



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 №1 - 24

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 №1	4
Практическая работа №2	6
Практическая работа №3	9
Практическая работа №4	15
Практическая работа №5	17
Практическая работа №6	19
Практическая работа №7	31
Практическая работа №8	36
Практическая работа №9	42
Практическая работа №10	46
Практическая работа №11	49
Практическая работа №12	52
Практическая работа №13	55
Практическая работа №14	57
Практическая работа №15	67
Практическая работа №16	78
Практическая работа №17	88
Практическая работа №18	99
Практическая работа №19	109
Практическая работа №20	119
Практическая работа №21	129
Практическая работа №22	141
Практическая работа №23	154

Практическая работа №24	165
Вывод.....	177
Список использованных источников	177

Практическая работа №1

Цель работы

Тема: Знакомство со встроенными функциональными интерфейсами Java. Возможности Java 8. Лямбда-выражения. Области действия, замыкания. Предикаты. Функции. Компараторы.

Постановка задачи: Имплементировать интерфейс Comparator, сравнивающий двух студентов по набранным за семестр баллов.

Листинг программы

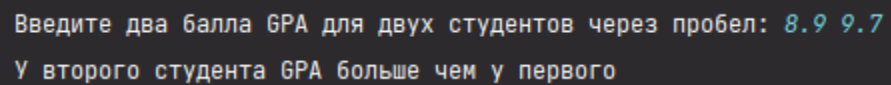
Student.java

```
public class Student {  
    private double gpa = 0;  
  
    public Student(double gpa) {  
        this.gpa = gpa;  
    }  
    public double getGPA() {  
        return gpa;  
    }  
    public void setGPA(double gpa) {  
        this.gpa = gpa;  
    }  
}
```

startStudent.java

```
import java.util.Comparator;  
import java.util.Scanner;  
  
public class startStudent {  
    public static void main(String[] args) {  
        System.out.print("Введите два балла GPA для двух студентов через пробел: ");  
        Scanner sc = new Scanner(System.in);  
        String[] arr = sc.nextLine().split(" ");  
  
        Student a = new Student(Double.parseDouble(arr[0]));  
        Student b = new Student(Double.parseDouble(arr[1]));  
  
        Comparator<Student> compGPA = (o1, o2) -> Double.compare(o1.getGPA(),  
o2.getGPA());  
        double result = compGPA.compare(a, b);  
  
        if(result >= 1)  
            System.out.println("У первого студента GPA больше чем у второго.");  
        else if(result == 0)  
            System.out.println("Оба студента имеют одинаковые баллы.");  
        else  
            System.out.println("У второго студента GPA больше чем у первого");  
    }  
}
```

Результат выполнения программы



Введите два балла GPA для двух студентов через пробел: 8.9 9.7
У второго студента GPA больше чем у первого

Рисунок 1.1 – Демонстрация работы программы

Практическая работа №2

Цель работы

Тема: Работа со Stream API в Java 8.

Постановка задачи: Уменьшение веса каждого объекта на 5, фильтрация по дате рождения меньшей, чем 3 февраля 1999, конкатенация фамилий в строку через пробел.

Листинг программы

Human.java

```
import java.time.LocalDate;

public class Human {
    private int age, weight;
    private String firstName, lastName;
    private LocalDate birthDate;

    public Human(int age, String firstName, String lastName, LocalDate birthDate, int
weight) {
        this.age = age;
        this.firstName = firstName;
        this.lastName = lastName;
        this.birthDate = birthDate;
        this.weight = weight;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }
}
```

```

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public LocalDate getBirthDate() {
        return birthDate;
    }

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

    @Override
    public String toString() {
        return firstName + ' ' + lastName + ", " + age + " years, " + birthDate + ", " +
weight + "kg";
    }
}

```

HumanBuilder.java

```

import java.time.LocalDate;
import java.util.*;
import java.util.stream.Stream;

public class HumanBuilder {
    private String[] firstNames = {"James", "Mary", "John", "Patricia", "Robert",
"Jennifer", "Michael", "Linda", "William", "Elizabeth", "David", "Barbara", "Richard",
"Susan", "Joseph", "Jessica", "Thomas", "Sarah", "Charles", "Karen", "Christopher",
"Nancy", "Daniel", "Lisa", "Matthew", "Margaret", "Anthony", "Betty", "Donald",
"Sandra", "Mark", "Ashley", "Paul", "Dorothy", "Steven", "Kimberly", "Andrew", "Emily",
"Kenneth", "Donna", "Joshua", "Michelle", "Kevin", "Carol", "Brian", "Amanda"};
    private String[] lastNames = {"Smith", "Johnson", "Williams", "Brown", "Jones",
"Garcia", "Miller", "Davis", "Rodriguez", "Martinez", "Hernandez", "Lopez", "Gonzalez",
"Wilson", "Anderson", "Thomas", "Taylor", "Moore", "Jackson", "Martin", "Lee", "Perez",
"Thompson", "White", "Harris", "Sanchez", "Clark", "Ramirez", "Lewis", "Robinson"};

    private List<Human> generateList(int size) {
        List<Human> result = new ArrayList<>();
        Random rand = new Random();

        for(int i = 0; i < size; i++) {
            LocalDate ld = LocalDate.of(rand.nextInt(70) + 1950, rand.nextInt(12) + 1,
rand.nextInt(27) + 1);
            result.add(new Human(rand.nextInt(99),
firstNames[rand.nextInt(firstNames.length)], lastNames[rand.nextInt(lastNames.length)],
ld, rand.nextInt(100) + 20));
        }

        return result;
    }

    public void humanTask(int size) {
        List<Human> harr = generateList(size);

        System.out.println("Original list:");
        harr.forEach(System.out::println);

        System.out.println("\nList after applying stream methods:");
    }
}

```

```

Stream<Human> s1 = harr.stream();
Stream<Human> s2 = harr.stream();
Stream<Human> s3 = harr.stream();

System.out.println("\nFirst stream: ");
s1.peek(o -> o.setWeight(o.getWeight() - 5)).forEach(System.out::println);

System.out.println("\nSecond stream: ");
s2.filter(o -> o.getBirthDate().isBefore(LocalDate.of(1999, 2,
3))).forEach(System.out::println);

System.out.println("\nThird stream: ");
s3.map(o -> o.getLastName() + ' ').forEach(System.out::print);
}
}

```

Main.java

```

public class main {
    public static void main(String[] args) {
        HumanBuilder hb = new HumanBuilder();
        hb.humanTask(10);
    }
}

```

Результат выполнения программы

```

Original list:
Susan Clark, 94 years, 2004-03-22, 91kg
John Gonzalez, 90 years, 1990-07-21, 85kg
Donald Perez, 9 years, 1975-10-03, 61kg
Steven Taylor, 78 years, 1970-06-11, 51kg
Joseph Williams, 80 years, 2011-03-09, 89kg

List after applying stream methods:

First stream:
Susan Clark, 94 years, 2004-03-22, 86kg
John Gonzalez, 90 years, 1990-07-21, 80kg
Donald Perez, 9 years, 1975-10-03, 56kg
Steven Taylor, 78 years, 1970-06-11, 46kg
Joseph Williams, 80 years, 2011-03-09, 84kg

Second stream:
John Gonzalez, 90 years, 1990-07-21, 80kg
Donald Perez, 9 years, 1975-10-03, 56kg
Steven Taylor, 78 years, 1970-06-11, 46kg

Third stream:
Clark Gonzalez Perez Taylor Williams
Process finished with exit code 0

```

Рисунок 2.1 – Демонстрация работы программы

Практическая работа №3

Цель работы

Тема: Знакомство с конкурентным программированием в Java. Потокобезопасность, ключевое слово synchronized, мьютексы, семафоры, мониторы, барьеры.

Постановка задачи: Создать свои потокобезопасные имплементации интерфейсов в соответствии с вариантом индивидуального задания. Set с использованием Lock, Map с использованием Semaphore.

Листинг программы

MapSemaphore.java

```
import java.util.Collection;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.Semaphore;

public class MapSemaphore<K, V> extends HashMap<K, V> implements Map<K, V> {
    private static final Semaphore sem = new Semaphore(1);

    @Override
    public int size() {
        return super.size();
    }

    @Override
    public boolean isEmpty() {
        return super.isEmpty();
    }

    @Override
    public boolean containsKey(Object key) {
        boolean b = false;
        try {
            sem.acquire();
            b = super.containsKey(key);
            sem.release();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return b;
    }

    @Override
    public boolean containsValue(Object value) {
        boolean b = false;
        try {
            sem.acquire();
            b = super.containsValue(value);
            sem.release();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```

        e.printStackTrace();
    }
    return b;
}

@Override
public V get(Object key) {
    V k = null;
    try {
        sem.acquire();
        k = super.get(key);
        sem.release();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return k;
}

@Override
public V put(K key, V value) {
    V k = null;
    try {
        sem.acquire();
        k = super.put(key, value);
        sem.release();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return k;
}

@Override
public V remove(Object key) {
    V k = null;
    try {
        sem.acquire();
        k = super.remove(key);
        sem.release();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return k;
}

@Override
public void putAll(Map<? extends K, ? extends V> m) {
    try {
        sem.acquire();
        super.putAll(m);
        sem.release();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

@Override
public void clear() {
    try {
        sem.acquire();
        super.clear();
        sem.release();
    } catch (InterruptedException e) {

```

```

        e.printStackTrace();
    }
}

@Override
public Set<K> keySet() {
    Set<K> k = null;
    try {
        sem.acquire();
        k = super.keySet();
        sem.release();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return k;
}

@Override
public Collection<V> values() {
    Collection<V> k = null;
    try {
        sem.acquire();
        k = super.values();
        sem.release();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return k;
}

@Override
public Set<Entry<K, V>> entrySet() {
    Set<Entry<K, V>> k = null;
    try {
        sem.acquire();
        k = super.entrySet();
        sem.release();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    assert k != null;
    return k;
}
}

```

SetLock.java

```

import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.Set;
import java.util.concurrent.locks.ReentrantLock;

public class SetLock<E> implements Set {
    private ArrayList<E> arr = new ArrayList<>();
    private static final ReentrantLock re = new ReentrantLock();

    @Override
    public boolean add(Object o) {
        re.lock();

```

```

        boolean b = arr.add((E) o);
        re.unlock();
        return b;
    }

    @Override
    public boolean remove(Object o) {
        re.lock();
        boolean b = arr.remove(o);
        re.unlock();
        return b;
    }

    @Override
    public boolean addAll(Collection c) {
        re.lock();
        boolean b = arr.addAll(c);
        re.unlock();
        return b;
    }

    @Override
    public void clear() {
        re.lock();
        arr.clear();
        re.unlock();
    }

    @Override
    public boolean removeAll(Collection c) {
        re.lock();
        boolean b = arr.removeAll(c);
        re.unlock();
        return b;
    }

    @Override
    public boolean retainAll(Collection c) {
        re.lock();
        boolean b = arr.retainAll(c);
        re.unlock();
        return b;
    }

    @Override
    public boolean containsAll(Collection c) {
        re.lock();
        boolean b = arr.containsAll(c);
        re.unlock();
        return b;
    }

    @Override
    public boolean contains(Object o) {
        re.lock();
        boolean b = arr.contains((E) o);
        re.unlock();
        return b;
    }

    @Override
    public int size() {

```

```

        return arr.size();
    }

    @Override
    public boolean isEmpty() {
        return arr.isEmpty();
    }

    @Override
    public Iterator iterator() {
        return arr.iterator();
    }

    @Override
    public Object[] toArray() {
        return arr.toArray();
    }

    @Override
    public Object[] toArray(Object[] a) {
        return arr.toArray(a);
    }

    @Override
    public String toString() {
        return "SetLock{" +
            "arr=" + arr +
            '}';
    }
}

```

TypesTester.java

```

public class TypesTester {
    static class t1 extends Thread {
        @Override
        public void run() {
            SetLock<Integer> sl = new SetLock<>();

            sl.add(5);
            sl.add(15);
            sl.add(25);

            System.out.println(sl);
            System.out.println(sl.contains(25));
            sl.remove(15);
            System.out.println(sl.size());
        }
    }

    static class t2 extends Thread {
        @Override
        public void run() {
            MapSemaphore<Integer, String> ms = new MapSemaphore<>();

            ms.put(1, "Ivan");
            ms.put(2, "Petr");
            ms.put(3, "Mikhail");

            System.out.println(ms);
        }
    }
}

```

```

        System.out.println(ms.values());
        System.out.println(ms.keySet());

        ms.remove(2);

        System.out.println(ms.size());
    }
}

public static void main(String[] args) {
    t1 th11 = new t1();
    t1 th12 = new t1();
    th11.start();
    th12.start();

    try {
        Thread.sleep(900);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    System.out.println("-----");

    t2 th21 = new t2();
    t2 th22 = new t2();
    th21.start();
    th22.start();
}
}

```

Результат выполнения программы

```

SetLock{arr=[5, 15, 25]}
true
2
SetLock{arr=[5, 15, 25]}
true
2
-----
{1=Ivan, 2=Petr, 3=Mikhail}
{1=Ivan, 2=Petr, 3=Mikhail}
[Ivan, Petr, Mikhail]
[Ivan, Petr, Mikhail]
[1, 2, 3]
[1, 2, 3]
2
2

```

Рисунок 3.1 – Демонстрация работы программы

Практическая работа №4

Цель работы

Тема: Работа с ExecutorService, CompletableFuture.

Постановка задачи: Реализовать собственную имплементацию ExecutorService с единственным параметром конструктора — количеством потоков.

Листинг программы

RandomWordThread.java

```
import java.util.Random;

public class RandomWordThread implements Runnable {
    private String[] words = {"capricious", "roasted", "check", "box", "day",
        "debonair", "coordinated", "observe", "beds", "jail", "wide-eyed", "anger",
        "attraction", "slimy", "thoughtless", "time", "drab", "pushy", "smiling", "helpful",
        "understood", "truck", "hobbies", "delight", "launch", "acoustic", "troubled", "thick",
        "cattle", "explode", "large", "melt", "release", "bashful", "hurried", "animal", "bite-
        sized", "kneel", "unaccountable", "squealing", "show", "drown", "telling", "aunt",
        "low", "superficial", "wave", "breath", "succeed", "complain"};
    private Random random = new Random();

    @Override
    public void run() {
        try {
            System.out.println(words[random.nextInt(words.length)]);
            Thread.sleep(random.nextInt(5001) + 500);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

ExecutorServiceHandler.java

```
import java.util.Random;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ExecutorServiceHandler {
    private ExecutorService exec;
    private Random random = new Random();

    public ExecutorServiceHandler(int number) {
        exec = Executors.newFixedThreadPool(number);

        int bound = random.nextInt(30) + 1;
        System.out.println("Запуск " + bound + " потоков, при возможных " + number + "
        потоках..\n");
    }
}
```

```

        for(int i = 0; i < bound; i++) {
            System.out.println("Запуск потока №" + (i + 1));
            exec.execute(new RandomWordThread());
        }
        exec.shutdown();
        System.out.println("Завершение работы потоков...");
    }
}

```

Main.java

```

public class Main {
    public static void main(String[] args) {
        ExecutorServiceHandler esh = new ExecutorServiceHandler(3);
    }
}

```

Результат выполнения программы

```

Запуск 12 потоков, при возможных 3 потоках..

Запуск потока №1
Запуск потока №2
Запуск потока №3
delight
Запуск потока №4
helpful
Запуск потока №5
Запуск потока №6
Запуск потока №7
Запуск потока №8
Запуск потока №9
Запуск потока №10
Запуск потока №11
animal
Запуск потока №12
Завершение работы потоков...
animal
cattle
succeed
explode
acoustic
complain
pushy
bashful
slimy

```

Рисунок 4.1 – Демонстрация работы программы

Практическая работа №5

Цель работы

Тема: Познакомиться с паттернами проектирования, их определением и классификацией. Обзор паттернов GoF. Паттерн Синглтон.

Постановка задачи: Реализовать паттерн Singleton как минимум 3-мя способами.

Листинг программы

FirstSingletonImpl.java

```
public class FirstSingletonImpl {
    private static FirstSingletonImpl instance;

    private FirstSingletonImpl() {
        System.out.println("FirstSingletonImpl");
    }

    public static FirstSingletonImpl getInstance() {
        if(instance == null)
            instance = new FirstSingletonImpl();
        return instance;
    }
}
```

SecondSingletonImpl.java

```
public class SecondSingletonImpl {
    private static SecondSingletonImpl instance = new SecondSingletonImpl();

    private SecondSingletonImpl() {
        System.out.println("SecondSingletonImpl");
    }

    public static SecondSingletonImpl getInstance() {
        return instance;
    }
}
```

ThirdSingletonImpl.java

```
public class ThirdSingletonImpl {
    private ThirdSingletonImpl() {
        System.out.println("ThirdSingletonImpl");
    }

    private static class ThirdSingletonImplHolder {
        private final static ThirdSingletonImpl instance = new ThirdSingletonImpl();
    }

    public static ThirdSingletonImpl getInstance() {
        return ThirdSingletonImplHolder.instance;
    }
}
```

```
}  
}
```

FourthSingletonImpl.java

```
public class FourthSingletonImpl {  
    private static volatile FourthSingletonImpl instance;  
  
    private FourthSingletonImpl() {  
        System.out.println("FourthSingletonImpl");  
    }  
  
    public static FourthSingletonImpl getInstance() {  
        if (instance == null)  
            synchronized (FourthSingletonImpl.class) {  
                if (instance == null)  
                    instance = new FourthSingletonImpl();  
            }  
        return instance;  
    }  
}
```

Main.java

```
public class Main {  
    public static void main(String[] args) {  
        FirstSingletonImpl.getInstance();  
        SecondSingletonImpl.getInstance();  
        ThirdSingletonImpl.getInstance();  
        FourthSingletonImpl.getInstance();  
    }  
}
```

Результат выполнения программы

```
FirstSingletonImpl  
SecondSingletonImpl  
ThirdSingletonImpl  
FourthSingletonImpl
```

Рисунок 5.1 – Демонстрация работы программы

Практическая работа №6

Цель работы

Тема: Знакомство с реализацией порождающих паттернов проектирования.

Постановка задачи: Написать реализацию паттернов «Фабричный метод», «Абстрактная фабрика», «Строитель», «Прототип».

Листинг программы

AbstractFactory:

AfricanGarden.java

```
package AbstractFactory;

public class AfricanGarden implements GardenFactory {
    public AfricanGarden() {
        System.out.println("Создан африканский сад!");
    }

    @Override
    public Tree plantTree() {
        return new Baobab();
    }

    @Override
    public Flower plantFlower() {
        return new Gloriosa();
    }
}
```

Baobab.java

```
package AbstractFactory;

public class Baobab implements Tree {
    public Baobab() {
        System.out.println("Посажено Дерево - Баобаб");
    }
}
```

EuropeanGarden.java

```
package AbstractFactory;

public class EuropeanGarden implements GardenFactory {
    public EuropeanGarden() {
        System.out.println("Создан европейский сад!");
    }

    @Override
    public Tree plantTree() {
        return new Oak();
    }
}
```

```

    }

    @Override
    public Flower plantFlower() {
        return new Rose();
    }
}

```

Flower.java

