Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 3

Spring, работа с БД
Тема

Преподаватель _____ А.С. Черниговский
Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 _____ А. К. Никитин
Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2021

1 Цель

Ознакомиться с механизмами работы с базами данных в Spring Framework.

2 Задачи

В каждом варианте есть сущность базы данных. Список задач представлен ниже.

1) Описать класс сущности, который имеет как минимум три текстовых поля и два числовых (и, естественно, id). Она описывает некий товар (затем, эта сущность и БД пригодится нам в следующих работах).

2) Создать таблицу базы данных (студент может выбрать любую реляционную БД), соответствующую спроектированной сущности.

3) Реализовать консольное Spring приложение (должно иметь простейший консольный пользовательский интерфейс), которое должно позволять:

а) Вводить (консольный ввод) пользователю поля сущности и добавлять её в таблицу БД.

б) Выводить в консоль все записи из таблицы БД.

в) Редактировать запись таблицы БД по Id.

г) Удалять запись по Id.

д) Осуществлять поиск по любому из признаков (Студент самостоятельно выбирает поле для поиска.

4) Способ работы с БД (JdbcTemplate, Hibernate, JPA или др.) студентом выбирается самостоятельно, ограничение одно — должен использоваться Spring Framework.

3 Описание варианта

Класс «Канцтовары».

4 Листинги программ

На листингах 1-5 представлен программный код классов, реализующий задание.

Листинг 1 – Класс таблицы канцтоваров

```
package ru.nikitin.entities;

import lombok.AccessLevel;
import lombok.Setter;

import javax.persistence.*;

@Entity
public class Stationery {
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    @Setter(AccessLevel.NONE)
    @Column(updatable = false)
    private Integer id;

    @Column(nullable = false)
    private String type;

    @Column(nullable = false)
    private Double price;

    @Column(nullable = false)
    private Integer amount;

    @Column(nullable = false)
    private String subtype;

    @Column(nullable = false)
    private String manufacturer;

    public Stationery() {
    }

    public Stationery(String type, Double price, Integer amount, String subtype,
String manufacturer) {
        this.type = type;
```

Окончание листинга 1

```
        this.price = price;
        this.amount = amount;
        this.subtype = subtype;
        this.manufacturer = manufacturer;
    }

    @Override
    public String toString() {
        return "id: " + this.id + " | " + "type: " + this.type + " | " + "subtype:
" + this.subtype + " | "
                + "price: " + this.price + " | " + "manufacturer: " +
this.manufacturer
                + " | " + "amount: " + this.amount;
    }
}
```

Листинг 2 – Класс JPA-репозитория

```
package ru.nikitin.repos;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.nikitin.entities.Stationery;

import java.util.List;

@Repository
public interface StationeryRepository extends JpaRepository<Stationery, Integer>
{
    List<Stationery> findStationeryByManufacturer(String manufacturer);
}
```

Листинг 3 – Интерфейс сервиса базы данных канцтоваров

```
package ru.nikitin.services;

import ru.nikitin.entities.Stationery;

import java.util.List;

public interface StationeryService {
    void add(Stationery stationery);
    boolean delete(Integer id);
    boolean update(Integer id, Stationery stationery);
```

Окончание листинга 3

```
    List<Stationery> getByManufacturer(String manufacturer);
    List<Stationery> getAll();
}
```

Листинг 4 – Класс-сервис базы данных канцтоваров

```
package ru.nikitin.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import ru.nikitin.services.StationeryService;
import ru.nikitin.entities.Stationery;
import ru.nikitin.repos.StationeryRepository;

import java.util.List;

@Service
public class StationaryServiceImpl implements StationeryService {
    @Autowired
    private StationeryRepository stationeryRepository;

    @Override
    public void add(Stationery stationery) {
        this.stationeryRepository.save(stationery);
    }
    @Override
    public boolean delete(Integer id) {
        if (checkId(id)) {
            this.stationeryRepository.deleteById(id);
            return true;
        }
        return false;
    }
    @Override
    public boolean update(Integer id, Stationery stationery) {
        if (checkId(id)) {
            this.stationeryRepository.deleteById(id);
            this.stationeryRepository.save(stationery);
            return true;
        }
        return false;
```

Окончание листинга 4

```
    }
    @Override
    public List<Stationery> getByManufacturer(String manufacturer) {
        return
this.stationeryRepository.findStationeryByManufacturer(manufacturer);
    }
    @Override
    public List<Stationery> getAll() {
        return this.stationeryRepository.findAll(Sort.by("type"));
    }

    private boolean checkId(int id) {
        return this.stationeryRepository.findById(id).isPresent();
    }

}
```

Листинг 5 – Класс с конфигурацией Spring

```
package ru.nikitin;

import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.jpa.JpaTransactionManager;
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;
import org.springframework.transaction.PlatformTransactionManager;

import javax.persistence.EntityManagerFactory;
import javax.sql.DataSource;
import java.util.Objects;
import java.util.Properties;

@Configuration
@ComponentScan(“ru.nikitin”)
```

