/posts")

public void addPost(@RequestBody Post post) {

postService.addPost(post);

}

@GetMapping("/posts")

public List<Post> getAll() {

return postService.getPosts();

}

@DeleteMapping("/post/{id}")

public void delete(@PathVariable UUID id) {

postService.deletePost(id);

}

@GetMapping(value = "/post/{id}/user")

public @ResponseBody

User getGame(@PathVariable("id") UUID id) {

return postService.getUserByPost(id);

}

}

*Config.java*

package app.Application.Configuration;

import org.springframework.context.annotation.Configuration;

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

@Configuration

@EnableJpaRepositories(basePackages = {"app.Application"})

public class Config {

}

*application.yml (в следующих работах тоже присутствует, но меняется только ссылка на базу данных)*

**spring**:

**jpa**:

**database**: POSTGRESQL

**show-sql**: **true**

**hibernate**:

**ddl-auto**: create-drop

**datasource**:

**platform**: postgres

**url**: jdbc:postgresql://localhost:5432/pr18db

**username**: postgres

**password**: secret

**driverClassName**: org.postgresql.Driver

***Результат выполнения программы***

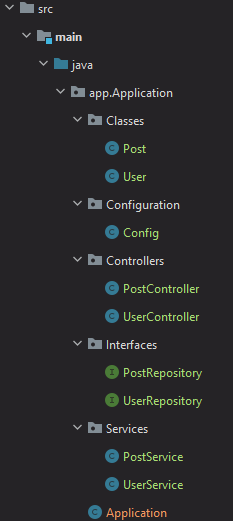


Рисунок 18.1 – Демонстрация работы программы

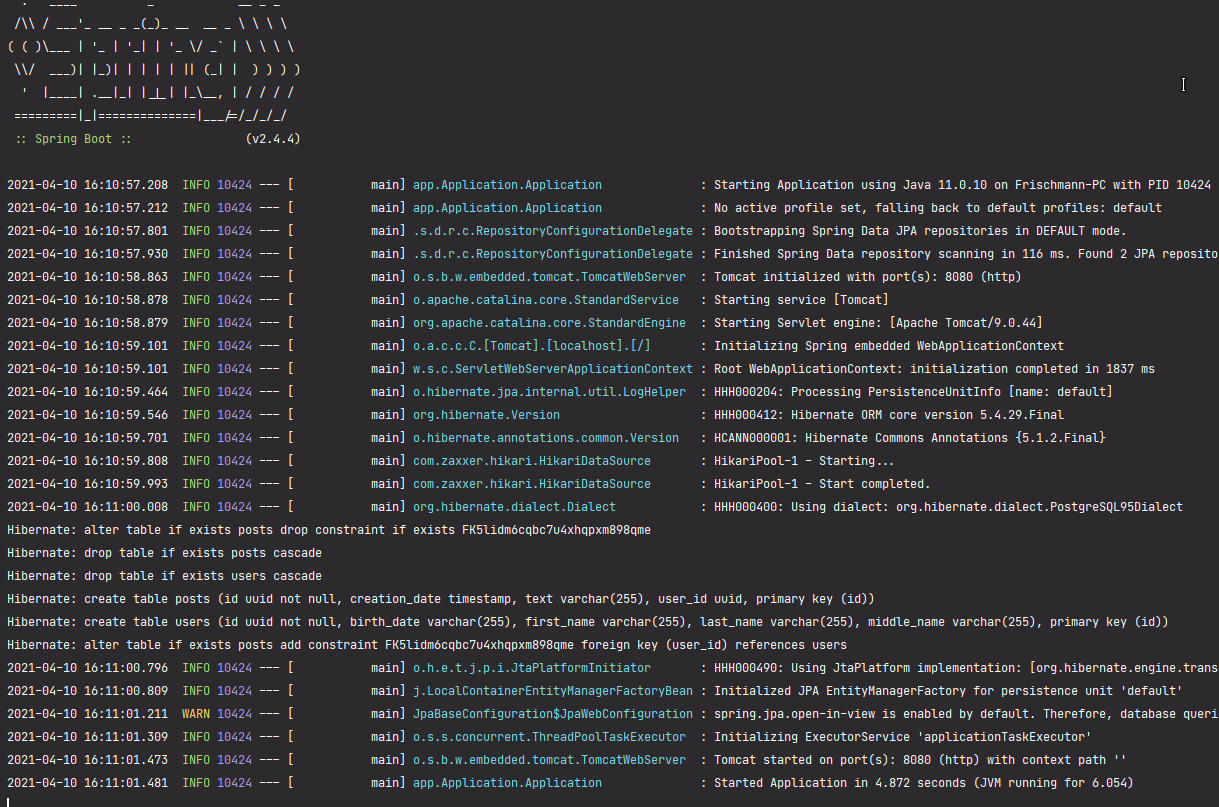


Рисунок 18.2 – Демонстрация работы программы

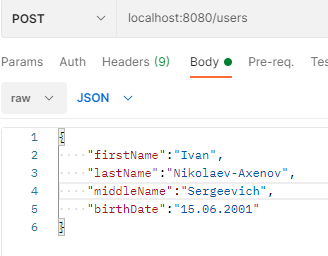


Рисунок 18.3 – Демонстрация работы программы

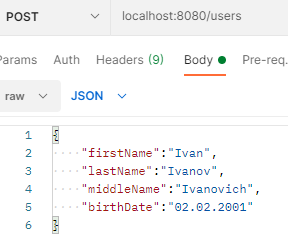


Рисунок 18.4 – Демонстрация работы программы

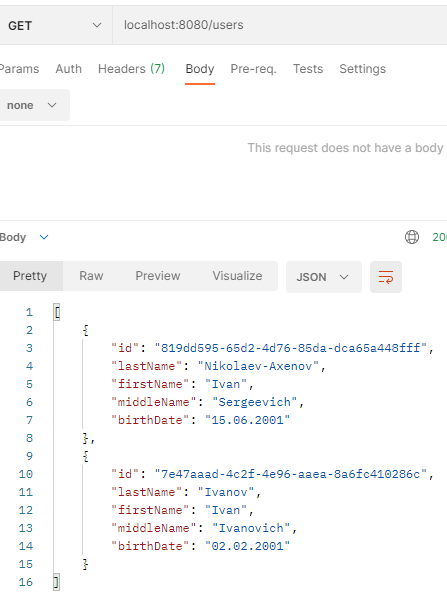


Рисунок 18.5 – Демонстрация работы программы

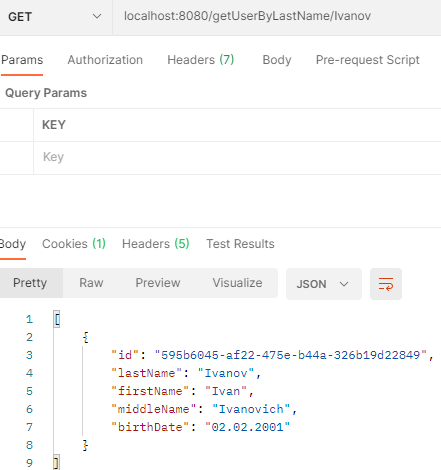


Рисунок 18.6 – Демонстрация работы программы

Практическая работа №19

***Цель работы***

Тема: Знакомство с логированием с использованием Logback в Spring.

Постановка задачи: Создать файл logback.xml, добавить логирование во все методы классов-сервисов.

***Листинг программы***

*User.java*

package app.Application.Classes;

import com.sun.istack.NotNull;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.\*;

@Entity

@Table(name = "users")

public class User implements Serializable {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "last\_name")

@NotNull

private String lastName;

@Column(name = "first\_name")

@NotNull

private String firstName;

@Column(name = "middle\_name")

@NotNull

private String middleName;

@Column(name = "birth\_date")

@NotNull

private String birthDate;

@OneToMany(mappedBy = "user")

private List<Post> posts = new ArrayList<>();

public User() {

}

public User(String lastName, String firstName, String middleName, String birthDate) {

this.lastName = lastName;

this.firstName = firstName;

this.middleName = middleName;

this.birthDate = birthDate;

}

public UUID getId() {

return id;

}

public String getLastName() {

return lastName;

}

public String getFirstName() {

return firstName;

}

public String getMiddleName() {

return middleName;

}

public String getBirthDate() {

return birthDate;

}

@Override

public String toString() {

return "Пользователь #" + id + " " + lastName + " " + firstName + " " + middleName + ", день рождения: " + birthDate;

}

}

*Post.java*

package app.Application.Classes;

import com.sun.istack.NotNull;

import org.hibernate.annotations.CreationTimestamp;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.time.LocalDateTime;

import java.util.UUID;

@Entity

@Table(name = "posts")

public class Post {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "text")

@NotNull

private String text;