```

package AbstractFactory;

public interface Flower {
}

```

GardenFactory.java

```

package AbstractFactory;

public interface GardenFactory {
    Tree plantTree();
    Flower plantFlower();
}

```

Gloriosa.java

```

package AbstractFactory;

public class Gloriosa implements Flower {
    public Gloriosa() {
        System.out.println("Посажен Цветок - Глориоза");
    }
}

```

HomeGarden.java

```

package AbstractFactory;

public class HomeGarden implements GardenFactory {
    public HomeGarden() {
        System.out.println("Создан домашний сад");
    }

    @Override
    public Tree plantTree() {
        System.out.println("Вы не можете посадить дома дерево!");
        return null;
    }

    @Override
    public Flower plantFlower() {
        return new Orchid();
    }
}

```

MainAbstractFactory.java

```
package AbstractFactory;

public class MainAbstractFactory {
    public static void main(String[] args) {
        AfricanGarden ag = new AfricanGarden();
        ag.plantFlower();
        ag.plantTree();

        System.out.print("\n");
        EuropeanGarden eg = new EuropeanGarden();
        eg.plantFlower();
        eg.plantTree();

        System.out.print("\n");
        HomeGarden hg = new HomeGarden();
        hg.plantFlower();
        hg.plantTree();
    }
}
```

Oak.java

```
package AbstractFactory;

public class Oak implements Tree {
    public Oak() {
        System.out.println("Посажено Дерево - Дуб");
    }
}
```

Orchid.java

```
package AbstractFactory;

public class Orchid implements Flower {
    public Orchid() {
        System.out.println("Посажен Цветок - Орхидея");
    }
}
```

Rose.java

```
package AbstractFactory;

public class Rose implements Flower {
    public Rose() {
        System.out.println("Посажен Цветок - Роза");
    }
}
```

Tree.java

```
package AbstractFactory;

public interface Tree {
}
```

Builder:

Animal.java

```
package Builder;

public class Animal {
    private final String species;
    private final int weight;
    private final String habitat;
    private final int lifespan;

    public static class AnimalBuilder {
        private final String species;
        private int weight = 0;
        private String habitat = "";
        private int lifespan = 0;

        public AnimalBuilder(String species) {
            this.species = species;
        }

        public AnimalBuilder weight(int weight) {
            this.weight = weight;
            return this;
        }

        public AnimalBuilder habitat(String habitat) {
            this.habitat = habitat;
            return this;
        }

        public AnimalBuilder lifespan(int lifespan) {
            this.lifespan = lifespan;
            return this;
        }

        public Animal create() {
            return new Animal(this);
        }
    }

    private Animal(AnimalBuilder builder) {
        species = builder.species;
        weight = builder.weight;
        habitat = builder.habitat;
        lifespan = builder.lifespan;
    }

    @Override
    public String toString() {
```

```

        return species + " обычно живет в " + habitat
            + ", весит примерно " + weight
            + " и живет " + lifespan + " лет.";
    }
}

```

MainBuilder.java

```

package Builder;

public class MainBuilder {
    public static void main(String[] args) {
        Animal lion = new Animal
            .AnimalBuilder("Лев")
            .weight(150)
            .habitat("Саванна")
            .lifespan(12)
            .create();

        Animal polarBear = new Animal
            .AnimalBuilder("Белый медведь")
            .weight(400)
            .habitat("Арктика")
            .lifespan(25)
            .create();

        Animal redFox = new Animal
            .AnimalBuilder("Обыкновенная лисица")
            .weight(10)
            .habitat("тундра, и степи, и леса разного типа, и пустыни и
высокогорья")
            .lifespan(4)
            .create();

        System.out.println(lion);
        System.out.println(polarBear);
        System.out.println(redFox);
    }
}

```

FactoryMethod:

AfricanCow.java

```

package FactoryMethod;

public class AfricanCow extends Cow {
    @Override
    public String getType() {
        return "Cow from Africa";
    }
}

```

AfricanLion.java

```
package FactoryMethod;

public class AfricanLion extends Lion {
    @Override
    public String getType() {
        return "Lion from Africa";
    }
}
```

AfricanOwl.java

```
package FactoryMethod;

public class AfricanOwl extends Owl {
    @Override
    public String getType() {
        return "Owl from Africa";
    }
}
```

AfricanZebra.java

```
package FactoryMethod;

public class AfricanZebra extends Zebra {
    @Override
    public String getType() {
        return "Zebra from Africa";
    }
}
```

AfricanZoo.java

```
package FactoryMethod;

public class AfricanZoo extends Zoo {
    @Override
    protected Animal createAnimal(AnimalType type) {
        Animal animal = null;

        switch (type) {
            case ZEBRA:
                animal = new AfricanZebra();
                break;
            case LION:
                animal = new AfricanLion();
                break;
            case COW:
                animal = new AfricanCow();
                break;
            case OWL:
                animal = new AfricanOwl();
                break;
            case BEAR:
                animal = new RussianBear();
                break;
        }
    }
}
```



```
        break;
    }
    return animal;
}
```

Animal.java

```
package FactoryMethod;

public interface Animal {
    String getType();
}
```

AnimalType.java

```
package FactoryMethod;

public enum AnimalType {
    ZEBRA,
    LION,
    COW,
    OWL,
    BEAR
}
```

Bear.java

```
package FactoryMethod;

public class Bear implements Animal {
    @Override
    public String getType() {
        return "Bear";
    }
}
```

Cow.java

```
package FactoryMethod;

public class Cow implements Animal {
    @Override
    public String getType() {
        return "Cow";
    }
}
```

Lion.java

```
package FactoryMethod;

public class Lion implements Animal {
    @Override
    public String getType() {
        return "Lion";
    }
}
```

MainFactoryMethod.java

```
package FactoryMethod;

public class MainFactoryMethod {
    public static void main(String[] args) {
        System.out.println("Zoo in Cape Town, Republic of South Africa");
        Zoo capeTown = new AfricanZoo();
        capeTown.buyNewAnimal(AnimalType.ZEBRA);
        capeTown.buyNewAnimal(AnimalType.OWL);
        capeTown.buyNewAnimal(AnimalType.COW);
        capeTown.buyNewAnimal(AnimalType.LION);
        capeTown.buyNewAnimal(AnimalType.BEAR);

        System.out.println("\nZoo in Moscow, Russian Federation");
        Zoo moscow = new RussianZoo();
        moscow.buyNewAnimal(AnimalType.BEAR);
        moscow.buyNewAnimal(AnimalType.OWL);
        moscow.buyNewAnimal(AnimalType.COW);
        moscow.buyNewAnimal(AnimalType.LION);
        moscow.buyNewAnimal(AnimalType.ZEBRA);
    }
}
```

Owl.java

```
package FactoryMethod;

public class Owl implements Animal {
    @Override
    public String getType() {
        return "Owl";
    }
}
```

RussianBear.java

```
package FactoryMethod;

public class RussianBear extends Bear {
    @Override
    public String getType() {
        return "Bear from Russia";
    }
}
```

RussianCow.java

```
package FactoryMethod;

public class RussianCow extends Cow {
    @Override
    public String getType() {
        return "Cow from Russia";
    }
}
```

RussianLion.java

```
package FactoryMethod;

public class RussianLion extends Lion {
    @Override
    public String getType() {
        return "Lion from Russia";
    }
}
```

RussianOwl.java

```
package FactoryMethod;

public class RussianOwl extends Owl {
    @Override
    public String getType() {
        return "Owl from Russia";
    }
}
```

RussianZoo.java

```
package FactoryMethod;

public class RussianZoo extends Zoo {
    @Override
    protected Animal createAnimal(AnimalType type) {
        Animal animal = null;

        switch (type) {
            case BEAR:
                animal = new RussianBear();
                break;
            case COW:
                animal = new RussianCow();
                break;
            case OWL:
                animal = new RussianOwl();
                break;
            case LION:
                animal = new RussianLion();
                break;
            case ZEBRA:
                animal = new AfricanZebra();
                break;
        }
    }
}
```

```

    }
    return animal;
}
}

```

Zebra.java

```

package FactoryMethod;

public class Zebra implements Animal {
    @Override
    public String getType() {
        return "Zebra from Africa";
    }
}

```

Zoo.java

```

package FactoryMethod;

public abstract class Zoo {
    public void buyNewAnimal(AnimalType type) {
        Animal animal = createAnimal(type);
        System.out.println("You just bought " + animal.getType() + " at the zoo!");
    }

    protected abstract Animal createAnimal(AnimalType type);
}

```

Prototype:

Life.java

```

package Prototype;

public class Life {
    private Type type;

    public Life() {
    }

    public Life(Type type) {
        this.type = type;
    }

    public Life copy() {
        return new Life();
    }

    public Type getType() {
        return type;
    }

    public void setType(Type type) {
        this.type = type;
    }

    @Override

```

```

    public String toString() {
        return "Life{" +
            "type=" + type +
            '}';
    }
}

```

Type.java

```

package Prototype;

public enum Type {
    BACTERIA,
    FUNGUS,
    ANIMAL,
    PLANT
}

```

MainPrototype.java

```

package Prototype;

public class MainPrototype {
    public static void main(String[] args) {
        Life bacteria = new Life(Type.BACTERIA);
        Life fungus = bacteria.copy();
        fungus.setType(Type.FUNGUS);

        System.out.println(bacteria);
        System.out.println(fungus);
    }
}

```

Результат выполнения программы

```

Создан африканский сад!
Посажен Цветок - Глориоза
Посажено Дерево - Баобаб

Создан европейский сад!
Посажен Цветок - Роза
Посажено Дерево - Дуб

Создан домашний сад
Посажен Цветок - Орхидея
Вы не можете посадить дома дерево!

```

Рисунок 6.1 – Демонстрация работы программы Abstract Factory

Лев обычно живет в Саванна, весит примерно 150 и живет 12 лет.
Белый медведь обычно живет в Арктика, весит примерно 400 и живет 25 лет.
Обыкновенная лисица обычно живет в тундра, и степи, и леса разного типа, и пустыни и высокогорья, весит примерно 10 и живет 4 лет.

Рисунок 6.2 – Демонстрация работы программы Main Builder

```
Zoo in Cape Town, Republic of South Africa
You just bought Zebra from Africa at the zoo!
You just bought Owl from Africa at the zoo!
You just bought Cow from Africa at the zoo!
You just bought Lion from Africa at the zoo!
You just bought Bear from Russia at the zoo!

Zoo in Moscow, Russian Federation
You just bought Bear from Russia at the zoo!
You just bought Owl from Russia at the zoo!
You just bought Cow from Russia at the zoo!
You just bought Lion from Russia at the zoo!
You just bought Zebra from Africa at the zoo!
```

Рисунок 6.3 – Демонстрация работы программы Factory Method

```
Life{type=BACTERIA}
Life{type=FUNGUS}
```

Рисунок 6.4 – Демонстрация работы программы Prototype

Практическая работа №7

Цель работы

Тема: Реализация структурных паттернов проектирования.

Постановка задачи: Написать реализацию паттерна в соответствии с вариантом индивидуального задания Легковес, Заместитель.

Листинг программы

Flyweight:

FirstHuman.java

```
package Flyweight;

public class FirstHuman extends Human {
    public FirstHuman() {
        firstName = "Иван";
        lastName = "Николаев-Аксенов";
        age = 19;
    }

    @Override
    public void getInfo() {
        System.out.println(firstName + ' ' + lastName + ", возраст " + age + " лет");
    }
}
```

FlyweightFactory.java

```
package Flyweight;

import java.util.HashMap;

public class FlyweightFactory {
    private HashMap<Integer, Human> people = new HashMap<>();

    public Human getHumanInfo(int number) {
        Human human = people.get(number);
        if (human == null) {
            switch (number) {
                case 1: {
                    human = new FirstHuman();
                    break;
                }
                case 2: {
                    human = new SecondHuman();
                    break;
                }
                case 3: {
                    human = new ThirdHuman();
                    break;
                }
            }
        }
    }
}
```

```

        people.put(number, human);
    }
    return human;
}

```

Human.java

```

package Flyweight;

public abstract class Human {
    protected String firstName;
    protected String lastName;
    protected int age;
    public abstract void getInfo();
}

```

MainFlyweight.java

```

package Flyweight;

public class MainFlyweight {
    public static void main(String[] args) {
        FlyweightFactory factory = new FlyweightFactory();

        int[] peopleList = {1, 2, 3, 3, 2};
        for(int i: peopleList) {
            Human h = factory.getHumanInfo(i);
            h.getInfo();
        }
    }
}

```

SecondHuman.java

```

package Flyweight;

public class SecondHuman extends Human {
    public SecondHuman() {
        firstName = "Иван";
        lastName = "ИВАНОВ";
        age = 23;
    }

    @Override
    public void getInfo() {
        System.out.println(firstName + ' ' + lastName + ", возраст " + age + " лет");
    }
}

```


ThirdHuman.java

```
package Flyweight;

public class ThirdHuman extends Human {
    public ThirdHuman() {
        firstName = "Петр";
        lastName = "Петров";
        age = 48;
    }

    @Override
    public void getInfo() {
        System.out.println(firstName + ' ' + lastName + ", возраст " + age + " лет");
    }
}
```

Proxy:

IUserChanger.java

```
package Proxy;

import Flyweight.Human;

public interface IUserChanger {
    void changeName(User user, String name);
    void changeAge(User user, int age);
}
```

MainProxy.java

```
package Proxy;

public class MainProxy {
    public static void main(String[] args) {
        User a = new User("Ivan", 19);
        User b = new User("Petr", 25);
        User c = new User("root", 0);

        UserChanger userChanger = new UserChanger();
        ProxyUserChanger proxyUserChanger = new ProxyUserChanger();

        userChanger.changeAge(a, 21);
        userChanger.changeName(b, "Pavel");

        proxyUserChanger.changeName(c, "Ivan&Pavel account");

        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }
}
```

ProxyUserChanger.java

```
package Proxy;

public class ProxyUserChanger implements IUserChanger {
    private UserChanger uc;

    @Override
    public void changeName(User user, String name) {
        System.out.println("Proxy...");
        UserChangerInitializer();
        uc.changeName(user, name);
    }

    @Override
    public void changeAge(User user, int age) {
        System.out.println("Proxy...");
        UserChangerInitializer();
        uc.changeAge(user, age);
    }

    private void UserChangerInitializer() {
        if(uc == null)
            uc = new UserChanger();
    }
}
```

User.java

```
package Proxy;

public class User {
    private String name;
    private int age;

    public User(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Пользователь " + name + ", возраст " + age + " лет.";
    }
}
```

UserChanger.java

```
package Proxy;

public class UserChanger implements IUserChanger {
    @Override
    public void changeName(User user, String name) {
        user.setName(name);
    }

    @Override
    public void changeAge(User user, int age) {
        user.setAge(age);
    }
}
```

Результат выполнения программы

```
Иван Николаев-Аксенов, возраст 19 лет
Иван Иванов, возраст 23 лет
Петр Петров, возраст 48 лет
Петр Петров, возраст 48 лет
Иван Иванов, возраст 23 лет
```

Рисунок 7.1 – Демонстрация работы программы Flyweight

```
Прoxy...
Пользователь Ivan, возраст 21 лет.
Пользователь Pavel, возраст 25 лет.
Пользователь Ivan&Pavel account, возраст 0 лет.
```

Рисунок 7.2 – Демонстрация работы программы Proxy

Практическая работа №8

Цель работы

Тема: Реализация поведенческих паттернов проектирования.

Постановка задачи: Написать реализацию паттерна в соответствии с вариантом индивидуального задания Наблюдатель, Состояние.

Листинг программы

Observer:

CollageRating.java

```
package Observer;

public class CollageRating implements Observer {
    private String name;
    private double gpa;

    public void printInfo() {
        System.out.println("Студент " + name + " имеет средний балл " + gpa);
    }

    @Override
    public void update(String name, double gpa) {
        this.name = name;
        this.gpa = gpa;
        printInfo();
    }
}
```

MainObserver.java

```
package Observer;

public class MainObserver {
    public static void main(String[] args) {
        Student student = new Student();
        Observer studentInfo = new CollageRating();

        student.addObserver(studentInfo);

        student.setNameAndGPA("Ivan", 7.9);
        student.setName("Petr");
        student.setGpa(5.2);
    }
}
```

Observable.java

```
package Observer;

public interface Observable {
    void addObserver(Observer o);
    void removeObserver(Observer o);
    void notifyObservers();
}
```

Observer.java

```
package Observer;

interface Observer {
    void update(String name, double gpa);
}
```

Student.java

```
package Observer;

import java.util.LinkedList;
import java.util.List;

public class Student implements Observable {
    private String name;
    private double gpa;
    private List<Observer> obs;

    public Student() {
        obs = new LinkedList<>();
    }

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

    public void setGpa(double gpa) {
        this.gpa = gpa;
        notifyObservers();
    }

    public void setNameAndGPA(String name, double gpa) {
        this.name = name;
        this.gpa = gpa;
        notifyObservers();
    }

    @Override
    public void addObserver(Observer o) {
        obs.add(o);
    }

    @Override
    public void removeObserver(Observer o) {
        obs.remove(o);
    }
}
```

```

@Override
public void notifyObservers() {
    for(Observer o: obs)
        o.update(name, gpa);
}
}

```

State:

Bread.java

```

package State;

public class Bread implements State {
    private static final String name = "хлеб";

    @Override
    public String getName() {
        return name;
    }

    @Override
    public void make(StateInfo stateInfo) {
        stateInfo.setState(new SandwichWithButter());
    }

    @Override
    public void eat(StateInfo stateInfo) {
        System.out.println("Сначала нужно приготовить бутерброд! Пока он на стадии: " +
name);
    }
}

```

HalfASandwich.java

```

package State;

public class HalfASandwich implements State {
    private static final String name = "половина бутерброда";

    @Override
    public String getName() {
        return name;
    }

    @Override
    public void make(StateInfo stateInfo) {
        System.out.println("Вы уже начали есть бутерброд!");
    }

    @Override
    public void eat(StateInfo stateInfo) {
        System.out.println("Вы съели бутерброд");
    }
}

```

MainState.java

```
package State;

public class MainState {
    public static void main(String[] args) {
        StateInfo stateInfo = new StateInfo();

        stateInfo.make();
        System.out.println();

        stateInfo.make();
        System.out.println();

        stateInfo.make();
        System.out.println();

        stateInfo.eat();
        System.out.println();

        stateInfo.eat();
    }
}
```

SandwichWithButter.java

```
package State;

public class SandwichWithButter implements State {
    private static final String name = "хлеб с маслом";

    @Override
    public String getName() {
        return name;
    }

    @Override
    public void make(StateInfo stateInfo) {
        stateInfo.setState(new SandwichWithButterAndSausage());
    }

    @Override
    public void eat(StateInfo stateInfo) {
        System.out.println("Сначала нужно приготовить бутерброд! Пока он на стадии: " +
name);
    }
}
```

SandwichWithButterAndSausage.java

```
package State;

public class SandwichWithButterAndSausage implements State {
    private static final String name = "бутерброд";

    @Override
    public String getName() {
        return name;
    }
}
```

```

@Override
public void make(StateInfo stateInfo) {
    System.out.println("Вы уже приготовили бутерброд!");
}

@Override
public void eat(StateInfo stateInfo) {
    stateInfo.setState(new HalfASandwich());
}
}

```

State.java

```

package State;

public interface State {
    String getName();
    void make(StateInfo stateInfo);
    void eat(StateInfo stateInfo);
}

```

StateInfo.java

```

package State;

public class StateInfo {
    private State state = new Bread();

    public void make() {
        System.out.println("Готовим бутерброд, текущее состояние " + state.getName());
        state.make(this);
    }

    public void eat() {
        System.out.println("Едим бутерброд, текущее состояние " + state.getName());
        state.eat(this);
    }

    public void setState(State state) {
        System.out.println("Изменяем состояние бутерброда на " + state.getName());
        this.state = state;
    }

    public State getState() {
        return state;
    }
}

```


Результат выполнения программы

```
Студент Ivan имеет средний балл 7.9  
Студент Petr имеет средний балл 7.9  
Студент Petr имеет средний балл 5.2
```

Рисунок 8.1 – Демонстрация работы программы Observer

```
Готовим бутерброд, текущее состояние хлеб  
Изменяем состояние бутерброда на хлеб с маслом  
  
Готовим бутерброд, текущее состояние хлеб с маслом  
Изменяем состояние бутерброда на бутерброд  
  
Готовим бутерброд, текущее состояние бутерброд  
Вы уже приготовили бутерброд!  
  
