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

Институт Космических и информационных технологий
институт
Кафедра «Информатика»
кафедра

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4

Работа с базами данных в Spring Framework
тема

Преподаватель		А.С. Черниговский
	подпись, дата	инициалы, фамилия
Студент КИ18-16б 031831229		В.А. Прекель
номер группы, зачетной книжки	подпись, дата	инициалы, фамилия

Красноярск 2020

1 Цель работы

Ознакомиться с механизмами работы с базами данных в
Spring Framework.

2 Общая постановка задачи

В каждом варианте есть сущность базы данных. Необходимо:

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:

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

• Выводить в консоль все записи из таблицы БД.

• Редактировать запись таблицы БД по Id.

• Удалять запись по Id.

• Осуществлять поиск по любому из признаков (Студент
самостоятельно выбирает поле для поиска. Например, поиск всех студентов,
средний балл которых выше введенного пользователем.)

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

Вариант 11. Мебель.

3 Исходный код

Листинг 1 – Lab04\src\main\java\com\github\prekel\JavaSpring\Lab04\Program.java

```
      package com.github.prekel.JavaSpring.Lab04;

import com.github.prekel.JavaSpring.Lab04.component.FurnitureDao;
import com.github.prekel.JavaSpring.Lab04.component.FurnitureJdbcDao;
import com.github.prekel.JavaSpring.Lab04.component.FurnitureRepository;
import com.github.prekel.JavaSpring.Lab04.entity.Furniture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

import java.math.BigDecimal;

@SpringBootApplication
public class Program implements CommandLineRunner {
    private static final Logger LOG = LoggerFactory.getLogger(Program.class);
    private final AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(SpringConfig.class);
    private final SpringConfig config = context.getBean("springConfig",
SpringConfig.class);
    private FurnitureDao furnitureDao;

    public static void main(String[] args) {
        LOG.info("Started");
        SpringApplication.run(Program.class, args);
        LOG.info("Ended");
    }

    @Override
    public void run(String... args) throws Exception {
        furnitureDao = new ReadWithCheckBuilder<Integer>()
                .hasMessage("1 - JdbcTemplate, 2 - JpaRepository: ")
                .hasChecker(number -> 1 <= number && number <= 2)
                .hasParser(Integer::parseInt)
                .readCycle() == 1
                ? context.getBean("furnitureJdbcDao", FurnitureJdbcDao.class)
                : context.getBean("furnitureRepository",
FurnitureRepository.class);

        System.out.println("1 - P’PIPµCЃC,Pë PïPsP»CЦ CЃCѓC‰PSPsCЃC,Pë Pë
PґPsP±P°PIPëC,CЊ PµC‘ PI C,P°P±P»PëC†Cѓ P‘P”");
        System.out.println("2 - P’C‹PIPµCЃC,Pë PICЃPµ P·P°PïPëCЃPë PëP·
C,P°P±P»PëC†C‹ P‘P”");
        System.out.println("3 - P PµPґP°PєC,PëCЂPsPIP°C,CЊ P·P°PïPëCЃCЊ
C,P°P±P»PëC†C‹ P‘P” PïPs Id");
        System.out.println("4 - PJPґP°P»PëC,CЊ P·P°PïPëCЃCЊ PïPs Id");
        System.out.println("5 - PћCЃCѓC‰PµCЃC,PIPëC,CЊ PïPsPëCЃPє PïPs
C,PëPïCѓ");
        System.out.println("6 - P’C‹PIPµCЃC,Pë P·P°PïPëCЃCЊ PëP· C,P°P±P»PëC†C‹
P‘P” PïPs Id");
        System.out.println("0 - P’C‹C…PsPґ PëP· PïCЂPsPiCЂP°PjPjC‹");

        while (true) {
            switch (new ReadWithCheckBuilder<Integer>()
                    .hasMessage("Р’РІРµРґРёС‚Рµ РЅРѕРјРµСЂ РєРѕРјР°РЅРґС‹: ")
                    .hasParser(Integer::parseInt)
                    .hasChecker(number -> 0 <= number && number <= 6)
                    .readCycle()) {
                case 1 -> furnitureDao
                        .insert(furnitureFromInput());
                case 2 -> furnitureDao
                        .findAll()
                        .forEach(System.out::println);
                case 3 -> furnitureDao
                        .findById(new ReadWithCheckBuilder<Integer>()