@CreationTimestamp

@Column(name = "creation\_date")

private LocalDateTime creationDate;

@ManyToOne

private User user;

public Post() {

}

public Post(String text) {

this.text = text;

}

public UUID getId() {

return id;

}

public String getText() {

return text;

}

public LocalDateTime getCreationDate() {

return creationDate;

}

public User getUser() {

return user;

}

}

*UserRepository.java*

package app.Application.Interfaces;

import app.Application.Classes.User;

import com.sun.istack.NotNull;

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

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.UUID;

@Repository("UserRepository")

public interface UserRepository extends JpaRepository<User,Long> {

List<User> findAllByFirstName(String firstName);

List<User> findAllByLastName(String lastName);

@NotNull List<User> findAll();

void deleteById(UUID id);

}

*PostRepository.java*

package app.Application.Interfaces;

import app.Application.Classes.Post;

import com.sun.istack.NotNull;

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

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.UUID;

@Repository("PostRepository")

public interface PostRepository extends JpaRepository<Post,Long> {

Post findById(UUID id);

@NotNull List<Post> findAll();

void deleteById(UUID id);

}

*UserService.java*

package app.Application.Services;

import app.Application.Classes.User;

import app.Application.Interfaces.UserRepository;

import lombok.extern.slf4j.Slf4j;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

@Slf4j

public class UserService {

@Autowired

private final UserRepository userRepository;

public UserService(UserRepository userRepository) {

this.userRepository = userRepository;

}

public void addUser(User user) {

userRepository.save(user);

}

public List<User> getUsers() {

return userRepository.findAll();

}

public void deleteUser(UUID id) {

userRepository.deleteById(id);

}

public List<User> getByFirstName(String firstName) {

return userRepository.findAllByFirstName(firstName);

}

public List<User> getByLastName(String lastName) {

return userRepository.findAllByLastName(lastName);

}

}

*PostService.java*

package app.Application.Services;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Interfaces.PostRepository;

import lombok.extern.slf4j.Slf4j;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

@Slf4j

public class PostService {

@Autowired

private final PostRepository postRepository;

public PostService(PostRepository postRepository) {

this.postRepository = postRepository;

}

public void addPost(Post post) {

postRepository.save(post);

}

public List<Post> getPosts() {

return postRepository.findAll();

}

public void deletePost(UUID id) {

postRepository.deleteById(id);

}

public User getUserByPost(UUID id) {

return postRepository.findById(id).getUser();

}

}

*UserController.java*

package app.Application.Controllers;

import app.Application.Classes.User;

import app.Application.Services.UserService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class UserController {

@Autowired

private UserService userService;

@PostMapping("/users")

public void addUser(@RequestBody User user) {

userService.addUser(user);

}

@GetMapping("/users")

public List<User> getAll() {

return userService.getUsers();

}

@DeleteMapping("/user/{id}")

public void delete(@PathVariable UUID id) {

userService.deleteUser(id);

}

@GetMapping("/getUserByFirstName/{firstName}")

public List<User> getByFirstName(@PathVariable String firstName){

return userService.getByFirstName(firstName);

}

@GetMapping("/getUserByLastName/{lastName}")

public List<User> getByLastName(@PathVariable String lastName){

return userService.getByLastName(lastName);

}

}

*PostController.java*

package app.Application.Controllers;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Services.PostService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class PostController {

@Autowired

private PostService postService;

@PostMapping("/posts")

public void addPost(@RequestBody Post post) {

postService.addPost(post);

}

@GetMapping("/posts")

public List<Post> getAll() {

return postService.getPosts();

}

@DeleteMapping("/post/{id}")

public void delete(@PathVariable UUID id) {

postService.deletePost(id);

}

@GetMapping(value = "/post/{id}/user")

public @ResponseBody

User getGame(@PathVariable("id") UUID id) {

return postService.getUserByPost(id);

}

}

*Config.java*

package app.Application.Configuration;

import org.springframework.context.annotation.Configuration;

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

@Configuration

@EnableJpaRepositories(basePackages = {"app.Application"})