Едим бутерброд, текущее состояние бутерброд  
Изменяем состояние бутерброда на половина бутерброда  
  
Едим бутерброд, текущее состояние половина бутерброда  
Вы съели бутерброд
```

Рисунок 8.2 – Демонстрация работы программы State

Практическая работа №9

Цель работы

Тема: Знакомство с системой сборки приложения. Gradle.

Постановка задачи: Создать приложение, которое выводит какое-то сообщение в консоль. Создать Gradle Task, который создает jar-файл приложения, переносит его в отдельную папку, в которой хранится Dockerfile для jar, а затем создает Docker контейнер из данного jar-файла и запускает его.

Листинг программы

Main.java

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello World from Java from Gradle from Docker!");  
    }  
}
```

build.gradle

```
plugins {  
    id 'java'  
}  
  
version '1.0'  
  
repositories {  
    mavenCentral()  
}  
  
jar {  
    manifest {  
        attributes(  
            'Class-Path': configurations.compile.collect { it.getName() }.join(' '),  
            'Main-Class': 'Main'  
        )  
    }  
}  
  
task toDocker {  
    doLast {  
        def stdout = new ByteArrayOutputStream()  
  
        println("Moving jar-file...")  
        ant.move file: "${buildDir}/libs/HelloWorldProject-1.0.jar", todir:  
"${projectDir}/Docker"  
  
        println("\nCreating Docker container...")  
        exec {  
            workingDir "${projectDir}/Docker"  
            commandLine 'docker', 'build', '-t', 'helloworld', '.'  
        }  
    }  
}
```

```

        println("\nLaunching Docker container...")
        exec {
            workingDir "${projectDir}/Docker"
            commandLine 'docker', 'run', 'helloworld'
            standardOutput = stdout
        }
        println "\nOutput from Docker container:\n$stdout"
    }
}

build.finalizedBy(toDocker)

```

Dockerfile

```

FROM openjdk:11.0.10
WORKDIR /
ADD HelloWorldProject-1.0.jar HelloWorldProject-1.0.jar
EXPOSE 8080
CMD ["java", "-jar", "HelloWorldProject-1.0.jar"]

```

Результат выполнения программы

```
frisc@FRISCHMANN-PC C:\Users\frisc\GoogleDrive\Study\JavaPatterns\Практическая работа 9
$ gradle build

> Task :toDocker
Moving jar-file...

Creating Docker container...
#1 [internal] load build definition from Dockerfile
#1 sha256:6c5c8e382d5ffc99fd11fe872beaf043be1ed3e70ef41bbb510364459de397b4
#1 transferring dockerfile: 32B 0.0s done
#1 DONE 0.0s

#2 [internal] load .dockerignore
#2 sha256:25102dd82253e6e3bb110fffb279cf849b80ba191760989a2482b6c73cb5034ef
#2 transferring context: 2B done
#2 DONE 0.0s

#3 [internal] load metadata for docker.io/library/openjdk:11.0.10
#3 sha256:c791bb7f23fefe5c5adff6530397c072a994ba739566a87564fdc5351fa2c3f7
#3 ...

#4 [auth] library/openjdk:pull token for registry-1.docker.io
#4 sha256:ed3a1316321d3728cfff11008cf06c3b1efca30e4b130ef743d80f105fbad572
#4 DONE 0.0s

#3 [internal] load metadata for docker.io/library/openjdk:11.0.10
#3 sha256:c791bb7f23fefe5c5adff6530397c072a994ba739566a87564fdc5351fa2c3f7
#3 DONE 2.1s

#6 [internal] load build context
#6 sha256:2179e4b2d6821d2188e14451ec4b9d7241a230ff747224d91390bc498029a626
#6 transferring context: 786B 0.0s done
#6 DONE 0.0s

#5 [1/3] FROM docker.io/library/openjdk:11.0.10@sha256:d112a532f300ce7d35f5cd196c22fced33d0a28a7dd0227
019da5fb430528428
#5 sha256:b3762d9b4b5c25472a7d73be65d92352989745a5defee86b0dbc82b6a2dc2fa2
#5 CACHED

#7 [2/3] ADD HelloWorldProject-1.0.jar HelloWorldProject-1.0.jar
#7 sha256:21d9bb277dcde7bd35d595c59612d099980ae7377089eb4213faa46ac864a901
#7 DONE 0.0s

#8 exporting to image
#8 sha256:e8c613e07b0b7ff33893b694f7759a10d42e180f2b4dc349fb57dc6b71dcab00
#8 exporting layers 0.0s done
#8 writing image sha256:077d11fbff279741454c3bc89476399fed405787a6639361d6cb6b1a59c5cc14
#8 writing image sha256:077d11fbff279741454c3bc89476399fed405787a6639361d6cb6b1a59c5cc14 done
#8 naming to docker.io/library/helloworld done
#8 DONE 0.0s

Launching Docker container...
```

Рисунок 9.1 – Демонстрация работы программы

```
Launching Docker container...

Output from Docker container:
Hello World from Java from Gradle from Docker!

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.8/userguide/command\_line\_interface.html#sec:command\_line\_warnings

BUILD SUCCESSFUL in 6s
3 actionable tasks: 2 executed, 1 up-to-date
```

Рисунок 9.2 – Демонстрация работы программы

Практическая работа №10

Цель работы

Тема: Введение в Spring. Container. Bean. Внедрение зависимостей, основанных на конструкторах и сеттерах. Конфигурация бинов. Автоматическое обнаружение и связывание классов.

Постановка задачи: Создать приложение, в котором создается ApplicationContext и из него берётся бин с названием, переданным в качестве аргумента к приложению, и вызывается метод интерфейса, который он имплементирует. Нужно создать по одному бину для каждого класса, определить им название. Проверить, что вызывается при вводе названия каждого из бинов. Классы и интерфейс определяются в соответствии с вариантом индивидуального задания Интерфейс Fighter с методом doFight(), его имплементации: StreetFighter, Boxer, Judoka.

Листинг программы

Boxer.java

```
package App;

import org.springframework.stereotype.Component;

@Component
public class Boxer implements Fighter {
    @Override
    public void doFight() {
        System.out.println("Boxer enters the ring");
    }
}
```

Fighter.java

```
package App;

public interface Fighter {
    void doFight();
}
```

Judoka.java

```
package App;

import org.springframework.stereotype.Component;

@Component
public class Judoka implements Fighter {
    @Override
    public void doFight() {
```

```

        System.out.println("Judoka fighter enters the ring");
    }
}

```

StreetFighter.java

```

package App;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext ac = new ClassPathXmlApplicationContext("fighters.xml");
        Fighter f = (Fighter) ac.getBean(args[0]);
        f.doFight();
    }
}

```

MainApp.java

```

package App;

import org.springframework.stereotype.Component;

@Component
public class StreetFighter implements Fighter {
    @Override
    public void doFight() {
        System.out.println("Street fighter enters the ring");
    }
}

```

Fighters.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="streetFighter" class="App.StreetFighter"/>
    <bean id="boxer" class="App.Boxer"/>
    <bean id="judoka" class="App.Judoka"/>
</beans>

```

Результат выполнения программы

```
$ java -jar "Практическая работа 10-1.0.jar" streetFighter
Street fighter enters the ring

frisc@FRISCHMANN-PC C:\Users\frisc\GoogleDrive\Study\JavaPatterns\libs
$ java -jar "Практическая работа 10-1.0.jar" boxer
Boxer enters the ring

frisc@FRISCHMANN-PC C:\Users\frisc\GoogleDrive\Study\JavaPatterns\libs
$ java -jar "Практическая работа 10-1.0.jar" judoka
Judoka fighter enters the ring
```

Рисунок 10.1 – Демонстрация работы программы

Практическая работа №11

Цель работы

Тема: Разобраться с использованием Spring boot.

Постановка задачи: Создать приложение с использованием Spring Boot Starter Initializr (<https://start.spring.io/>) с такими зависимостями:

- Spring Web;
- Lombok;
- Validation;
- Spring boot Actuator.

Запустить приложение и удостовериться, что не появилось никаких ошибок. Добавить все эндпоинты в Actuator, сделать HTTP-запрос на проверку состояния приложения. Собрать jar-файл приложения, запустить и проверить состояние при помощи REST-запроса.

Листинг программы

TestRest.java

```
package App.AppMain;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class TestRest {
    @RequestMapping("/test")
    public String open(){
        return "Hello, World!";
    }
}
```

AppMainApplication.java

```
package App.AppMain;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class AppMainApplication {

    public static void main(String[] args) {
        SpringApplication.run(AppMainApplication.class, args);

        TestRest tr = new TestRest();
        tr.open();
    }
}
```

Application.properties

```
management.endpoint.auditevents.enabled=true
management.endpoint.beans.enabled=true
management.endpoint.caches.enabled=true
management.endpoint.conditions.enabled=true
management.endpoint.configprops.enabled=true
management.endpoint.env.enabled=true
management.endpoint.flyway.enabled=true
management.endpoint.health.enabled=true
management.endpoint.httptrace.enabled=true
management.endpoint.info.enabled=true
management.endpoint.integrationgraph.enabled=true
management.endpoint.loggers.enabled=true
management.endpoint.liquibase.enabled=true
management.endpoint.metrics.enabled=true
management.endpoint.mappings.enabled=true
management.endpoint.scheduledtasks.enabled=true
management.endpoint.sessions.enabled=true
management.endpoint.shutdown.enabled=true
management.endpoint.threaddump.enabled=true
```

Результат выполнения программы

```

      ____          _            __ _ _
     /  _ \        | |          // (_) |
    / ___|         | |__   _//_/ (_)_|| |
   /___\           |_____|_//___(____)___/

:: Spring Boot ::                (v2.4.4)


2021-03-23 00:06:44.384 INFO 13560 --- [main] App.AppMain.AppMainApplication : Starting AppMainApplication using Java 11.0.10 on Frischmann-PC with PID 13560
2021-03-23 00:06:44.387 INFO 13560 --- [main] App.AppMain.AppMainApplication : No active profile set, falling back to default profiles: default
2021-03-23 00:06:45.851 INFO 13560 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-03-23 00:06:45.866 INFO 13560 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-03-23 00:06:45.867 INFO 13560 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.44]
2021-03-23 00:06:45.971 INFO 13560 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-03-23 00:06:45.971 INFO 13560 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1528 ms
2021-03-23 00:06:46.248 INFO 13560 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2021-03-23 00:06:46.526 INFO 13560 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base path '/actuator'
2021-03-23 00:06:46.573 INFO 13560 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-03-23 00:06:46.587 INFO 13560 --- [main] App.AppMain.AppMainApplication : Started AppMainApplication in 2.639 seconds (JVM running for 3.913)
2021-03-23 00:06:48.044 INFO 13560 --- [1]-172.29.128.1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-03-23 00:06:48.044 INFO 13560 --- [1]-172.29.128.1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-03-23 00:06:48.046 INFO 13560 --- [1]-172.29.128.1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms

```

Рисунок 11.1 – Демонстрация работы программы

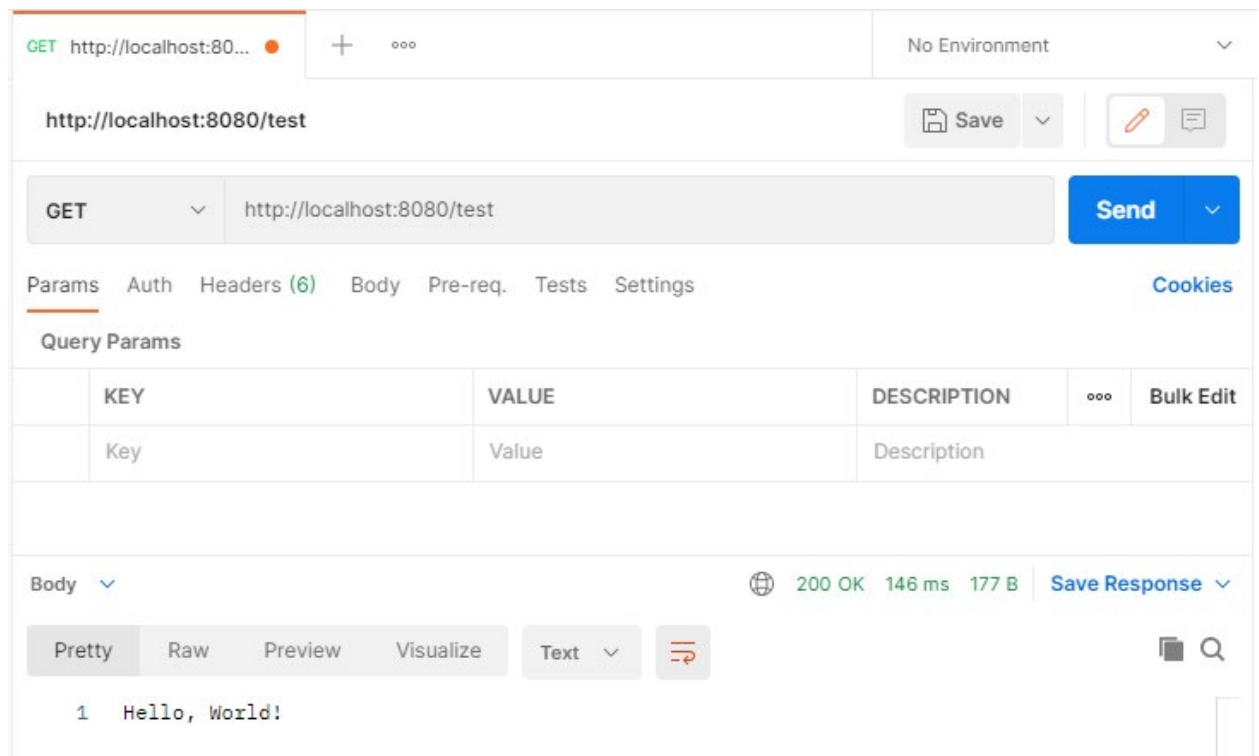


Рисунок 11.2 – Демонстрация работы программы

Практическая работа №12

Цель работы

Тема: Работа с жизненным циклом компонентов. Аннотации PostConstruct, PreDestroy.

Постановка задачи: Создать приложение, которое при запуске берет данные из одного файла, хеширует, а при остановке приложения удаляет исходный файл, оставляя только файл с захешированными данными. Названия первого и второго файла передаются в качестве аргументов при запуске. При отсутствии первого файла создает второй файл и записывает в него строку null. Реализовать с использованием аннотаций PostConstruct, PreDestroy.

Листинг программы

FileWorker.java

```
package App.Main;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.ApplicationArguments;
import org.springframework.stereotype.Component;
import org.springframework.util.DigestUtils;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.io.*;

@Component
public class FileWorker {
    @Autowired
    private ApplicationArguments arguments;

    private String hashed;

    public FileWorker(){
        hashed = "";
    }

    @PostConstruct
    public void init() throws IOException {
        try(InputStream file = new FileInputStream(arguments.getNonOptionArgs().get(0)))
        {
            hashed = DigestUtils.md5DigestAsHex(file);

            File secondFile = new File(arguments.getNonOptionArgs().get(1));
            if(!secondFile.exists())
                secondFile.createNewFile();

            FileWriter writer = new FileWriter(secondFile);
            writer.write(hashed);
            writer.close();

        } catch (FileNotFoundException e) {
```

```

        File file = new File(arguments.getNonOptionArgs().get(1));
        if(!file.exists())
            file.createNewFile();

        FileWriter writer = new FileWriter(file);
        writer.write("null");
        writer.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

@PreDestroy
public void deleteFirst(){
    File file = new File(arguments.getNonOptionArgs().get(0));

    if(file.exists()) {
        file.delete();
    }
}
}

```

MainApplication.java

```

package App.Main;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MainApplication {
    private final FileWorker worker;

    @Autowired
    public MainApplication(FileWorker worker) {
        this.worker = worker;
    }

    public static void main(String[] args) throws Exception {
        SpringApplication.run(MainApplication.class, args);
    }
}

```

Результат выполнения программы

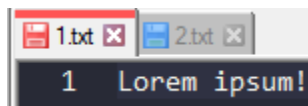


Рисунок 12.1 – Демонстрация работы программы

2.txt	23.03.2021 0:09	Файл "TXT"	1 КБ
Main-0.0.1.jar	22.03.2021 1:21	Файл "JAR"	8 360 КБ

Рисунок 12.2 – Демонстрация работы программы

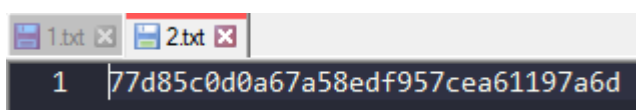


Рисунок 12.2 – Демонстрация работы программы

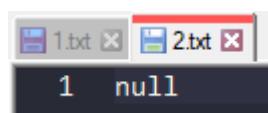


Рисунок 12.3 – Демонстрация работы программы

Практическая работа №13

Цель работы

Тема: Конфигурирование приложения. Environment.

Постановка задачи: Создать файл application.yml в папке resources, добавить в него такие свойства:

- student.name – имя студента;
- student.last_name – фамилия студента;
- student.group – название группы студента.

При запуске приложения выведите данные свойства в консоль при помощи интерфейса Environment или аннотации Value.

Листинг программы

Student.java

```
package App.Main;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;

@Component
public class Student {
    @Value("${student.name}")
    private String name;

    @Value("${student.last_name}")
    private String last_name;

    @Value("${student.group}")
    private String group;

    @PostConstruct
    public void init() {
        System.out.println("First name: " + name + "\nLast name: " + last_name +
            "\nGroup: " + group);
    }
}
```

MainApplication.java

```
package App.Main;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MainApplication {

    public static void main(String[] args) {
        SpringApplication.run(MainApplication.class, args);
    }
}
```

application.yml

```
student:
  name: Ivan
  last_name: Nikolaev-Axenov
  group: ИКБО-20-19
```

Результат выполнения программы