Продолжение листинга 5

```
@PropertySource("classpath:application.properties")
@EnableJpaRepositories
@RequiredArgsConstructor
public class SpringConfiguration {
    private final Environment env;

    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();

dataSource.setDriverClassName(Objects.requireNonNull(env.getProperty("spring.dat
asource.driverClassName")));
        dataSource.setUrl(env.getProperty("spring.dataSource.url"));
        dataSource.setUsername(env.getProperty("spring.dataSource.username"));
        dataSource.setPassword(env.getProperty("spring.dataSource.password"));
        return dataSource;
    }

    @Bean
    public Properties jpaProperties() {
        return new Properties();
    }

    @Bean
    @Autowired
    public EntityManagerFactory entityManagerFactory(DataSource dataSource,
Properties jpaProperties) {
        HibernateJpaVendorAdapter vendorAdapter = new
HibernateJpaVendorAdapter();
        LocalContainerEntityManagerFactoryBean factory = new
LocalContainerEntityManagerFactoryBean();
        factory.setJpaVendorAdapter(vendorAdapter);
        factory.setPackagesToScan("ru.nikitin");
        factory.setDataSource(dataSource);
        factory.setJpaProperties(jpaProperties);
        factory.afterPropertiesSet();
        return factory.getObject();
    }

    @Bean
    @Autowired
```

Окончание листинга 5

```
    public PlatformTransactionManager transactionManager(EntityManagerFactory
entityManagerFactory) {
        JpaTransactionManager txManager = new JpaTransactionManager();
        txManager.setEntityManagerFactory(entityManagerFactory);
        return txManager;
    }

}
```

Листинг 6 – Класс Main

```
package ru.nikitin;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
import ru.nikitin.services.impl.StationaryServiceImpl;
import ru.nikitin.entities.Stationery;

import java.util.Scanner;

public class Main {
    public static String inputString(boolean lower) {
        Scanner in = new Scanner(System.in);
        String text = in.nextLine();
        while (text.equals("")) {
            System.out.println("Поле не должно быть пустым!");
            text = in.nextLine();
        }
        if (lower)
            text = text.toLowerCase();
        return text;
    }

    public static int inputNat(boolean strict) {
        Scanner in = new Scanner(System.in);

        int x;
        while (true) {
            try {
                x = in.nextInt();
                in.nextLine();
```

Продолжение листинга 6

```
                if (x < 0)
                    throw new java.util.InputMismatchException("Value is below
zero!");

                if (strict && x <= 0)
                    throw new java.util.InputMismatchException("Value is below
zero!");
                break;

            } catch (java.util.InputMismatchException e) {
                System.out.println("Input the correct value!");
                in.nextLine();
            }
        }
        return x;
    }

    public static Double inputDouble(double bottomBound, double upperBound) {
        Scanner in = new Scanner(System.in);

        double x;
        while (true) {
            try {
                x = in.nextDouble();
                in.nextLine();

                if (x < bottomBound || x > upperBound)
                    throw    new    java.util.InputMismatchException("Incorrect
value!");
                break;

            } catch (java.util.InputMismatchException e) {
                System.out.println("Input the correct value!");
                in.nextLine();
            }
        }
        return x;
    }

    public static Stationery inputStationery() {
```

Продолжение листинга 6

```
        System.out.println("Введите тип канцтовара (обязательное поле):");
        String type = inputString(true);

        System.out.println("Введите подтип канцтовара:");
        String subtype = inputString(true).toLowerCase();

        System.out.println("Введите цену товара (обязательное поле):");
        Double price = inputDouble(0, Double.POSITIVE_INFINITY);

        System.out.println("Введите количество канцтовара на складе:");
        int amount = inputNat(false);

        System.out.println("Введите производителя канцтовара:");
        String manufacturer = inputString(true).toLowerCase();

        return new Stationery(type, price, amount, subtype, manufacturer);
    }

    public static int inputId() {
        System.out.println("Введите id элемента:");
        return inputNat(false);
    }

    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(SpringConfiguration.class);

        while (true) {
            System.out.println("Введите команду:");
            System.out.println("1. Добавить новый элемент;");
            System.out.println("2. Вывести содержимое БД;");
            System.out.println("3. Редактировать элемент;");
            System.out.println("4. Удалить элемент;");
            System.out.println("5. Поиск продукции определенного
производителя;");
            System.out.println("6. Завершение работы приложения.");

            int choice = inputNat(true);
            StationaryServiceImpl service =
context.getBean("stationaryServiceImpl", StationaryServiceImpl.class);
            switch (UserMenu.values()[choice - 1]) {
```

Продолжение листинга 6

```
case ADD -> {
    Stationery stationery = inputStationery();
    service.add(stationery);
    System.out.println("Элемент успешно добавлен.");
}

case OUTPUT -> {
    service.getAll().forEach(System.out::println);
}

case UPDATE -> {
    int id = inputId();
    Stationery stationery = inputStationery();
    if (service.update(id, stationery))
        System.out.println("Элемент успешно удален.");
    else
        System.out.println("Элемента с таким id нет!");
}

case REMOVE -> {
    int id = inputId();
    if (service.delete(id))
        System.out.println("Элемент успешно удален.");
    else
        System.out.println("Элемента с таким id нет!");
}

case FIND_MANUFACTURER -> {
    System.out.println("Введите производителя:");
    String manufacturer = inputString(true);
    System.out.println(service.getByManufacturer(manufacturer));
}

case EXIT -> {
    context.close();
    System.exit(1);
}

default -> {
    System.out.println("Введите значение от 1 до 6!");
}
```

Окончание листинга 6

```
                }
            }
        }
}
```