public class Config {

}

***Результат выполнения программы***

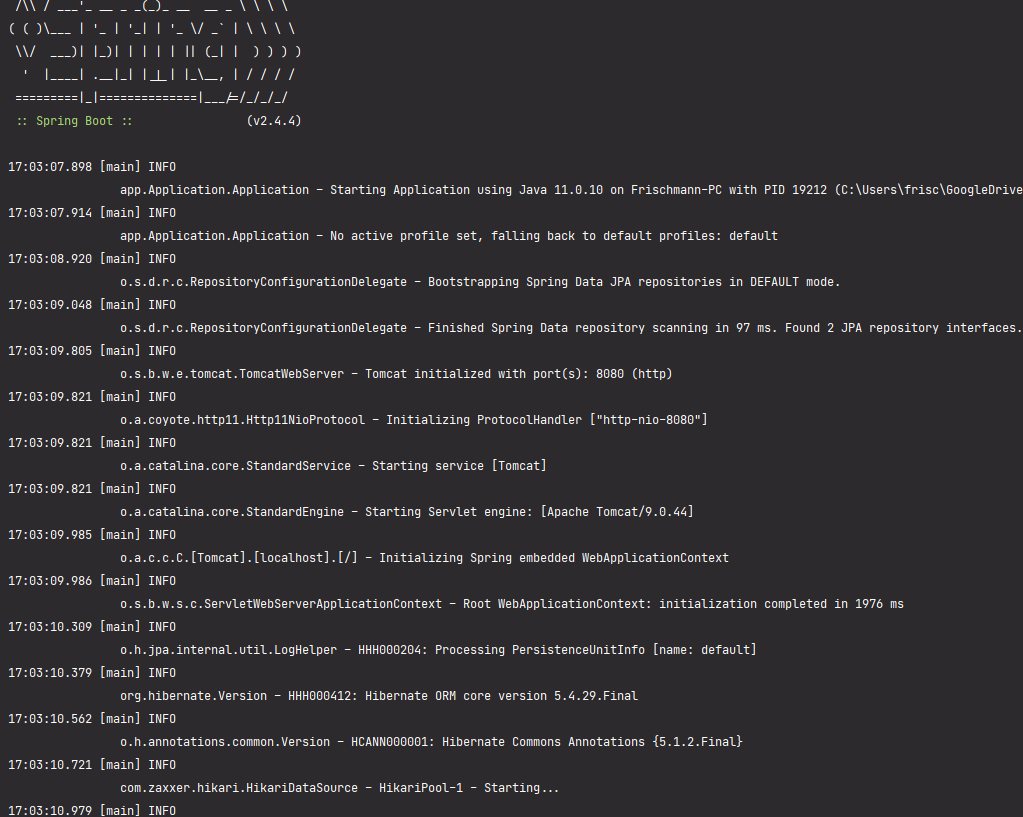


Рисунок 19.1 – Демонстрация работы программы

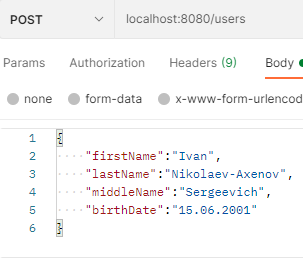


Рисунок 19.2 – Демонстрация работы программы

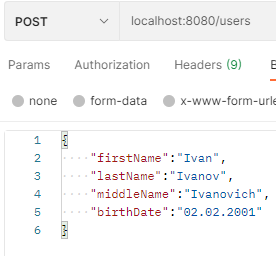


Рисунок 19.3 – Демонстрация работы программы

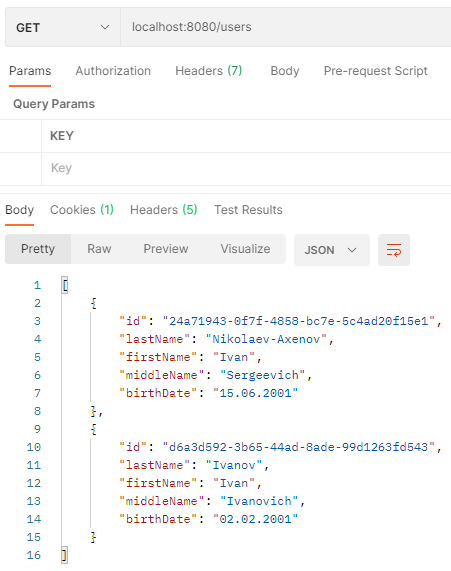


Рисунок 19.4 – Демонстрация работы программы

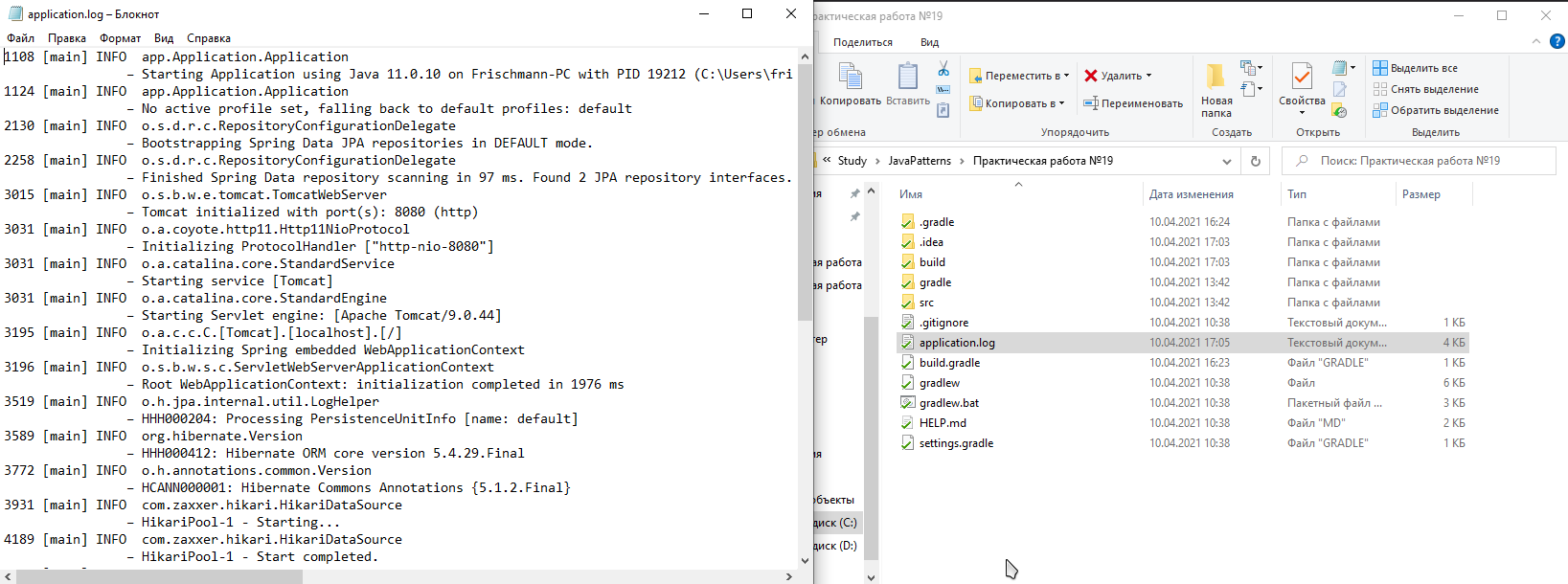


Рисунок 19.5 – Демонстрация работы программы

Практическая работа №20

***Цель работы***

Тема: Использование Spring AOP. Pointcut, JoinPoint. Advice.

Постановка задачи: Для приложения из предыдущего задания добавить логирование времени выполнения каждого метода сервиса с использованием Spring AOP.

***Листинг программы***

*User.java*

package app.Application.Classes;

import com.sun.istack.NotNull;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.\*;

@Entity

@Table(name = "users")

public class User implements Serializable {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "last\_name")

@NotNull

private String lastName;

@Column(name = "first\_name")

@NotNull

private String firstName;

@Column(name = "middle\_name")

@NotNull

private String middleName;

@Column(name = "birth\_date")

@NotNull

private String birthDate;

@OneToMany(mappedBy = "user")

private List<Post> posts = new ArrayList<>();

public User() {

}

public User(String lastName, String firstName, String middleName, String birthDate) {

this.lastName = lastName;

this.firstName = firstName;

this.middleName = middleName;

this.birthDate = birthDate;

}

public UUID getId() {

return id;

}

public String getLastName() {

return lastName;

}

public String getFirstName() {

return firstName;

}

public String getMiddleName() {

return middleName;

}

public String getBirthDate() {

return birthDate;

}

@Override

public String toString() {

return "Пользователь #" + id + " " + lastName + " " + firstName + " " + middleName + ", день рождения: " + birthDate;

}

}

*Post.java*

package app.Application.Classes;

import com.sun.istack.NotNull;

import org.hibernate.annotations.CreationTimestamp;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.time.LocalDateTime;

import java.util.UUID;

@Entity

@Table(name = "posts")

public class Post {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "text")

@NotNull

private String text;

@CreationTimestamp

@Column(name = "creation\_date")

private LocalDateTime creationDate;

@ManyToOne

private User user;

public Post() {

}

public Post(String text) {

this.text = text;

}

public UUID getId() {

return id;

}

public String getText() {

return text;

}

public LocalDateTime getCreationDate() {

return creationDate;

}

public User getUser() {

return user;

}

}

*UserRepository.java*

package app.Application.Interfaces;

import app.Application.Classes.User;

import com.sun.istack.NotNull;

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

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.UUID;

@Repository("UserRepository")

public interface UserRepository extends JpaRepository<User,Long> {

List<User> findAllByFirstName(String firstName);

List<User> findAllByLastName(String lastName);

@NotNull List<User> findAll();

void deleteById(UUID id);

}

*PostRepository.java*

package app.Application.Interfaces;

import app.Application.Classes.Post;

import com.sun.istack.NotNull;

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

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.UUID;

@Repository("PostRepository")

public interface PostRepository extends JpaRepository<Post,Long> {

Post findById(UUID id);

@NotNull List<Post> findAll();

void deleteById(UUID id);

}

*UserService.java*

package app.Application.Services;

import app.Application.Classes.User;

import app.Application.Interfaces.UserRepository;

import lombok.extern.slf4j.Slf4j;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

@Slf4j

public class UserService {

@Autowired

private final UserRepository userRepository;

public UserService(UserRepository userRepository) {

this.userRepository = userRepository;

}

public void addUser(User user) {

userRepository.save(user);

}

public List<User> getUsers() {

return userRepository.findAll();

}

public void deleteUser(UUID id) {

userRepository.deleteById(id);

}

public List<User> getByFirstName(String firstName) {

return userRepository.findAllByFirstName(firstName);

}

public List<User> getByLastName(String lastName) {

return userRepository.findAllByLastName(lastName);

}

}

*PostService.java*

package app.Application.Services;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Interfaces.PostRepository;

import lombok.extern.slf4j.Slf4j;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

@Slf4j

public class PostService {

@Autowired

private final PostRepository postRepository;

public PostService(PostRepository postRepository) {

this.postRepository = postRepository;

}

public void addPost(Post post) {

postRepository.save(post);

}

public List<Post> getPosts() {

return postRepository.findAll();

}

public void deletePost(UUID id) {

postRepository.deleteById(id);

}

public User getUserByPost(UUID id) {

return postRepository.findById(id).getUser();

}

}

*UserController.java*

package app.Application.Controllers;

import app.Application.Classes.User;

import app.Application.Services.UserService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class UserController {

@Autowired

private UserService userService;

@PostMapping("/users")

public void addUser(@RequestBody User user) {

userService.addUser(user);

}

@GetMapping("/users")

public List<User> getAll() {

return userService.getUsers();

}

@DeleteMapping("/user/{id}")

public void delete(@PathVariable UUID id) {

userService.deleteUser(id);

}

@GetMapping("/getUserByFirstName/{firstName}")

public List<User> getByFirstName(@PathVariable String firstName){

return userService.getByFirstName(firstName);

}

@GetMapping("/getUserByLastName/{lastName}")

public List<User> getByLastName(@PathVariable String lastName){

return userService.getByLastName(lastName);

}

}

*PostController.java*

package app.Application.Controllers;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Services.PostService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class PostController {

@Autowired

private PostService postService;

@PostMapping("/posts")

public void addPost(@RequestBody Post post) {

postService.addPost(post);

}

@GetMapping("/posts")

public List<Post> getAll() {

return postService.getPosts();

}

@DeleteMapping("/post/{id}")

public void delete(@PathVariable UUID id) {

postService.deletePost(id);

}

@GetMapping(value = "/post/{id}/user")

public @ResponseBody

User getGame(@PathVariable("id") UUID id) {

return postService.getUserByPost(id);

}

}

*Config.java*

package app.Application.Configuration;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.EnableAspectJAutoProxy;

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

@Configuration

@EnableAspectJAutoProxy

@EnableJpaRepositories(basePackages = {"app.Application"})

public class Config {

}

*Aspect.java*

package app.Application;

import lombok.extern.slf4j.Slf4j;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Pointcut;

import org.springframework.stereotype.Component;

import java.util.logging.Logger;

@Slf4j

@Component

@org.aspectj.lang.annotation.Aspect

public class Aspect {

private Logger log = Logger.getLogger(Aspect.class.getName());

@Around("allServiceMethods()")

public Object logExecutionTime (ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.currentTimeMillis();