```
. _ _ _ _ _  
/\ / ____' _ _ _ _ _ \ \ \ \ \  
( ( \_ _ _ | ' _ | ' _ | ' _ V _ | \ \ \ \ \  
\W _ _ _ )| |_) | | | | | | | | C | | ) ) ) )  
 ' | _ _ | _ _ | | _ | | \_ _ , / / / /  
=====|_|=====|___#/_/_/_/  
  
:: Spring Boot ::                (v2.4.4)  
  
2021-03-23 00:11:45.214 INFO 16072 --- [main] App.Main.MainApplication : Starting MainApplication using Java 11.0.10 on Frischmann-PC with  
2021-03-23 00:11:45.217 INFO 16072 --- [main] App.Main.MainApplication : No active profile set, falling back to default profiles: default  
First name: Ivan  
Last name: Nikolaev-Axenov  
Group: WK50-20-19  
2021-03-23 00:11:45.666 INFO 16072 --- [main] App.Main.MainApplication : Started MainApplication in 0.834 seconds (JVM running for 1.672)
```

Рисунок 13.1 – Демонстрация работы программы

Практическая работа №14

Цель работы

Тема: Знакомство со Spring MVC. Работа с Rest API в Spring.

Постановка задачи: Создать отдельный репозиторий Git. Создать простой html-документ, который будет содержать вашу фамилию, имя, номер группы, номер варианта. Создать контроллер, который будет возвращать данный статический документ при переходе на url «/home». Выполнить задание в зависимости с вариантом индивидуального задания Создать класс Post с полями text, creationDate. Создать класс User с полями firstName, lastName, middleName, birthDate. Создать классы-контроллеры для создания, удаления объектов и получения всех объектов каждого типа. Сами объекты хранить в памяти.

Листинг программы

Application.java

```
package PR14.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);
    }

}
```

Post.java

```
package PR14.Application.model;

import com.fasterxml.jackson.annotation.JsonProperty;
import java.util.Date;

public class Post {
    private final String text;
    private final Date creationDate;

    public Post(@JsonProperty("text") String text) {
        this.text = text;
        this.creationDate = new Date();
    }

    public String getText() {
        return text;
    }

    public Date getCreationDate() {
        return creationDate;
    }

    @Override
    public String toString() {
        return "Пост от " + creationDate + "\nТекст: " + text;
    }
}
```

User.java

```
package PR14.Application.model;

import com.fasterxml.jackson.annotation.JsonProperty;

import java.util.ArrayList;
import java.util.List;

public class User {
    private final String firstName;
    private final String lastName;
    private final String middleName;
    private final String birthDate;

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

    public User(@JsonProperty("firstName") String firstName,
                @JsonProperty("lastName") String lastName,
                @JsonProperty("middleName") String middleName,
                @JsonProperty("birthDate") String birthDate) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.middleName = middleName;
        this.birthDate = birthDate;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    public List<Post> getPosts() {
        return posts;
    }

    public void addPost(Post post) {
        this.posts.add(post);
    }

    public void deletePost(Post post) {
        this.posts.remove(post);
    }

    @Override
    public String toString() {
        return "Пользователь " + lastName + " " + firstName + " " + middleName + ", день  
рождения: " + birthDate + "\nОпубликовал следующие посты: " + posts;
    }
}
```

UserPostHolder.java

```
package PR14.Application.model;

import com.fasterxml.jackson.annotation.JsonProperty;

public class UserPostHolder {
    private final User user;
    private final String text;

    public UserPostHolder(@JsonProperty("user") User user,
        @JsonProperty("text") String text) {
        this.user = user;
        this.text = text;
    }

    public User getUser() {
        return user;
    }

    public String getText() {
        return text;
    }
}
```

UserDataAccessService.java

```
package PR14.Application.service;

import PR14.Application.model.Post;
import PR14.Application.model.User;
import PR14.Application.model.UserPostHolder;
import org.springframework.stereotype.Repository;

import java.util.ArrayList;
import java.util.List;

@Repository
public class UserDataAccessService {
    private static List<User> DB = new ArrayList<>();

    public int insertUser(User user) {
        DB.add(user);
        return 1;
    }

    public int insertPost(UserPostHolder userPostHolder) {
        User user = userPostHolder.getUser();
        String text = userPostHolder.getText();
        for(User i : DB) {
            if(i.getFirstName().equals(user.getFirstName()) &&
i.getLastName().equals(user.getLastName()) &&
i.getMiddleName().equals(user.getMiddleName()) &&
i.getBirthDate().equals(user.getBirthDate())) {
                i.addPost(new Post(text));
            }
        }
        return 1;
    }
}
```

```

    public int deleteUser(User user) {
        DB.removeIf(i -> i.getFirstName().equals(user.getFirstName()) &&
i.getLastName().equals(user.getLastName()) &&
i.getMiddleName().equals(user.getMiddleName()) &&
i.getBirthDate().equals(user.getBirthDate()));
        return 1;
    }

    public int deletePost(UserPostHolder userPostHolder) {
        User user = userPostHolder.getUser();
        String text = userPostHolder.getText();
        for(User i : DB) {
            if(i.getFirstName().equals(user.getFirstName()) &&
i.getLastName().equals(user.getLastName()) &&
i.getMiddleName().equals(user.getMiddleName()) &&
i.getBirthDate().equals(user.getBirthDate())) {
                for(Post j : i.getPosts()) {
                    if(j.getText().equals(text)) {
                        i.deletePost(j);
                    }
                }
            }
        }
        return 1;
    }

    public List<User> getAllUsers() {
        return DB;
    }
}

```

UserService.java

```

package PR14.Application.service;

import PR14.Application.model.User;
import PR14.Application.model.UserPostHolder;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class UserService {
    private final UserDataAccessService userDataAccessService;

    @Autowired
    public UserService(UserDataAccessService userDataAccessService) {
        this.userDataAccessService = userDataAccessService;
    }

    public int insertUser(User user) {
        return userDataAccessService.insertUser(user);
    }

    public int insertPost(UserPostHolder userPostHolder) {
        return userDataAccessService.insertPost(userPostHolder);
    }

    public int deleteUser(User user) {

```

```

        return userDataAccessService.deleteUser(user);
    }

    public int deletePost(UserPostHolder userPostHolder) {
        return userDataAccessService.deletePost(userPostHolder);
    }

    public List<User> getAllUsers() {
        return userDataAccessService.getAllUsers();
    }
}

```

HomeController.java

```

package PR14.Application.controller;

import org.springframework.http.MediaType;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class HomeController {
    @GetMapping(value = "/home", produces = MediaType.TEXT_HTML_VALUE)
    @ResponseBody
    public String homePage() {
        return "<html>\n" +
            "    <head><title>Home</title></head>\n" +
            "    <body>\n" +
            "        Фамилия: Николаев-Аксенов<br><hr>\nИмя: Иван<br><hr>\nНомер группы:  
ИКБО-20-19<br><hr>\nНомер варианта: 21(6)<hr>" +
            "    </body>\n" +
            "    </html>";
    }
}

```

UserController.java

```
package PR14.Application.controller;

import PR14.Application.model.User;
import PR14.Application.model.UserPostHolder;
import PR14.Application.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class UserController {
    private final UserService userService;

    @Autowired
    public UserController(UserService userService) {
        this.userService = userService;
    }

    @PostMapping("/users")
    public int insertUser(@RequestBody User user) {
        return userService.insertUser(user);
    }

    @PostMapping("/posts")
    public int insertPost(@RequestBody UserPostHolder userPostHolder) {
        return userService.insertPost(userPostHolder);
    }

    @DeleteMapping("/users")
    public int deleteUser(@RequestBody User user) {
        return userService.deleteUser(user);
    }

    @DeleteMapping("/posts")
    public int deletePost(@RequestBody UserPostHolder userPostHolder) {
        return userService.deletePost(userPostHolder);
    }

    @GetMapping("/users")
    public List<User> getAllUsers() {
        return userService.getAllUsers();
    }
}
```

Результат выполнения программы

```
"C:\Program Files\OpenJDK\openjdk-11.0.10_9\bin\java.exe" ...  
  
  _   _      _   _      _   _  
 / \ / \    / \ / \    / \ / \  
( ) ( )    ( ) ( )    ( ) ( )  
W  W     W  W     W  W     W  W  
 '  |_____|  |_____|  |_____|  |_____|  
=====|_|=====|_|#/_/_/_/  
  
:: Spring Boot ::                (v2.4.4)  
  
2021-04-05 21:41:01.166 INFO 10800 --- [main] PR14.Application.Application : Starting Application using Java 11.0.10 on Frischmann-PC with PID 10800  
2021-04-05 21:41:01.172 INFO 10800 --- [main] PR14.Application.Application : No active profile set, falling back to default profiles: default  
2021-04-05 21:41:03.073 INFO 10800 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)  
2021-04-05 21:41:03.091 INFO 10800 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]  
2021-04-05 21:41:03.091 INFO 10800 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.44]  
2021-04-05 21:41:03.241 INFO 10800 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext  
2021-04-05 21:41:03.241 INFO 10800 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1971 ms  
2021-04-05 21:41:03.508 INFO 10800 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'  
2021-04-05 21:41:03.707 INFO 10800 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''  
2021-04-05 21:41:03.717 INFO 10800 --- [main] PR14.Application.Application : Started Application in 3.311 seconds (JVM running for 4.793)
```

Рисунок 14.1 – Демонстрация работы программы

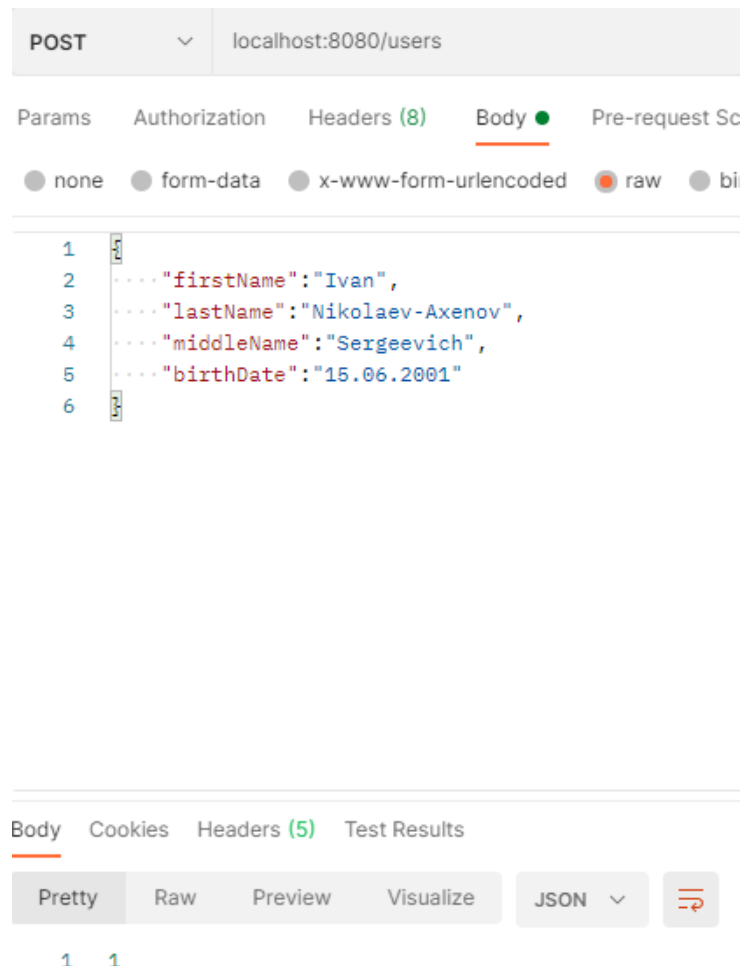


Рисунок 14.2 – Демонстрация работы программы



Рисунок 14.3 – Демонстрация работы программы

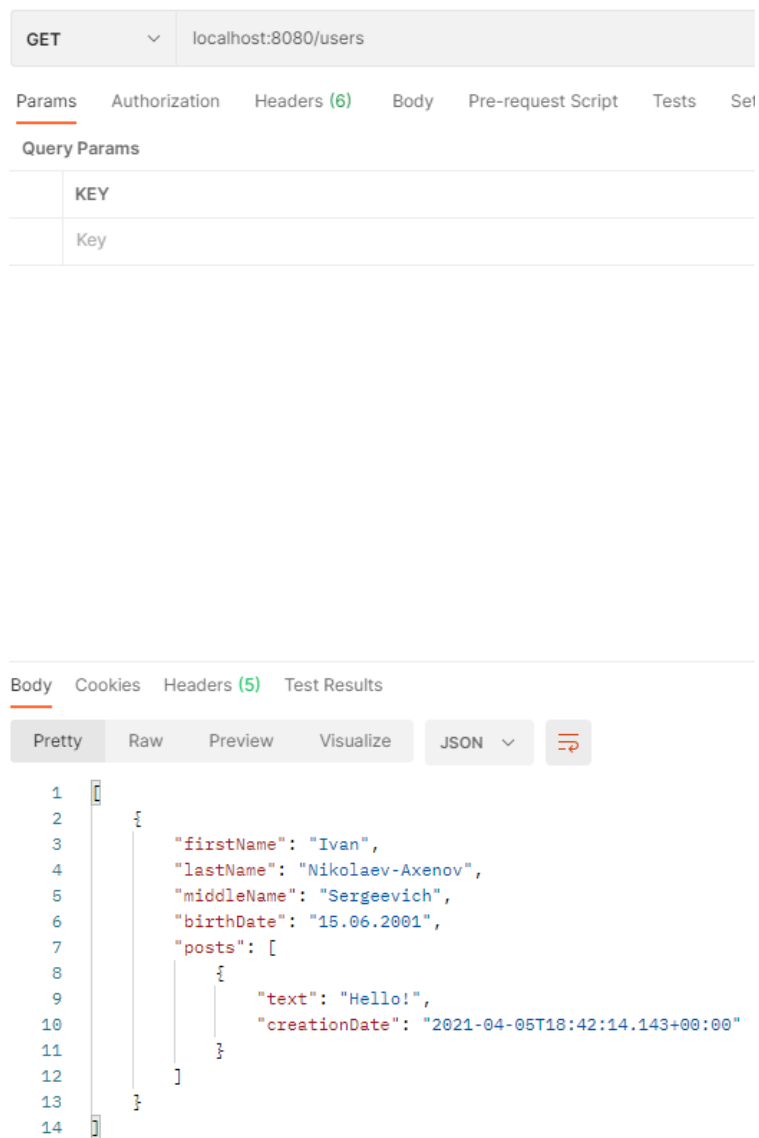


Рисунок 14.4 – Демонстрация работы программы

Практическая работа №15

Цель работы

Тема: Использование Hibernate в Spring framework.

Постановка задачи: Изменить программу с предыдущего задания так, чтобы объекты хранились в базе данных PostgreSQL вместо памяти компьютера.

Листинг программы

Application.java

```
package PR15.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 PR15.Application.model;

import com.sun.istack.NotNull;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.io.Serializable;
import java.util.UUID;

@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;

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 PR15.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;

    @Column(name = "owner")
    @NotNull
    private UUID owner;

    public Post() {
    }

    public Post(String text, UUID owner) {
        this.text = text;
        this.owner = owner;
    }

    public UUID getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public LocalDateTime getCreationDate() {
        return creationDate;
    }

    public UUID getOwner() {
        return owner;
    }
}
```

UserService.java

```
package PR15.Application.service;

import PR15.Application.model.User;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.PostMapping;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.util.List;
import java.util.UUID;

@Service
public class UserService {
    @Autowired
    private final SessionFactory sessionFactory;

    private Session session;

    public UserService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @PostConstruct
    public void init() {
        session = sessionFactory.openSession();
    }

    @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 p 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();
    }
}
```

PostService.java

```
package PR15.Application.service;

import PR15.Application.model.Post;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.util.List;
import java.util.UUID;

@Service
public class PostService {
    @Autowired
    private final SessionFactory sessionFactory;

    private Session session;

    public PostService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @PostConstruct
    public void init() {
        session = sessionFactory.openSession();
    }

    @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 List<Post> getPost(UUID id) {
        return session.createQuery("select p from Post p where p.id = '" + id + "'",
Post.class).list();
    }

    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();
    }
}

```

UserController.java

```

package PR15.Application.controller;

import PR15.Application.model.User;
import PR15.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);
    }
}

```


PostController.java

```
package PR15.Application.controller;

import PR15.Application.model.Post;
import PR15.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> getAll() {
        return postService.getPosts();
    }

    @GetMapping("/post/{id}")
    public List<Post> getPost(@PathVariable UUID id) {
        return postService.getPost(id);
    }

    @DeleteMapping("/post/{id}")
    public void deletePost(@PathVariable UUID id) {
        postService.deletePost(id);
    }
}
```

Config.java

```
package PR15.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/pr15db");
        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("PR15.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

[illegible]

Рисунок 15.1 – Демонстрация работы программы



Рисунок 15.2 – Демонстрация работы программы

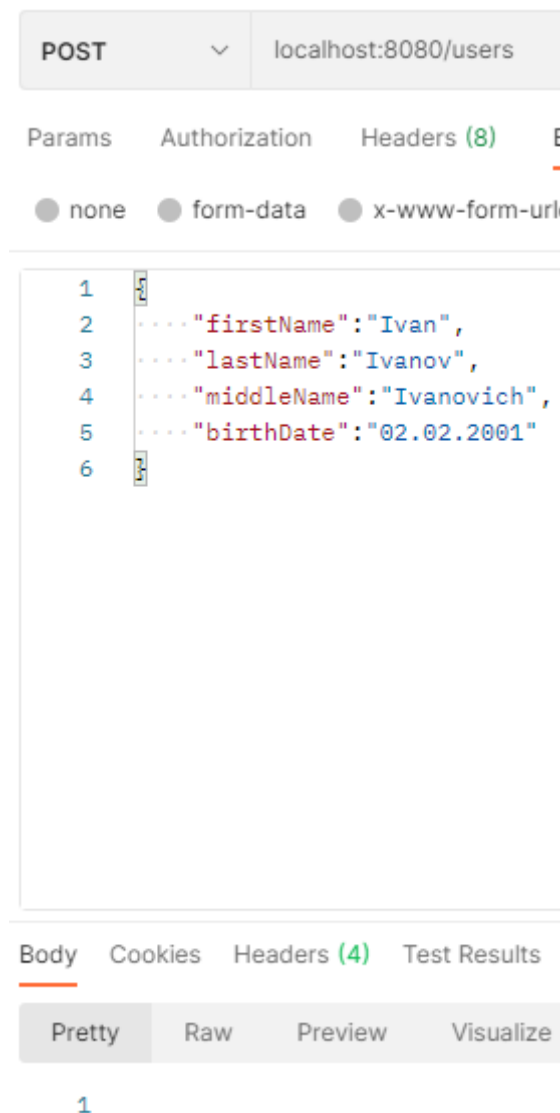


Рисунок 15.3 – Демонстрация работы программы

	id	last_name	first_name	middle_name	birth_date
1	c1e2e561-b3e2-4171-a8b8-e73be815d13f	Nikolaev-Axenov	Ivan	Sergeevich	15.06.2001
2	ada17906-0fb9-446b-b238-b05b05394d0b	Ivanov	Ivan	Ivanovich	02.02.2001