                                .hasMessage("Р’РІРµРґРёС‚Рµ Id РґР»СЏ
СЂРµРґР°РєС‚РёСЂРѕРІР°РЅРёСЏ: ")
                                .hasChecker(id -> id > 0)
                                .hasParser(Integer::parseInt)
                                .readCycle())
                        .ifPresentOrElse(
                                furniture ->
furnitureDao.updateById(furniture.getId(), furnitureFromInput()),
                                () -> System.out.println("РќРµС‚ С‚Р°РєРѕР№
Р·Р°РїРёСЃРё")
                        );
                case 4 -> furnitureDao
                        .findById(new ReadWithCheckBuilder<Integer>()
                                .hasMessage("Р’РІРµРґРёС‚Рµ Id РґР»СЏ
СѓРґР°Р»РµРЅРёСЏ: ")
                                .hasChecker(id -> id > 0)
                                .hasParser(Integer::parseInt)
                                .readCycle())
                        .ifPresentOrElse(
                                furniture ->
furnitureDao.removeById(furniture.getId()),
                                () -> System.out.println("РќРµС‚ С‚Р°РєРѕР№
Р·Р°РїРёСЃРё")
                        );
                case 5 -> furnitureDao
                        .findByType(new ReadWithCheckBuilder<String>()
                                .hasMessage("Р’РІРµРґРёС‚Рµ С‚РёРї РґР»СЏ
РїРѕРёСЃРєР°: ")
                                .hasChecker(string -> !string.isBlank())
                                .readCycle()
                        )
                        .forEach(System.out::println);
                case 6 -> furnitureDao
                        .findById(new ReadWithCheckBuilder<Integer>()
                                .hasMessage("Р’РІРµРґРёС‚Рµ Id Р·Р°РїРёСЃРё: ")
                                .hasChecker(id -> id > 0)
                                .hasParser(Integer::parseInt)
                                .readCycle())
                        .ifPresentOrElse(
                                System.out::println,
                                () -> System.out.println("РќРµС‚ С‚Р°РєРѕР№
Р·Р°РїРёСЃРё")
                        );
                case 0 -> {
                    return;
                }
            }
        }
    }

    private Furniture furnitureFromInput() {
        return new Furniture(
                new ReadWithCheckBuilder<String>()
                        .hasMessage("Р’РІРµРґРёС‚Рµ С‚РёРї: ")
                        .hasChecker(string -> !string.isBlank())
                        .readCycle(),
                new ReadWithCheckBuilder<String>()
                        .hasMessage("Р’РІРµРґРёС‚Рµ РјРѕРґРµР»СЊ: ")
                        .hasChecker(string -> !string.isBlank())
                        .readCycle(),
                new ReadWithCheckBuilder<String>()
                        .hasMessage("Р’РІРµРґРёС‚Рµ РїСЂРѕРёР·РІРѕРґРёС‚РµР»СЏ:
")
                        .hasChecker(string -> !string.isBlank())
                        .readCycle(),
                BigDecimal.valueOf(new ReadWithCheckBuilder<Integer>()
                        .hasMessage("Р’РІРµРґРёС‚Рµ С†РµРЅСѓ (С†РµР»Р°СЏ
С‡Р°СЃС‚СЊ, СЂСѓР±Р»Рё): ")
                        .hasChecker(number -> 0 < number)
                        .hasParser(Integer::parseInt)
                        .readCycle() +
                        new ReadWithCheckBuilder<Integer>()
                                .hasMessage("Р’РІРµРґРёС‚Рµ С†РµРЅСѓ
(РєРѕРїРµР№РєРё): ")
                                .hasChecker(number -> 0 < number && number <
100)
                                .hasParser(Integer::parseInt)
                                .readCycle() / 100.0),
                new ReadWithCheckBuilder<Double>()
                        .hasMessage("Р’РІРµРґРёС‚Рµ РІС‹СЃРѕС‚Сѓ
(СЃР°РЅС‚РёРјРµС‚СЂС‹): ")
                        .hasChecker(number -> 0 < number && number < 10000)
                        .hasParser(Double::parseDouble)
                        .readCycle()
        );
    }
}
```