Object proceed = joinPoint.proceed();

long executionTime = System.currentTimeMillis() - start;

log.info(joinPoint.getSignature() + " выполнен за " + executionTime + "мс");

return proceed;

}

@Pointcut("within(app.Application.Services.\*)")

public void allServiceMethods() {}

}

***Результат выполнения программы***

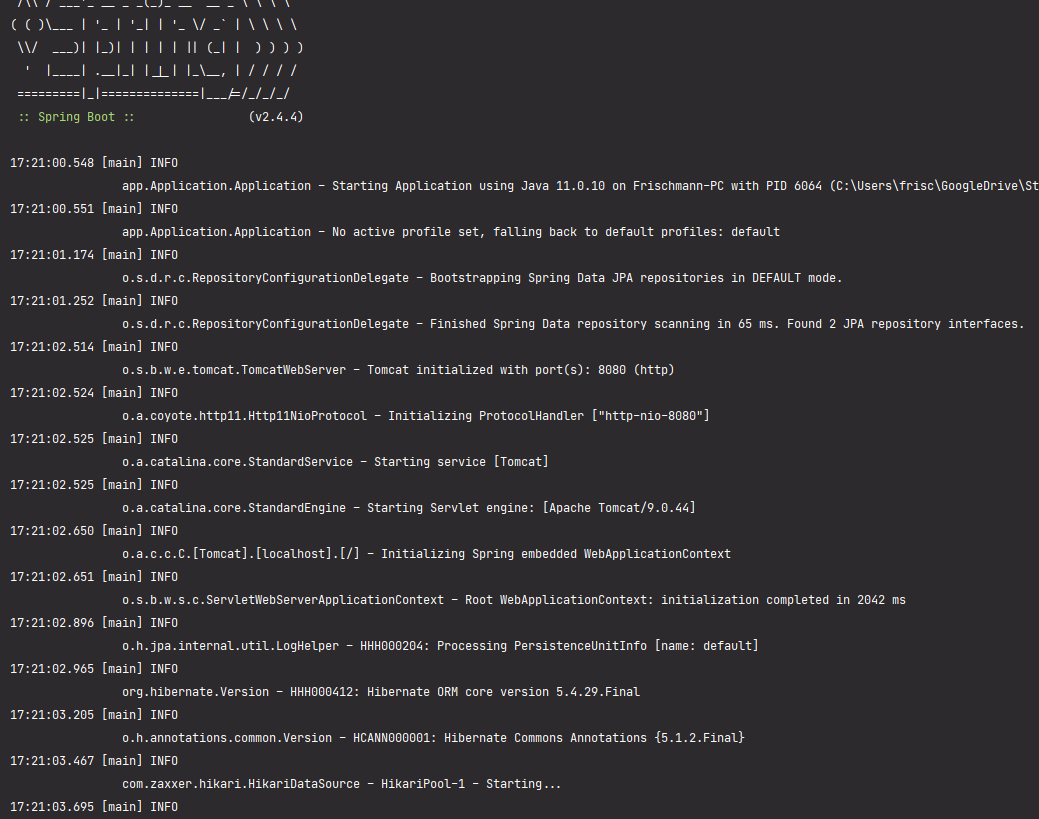


Рисунок 20.1 – Демонстрация работы программы

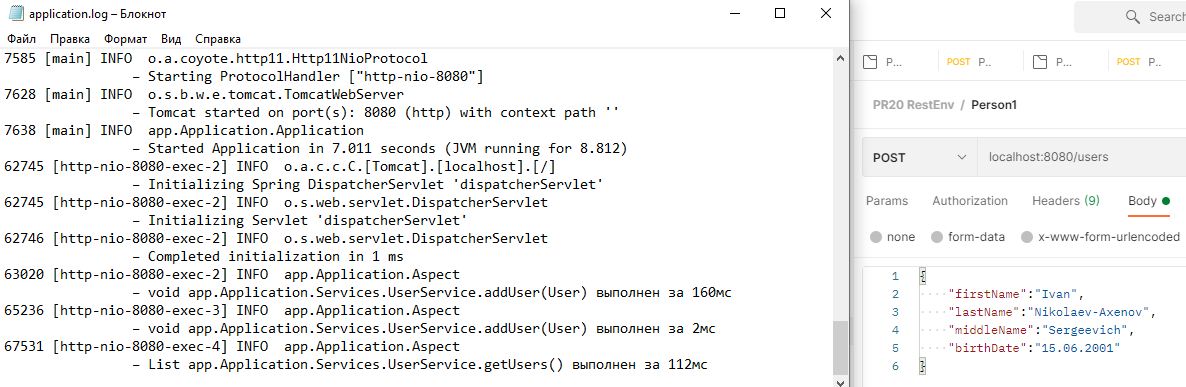


Рисунок 20.2 – Демонстрация работы программы

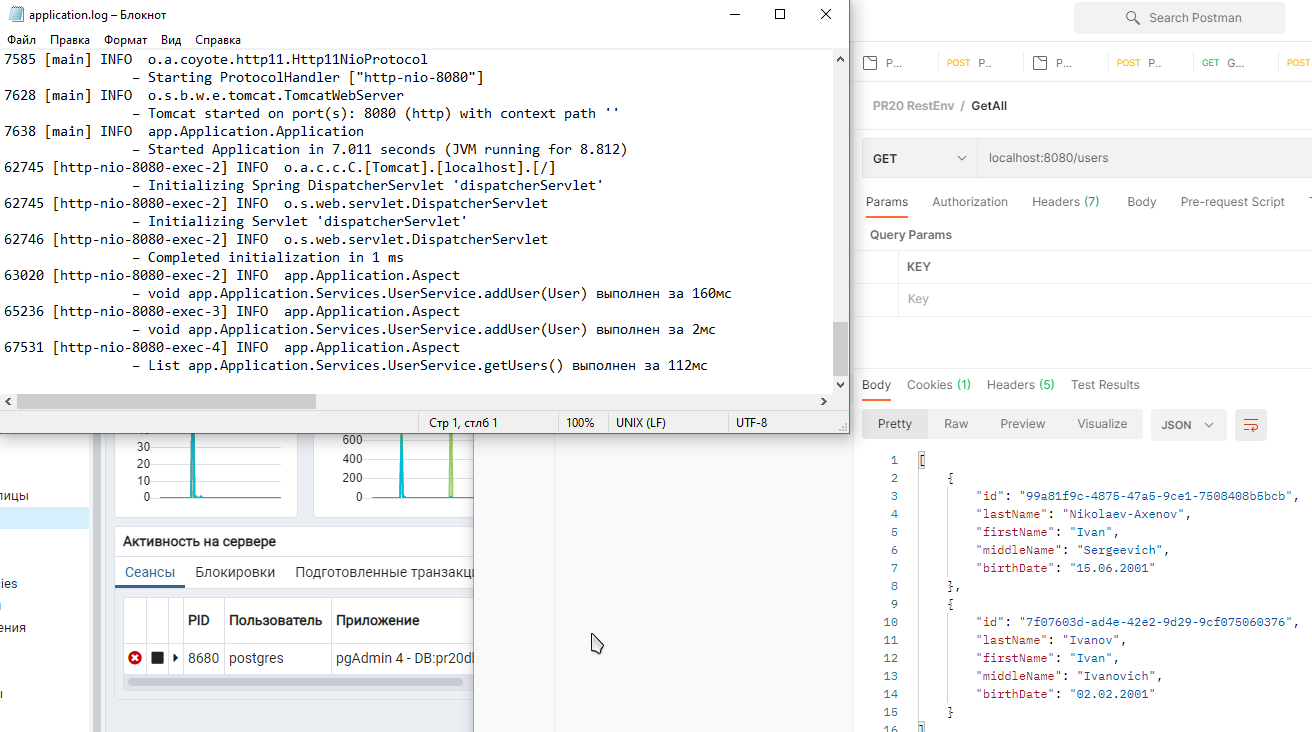


Рисунок 20.3 – Демонстрация работы программы

Практическая работа №21

***Цель работы***

Тема: Проксирование. Аннотация Transactional. Аннотация Async.

Постановка задачи: Для приложения из предыдущего задания пометить все классы сервисов, в которых происходит взаимодействие с базой данных, как Transactional. Добавить отправку информации о сохранении каждого объекта по электронной почте, создав отдельный класс EmailService с асинхронными методами отправки сообщений. Для асинхронности методов используйте аннотацию Async.