Рисунок 15.4 – Демонстрация работы программы

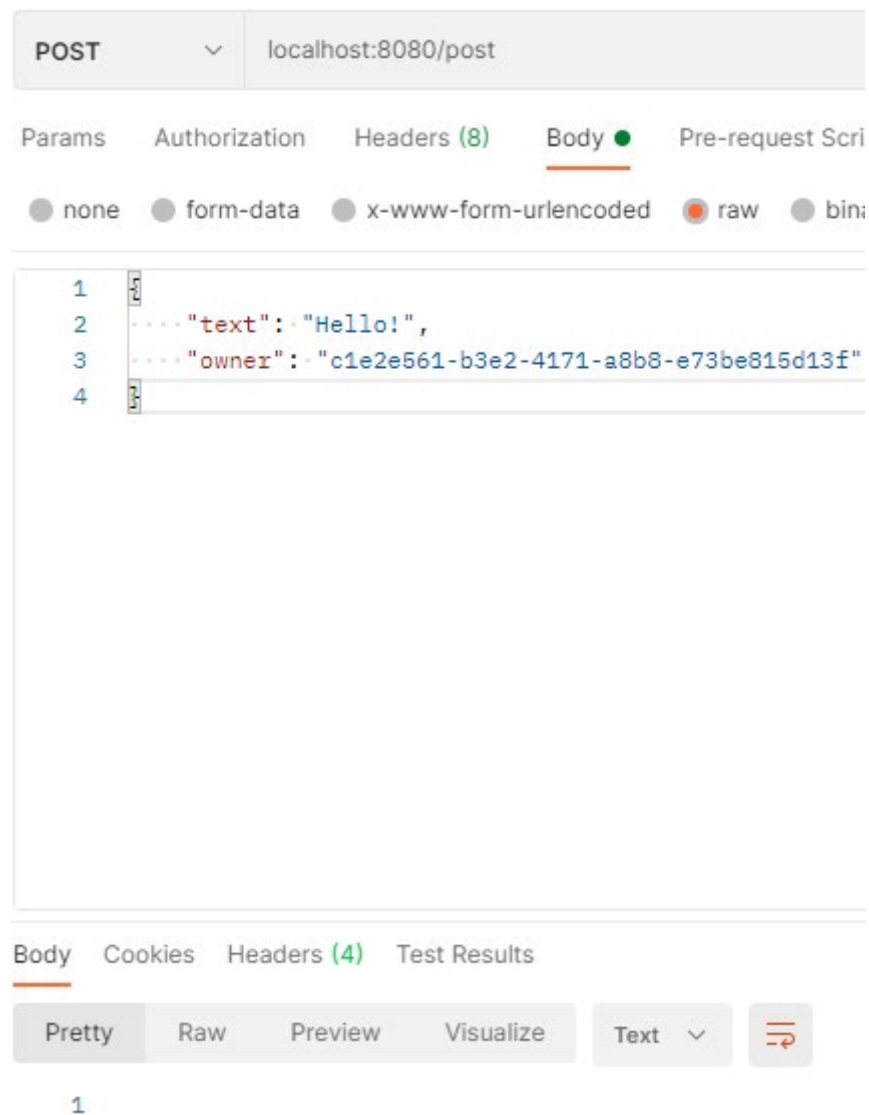


Рисунок 15.5 – Демонстрация работы программы

	id	text	creation_date	owner
1	aed67a07-663d-48eb-8520-0993f7b65019	Hello!	2021-04-05	c1e2e561-b3e2-4171-a8b8-e73be815d13f

Рисунок 15.6 – Демонстрация работы программы

Практическая работа №16

Цель работы

Тема: Изучение видов связей между сущностями в Hibernate. Использование транзакций.

Постановка задачи: Создать связь Один-ко-многим между сущностями из предыдущего задания и проверить работу lazy loading.

Листинг программы

Application.java

```
package PR16.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 PR16.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", cascade = CascadeType.ALL, orphanRemoval = true)
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 void addPost(Post post) {
    posts.add(post);
    post.setUser(this);
}

public void removePost(Post post) {
    posts.remove(post);
    post.setUser(null);
}

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 PR16.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(fetch = FetchType.LAZY)
    private User user;

    public Post() {
    }

    public Post(String text) {
        this.text = text;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public UUID getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public LocalDateTime getCreationDate() {
        return creationDate;
    }
}
```


UserService.java

```
package PR16.Application.service;

import PR16.Application.model.Post;
import PR16.Application.model.User;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.util.List;
import java.util.UUID;

@Service
public class UserService {
    @Autowired
    private final SessionFactory sessionFactory;

    private Session session;

    public UserService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @PostConstruct
    public void init() {
        session = sessionFactory.openSession();
    }

    @PreDestroy
    public void unSession() {
        session.close();
    }

    public void addUser(User user) {
        session.beginTransaction();
        session.saveOrUpdate(user);
        session.getTransaction().commit();
    }

    public void addPost(UUID id, Post post) {
        session.beginTransaction();

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

        session.getTransaction().commit();
    }

    public void removePost(UUID id, Post post) {
        session.beginTransaction();

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

        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();
    }
}

```

PostService.java

```

package PR16.Application.service;

import PR16.Application.model.Post;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.util.List;
import java.util.UUID;

@Service
public class PostService {
    @Autowired
    private final SessionFactory sessionFactory;

    private Session session;

    public PostService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @PostConstruct
    public void init() {
        session = sessionFactory.openSession();
    }

    @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 List<Post> getPost(UUID id) {
        return session.createQuery("select p from Post p where p.id = '" + id + "'",
Post.class).list();
    }

    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();
    }
}

```

UserController.java

```

package PR16.Application.controller;

import PR16.Application.model.Post;
import PR16.Application.model.User;
import PR16.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();
    }
}

```

```

@PostMapping("/userpost/{id}")
public void addPost(@PathVariable UUID id, @RequestBody String text) {
    userService.addPost(id, new Post(text));
}

@DeleteMapping("/userpost/{id}")
public void deletePost(@PathVariable UUID id, Post post) {
    userService.removePost(id, post);
}

@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);
}
}

```

PostController.java

```

package PR16.Application.controller;

import PR16.Application.model.Post;
import PR16.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();
    }

    @GetMapping("/post/{id}")
    public List<Post> getPost(@PathVariable UUID id) {
        return postService.getPost(id);
    }

    @DeleteMapping("/post/{id}")
    public void deletePost(@PathVariable UUID id) {
        postService.deletePost(id);
    }
}

```

Config.java

```
package PR16.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/pr16db");
        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("PR16.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

```
      .   _.._          _._..._.  
    /\ / ____/_.._(C)_____\ \\\ \  
  ((\___|_|'|_|'|\_/_'\\\ \\\  
  (\ ___)|_|_|_|_|_|(|_|)) ) )  
    '|_____||_|_|_|_|_|_|_|_|/  
=====|_|=====|_|=/_/_/_/  
  
:: Spring Boot ::                (v2.4.4)  
  
2021-04-05 22:05:15.072 INFO 20472 --- [main] PR16.Application.Application : Starting Application using Java 11.0.10 on Frischmann-PC with PID 20472  
2021-04-05 22:05:15.077 INFO 20472 --- [main] PR16.Application.Application : No active profile set, falling back to default profiles: default  
2021-04-05 22:05:16.177 INFO 20472 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)  
2021-04-05 22:05:16.188 INFO 20472 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]  
2021-04-05 22:05:16.188 INFO 20472 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.44]  
2021-04-05 22:05:16.302 INFO 20472 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext  
2021-04-05 22:05:16.302 INFO 20472 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1161 ms  
2021-04-05 22:05:16.358 INFO 20472 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...  
2021-04-05 22:05:16.548 INFO 20472 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.  
2021-04-05 22:05:16.691 INFO 20472 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.4.28.Final  
2021-04-05 22:05:16.920 INFO 20472 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}  
2021-04-05 22:05:17.172 INFO 20472 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect  
2021-04-05 22:05:17.813 INFO 20472 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]  
2021-04-05 22:05:17.931 WARN 20472 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may run longer than you want; add spring.jpa.open-in-view=false to your application configuration to turn it off.  
2021-04-05 22:05:18.008 INFO 20472 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'  
2021-04-05 22:05:18.200 INFO 20472 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path '/'  
2021-04-05 22:05:18.211 INFO 20472 --- [main] PR16.Application.Application : Started Application in 3.603 seconds (JVM running for 4.473)
```

Рисунок 15.1 – Демонстрация работы программы

POST

localhost:8080/users

Params

Authorization

Headers (8)

Body

none

form-data

x-www-form-urlencoded

```
1 {  
2   ... "firstName": "Ivan",  
3   ... "lastName": "Nikolaev-Axenov",  
4   ... "middleName": "Sergeevich",  
5   ... "birthDate": "15.06.2001"  
6 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

Text

Рисунок 15.2 – Демонстрация работы программы

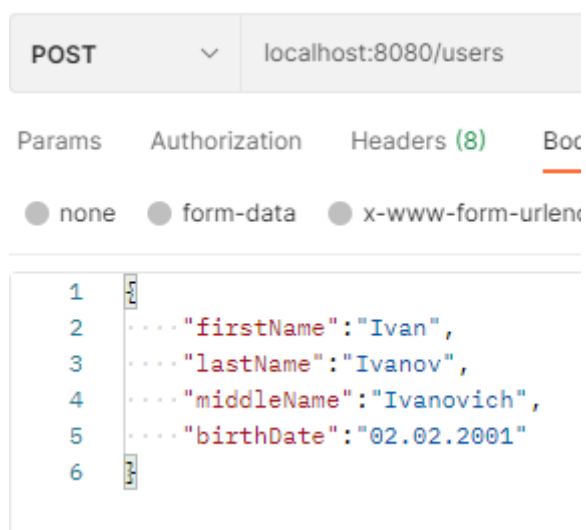


Рисунок 15.3 – Демонстрация работы программы

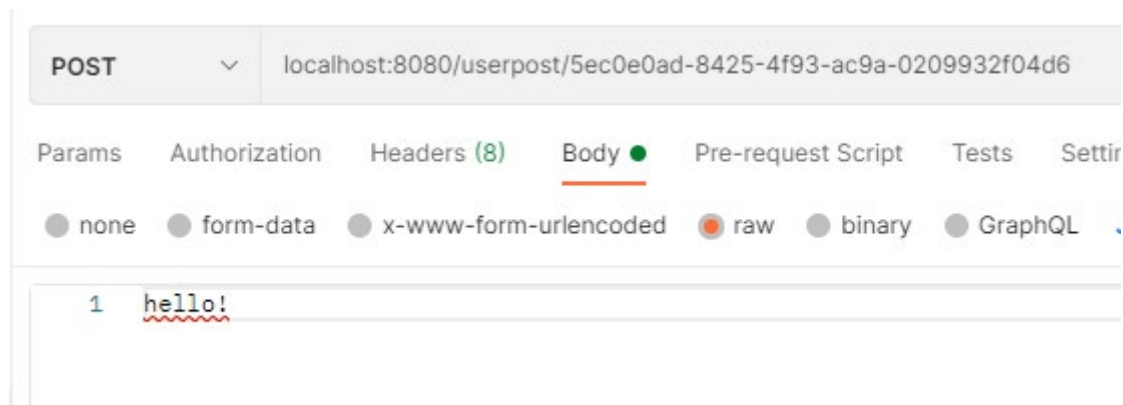


Рисунок 15.4 – Демонстрация работы программы

	id	text	creation_date	user_id
1	0a57064f-be5e-42e3-9c87-6a64bdbf527a	hello!	2021-04-05	5ec0e0ad-8425-4f93-ac9a-0209932f04d6

Рисунок 15.5 – Демонстрация работы программы

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

```

      ____          _            __ _ _
     /  _ \        | |          // \\ \\
    /  ___ \       | |__   _// \\_\\ \\
   /  ___  \      |  __| |//   ___// \\
  /  ___  \      |  | | |//   ___// \\
 /  ___  \      |  |_| |//   ___// \\
/  ___  \      |_____|//   ___// \\
=====|_|=====|_|_/___/_/

:: Spring Boot ::                (v2.4.4)


2021-04-10 15:09:36.311 INFO 9856 --- [main] app.Application.Application : Starting Application using Java 11.0.10 on Frischmann-PC with PID 9856
2021-04-10 15:09:36.314 INFO 9856 --- [main] app.Application.Application : No active profile set, falling back to default profiles: default
2021-04-10 15:09:37.458 INFO 9856 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-04-10 15:09:37.480 INFO 9856 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 11 ms. Found 0 JPA reposi
2021-04-10 15:09:38.169 INFO 9856 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-04-10 15:09:38.182 INFO 9856 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-04-10 15:09:38.182 INFO 9856 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.44]
2021-04-10 15:09:38.327 INFO 9856 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-04-10 15:09:38.328 INFO 9856 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1890 ms
2021-04-10 15:09:38.410 INFO 9856 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2021-04-10 15:09:38.626 INFO 9856 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2021-04-10 15:09:38.753 INFO 9856 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.4.29.Final
2021-04-10 15:09:39.140 INFO 9856 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2021-04-10 15:09:39.374 INFO 9856 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect
2021-04-10 15:09:40.968 INFO 9856 --- [main] o.h.e.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2021-04-10 15:09:41.253 WARN 9856 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may run longer than you need. Read https://github.com/spring-projects/spring-data-jpa/issues/1501 for details.
2021-04-10 15:09:41.365 INFO 9856 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2021-04-10 15:09:41.584 INFO 9856 --- [main] .s.s.UserDetailsServiceAutoConfiguration :

Using generated security password: e34df85d-724d-4c59-9e67-4cf94f423cf6


2021-04-10 15:09:41.709 INFO 9856 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework.security.web.context.request.async.SecurityMetadataContextHolderImpl, org.springframework.security.web.authentication.logout.LogoutFilter, org.springframework.security.web.csrf.CsrfFilter, org.springframework.security.web.header.HeaderWriterFilter, org.springframework.security.web.savedrequest.RequestCacheAwareFilter, org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter, org.springframework.security.web.util.matcher.AntPathRequestMatcher, org.springframework.security.web.access.ExceptionTranslationFilter, org.springframework.security.web.access.intercept.AuthorizationFilter, org.springframework.security.web.access.authorization.method.AuthorizedRequestMethodNotPermitted, org.springframework.security.web.access.authorization.method.RequireAuthorizationFilter, org.springframework.security.web.access.authorization.method.RequireFullyAuthenticated, org.springframework.security.web.access.authorization.method.RequireLoggedIn, org.springframework.security.web.access.authorization.method.RequirePersistentSession, org.springframework.security.web.access.authorization.method.RequireValidCredentials, org.springframework.security.web.access.authorization.method.RequireValidMethod, org.springframework.security.web.access.authorization.method.RequireValidOrigin, org.springframework.security.web.access.authorization.method.RequireValidScheme, org.springframework.security.web.access.authorization.method.RequireValidUserAgent, org.springframework.security.web.access.authorization.method.RequireValidXFrameOptions, org.springframework.security.web.access.authorization.method.RequireValidXssProtection, org.springframework.security.web.access.authorization.method.RequireValidXxssProtection]
2021-04-10 15:09:41.850 INFO 9856 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-04-10 15:09:41.861 INFO 9856 --- [main] app.Application.Application : Started Application in 8.508 seconds (JVM running for 10.932)
```