Листинг 2 – Lab04\src\main\java\com\github\prekel\JavaSpring\Lab04\ReadWithCheckBuilder.java

```
          package com.github.prekel.JavaSpring.Lab04;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.function.Function;

public class ReadWithCheckBuilder<T> {
    private final BufferedReader reader;
    private String message = "";
    private Function<String, T> parser = (s -> (T) s);
    private Function<T, Boolean> checker = (obj -> true);

    public ReadWithCheckBuilder() {
        this(System.in);
    }

    public ReadWithCheckBuilder(InputStream in) {
        this(new BufferedReader(new InputStreamReader(in)));
    }

    public ReadWithCheckBuilder(BufferedReader reader) {
        this.reader = reader;
    }

    public ReadWithCheckBuilder<T> hasMessage(String message) {
        this.message = message;
        return this;
    }

    public ReadWithCheckBuilder<T> hasParser(Function<String, T> parser) {
        this.parser = parser;
        return this;
    }

    public ReadWithCheckBuilder<T> hasChecker(Function<T, Boolean> checker) {
        this.checker = checker;
        return this;
    }

    public T readCycle() {
        while (true) {
            System.out.print(message);
            try {
                var parsed = parser.apply(reader.readLine());
                if (!checker.apply(parsed)) {
                    throw new Exception("РќРµ РІ РїСЂРѕРјРµР¶СѓС‚РєРµ РёР»Рё
РїСѓСЃС‚Р°СЏ СЃС‚СЂРѕРєР°");
                }
                return parsed;
            } catch (Exception e) {
                System.err.println(e.getMessage());
            }
        }
    }
}
```

Листинг 3 – Lab04\src\main\java\com\github\prekel\JavaSpring\Lab04\SpringConfig.java

```
      package com.github.prekel.JavaSpring.Lab04;

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 org.springframework.transaction.annotation.EnableTransactionManagement;

import javax.persistence.EntityManagerFactory;
import javax.sql.DataSource;

@Configuration("springConfig")
@ComponentScan("com.github.prekel.JavaSpring.Lab04.component")
@PropertySource("classpath:application.properties")
@EnableTransactionManagement
@EnableJpaRepositories
public class SpringConfig {
    @Autowired
    private Environment env;

    @Bean
    public PlatformTransactionManager transactionManager() {
        var txManager = new JpaTransactionManager();
        txManager.setEntityManagerFactory(entityManagerFactory());
        return txManager;
    }

    @Bean
    public EntityManagerFactory entityManagerFactory() {
        var vendorAdapter = new HibernateJpaVendorAdapter();
        vendorAdapter.setGenerateDdl(true);
        var factory = new LocalContainerEntityManagerFactoryBean();
        factory.setJpaVendorAdapter(vendorAdapter);
        factory.setPackagesToScan("com.github.prekel.JavaSpring.Lab04.entity");
        factory.setDataSource(dataSource());
        factory.afterPropertiesSet();
        return factory.getObject();
    }

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

dataSource.setDriverClassName(env.getProperty("dataSource.driverClassName"));
        dataSource.setUrl(env.getProperty("dataSource.url"));
        dataSource.setUsername(env.getProperty("dataSource.username"));
        dataSource.setPassword(env.getProperty("dataSource.password"));

        return dataSource;
    }
}
```

Листинг 4 – Lab04\src\main\java\com\github\prekel\JavaSpring\Lab04\component\FurnitureDao.java

```
      package com.github.prekel.JavaSpring.Lab04.component;

import com.github.prekel.JavaSpring.Lab04.entity.Furniture;
```

```java
import java.util.List;
import java.util.Optional;

public interface FurnitureDao {
    List<Furniture> findAll();
    List<Furniture> findByType(String type);
    Optional<Furniture> findById(int id);
    void updateById(int id, Furniture furniture);
    void insert(Furniture furniture);
    void removeById(int id);
}
```