***Листинг программы***

*User.java*

package app.Application.Classes;

import com.sun.istack.NotNull;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.\*;

@Entity

@Table(name = "users")

public class User implements Serializable {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "last\_name")

@NotNull

private String lastName;

@Column(name = "first\_name")

@NotNull

private String firstName;

@Column(name = "middle\_name")

@NotNull

private String middleName;

@Column(name = "birth\_date")

@NotNull

private String birthDate;

@OneToMany(mappedBy = "user")

private List<Post> posts = new ArrayList<>();

public User() {

}

public User(String lastName, String firstName, String middleName, String birthDate) {

this.lastName = lastName;

this.firstName = firstName;

this.middleName = middleName;

this.birthDate = birthDate;

}

public UUID getId() {

return id;

}

public String getLastName() {

return lastName;

}

public String getFirstName() {

return firstName;

}

public String getMiddleName() {

return middleName;

}

public String getBirthDate() {

return birthDate;

}

@Override

public String toString() {

return "Пользователь #" + id + " " + lastName + " " + firstName + " " + middleName + ", день рождения: " + birthDate;

}

}

*Post.java*

package app.Application.Classes;

import com.sun.istack.NotNull;

import org.hibernate.annotations.CreationTimestamp;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.time.LocalDateTime;

import java.util.UUID;

@Entity

@Table(name = "posts")

public class Post {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "text")

@NotNull

private String text;

@CreationTimestamp

@Column(name = "creation\_date")

private LocalDateTime creationDate;

@ManyToOne

private User user;

public Post() {

}

public Post(String text) {

this.text = text;

}

public UUID getId() {

return id;

}

public String getText() {

return text;

}

public LocalDateTime getCreationDate() {

return creationDate;

}

public User getUser() {

return user;

}

}

*UserRepository.java*

package app.Application.Interfaces;

import app.Application.Classes.User;

import com.sun.istack.NotNull;

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

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.UUID;

@Repository("UserRepository")

public interface UserRepository extends JpaRepository<User,Long> {

List<User> findAllByFirstName(String firstName);

List<User> findAllByLastName(String lastName);

@NotNull List<User> findAll();

void deleteById(UUID id);

}

*PostRepository.java*

package app.Application.Interfaces;

import app.Application.Classes.Post;

import com.sun.istack.NotNull;

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

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.UUID;

@Repository("PostRepository")

public interface PostRepository extends JpaRepository<Post,Long> {

Post findById(UUID id);

@NotNull List<Post> findAll();

void deleteById(UUID id);

}

*UserService.java*

package app.Application.Services;

import app.Application.Classes.User;

import app.Application.Interfaces.UserRepository;

import lombok.extern.slf4j.Slf4j;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

@Slf4j

public class UserService {

@Autowired

private final UserRepository userRepository;

public UserService(UserRepository userRepository) {

this.userRepository = userRepository;

}

public void addUser(User user) {

userRepository.save(user);

}

public List<User> getUsers() {

return userRepository.findAll();

}

public void deleteUser(UUID id) {

userRepository.deleteById(id);

}

public List<User> getByFirstName(String firstName) {

return userRepository.findAllByFirstName(firstName);

}

public List<User> getByLastName(String lastName) {

return userRepository.findAllByLastName(lastName);

}

}

*PostService.java*

package app.Application.Services;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Interfaces.PostRepository;

import lombok.extern.slf4j.Slf4j;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

@Slf4j

public class PostService {

@Autowired

private final PostRepository postRepository;

public PostService(PostRepository postRepository) {

this.postRepository = postRepository;

}

public void addPost(Post post) {

postRepository.save(post);

}

public List<Post> getPosts() {

return postRepository.findAll();

}

public void deletePost(UUID id) {

postRepository.deleteById(id);

}

public User getUserByPost(UUID id) {

return postRepository.findById(id).getUser();

}

}

*UserController.java*

package app.Application.Controllers;

import app.Application.Classes.User;

import app.Application.EmailService;

import app.Application.Services.UserService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class UserController {

@Autowired

private UserService userService;

@Autowired

private EmailService emailService;

@PostMapping("/users")

public void addUser(@RequestBody User user) {

userService.addUser(user);

}

@GetMapping("/users")

public List<User> getAll() {

emailService.SendEmail();

return userService.getUsers();

}

@DeleteMapping("/user/{id}")

public void delete(@PathVariable UUID id) {

userService.deleteUser(id);

}

@GetMapping("/getUserByFirstName/{firstName}")

public List<User> getByFirstName(@PathVariable String firstName){

return userService.getByFirstName(firstName);

}

@GetMapping("/getUserByLastName/{lastName}")

public List<User> getByLastName(@PathVariable String lastName){

return userService.getByLastName(lastName);

}

}

*PostController.java*

package app.Application.Controllers;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Services.PostService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class PostController {

@Autowired

private PostService postService;

@PostMapping("/posts")

public void addPost(@RequestBody Post post) {

postService.addPost(post);

}

@GetMapping("/posts")

public List<Post> getAll() {

return postService.getPosts();

}

@DeleteMapping("/post/{id}")

public void delete(@PathVariable UUID id) {

postService.deletePost(id);

}

@GetMapping(value = "/post/{id}/user")

public @ResponseBody

User getGame(@PathVariable("id") UUID id) {

return postService.getUserByPost(id);

}

}

*Config.java*

package app.Application.Configuration;

import app.Application.EmailService;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.EnableAspectJAutoProxy;

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

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.mail.javamail.JavaMailSenderImpl;

import org.springframework.scheduling.annotation.EnableAsync;

import java.util.Properties;

@Configuration

@EnableAspectJAutoProxy

@EnableJpaRepositories(basePackages = {"app.Application"})

@EnableAsync

public class Config {

@Bean

public JavaMailSender getJavaMailSender() {

JavaMailSenderImpl mailSender = new JavaMailSenderImpl();

mailSender.setHost("smtp.mail.ru");

mailSender.setPort(465);

mailSender.setUsername("lorememail@bk.ru");

mailSender.setPassword("secret");

Properties props = mailSender.getJavaMailProperties();

props.put("mail.transport.protocol", "smtps");

props.put("mail.smtp.auth", "true");

props.put("smtp.ssl.enable", "true");

props.put("mail.smtp.starttls.enable", "true");

props.put("mail.debug", "true");

return mailSender;

}

@Bean

public EmailService getEmailService(){

return new EmailService();

}

}

*Aspect.java*

package app.Application;

import lombok.extern.slf4j.Slf4j;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Pointcut;

import org.springframework.stereotype.Component;

import java.util.logging.Logger;

@Slf4j

@Component

@org.aspectj.lang.annotation.Aspect

public class Aspect {

private Logger log = Logger.getLogger(Aspect.class.getName());

@Around("allServiceMethods()")

public Object logExecutionTime (ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.currentTimeMillis();

Object proceed = joinPoint.proceed();

long executionTime = System.currentTimeMillis() - start;

log.info(joinPoint.getSignature() + " выполнен за " + executionTime + "мс");

return proceed;

}

@Pointcut("within(app.Application.Services.\*)")

public void allServiceMethods() {}

}

*EmailService.java*

package app.Application;

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

import org.springframework.mail.SimpleMailMessage;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.scheduling.annotation.Async;

public class EmailService {

@Autowired

public JavaMailSender emailSender;

@Async

public void SendEmail(){

SimpleMailMessage message = new SimpleMailMessage();

message.setFrom("lorememail@bk.ru");

message.setTo("ghost777t@ya.ru");

message.setSubject("Test email message");

message.setText("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla feugiat eget sapien sed lacinia.");

this.emailSender.send(message);

System.out.println("Email successfully sent!");