Рисунок 17.1 – Демонстрация работы программы

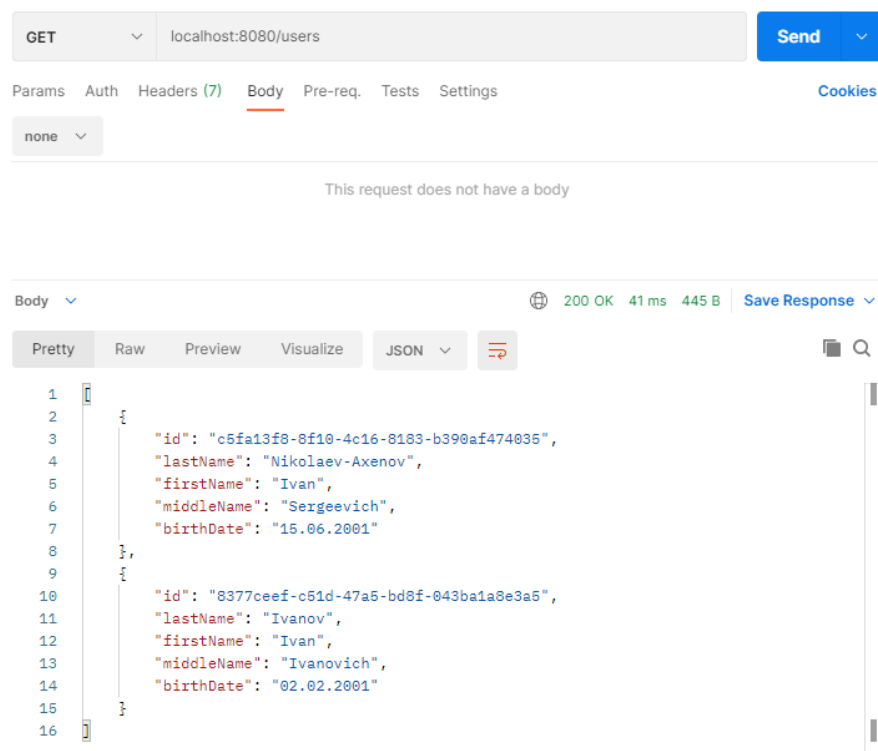


Рисунок 17.2 – Демонстрация работы программы

GET localhost:8080/getByLastName

Params Auth Headers (7) Body Pre-req. Tests Settings

Query Params

KEY	VALUE
Key	Value

Body 200 C

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": "8377ceef-c51d-47a5-bd8f-043ba1a8e3a5",
4     "lastName": "Ivanov",
5     "firstName": "Ivan",
6     "middleName": "Ivanovich",
7     "birthDate": "02.02.2001"
8   },
9   {
10    "id": "c5fa13f8-8f10-4c16-8183-b390af474035",
11    "lastName": "Nikolaev-Axenov",
12    "firstName": "Ivan",
13    "middleName": "Sergeevich",
14    "birthDate": "15.06.2001"
15  }
16 ]
```

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

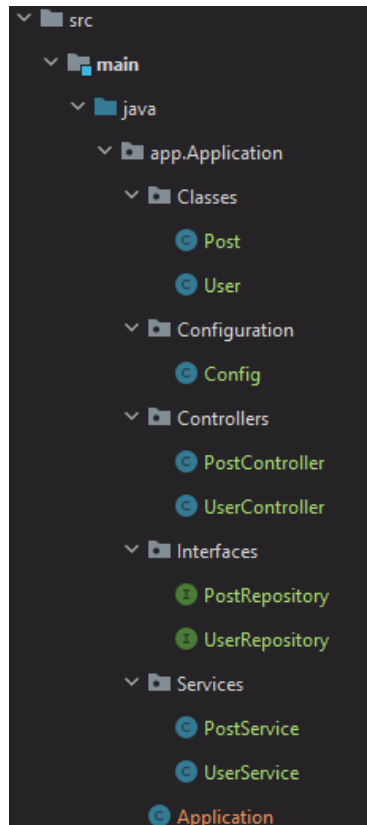


Рисунок 18.1 – Демонстрация работы программы

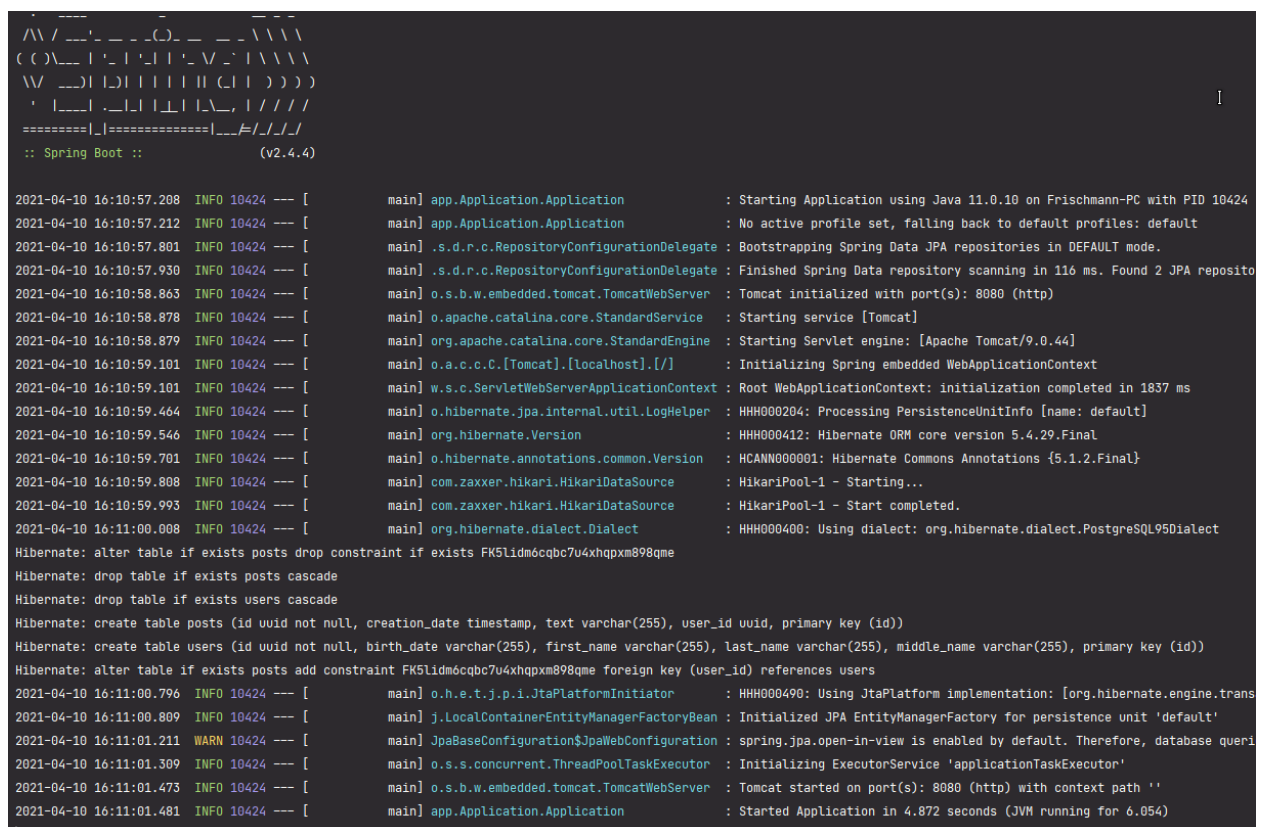


Рисунок 18.2 – Демонстрация работы программы

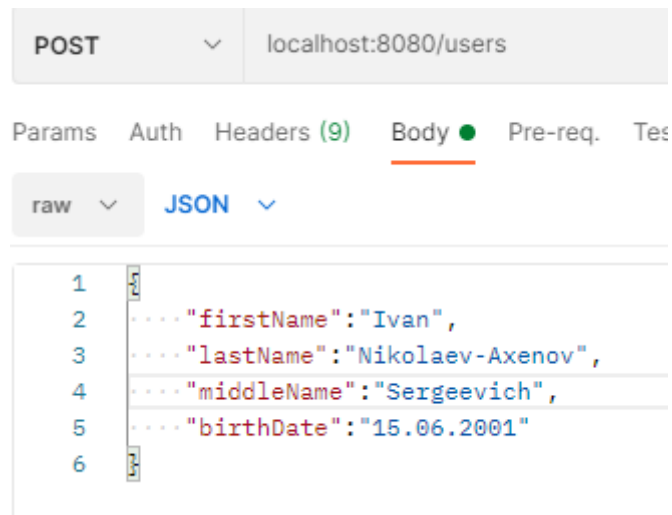


Рисунок 18.3 – Демонстрация работы программы

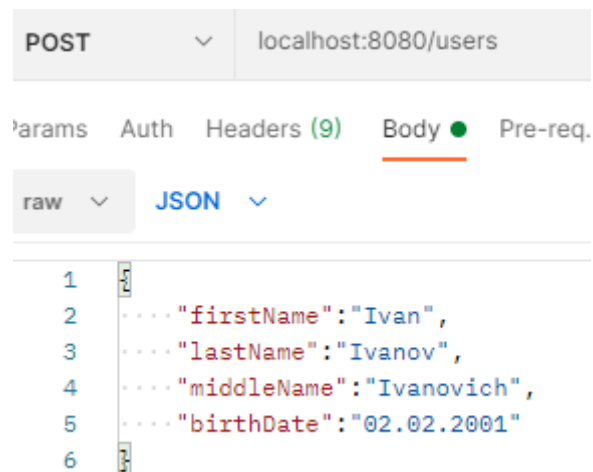


Рисунок 18.4 – Демонстрация работы программы

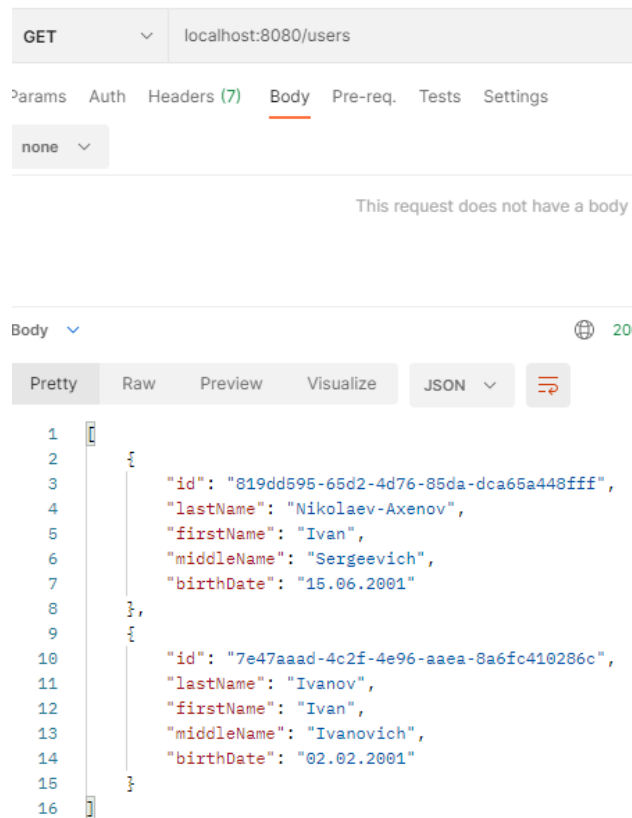


Рисунок 18.5 – Демонстрация работы программы

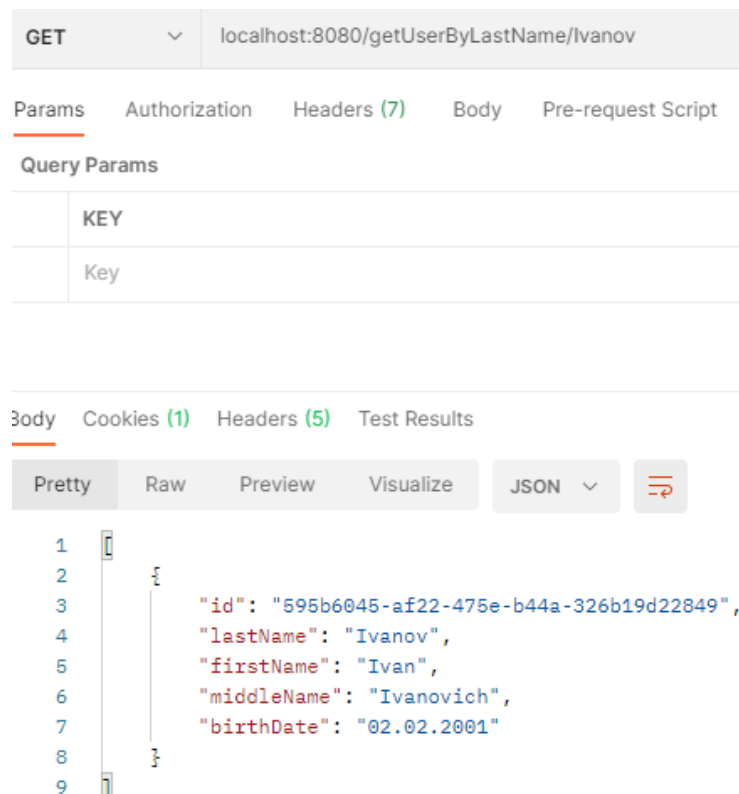


Рисунок 18.6 – Демонстрация работы программы

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

```

  /\_/\  _/_/ _/_/ _/_/ _/_/ _/_/
 ( ( )\__/_/ _/_/ _/_/ _/_/ _/_/ _/_/
  \/_/ _/_/ _/_/ _/_/ _/_/ _/_/ _/_/
   ' _/_/ _/_/ _/_/ _/_/ _/_/ _/_/
  =====|_|=====|_|_/_/_/_/
  :: Spring Boot ::                (v2.4.4)

17:03:07.898 [main] INFO
    app.Application.Application - Starting Application using Java 11.0.10 on Frischmann-PC with PID 19212 (C:\Users\frisc\GoogleDrive
17:03:07.914 [main] INFO
    app.Application.Application - No active profile set, falling back to default profiles: default
17:03:08.920 [main] INFO
    o.s.d.r.c.RepositoryConfigurationDelegate - Bootstrapping Spring Data JPA repositories in DEFAULT mode.
17:03:09.048 [main] INFO
    o.s.d.r.c.RepositoryConfigurationDelegate - Finished Spring Data repository scanning in 97 ms. Found 2 JPA repository interfaces.
17:03:09.805 [main] INFO
    o.s.b.w.e.tomcat.TomcatWebServer - Tomcat initialized with port(s): 8080 (http)
17:03:09.821 [main] INFO
    o.a.coyote.http11.Http11NioProtocol - Initializing ProtocolHandler ["http-nio-8080"]
17:03:09.821 [main] INFO
    o.a.catalina.core.StandardService - Starting service [Tomcat]
17:03:09.821 [main] INFO
    o.a.catalina.core.StandardEngine - Starting Servlet engine: [Apache Tomcat/9.0.44]
17:03:09.985 [main] INFO
    o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext
17:03:09.986 [main] INFO
    o.s.b.w.s.c.ServletWebServerApplicationContext - Root WebApplicationContext: initialization completed in 1976 ms
17:03:10.309 [main] INFO
    o.h.jpa.internal.util.LogHelper - HHH000204: Processing PersistenceUnitInfo [name: default]
17:03:10.379 [main] INFO
    org.hibernate.Version - HHH000412: Hibernate ORM core version 5.4.29.Final
17:03:10.562 [main] INFO
    o.h.annotations.common.Version - HCANW000001: Hibernate Commons Annotations {5.1.2.Final}
17:03:10.721 [main] INFO
    com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
17:03:10.979 [main] INFO
```

Рисунок 19.1 – Демонстрация работы программы

The image shows a REST client interface. At the top, the method is set to 'POST' and the URL is 'localhost:8080/users'. Below this, there are tabs for 'Params', 'Authorization', 'Headers (9)', and 'Body'. The 'Body' tab is selected and underlined. Under the 'Body' tab, there are three radio buttons: 'none', 'form-data', and 'x-www-form-urlencoded'. The 'none' radio button is selected. The body content is a JSON object with the following fields: 'firstName' with value 'Ivan', 'lastName' with value 'Nikolaev-Axenov', 'middleName' with value 'Sergeevich', and 'birthDate' with value '15.06.2001'. The JSON is displayed in a code editor with line numbers 1 through 6.

```
1 {
2   .... "firstName": "Ivan",
3   .... "lastName": "Nikolaev-Axenov",
4   .... "middleName": "Sergeevich",
5   .... "birthDate": "15.06.2001"
6 }
```

Рисунок 19.2 – Демонстрация работы программы

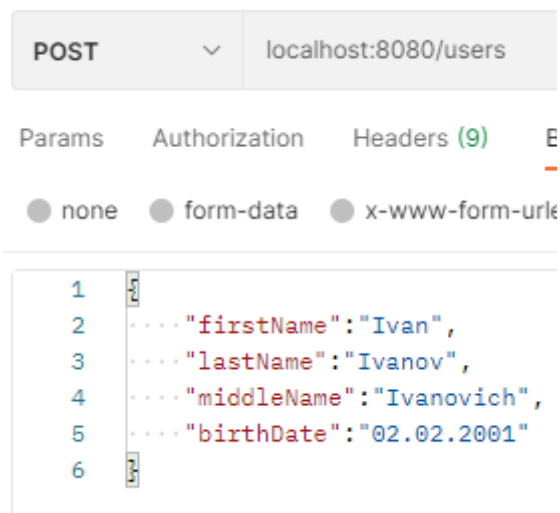


Рисунок 19.3 – Демонстрация работы программы

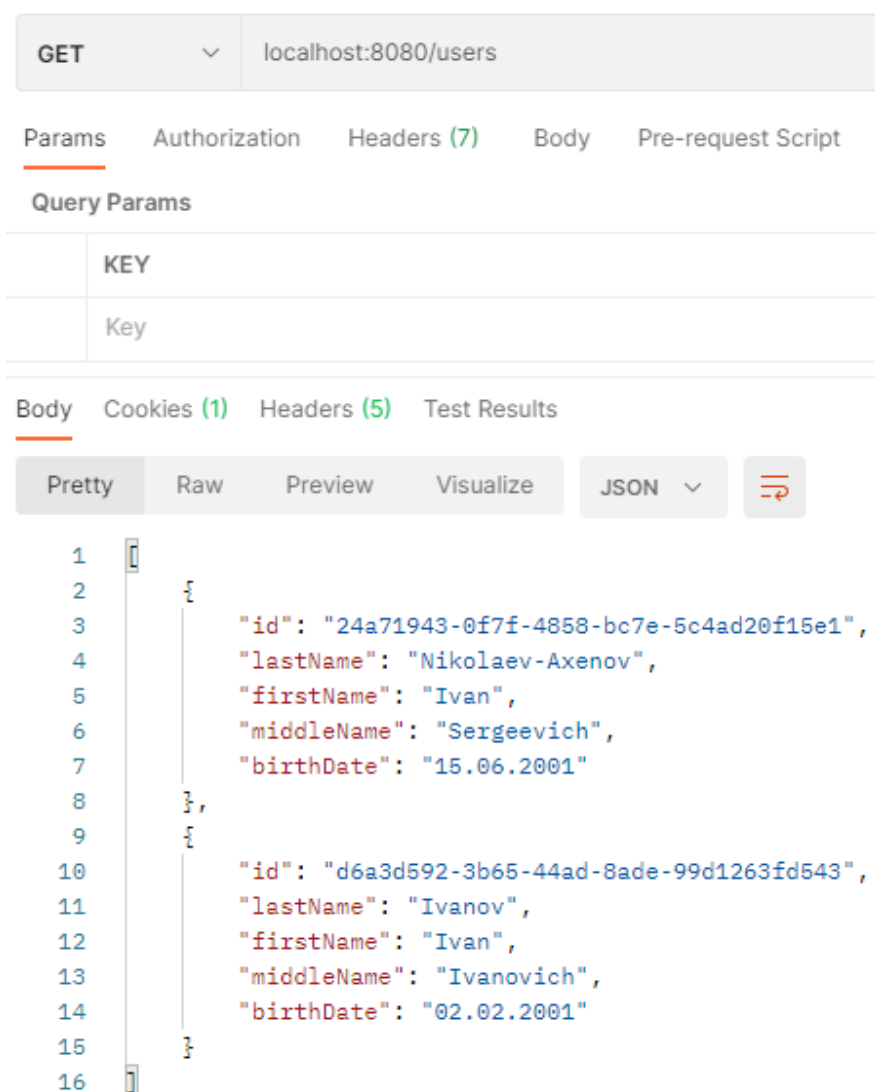


Рисунок 19.4 – Демонстрация работы программы

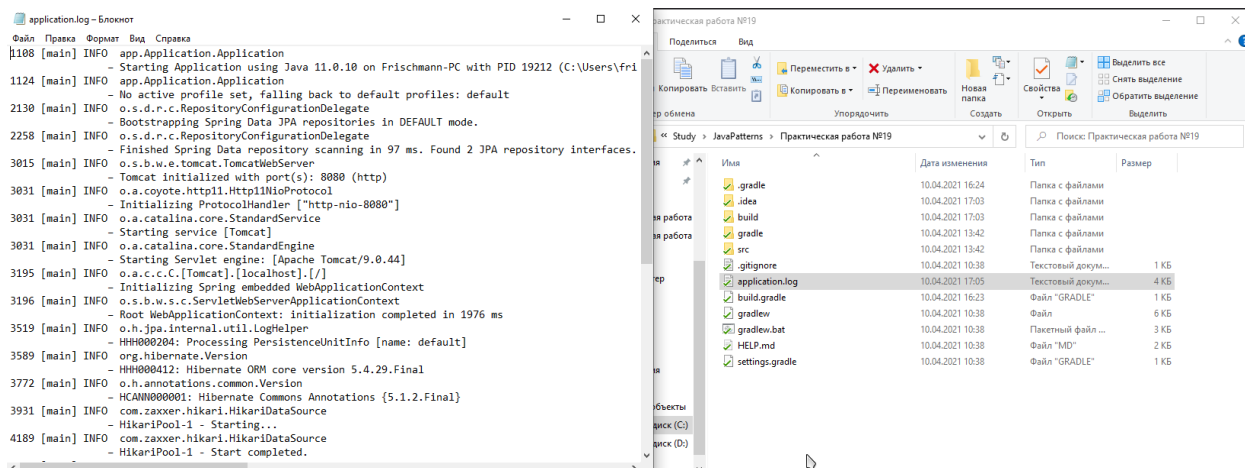


Рисунок 19.5 – Демонстрация работы программы

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

```
( ) \_ _ | _ | '_| |'_\ V'_| \| \\ \ \  
\\/_ __) | |_)| ||||| H |( |) )) ) )  
'| ____|_.|_| |_| | |\_, / / / /  
=====|_|=====|__#/. /. ./.  
  