Листинг 5 – Lab04\src\main\java\com\github\prekel\JavaSpring\Lab04\component\FurnitureJdbcDao.java

```java
      package com.github.prekel.JavaSpring.Lab04.component;

import com.github.prekel.JavaSpring.Lab04.entity.Furniture;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Component;

import javax.sql.DataSource;
import java.util.List;
import java.util.Optional;

@Component("furnitureJdbcDao")
public class FurnitureJdbcDao implements FurnitureDao {
    private JdbcTemplate jdbcTemplate;

    @Autowired
    public void setDataSource(DataSource dataSource) {
        this.jdbcTemplate = new JdbcTemplate(dataSource);
    }

    @Override
    public List<Furniture> findAll() {
        return jdbcTemplate.query("SELECT * FROM Furniture", new
BeanPropertyRowMapper<>(Furniture.class));
    }

    @Override
    public List<Furniture> findByType(String type) {
        return jdbcTemplate.query("SELECT * FROM Furniture WHERE type = ?", new
Object[]{type}, new BeanPropertyRowMapper<>(Furniture.class));
    }

    @Override
    public Optional<Furniture> findById(int id) {
        var ret = jdbcTemplate.query("SELECT * FROM Furniture WHERE id = ?", new
Object[]{id}, new BeanPropertyRowMapper<>(Furniture.class));
        return ret.stream().findFirst();
    }
    @Override
    public void updateById(int id, Furniture furniture) {
        jdbcTemplate.update("UPDATE Furniture SET type = ?, model = ?,
manufacturer = ?, cost = ?, height = ? WHERE id = ?",
                furniture.getType(), furniture.getModel(),
furniture.getManufacturer(), furniture.getCost(), furniture.getHeight(), id);
    }

    @Override
    public void insert(Furniture furniture) {
        jdbcTemplate.update("INSERT INTO Furniture (id, type, model,
manufacturer, cost, height) VALUES (DEFAULT,?,?,?,?,?)",
                furniture.getType(), furniture.getModel(),
furniture.getManufacturer(), furniture.getCost(), furniture.getHeight());
    }

    @Override
    public void removeById(int id) {
        jdbcTemplate.update("DELETE FROM Furniture WHERE id = ?", id);
    }
}
```

Листинг 6 – Lab04\src\main\java\com\github\prekel\JavaSpring\Lab04\component\FurnitureRepository.java

```
      package com.github.prekel.JavaSpring.Lab04.component;

import com.github.prekel.JavaSpring.Lab04.entity.Furniture;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.Optional;

@Repository("furnitureRepository")
public interface FurnitureRepository extends JpaRepository<Furniture, Integer>,
FurnitureDao {
    List<Furniture> findByType(String type);

    @Transactional
    void removeById(int id);

    Optional<Furniture> findById(int id);

    default void updateById(int id, Furniture furniture) {
        furniture.setId(id);
        save(furniture);
    }

    default void insert(Furniture furniture) {
        save(furniture);
    }
}
```

Листинг 7 – Lab04\src\main\java\com\github\prekel\JavaSpring\Lab04\entity\Furniture.java

```
      package com.github.prekel.JavaSpring.Lab04.entity;

import javax.persistence.*;
import java.math.BigDecimal;
import java.util.StringJoiner;

@Entity
public class Furniture {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column
    private String type;
    @Column
    private String model;
    @Column
    private String manufacturer;
    @Column
    private BigDecimal cost;
    @Column
    private double height;

    public Furniture() {

    }

    public Furniture(String type, String model, String manufacturer, BigDecimal
cost, double height) {
        this.type = type;
        this.model = model;
        this.manufacturer = manufacturer;
        this.cost = cost;
        this.height = height;
    }

    public int getId() {
        return id;
    }

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

    public String getType() {
        return type;
    }

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

    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    public BigDecimal getCost() {
        return cost;
    }

    public void setCost(BigDecimal cost) {
        this.cost = cost;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }

    @Override
    public String toString() {
        return new StringJoiner(", ", Furniture.class.getSimpleName() + "[",
"]")
                .add("id=" + id)
                .add("type='" + type + "'")
                .add("model='" + model + "'")
                .add("manufacturer='" + manufacturer + "'")
                .add("cost=" + cost)
                .add("height=" + height)
                .toString();
    }
}
```

Листинг 8 – Lab04\src\main\resources\application.properties

```
      #--- Postgres ---
dataSource.driverClassName=org.postgresql.Driver
jpa.database=POSTGRESQL
dataSource.url=jdbc:postgresql://localhost:5432/javaspring
dataSource.username=postgres
dataSource.password=qwerty123
```