}

}

***Результат выполнения программы***

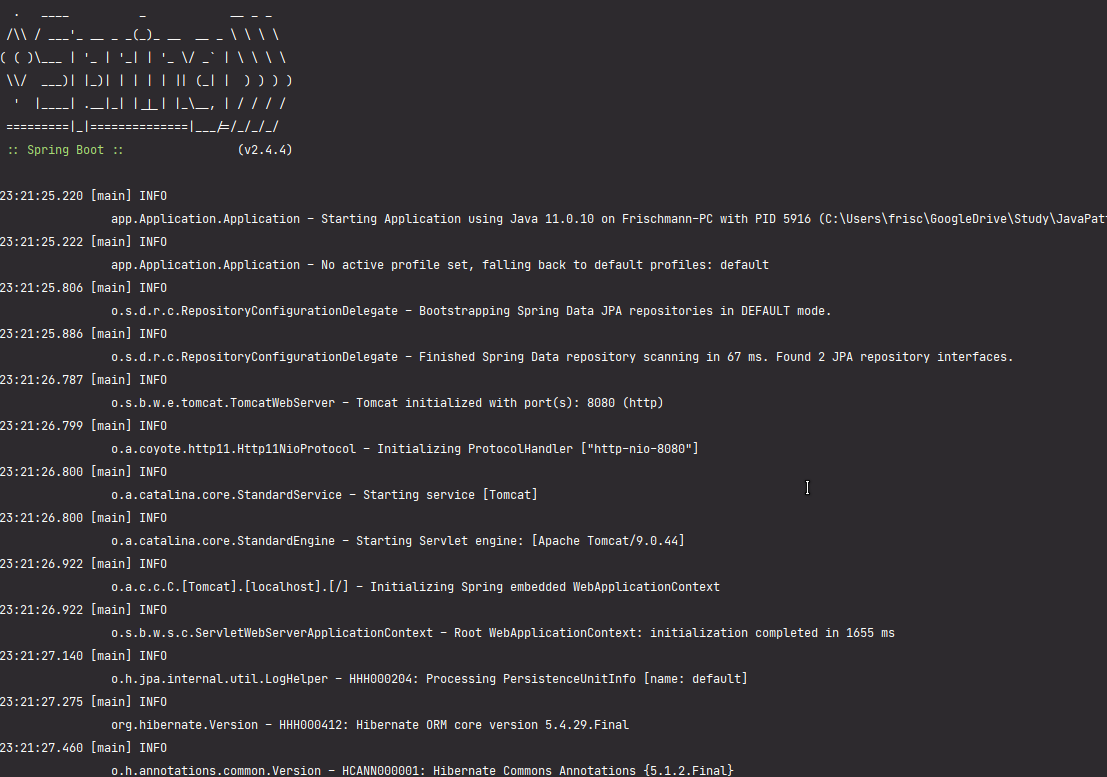


Рисунок 21.1 – Демонстрация работы программы

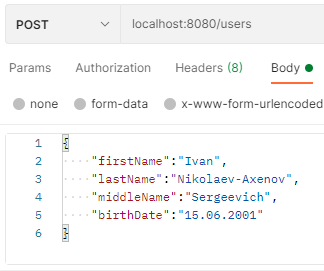


Рисунок 21.2 – Демонстрация работы программы

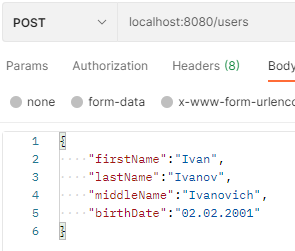


Рисунок 21.3 – Демонстрация работы программы

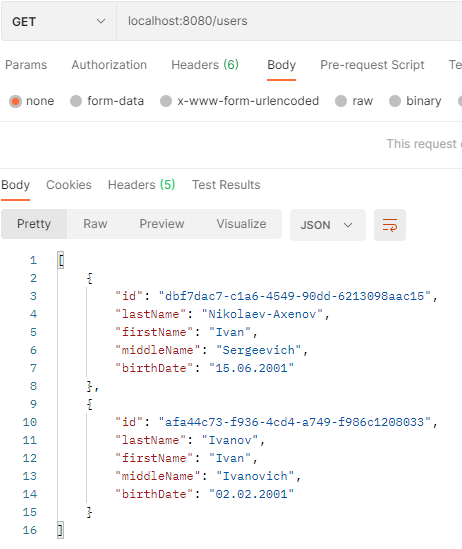


Рисунок 21.4 – Демонстрация работы программы

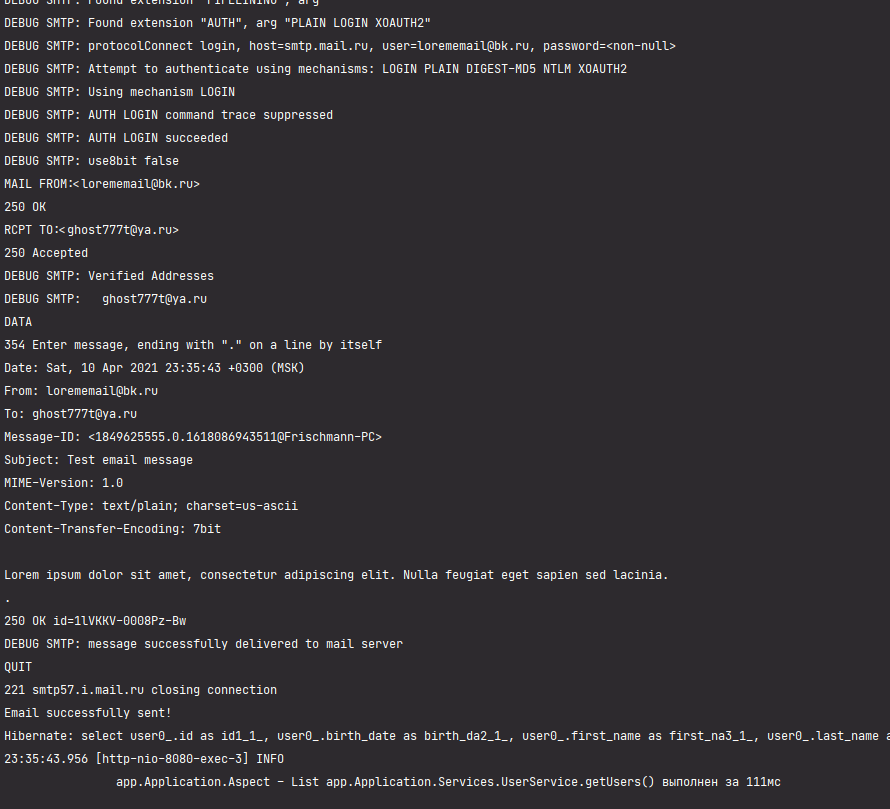


Рисунок 21.5 – Демонстрация работы программы

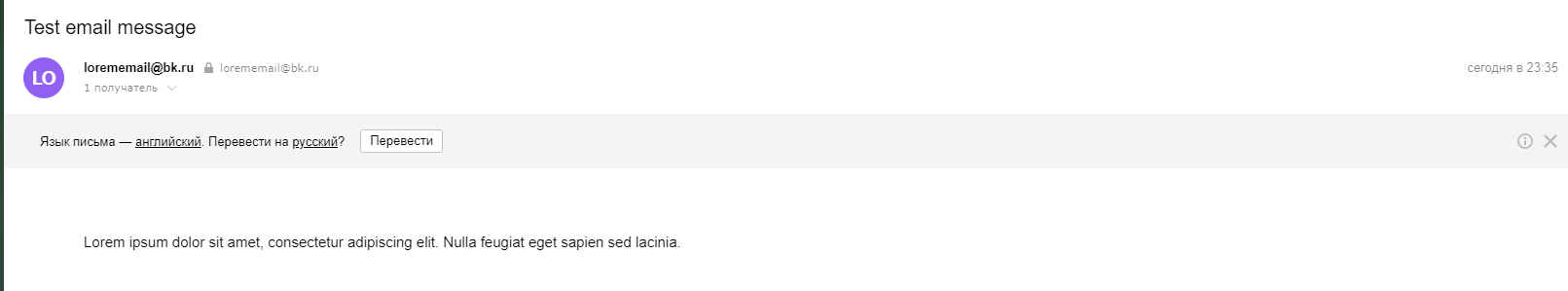


Рисунок 21.6 – Демонстрация работы программы

Практическая работа №22

***Цель работы***

Тема: Планирование заданий. Scheduler в Spring.

Постановка задачи: Для приложения из предыдущего задания создать класс-сервис с методом, который будет вызываться каждые 30 минут и очищать определённую директорию, а затем создавать по файлу для каждой из сущностей и загружать туда все данные из базы данных. Также добавить возможность вызывать данный метод с использованием Java Management Extensions (JMX).

***Листинг программы***

*User.java*

package app.Application.Classes;

import com.sun.istack.NotNull;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.\*;

@Entity

@Table(name = "users")

public class User implements Serializable {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "last\_name")

@NotNull

private String lastName;

@Column(name = "first\_name")

@NotNull

private String firstName;

@Column(name = "middle\_name")

@NotNull

private String middleName;

@Column(name = "birth\_date")

@NotNull

private String birthDate;

@OneToMany(mappedBy = "user")

private List<Post> posts = new ArrayList<>();

public User() {

}

public User(String lastName, String firstName, String middleName, String birthDate) {

this.lastName = lastName;

this.firstName = firstName;

this.middleName = middleName;

this.birthDate = birthDate;

}

public UUID getId() {

return id;

}

public String getLastName() {

return lastName;

}

public String getFirstName() {

return firstName;

}

public String getMiddleName() {

return middleName;

}

public String getBirthDate() {

return birthDate;

}

@Override

public String toString() {

return "User{" +

"id=" + id +

", lastName='" + lastName + '\'' +

", firstName='" + firstName + '\'' +

", middleName='" + middleName + '\'' +

", birthDate='" + birthDate + '\'' +

'}';

}

}

*Post.java*

package app.Application.Classes;

import com.sun.istack.NotNull;

import org.hibernate.annotations.CreationTimestamp;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.time.LocalDateTime;

import java.util.UUID;

@Entity

@Table(name = "posts")

public class Post {

@Id

@GeneratedValue(generator = "UUID")

@GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")

@Column(name = "id", updatable = false, nullable = false)

private UUID id;

@Column(name = "text")

@NotNull

private String text;

@CreationTimestamp

@Column(name = "creation\_date")

private LocalDateTime creationDate;

@ManyToOne

private User user;

public Post() {

}

public Post(User user, String text) {

this.text = text;

this.user = user;

}

public UUID getId() {

return id;

}

public String getText() {

return text;

}

public LocalDateTime getCreationDate() {

return creationDate;

}

public User getUser() {

return user;

}

@Override

public String toString() {

return "Post{" +

"id=" + id +

", text='" + text + '\'' +

", creationDate=" + creationDate +

", user=" + user +

'}';

}

}

*UserRepository.java*

package app.Application.Interfaces;

import app.Application.Classes.User;

import com.sun.istack.NotNull;

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

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.UUID;

@Repository("UserRepository")

public interface UserRepository extends JpaRepository<User,Long> {

List<User> findAllByFirstName(String firstName);

List<User> findAllByLastName(String lastName);

@NotNull List<User> findAll();

void deleteById(UUID id);

}

*PostRepository.java*

package app.Application.Interfaces;

import app.Application.Classes.Post;

import com.sun.istack.NotNull;

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

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.UUID;

@Repository("PostRepository")

public interface PostRepository extends JpaRepository<Post,Long> {

Post findById(UUID id);

List<Post> findAll();

void deleteById(UUID id);

}

*UserService.java*

package app.Application.Services;

import app.Application.Classes.User;

import app.Application.Interfaces.UserRepository;

import lombok.extern.slf4j.Slf4j;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

@Slf4j

public class UserService {

@Autowired

private final UserRepository userRepository;

public UserService(UserRepository userRepository) {

this.userRepository = userRepository;

}

public void addUser(User user) {

userRepository.save(user);

}

public List<User> getUsers() {

return userRepository.findAll();

}

public void deleteUser(UUID id) {

userRepository.deleteById(id);

}

public List<User> getByFirstName(String firstName) {

return userRepository.findAllByFirstName(firstName);

}

public List<User> getByLastName(String lastName) {

return userRepository.findAllByLastName(lastName);

}

}

*PostService.java*

package app.Application.Services;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Interfaces.PostRepository;

import lombok.extern.slf4j.Slf4j;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import java.util.List;

import java.util.UUID;

@Service

@Slf4j

public class PostService {

@Autowired

private final PostRepository postRepository;

public PostService(PostRepository postRepository) {

this.postRepository = postRepository;

}

public void addPost(Post post) {

postRepository.save(post);

}

public List<Post> getPosts() {

return postRepository.findAll();

}

public void deletePost(UUID id) {

postRepository.deleteById(id);

}

public User getUserByPost(UUID id) {

return postRepository.findById(id).getUser();

}

}

*UserController.java*

package app.Application.Controllers;

import app.Application.Classes.User;

import app.Application.Services.EmailService;

import app.Application.Services.UserService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class UserController {

@Autowired

private UserService userService;

@Autowired

private EmailService emailService;

@PostMapping("/users")

public void addUser(@RequestBody User user) {

userService.addUser(user);

}

@GetMapping("/users")

public List<User> getAll() {

//emailService.SendEmail();

return userService.getUsers();

}

@DeleteMapping("/user/{id}")

public void delete(@PathVariable UUID id) {

userService.deleteUser(id);

}

@GetMapping("/getUserByFirstName/{firstName}")

public List<User> getByFirstName(@PathVariable String firstName){

return userService.getByFirstName(firstName);

}

@GetMapping("/getUserByLastName/{lastName}")

public List<User> getByLastName(@PathVariable String lastName){

return userService.getByLastName(lastName);

}

}

*PostController.java*

package app.Application.Controllers;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Services.PostService;

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

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

import java.util.List;

import java.util.UUID;

@RestController

public class PostController {

@Autowired

private PostService postService;

@PostMapping("/posts")

public void addPost(@RequestBody Post post) {

postService.addPost(post);

}

@GetMapping("/posts")

public List<Post> getAll() {

return postService.getPosts();

}

@DeleteMapping("/post/{id}")

public void delete(@PathVariable UUID id) {

postService.deletePost(id);

}

@GetMapping(value = "/post/{id}/user")

public @ResponseBody

User getGame(@PathVariable("id") UUID id) {

return postService.getUserByPost(id);

}

}

*Config.java*

package app.Application.Configuration;

import app.Application.Services.EmailService;

import app.Application.Services.Schedule;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.EnableAspectJAutoProxy;

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

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.mail.javamail.JavaMailSenderImpl;

import org.springframework.scheduling.annotation.EnableAsync;

import org.springframework.scheduling.annotation.EnableScheduling;

import java.util.Properties;

@Configuration

@EnableAspectJAutoProxy

@EnableJpaRepositories(basePackages = {"app.Application"})

@EnableAsync

@EnableScheduling

public class Config {

@Bean

public JavaMailSender getJavaMailSender() {

JavaMailSenderImpl mailSender = new JavaMailSenderImpl();

mailSender.setHost("smtp.mail.ru");

mailSender.setPort(465);

mailSender.setUsername("lorememail@bk.ru");

mailSender.setPassword("secret");

Properties props = mailSender.getJavaMailProperties();

props.put("mail.transport.protocol", "smtps");

props.put("mail.smtp.auth", "true");

props.put("smtp.ssl.enable", "true");

props.put("mail.smtp.starttls.enable", "true");

props.put("mail.debug", "true");

return mailSender;

}

@Bean

public EmailService getEmailService(){

return new EmailService();

}

}

*Aspect.java*

package app.Application;

import lombok.extern.slf4j.Slf4j;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Pointcut;

import org.springframework.stereotype.Component;

import java.util.logging.Logger;

@Slf4j

@Component

@org.aspectj.lang.annotation.Aspect

public class Aspect {

private Logger log = Logger.getLogger(Aspect.class.getName());

@Around("allServiceMethods()")

public Object logExecutionTime (ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.currentTimeMillis();

Object proceed = joinPoint.proceed();

long executionTime = System.currentTimeMillis() - start;

log.info(joinPoint.getSignature() + " выполнен за " + executionTime + "мс");

return proceed;

}

@Pointcut("within(app.Application.Services.\*)")

public void allServiceMethods() {}

}

*EmailService.java*

package app.Application.Services;

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

import org.springframework.mail.SimpleMailMessage;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.scheduling.annotation.Async;

public class EmailService {

@Autowired

public JavaMailSender emailSender;

@Async

public void SendEmail(){

SimpleMailMessage message = new SimpleMailMessage();

message.setFrom("lorememail@bk.ru");

message.setTo("ghost777t@ya.ru");

message.setSubject("Test email message");

message.setText("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla feugiat eget sapien sed lacinia.");

this.emailSender.send(message);

System.out.println("Email successfully sent!");

}

}

*ScheduleMXBean.java*

package app.Application;

import java.io.IOException;

import app.Application.Interfaces.UserRepository;

import app.Application.Interfaces.PostRepository;

public interface ScheduleMXBean {

void doScheduledTask() throws IOException;

}

*Schedule.java*

package app.Application.Services;

import app.Application.Classes.Post;

import app.Application.Classes.User;

import app.Application.Controllers.PostController;

import app.Application.Controllers.UserController;

import app.Application.Interfaces.PostRepository;

import app.Application.Interfaces.UserRepository;

import app.Application.ScheduleMXBean;

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

import org.springframework.jmx.export.annotation.ManagedOperation;

import org.springframework.scheduling.annotation.Scheduled;

import org.springframework.stereotype.Service;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.List;

import java.util.Objects;

@Service

public class Schedule implements ScheduleMXBean {

@Autowired

private final UserRepository userRepository;

@Autowired

private final PostRepository postRepository;

public Schedule(UserRepository userRepository, PostRepository postRepository) {

this.userRepository = userRepository;

this.postRepository = postRepository;

}

private Boolean isEmpty(final File file) {

return (file.isDirectory() && (file.list().length > 0));

}

@ManagedOperation

@Scheduled(cron = "0 0/2 \* \* \* \*")

public void doScheduledTask() throws IOException {

if(isEmpty(new File("C:\\Users\\frisc\\GoogleDrive\\Study\\JavaPatterns\\Практическая работа №22\\testDirectory"))){

for (File myFile : new File("C:\\Users\\frisc\\GoogleDrive\\Study\\JavaPatterns\\Практическая работа №22\\testDirectory").listFiles()) {

if (myFile.isFile()) myFile.delete();

}

}

List <Post> posts = postRepository.findAll();

List <User> users = userRepository.findAll();

for (int i = 0; i < users.size(); i++) {

File user = new File("C:\\Users\\frisc\\GoogleDrive\\Study\\JavaPatterns\\Практическая работа №22\\testDirectory\\user\_" + i + ".txt");