:: Spring Boot ::           (v2.4.4)  
  
17:21:00.548 [main] INFO  
    app.Application.Application - Starting Application using Java 11.0.10 on Frischmann-PC with PID 6064 (C:\Users\frisc\GoogleDrive\St  
17:21:00.551 [main] INFO  
    app.Application.Application - No active profile set, falling back to default profiles: default  
17:21:01.174 [main] INFO  
    o.s.d.r.c.RepositoryConfigurationDelegate - Bootstrapping Spring Data JPA repositories in DEFAULT mode.  
17:21:01.252 [main] INFO  
    o.s.d.r.c.RepositoryConfigurationDelegate - Finished Spring Data repository scanning in 65 ms. Found 2 JPA repository interfaces.  
17:21:02.514 [main] INFO  
    o.s.b.w.e.tomcat.TomcatWebServer - Tomcat initialized with port(s): 8080 (http)  
17:21:02.524 [main] INFO  
    o.a.coyote.http11.Http11NioProtocol - Initializing ProtocolHandler ["http-nio-8080"]  
17:21:02.525 [main] INFO  
    o.a.catalina.core.StandardService - Starting service [Tomcat]  
17:21:02.525 [main] INFO  
    o.a.catalina.core.StandardEngine - Starting Servlet engine: [Apache Tomcat/9.0.44]  
17:21:02.650 [main] INFO  
    o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext  
17:21:02.651 [main] INFO  
    o.s.b.w.s.c.ServletWebServerApplicationContext - Root WebApplicationContext: initialization completed in 2042 ms  
17:21:02.896 [main] INFO  
    o.h.jpa.internal.util.LogHelper - HHH000204: Processing PersistenceUnitInfo [name: default]  
17:21:02.965 [main] INFO  
    org.hibernate.Version - HHH000412: Hibernate ORM core version 5.4.29.Final  
17:21:03.205 [main] INFO  
    o.h.annotations.common.Version - HCANN000001: Hibernate Commons Annotations {5.1.2.Final}  
17:21:03.467 [main] INFO  
    com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...  
17:21:03.695 [main] INFO
```

Рисунок 20.1 – Демонстрация работы программы

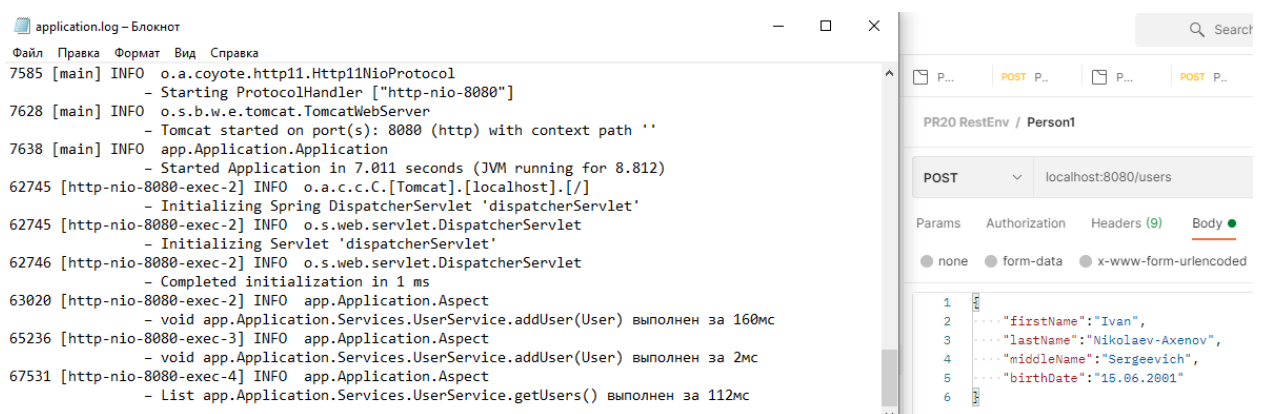


Рисунок 20.2 – Демонстрация работы программы

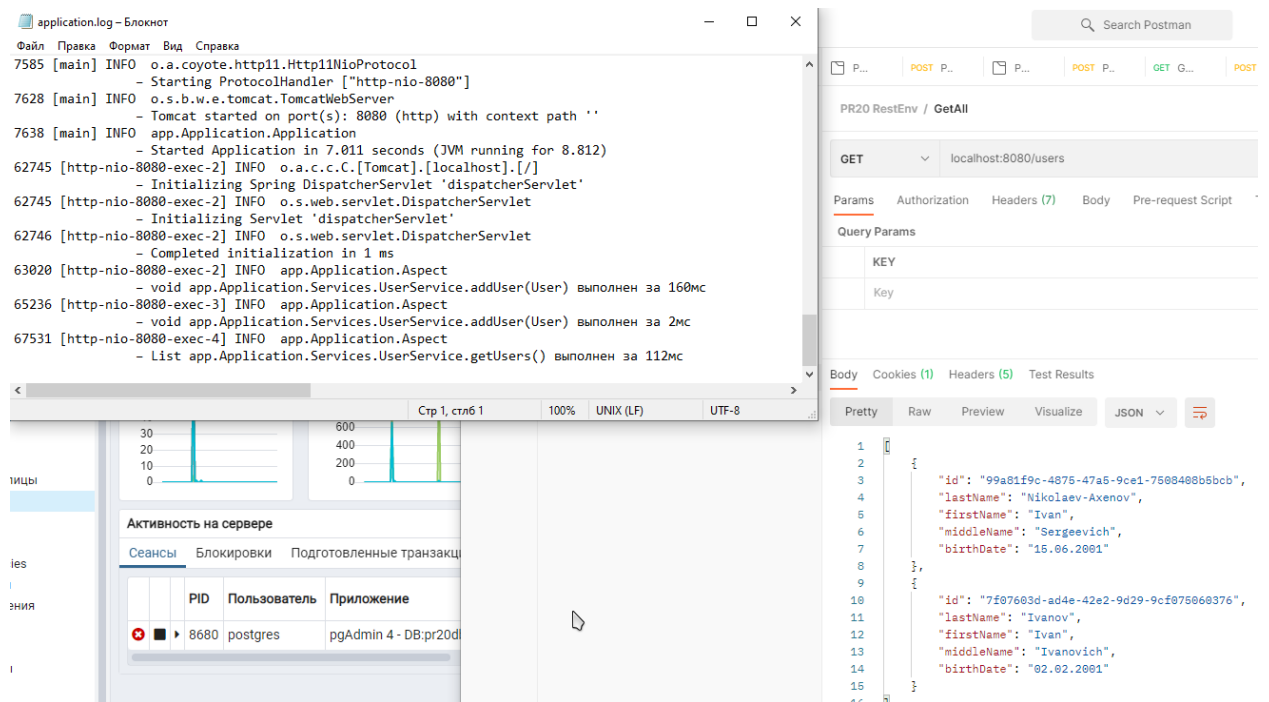


Рисунок 20.3 – Демонстрация работы программы

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

```

  ____  _
 / __ \| | | |
( (___ \| |_| |
 \___) /____|_|_|
      | | | |
      | |_| |
      |_|_|_|

:: Spring Boot ::                (v2.4.4)

23:21:25.220 [main] INFO
    app.Application.Application - Starting Application using Java 11.0.10 on Frischmann-PC with PID 5916 (C:\Users\frisc\GoogleDrive\Study\JavaPat
23:21:25.222 [main] INFO
    app.Application.Application - No active profile set, falling back to default profiles: default
23:21:25.806 [main] INFO
    o.s.d.r.c.RepositoryConfigurationDelegate - Bootstrapping Spring Data JPA repositories in DEFAULT mode.
23:21:25.886 [main] INFO
    o.s.d.r.c.RepositoryConfigurationDelegate - Finished Spring Data repository scanning in 67 ms. Found 2 JPA repository interfaces.
23:21:26.787 [main] INFO
    o.s.b.w.e.tomcat.TomcatWebServer - Tomcat initialized with port(s): 8080 (http)
23:21:26.799 [main] INFO
    o.a.coyote.http11.Http11NioProtocol - Initializing ProtocolHandler ["http-nio-8080"]
23:21:26.800 [main] INFO
    o.a.catalina.core.StandardService - Starting service [Tomcat]
23:21:26.800 [main] INFO
    o.a.catalina.core.StandardEngine - Starting Servlet engine: [Apache Tomcat/9.0.44]
23:21:26.922 [main] INFO
    o.a.c.c.c.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext
23:21:26.922 [main] INFO
    o.s.b.w.s.c.ServletWebServerApplicationContext - Root WebApplicationContext: initialization completed in 1655 ms
23:21:27.140 [main] INFO
    o.h.jpa.internal.util.LogHelper - HHH000204: Processing PersistenceUnitInfo [name: default]
23:21:27.275 [main] INFO
    org.hibernate.Version - HHH000412: Hibernate ORM core version 5.4.29.Final
23:21:27.460 [main] INFO
    o.h.annotations.common.Version - HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
```

Рисунок 21.1 – Демонстрация работы программы

POST

localhost:8080/users

Params

Authorization

Headers (8)

Body

☐ none

☐ form-data

☐ x-www-form-urlencoded

1

2

3

4

5

6

.....

.....

.....

.....

.....

.....

"firstName": "Ivan",

"lastName": "Nikolaev-Axenov",

"middleName": "Sergeevich",

"birthDate": "15.06.2001"

Рисунок 21.2 – Демонстрация работы программы

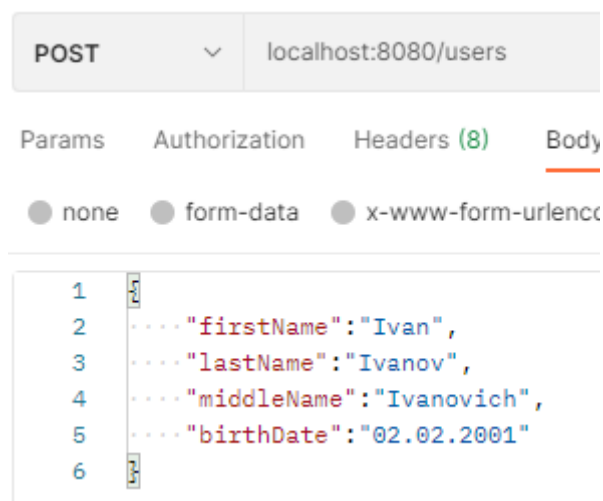


Рисунок 21.3 – Демонстрация работы программы

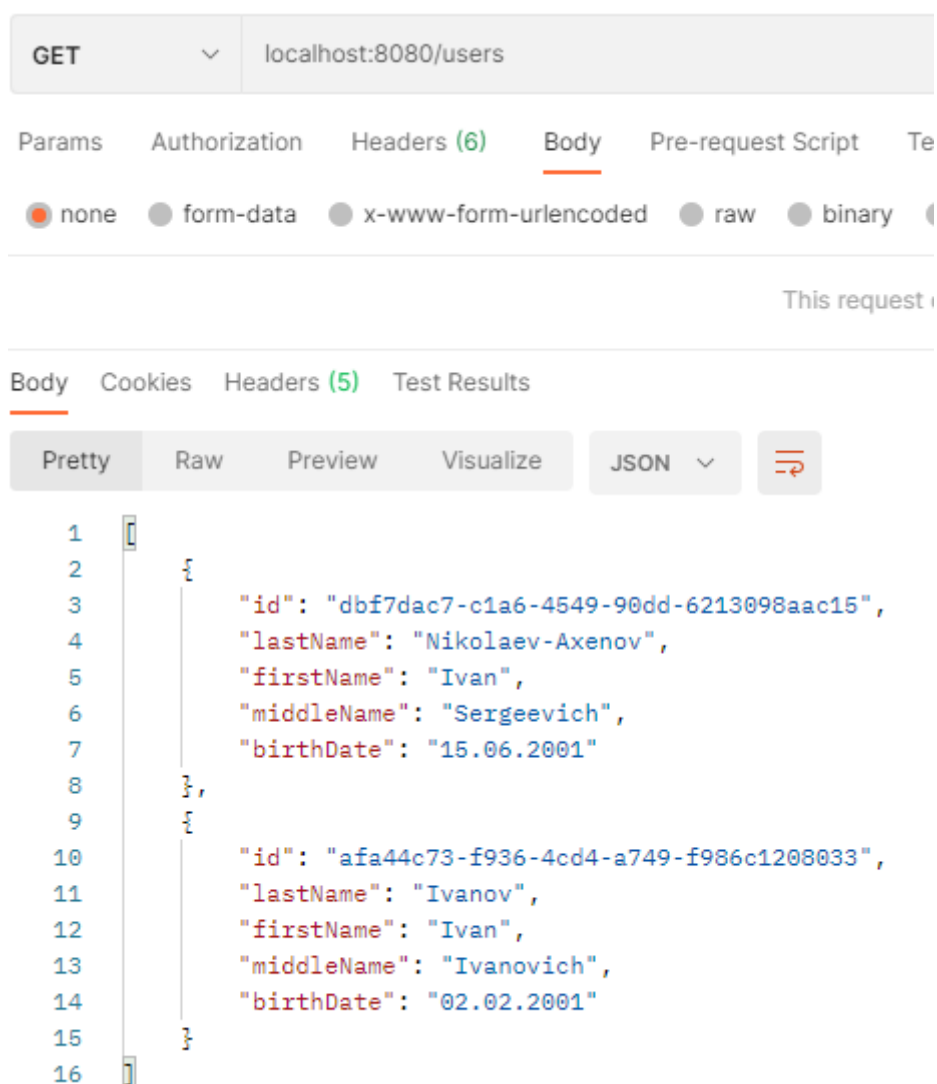


Рисунок 21.4 – Демонстрация работы программы

```

DEBUG SMTP: Found extension "PIPELINING", arg
DEBUG SMTP: Found extension "AUTH", arg "PLAIN LOGIN XOAUTH2"
DEBUG SMTP: protocolConnect login, host=smtp.mail.ru, user=lorememail@bk.ru, password=<non-null>
DEBUG SMTP: Attempt to authenticate using mechanisms: LOGIN PLAIN DIGEST-MD5 NTLM XOAUTH2
DEBUG SMTP: Using mechanism LOGIN
DEBUG SMTP: AUTH LOGIN command trace suppressed
DEBUG SMTP: AUTH LOGIN succeeded
DEBUG SMTP: use8bit false
MAIL FROM:<lorememail@bk.ru>
250 OK
RCPT TO:<ghost777t@ya.ru>
250 Accepted
DEBUG SMTP: Verified Addresses
DEBUG SMTP:  ghost777t@ya.ru
DATA
354 Enter message, ending with "." on a line by itself
Date: Sat, 10 Apr 2021 23:35:43 +0300 (MSK)
From: lorememail@bk.ru
To: ghost777t@ya.ru
Message-ID: <1849625555.0.1618086943511@Frischmann-PC>
Subject: Test email message
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla feugiat eget sapien sed lacinia.
.
250 OK id=1LVKKV-0008Pz-Bw
DEBUG SMTP: message successfully delivered to mail server
QUIT
221 smtp57.i.mail.ru closing connection
Email successfully sent!
Hibernate: select user0_.id as id1_1_, user0_.birth_date as birth_da2_1_, user0_.first_name as first_na3_1_, user0_.last_name as last_na4_1_
23:35:43.956 [http-nio-8080-exec-3] INFO
app.Application.Aspect - List app.Application.Services.UserService.getUsers() выполнен за 111мс

```

Рисунок 21.5 – Демонстрация работы программы

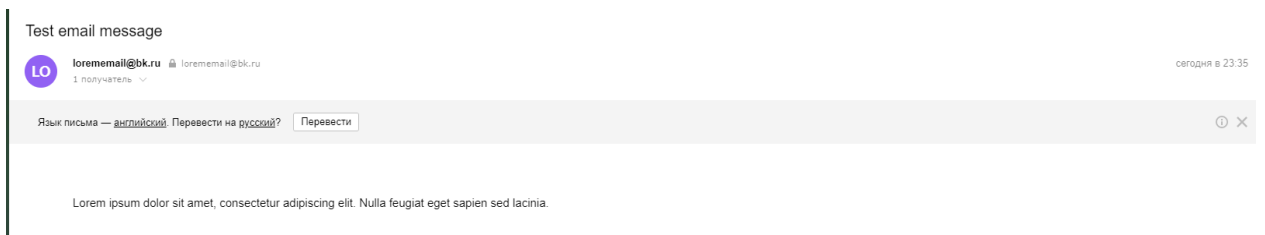


Рисунок 21.6 – Демонстрация работы программы

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

[illegible]

Рисунок 22.1 – Демонстрация работы программы

GET

localhost:8080/users

Params

Authorization

Headers (6)

Body

Pre-request Script

Test Results

Query Params

KEY
Key

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```

1  [
2      {
3          "id": "62b41408-9abd-11eb-abb6-ffc39c4642e7",
4          "lastName": "Nikolaev-Axenov",
5          "firstName": "Ivan",
6          "middleName": "Sergeevich",
7          "birthDate": "15.06.2001"
8      },
9      {
10         "id": "62b4621e-9abd-11eb-abb7-c73b0245ed12",
11         "lastName": "Ivanov",
12         "firstName": "Ivan",
13         "middleName": "Ivanovich",
14         "birthDate": "01.01.2000"
15     }
16 ]

```

Рисунок 22.2 – Демонстрация работы программы

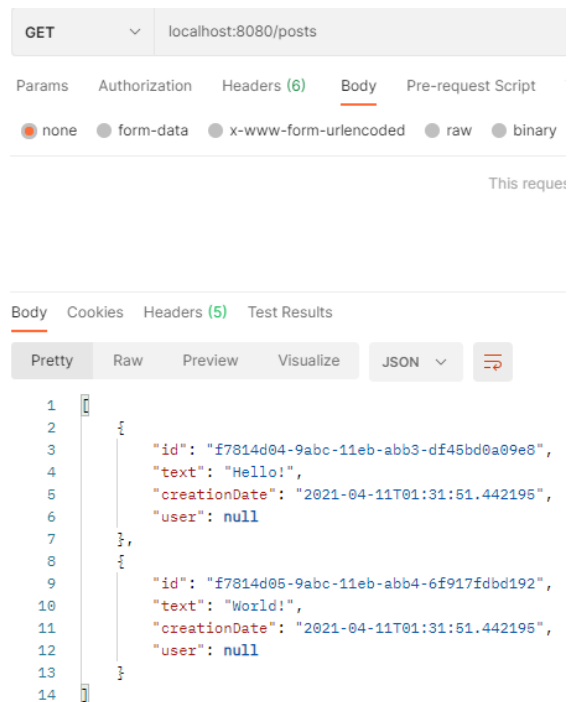


Рисунок 22.3 – Демонстрация работы программы

```

Hibernate: select user0_.id as id1_1_, user0_.birth_date as birth_da2_1_, user0_.first_name as first_na3_1_, user0_.last_name as last_na4_1_, user0_.middle_name as middle_n5_1_ from users user0_
14:59:43.869 [http-nio-8080-exec-1] INFO
app.Application.Aspect - List app.Application.Services.UserService.getUsers() выполнен за 142мс
Hibernate: select post0_.id as id1_0_, post0_.creation_date as creation2_0_, post0_.text as text3_0_, post0_.user_id as user_id4_0_ from posts post0_
Hibernate: select user0_.id as id1_1_, user0_.birth_date as birth_da2_1_, user0_.first_name as first_na3_1_, user0_.last_name as last_na4_1_, user0_.middle_name as middle_n5_1_ from users user0_
User{id=62b41408-9abd-11eb-abb6-ffc39c4642e7, lastName='Nikolaev-Axenov', firstName='Ivan', middleName='Sergeevich', birthDate='15.06.2001'}
15:00:00.035 [scheduled-1] INFO |
app.Application.Aspect - void app.Application.Services.Schedule.doScheduledTask() выполнен за 31мс

```

Рисунок 22.4 – Демонстрация работы программы

JavaPatterns > Практическая работа №22 > testDirectory					Поиск: testDirectory	
	Имя	Дата изменения	Тип	Размер		
Доступ	post_0.txt	11.04.2021 15:00	Файл "TXT"	1 КБ		
	post_1.txt	11.04.2021 15:00	Файл "TXT"	1 КБ		
	user_0.txt	11.04.2021 15:00	Файл "TXT"	1 КБ		
	user_1.txt	11.04.2021 15:00	Файл "TXT"	1 КБ		

Рисунок 22.5 – Демонстрация работы программы

```

1 |User{id=62b41408-9abd-11eb-abb6-ffc39c4642e7, lastName='Nikolaev-Axenov', firstName='Ivan', middleName='Sergeevich', birthDate='15.06.2001'}

```

Рисунок 22.6 – Демонстрация работы программы

Практическая работа №23

Цель работы

Тема: Использование Spring Security для аутентификации и авторизации пользователей.

Постановка задачи: В приложении из предыдущего задания добавить возможность регистрации и авторизации пользователей, хранение cookie сессий в базе данных PostgreSQL, хеширование паролей алгоритмом Bcrypt, защиту всех запросов, кроме запросов на авторизацию и регистрацию, от неавторизованных пользователей.

Листинг программы

User.java

```
package app.Application.Classes;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import javax.validation.constraints.Size;
import java.util.Collection;
import java.util.Set;

@Entity
@Table(name = "t_user")
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Size(min=2, message = "Не меньше 5 знаков")
    private String username;

    @Size(min=2, message = "Не меньше 5 знаков")
    private String password;

    @Transient
    private String passwordConfirm;

    @ManyToMany(fetch = FetchType.EAGER)
    private Set<Role> roles;

    public User() {
    }

    public Long getId() {
        return id;
    }

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

    @Override
```

```

public String getUsername() {
    return username;
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return true;
}

public void setUsername(String username) {
    this.username = username;
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    return getRoles();
}

@Override
public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getPasswordConfirm() {
    return passwordConfirm;
}

public void setPasswordConfirm(String passwordConfirm) {
    this.passwordConfirm = passwordConfirm;
}

public Set<Role> getRoles() {
    return roles;
}

public void setRoles(Set<Role> roles) {
    this.roles = roles;
}
}

```

Role.java

```
package app.Application.Classes;

import org.springframework.security.core.GrantedAuthority;

import javax.persistence.*;
import java.util.Set;

@Entity
@Table(name = "t_role")
public class Role implements GrantedAuthority {
    @Id
    private Long id;

    private String name;

    @Transient
    @ManyToMany(mappedBy = "roles")
    private Set<User> users;

    public Role() {
    }

    public Role(Long id) {
        this.id = id;
    }

    public Role(Long id, String name) {
        this.id = id;
        this.name = name;
    }

    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 Set<User> getUsers() {
        return users;
    }

    public void setUsers(Set<User> users) {
        this.users = users;
    }

    @Override
    public String getAuthority() {
        return getName();
    }
}
```

UserRepository.java

```
package app.Application.Interfaces;

import app.Application.Classes.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

RoleRepository.java

```
package app.Application.Interfaces;

import app.Application.Classes.Role;
import org.springframework.data.jpa.repository.JpaRepository;

public interface RoleRepository extends JpaRepository<Role, Long> {
}
```

UserService.java

```
package app.Application.Services;

import app.Application.Classes.Role;
import app.Application.Classes.User;
import app.Application.Interfaces.RoleRepository;
import app.Application.Interfaces.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import java.util.Collections;
import java.util.List;
import java.util.Optional;

@Service
public class UserService implements UserDetailsService {
    @PersistenceContext
    private EntityManager em;

    @Autowired
    UserRepository userRepository;

    @Autowired
    RoleRepository roleRepository;

    @Autowired
    BCryptPasswordEncoder bCryptPasswordEncoder;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }
}
```

```

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userRepository.findByUsername(username);

        if (user == null) {
            throw new UsernameNotFoundException("User not found");
        }

        return user;
    }

    public User findUserById(Long userId) {
        Optional<User> userFromDb = userRepository.findById(userId);
        return userFromDb.orElse(new User());
    }

    public List<User> allUsers() {
        return userRepository.findAll();
    }

    public boolean saveUser(User user) {
        User userFromDB = userRepository.findByUsername(user.getUsername());

        if (userFromDB != null) {
            return false;
        }

        user.setRoles(Collections.singleton(new Role(1L, "ROLE_USER")));
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        userRepository.save(user);
        return true;
    }

    public boolean deleteUser(Long userId) {
        if (userRepository.findById(userId).isPresent()) {
            userRepository.deleteById(userId);
            return true;
        }
        return false;
    }

    public List<User> usertList(Long idMin) {
        return em.createQuery("SELECT u FROM User u WHERE u.id > :paramId", User.class)
            .setParameter("paramId", idMin).getResultList();
    }
}

```

AdminController.java

```
package app.Application.Controllers;

import app.Application.Services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class AdminController {
    @Autowired
    private UserService userService;

    @GetMapping("/admin")
    public String userList(Model model) {
        model.addAttribute("allUsers", userService.allUsers());
        return "admin";
    }

    @PostMapping("/admin")
    public String deleteUser(@RequestParam(required = true, defaultValue = "" ) Long
userId,
                           @RequestParam(required = true, defaultValue = "" ) String
action,
                           Model model) {
        if (action.equals("delete")){
            userService.deleteUser(userId);
        }
        return "redirect:/admin";
    }

    @GetMapping("/admin/gt/{userId}")
    public String gtUser(@PathVariable("userId") Long userId, Model model) {
        model.addAttribute("allUsers", userService.usergtList(userId));
        return "admin";
    }
}
```

CookieController.java

```
package app.Application.Controllers;

import org.springframework.web.bind.annotation.RestController;

@RestController
public class CookieController {
}
```

RegistrationController.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.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

import javax.validation.Valid;

@Controller
public class RegistrationController {

    @Autowired
    private UserService userService;

    @GetMapping("/registration")
    public String registration(Model model) {
        model.addAttribute("userForm", new User());

        return "registration";
    }

    @PostMapping("/registration")
    public String addUser(@ModelAttribute("userForm") @Valid User userForm,
        BindingResult bindingResult, Model model) {

        if (bindingResult.hasErrors()) {
            return "registration";
        }
        if (!userForm.getPassword().equals(userForm.getPasswordConfirm())){
            model.addAttribute("passwordError", "Пароли не совпадают");
            return "registration";
        }
        if (!userService.saveUser(userForm)){
            model.addAttribute("usernameError", "Пользователь с таким именем уже
существует");
            return "registration";
        }

        return "redirect:/";
    }
}
```


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;
import org.springframework.scheduling.annotation.EnableAsync;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration
@EnableAspectJAutoProxy
@EnableJpaRepositories(basePackages = {"app.Application"})
@EnableAsync
public class Config extends WebSecurityConfigurerAdapter {
}
```

MvcConfig.java

```
package app.Application.Configuration;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class MvcConfig implements WebMvcConfigurer {

    @Override
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/login").setViewName("login");
        registry.addViewController("/news").setViewName("news");
    }
}
```

WebSecurityConfig.java

```
package app.Application.Configuration;

import app.Application.Services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.annotation.Order;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
```

```

@EnableWebSecurity
@Order(1000)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    UserService userService;
    @Bean("authenticationManager")
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }
    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception {
        httpSecurity
            .csrf().disable().cors().disable()
            .authorizeRequests()
            .antMatchers("/registration").permitAll()
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .loginPage("/login")
            .defaultSuccessUrl("/")
            .permitAll()
            .and()
            .logout().deleteCookies("JSESSIONID")
            .permitAll()
            .logoutSuccessUrl("/")
            .and()
            .rememberMe().key("uniqueAndSecret");
    }

    @Autowired
    protected void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userService).passwordEncoder(bCryptPasswordEncoder());
    }
}

```

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.log(Level.INFO, joinPoint.getSignature() + " выполнен за " + executionTime
+ "мс");
        log.info(joinPoint.getSignature() + " выполнен за " + executionTime + "мс");
        return proceed;
    }

    @Pointcut("within(Homework.twentieth.Services.*)")
    public void allServiceMethods() {}
}
```

Результат выполнения программы

```
. _ _ _ . _ _  
/\ / ___' _ _ _ (\) _ _ \\\ \  
(C)\___ | '_|'_||'_|\_.' |\ \\\  
\/_ __)|_|)|| || || || (| ) )))  
'|__||_|_|_|_|_\_, |// // //  
=====|_|=====|____#/_/_/  
  
:: Spring Boot ::                (v2.4.4)
```

```
21:15:06.903 [background-preinit] INFO  
    o.h.validator.internal.util.Version - HV000001: Hibernate Validator 6.1.7.Final  
21:15:06.903 [main] INFO  
    app.Application.Application - Starting Application using Java 11.0.10 on Frischmann-PC with PID 5916 (C:\Users\frisc\GoogleDrive\Stu  
21:15:06.906 [main] INFO  
    app.Application.Application - No active profile set, falling back to default profiles: default  
21:15:07.399 [main] INFO  
    o.s.d.r.c.RepositoryConfigurationDelegate - Bootstrapping Spring Data JPA repositories in DEFAULT mode.  
21:15:07.463 [main] INFO  
    o.s.d.r.c.RepositoryConfigurationDelegate - Finished Spring Data repository scanning in 55 ms. Found 2 JPA repository interfaces.  
21:15:08.174 [main] INFO  
    o.s.b.w.e.tomcat.TomcatWebServer - Tomcat initialized with port(s): 8081 (http)  
21:15:08.182 [main] INFO  
    o.a.coyote.http11.Http11NioProtocol - Initializing ProtocolHandler ["http-nio-8081"]  
21:15:08.182 [main] INFO  
    o.a.catalina.core.StandardService - Starting service [Tomcat]  
21:15:08.182 [main] INFO  
    o.a.catalina.core.StandardEngine - Starting Servlet engine: [Apache Tomcat/9.0.44]  
21:15:08.302 [main] INFO  
    o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext  
21:15:08.302 [main] INFO  
    o.s.b.w.s.c.ServletWebServerApplicationContext - Root WebApplicationContext: initialization completed in 1356 ms  
21:15:08.465 [main] INFO  
    o.h.jpa.internal.util.LogHelper - HHH0000204: Processing PersistenceUnitInfo [name: default]
```

Рисунок 23.1 – Демонстрация работы программы

Please sign in

Sign in

Рисунок 23.2 – Демонстрация работы программы

Практическая работа №24

Цель работы

Тема: Тестирование в Spring Framework с использованием Junit.

Постановка задачи: Написать модульное тестирование для всех классов сервисов приложения из предыдущего задания.

Листинг программы

User.java

```
package app.Application.Classes;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import javax.validation.constraints.Size;
import java.util.Collection;
import java.util.Set;

@Entity
@Table(name = "t_user")
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Size(min=2, message = "Не меньше 5 знаков")
    private String username;

    @Size(min=2, message = "Не меньше 5 знаков")
    private String password;

    @Transient
    private String passwordConfirm;

    @ManyToMany(fetch = FetchType.EAGER)
    private Set<Role> roles;

    public User() {
    }

    public Long getId() {
        return id;
    }

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

    @Override
    public String getUsername() {
        return username;
    }
}
```

```

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return true;
}

public void setUsername(String username) {
    this.username = username;
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    return getRoles();
}

@Override
public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getPasswordConfirm() {
    return passwordConfirm;
}

public void setPasswordConfirm(String passwordConfirm) {
    this.passwordConfirm = passwordConfirm;
}

public Set<Role> getRoles() {
    return roles;
}

public void setRoles(Set<Role> roles) {
    this.roles = roles;
}
}

```

Role.java

```
package app.Application.Classes;

import org.springframework.security.core.GrantedAuthority;

import javax.persistence.*;
import java.util.Set;

@Entity
@Table(name = "t_role")
public class Role implements GrantedAuthority {
    @Id
    private Long id;

    private String name;

    @Transient
    @ManyToMany(mappedBy = "roles")
    private Set<User> users;

    public Role() {
    }

    public Role(Long id) {
        this.id = id;
    }

    public Role(Long id, String name) {
        this.id = id;
        this.name = name;
    }

    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 Set<User> getUsers() {
        return users;
    }

    public void setUsers(Set<User> users) {
        this.users = users;
    }

    @Override
    public String getAuthority() {
        return getName();
    }
}
```

UserRepository.java

```
package app.Application.Interfaces;

import app.Application.Classes.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

RoleRepository.java

```
package app.Application.Interfaces;

import app.Application.Classes.Role;
import org.springframework.data.jpa.repository.JpaRepository;

public interface RoleRepository extends JpaRepository<Role, Long> {
}
```

UserService.java

```
package app.Application.Services;

import app.Application.Classes.Role;
import app.Application.Classes.User;
import app.Application.Interfaces.RoleRepository;
import app.Application.Interfaces.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import java.util.Collections;
import java.util.List;
import java.util.Optional;

@Service
public class UserService implements UserDetailsService {
    @PersistenceContext
    private EntityManager em;

    @Autowired
    UserRepository userRepository;

    @Autowired
    RoleRepository roleRepository;

    @Autowired
    BCryptPasswordEncoder bCryptPasswordEncoder;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }
}
```



```

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userRepository.findByUsername(username);

        if (user == null) {
            throw new UsernameNotFoundException("User not found");
        }

        return user;
    }

    public User findUserById(Long userId) {
        Optional<User> userFromDb = userRepository.findById(userId);
        return userFromDb.orElse(new User());
    }

    public List<User> allUsers() {
        return userRepository.findAll();
    }

    public boolean saveUser(User user) {
        User userFromDB = userRepository.findByUsername(user.getUsername());

        if (userFromDB != null) {
            return false;
        }

        user.setRoles(Collections.singleton(new Role(1L, "ROLE_USER")));
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        userRepository.save(user);
        return true;
    }

    public boolean deleteUser(Long userId) {
        if (userRepository.findById(userId).isPresent()) {
            userRepository.deleteById(userId);
            return true;
        }
        return false;
    }

    public List<User> usertList(Long idMin) {
        return em.createQuery("SELECT u FROM User u WHERE u.id > :paramId", User.class)
            .setParameter("paramId", idMin).getResultList();
    }
}

```

AdminController.java

```
package app.Application.Controllers;

import app.Application.Services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class AdminController {
    @Autowired
    private UserService userService;

    @GetMapping("/admin")
    public String userList(Model model) {
        model.addAttribute("allUsers", userService.allUsers());
        return "admin";
    }

    @PostMapping("/admin")
    public String deleteUser(@RequestParam(required = true, defaultValue = "" ) Long
userId,
                           @RequestParam(required = true, defaultValue = "" ) String
action,
                           Model model) {
        if (action.equals("delete")){
            userService.deleteUser(userId);
        }
        return "redirect:/admin";
    }

    @GetMapping("/admin/gt/{userId}")
    public String gtUser(@PathVariable("userId") Long userId, Model model) {
        model.addAttribute("allUsers", userService.usergtList(userId));
        return "admin";
    }
}
```

CookieController.java

```
package app.Application.Controllers;

import org.springframework.web.bind.annotation.RestController;

@RestController
public class CookieController {
}
```

RegistrationController.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.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

import javax.validation.Valid;

@Controller
public class RegistrationController {

    @Autowired
    private UserService userService;

    @GetMapping("/registration")
    public String registration(Model model) {
        model.addAttribute("userForm", new User());

        return "registration";
    }

    @PostMapping("/registration")
    public String addUser(@ModelAttribute("userForm") @Valid User userForm,
        BindingResult bindingResult, Model model) {

        if (bindingResult.hasErrors()) {
            return "registration";
        }
        if (!userForm.getPassword().equals(userForm.getPasswordConfirm())){
            model.addAttribute("passwordError", "Пароли не совпадают");
            return "registration";
        }
        if (!userService.saveUser(userForm)){
            model.addAttribute("usernameError", "Пользователь с таким именем уже существует");
            return "registration";
        }

        return "redirect:/";
    }
}
```

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;
import org.springframework.scheduling.annotation.EnableAsync;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration
@EnableAspectJAutoProxy
@EnableJpaRepositories(basePackages = {"app.Application"})
@EnableAsync
public class Config extends WebSecurityConfigurerAdapter {
}
```

MvcConfig.java

```
package app.Application.Configuration;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class MvcConfig implements WebMvcConfigurer {

    @Override
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/login").setViewName("login");
        registry.addViewController("/news").setViewName("news");
    }
}
```

WebSecurityConfig.java

```
package app.Application.Configuration;

import app.Application.Services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.annotation.Order;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
```

```

@EnableWebSecurity
@Order(1000)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    UserService userService;
    @Bean("authenticationManager")
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }
    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception {
        httpSecurity
            .csrf().disable().cors().disable()
            .authorizeRequests()
            .antMatchers("/registration").permitAll()
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .loginPage("/login")
            .defaultSuccessUrl("/")
            .permitAll()
            .and()
            .logout().deleteCookies("JSESSIONID")
            .permitAll()
            .logoutSuccessUrl("/")
            .and()
            .rememberMe().key("uniqueAndSecret");
    }

    @Autowired
    protected void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userService).passwordEncoder(bCryptPasswordEncoder());
    }
}

```

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.log(Level.INFO, joinPoint.getSignature() + " выполнен за " + executionTime
+ "мс");
        log.info(joinPoint.getSignature() + " выполнен за " + executionTime + "мс");
        return proceed;
    }

    @Pointcut("within(Homework.twentieth.Services.*)")
    public void allServiceMethods() {}
}
```

UserServiceImplTest.java

```
package app.Application.Test;

import app.Application.Classes.User;
import app.Application.Interfaces.UserRepository;
import app.Application.Services.UserService;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.junit.jupiter.MockitoExtension;

import java.util.List;

import static org.mockito.Mockito.mock;

@ExtendWith(MockitoExtension.class)
public class UserServiceImplTest {
    @Mock
    private UserRepository userRepository;

    @Test
    public void getGame() {
        userRepository=mock(UserRepository.class);

        User user = new User();
        user.setUsername("Ivan");

        User user1 = new User();
        user1.setUsername("Petr");

        Mockito.when(userRepository.findAll()).thenReturn(List.of(user,user1));
        UserService userService = new UserService(userRepository);
        Assertions.assertEquals(2, userService.allUsers().size());
        Assertions.assertEquals("Petr",
            userService.allUsers().get(0).getUsername());
    }
}
```

Результат выполнения программы

```
> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava NO-SOURCE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test NO-SOURCE
BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 up-to-date
21:42:03: Task execution finished 'test'.
```

Рисунок 24.1 – 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 "Вильямс", 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.