FileWriter writer = new FileWriter(user, true);

System.out.println(users.get(i).toString());

writer.write(users.get(i).toString());

writer.close();

}

for (int i = 0; i < posts.size(); i++) {

File post = new File("C:\\Users\\frisc\\GoogleDrive\\Study\\JavaPatterns\\Практическая работа №22\\testDirectory\\post\_" + i + ".txt");

FileWriter writer = new FileWriter(post, true);

writer.write(posts.get(i).toString());

writer.close();

}

}

}

***Результат выполнения программы***

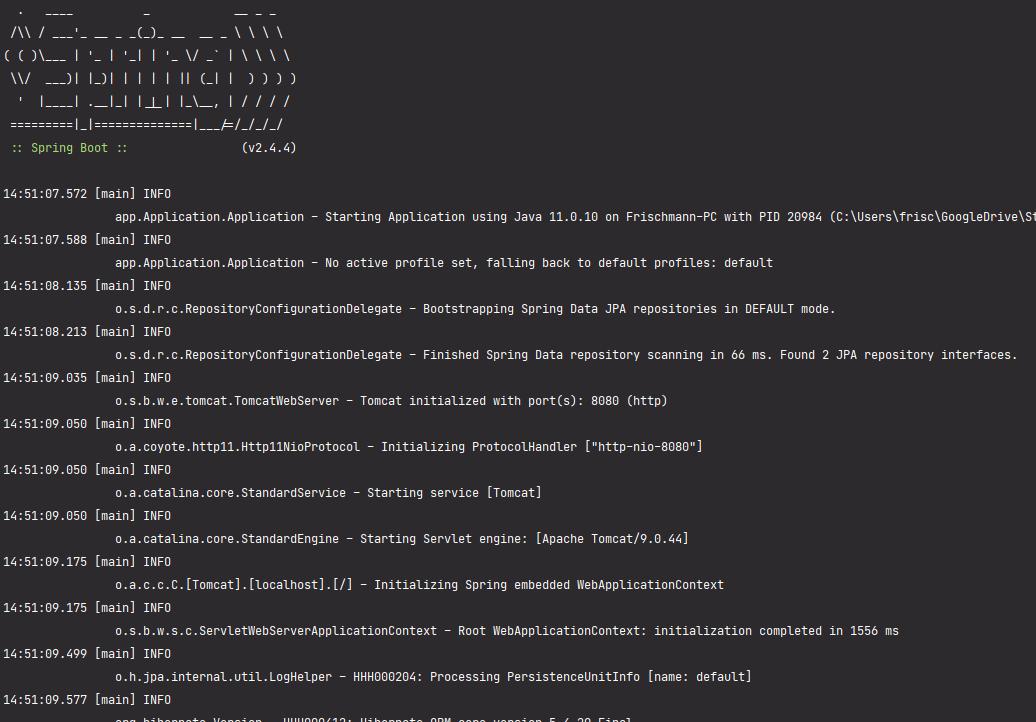


Рисунок 22.1 – Демонстрация работы программы

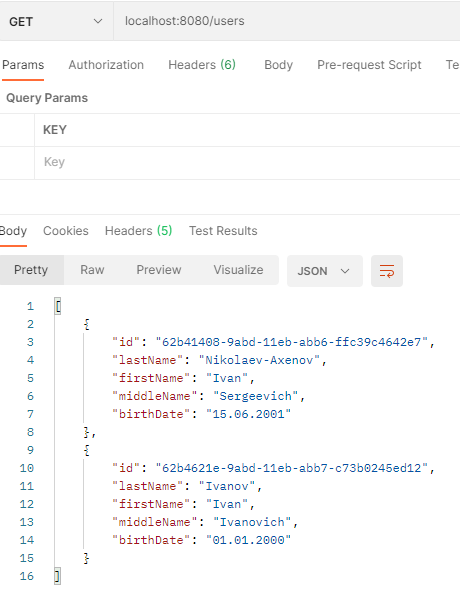


Рисунок 22.2 – Демонстрация работы программы

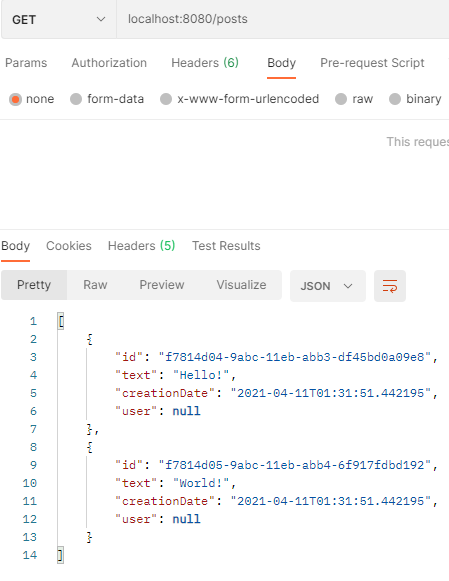


Рисунок 22.3 – Демонстрация работы программы

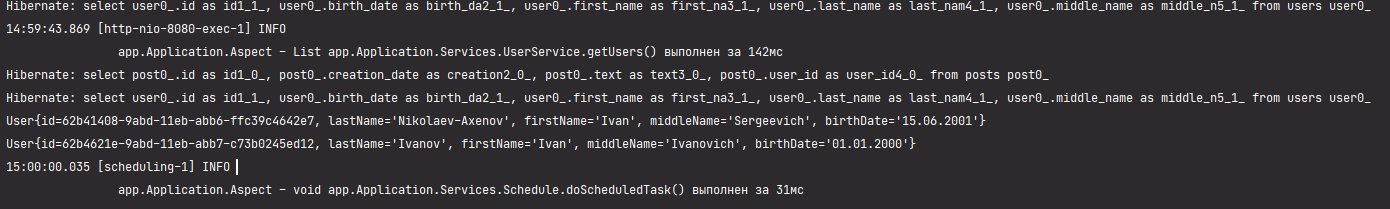


Рисунок 22.4 – Демонстрация работы программы

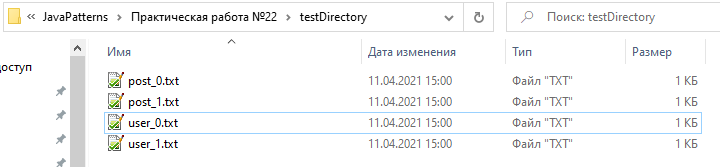


Рисунок 22.5 – Демонстрация работы программы

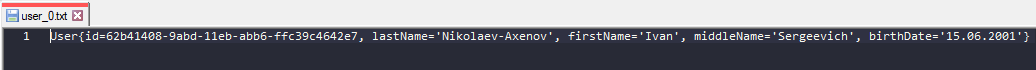


Рисунок 22.6 – Демонстрация работы программы

Вывод

В ходе выполнения данных практических работ были получены навыки работы с основными технологиями, необходимыми для создания клиент-серверных приложений. Также были получены навыки работы с фреймворком Spring.

Список использованных источников

1. Стелтинг С., Маасен О. Применение шаблонов Java. Библиотека профессионала.: Пер. с англ. — М.: Издательский дом "Вильяме", 2002. — 576 с.: ил. — Парал. тит. англ.
2. Functional Interfaces in Java: Fundamentals and Examples 1st ed. Edition, Kindle Edition [Электронный ресурс]. URL: https://www.amazon.com/Functional-Interfaces-Java-Fundamentals-Examples-ebook/dp/B07NRHQSCW (дата обращения: 29.01.21). Заголовок с экрана.
3. Hibernate Search 6.0.0.Final: Reference Documentation [Электронный ресурс]. URL: https://docs.jboss.org/hibernate/stable/search/reference/en-US/html\_single/ (дата обращения: 29.01.21). Заголовок с экрана.
4. Паттерны проектирования на Java. Каталог Java-примеров. [Электронный ресурс]. URL: https://refactoring.guru/ru/design-patterns/java (дата обращения: 29.01.21). Заголовок с экрана.
5. Руководство по Spring [Электронный ресурс]. URL: https://proselyte.net/tutorials/spring-tutorial-full-version/ (дата обращения: 29.01.21). Заголовок с экрана.
6. The Reactive Manifesto [Электронный ресурс]. URL: https://www.reactivemanifesto.org/ (дата обращения: 29.01.21). Заголовок с экрана.
7. Spring Framework Documentation [Электронный ресурс]. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/web.html (дата обращения: 29.01.21). Заголовок с экрана.
8. Hibernate Search 6.0.0. Final: Reference Documentation [Электронный ресурс]. URL: https://docs.jboss.org/hibernate/stable/search/reference/en-US/html\_single/ (дата обращения: 29.01.21). Заголовок с экрана.