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

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 4

Spring MVC
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

Познакомиться с шаблоном MVC в Spring и тем, как он используется при создании web-приложений.

2 Задачи

Изменить практическую работу №3 таким образом, чтобы она представляла собой web-приложение.

Web-приложение должно иметь следующие страницы:

1) Главная страница, содержит приветствие и ссылки на другие (которые дублируют по функционалу пункты меню из работы №4).

2) Страница просмотра таблицы записей.

3) Страница добавления новой записи в таблицу.

4) Страница редактирования записи.

5) Страница удаления записи из таблицы БД.

6) Страница просмотра записей согласно некоторому критерию (аналогично пункту в работе №4).

Помимо всего должны быть осуществлены проверки (не менее двух) входных данных, сопровождающиеся соответствующими сообщениями об ошибках.

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;
```

Продолжение листинга 1

```
import lombok.Setter;

import javax.persistence.*;
import javax.validation.constraints.Min;

@Entity
@Table(name = "stationery")
public class Stationery {
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    @Setter(AccessLevel.NONE)
    private Integer id;

    @Column(name = "type")
    private String type;

    @Column(name = "price")
    @Min(0)
    private Double price;

    @Column(name = "amount")
    @Min(0)
    private Integer amount;

    @Column(name = "subtype")
    private String subtype;

    @Column(name = "manufacturer")
    private String manufacturer;

    public Stationery() {
    }

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

Продолжение листинга 1

```
}

public Integer getId() {
    return id;
}

public String getType() {
    return type;
}

public Double getPrice() {
    return price;
}

public Integer getAmount() {
    return amount;
}

public String getSubtype() {
    return subtype;
}

public String getManufacturer() {
    return manufacturer;
}

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

public void setPrice(Double price) {
    this.price = price;
}

public void setAmount(Integer amount) {
    this.amount = amount;
}

public void setSubtype(String subtype) {
    this.subtype = subtype;
}
```

Окончание листинга 1

```
    public void setManufacturer(String manufacturer) {
        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;
    }
}
        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
```

Окончание листинга 2

```
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);
    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
```

Окончание листинга 4

```
    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;
    }
    @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;
```

Продолжение листинга 5

```
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”)
@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
```

Окончание листинга 5

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

}
```

Листинг 6 – Класс веб-конфигурации

```
package ru.nikitin.configs;

import lombok.Data;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import
org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.ViewResolverRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.thymeleaf.spring5.SpringTemplateEngine;
import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;
```

Окончание листинга 6

```
import org.thymeleaf.spring5.view.ThymeleafViewResolver;

@Configuration
@EnableWebMvc
@ComponentScan("ru.nikitin.controllers")
public class WebConfig implements WebMvcConfigurer {
    private final ApplicationContext applicationContext;
    @Autowired
    public WebConfig(ApplicationContext applicationContext) {
        this.applicationContext = applicationContext;
    }

    @Bean
    public SpringResourceTemplateResolver templateResolver() {
        SpringResourceTemplateResolver templateResolver = new
                SpringResourceTemplateResolver();
        templateResolver.setApplicationContext(applicationContext);
        templateResolver.setPrefix("/WEB-INF/views/");
        templateResolver.setSuffix(".html");
        return templateResolver;
    }

    @Bean
    public SpringTemplateEngine templateEngine() {
        SpringTemplateEngine templateEngine = new SpringTemplateEngine();
        SpringResourceTemplateResolver resolver = templateResolver();
        resolver.setCharacterEncoding("UTF-8");
        templateEngine.setTemplateResolver(resolver);
        templateEngine.setEnableSpringELCompiler(true);
        return templateEngine;
    }
    @Override
    public void configureViewResolvers(ViewResolverRegistry registry) {
        ThymeleafViewResolver resolver = new ThymeleafViewResolver();
        resolver.setTemplateEngine(templateEngine());
        resolver.setCharacterEncoding("UTF-8");
        resolver.setContentType("text/html;charset=UTF-8");
        registry.viewResolver(resolver);
    }
}
```

Листинг 7 – Класс конфигурации диспетчер-сервлета

```
package ru.nikitin.configs;

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServle
tInitializer;

public class DispatcherConfig extends
AbstractAnnotationConfigDispatcherServletInitializer {
    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class<?>[] { AppConfig.class };
    }
    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class<?>[] { WebConfig.class };
    }
    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }
}
```

Листинг 8 – Класс контроллера

```
package ru.nikitin.controllers;

import lombok.Data;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import ru.nikitin.entities.Stationery;
import ru.nikitin.services.impl.StationeryServiceImpl;

import javax.validation.Valid;

@Controller
public @Data
class MainController {
```

Продолжение листинга 8

```
    private final ApplicationContext context;
    private final StationeryServiceImpl stationeryService;

    @Autowired
    public MainController(ApplicationContext applicationContext) {
        this.context = applicationContext;
        this.stationeryService = context.getBean("stationeryServiceImpl",
StationeryServiceImpl.class);
    }

    @GetMapping("/")
    public String mainPage(Model model) {
        model.addAttribute("stationery", stationeryService.getAll());
        return "home";
    }

    @GetMapping("/add")
    public String addPage(Model model) {
        model.addAttribute("stationery", new Stationery());
        return "add";
    }
    @PostMapping("/add")
    public String addFormHandler(@ModelAttribute("stationery") Stationery
stationery) {
        stationeryService.add(stationery);
        return "redirect:/";
    }

    @GetMapping("/edit")
    public String editPage(Model model) {
        model.addAttribute("stationery", new Stationery());
        return "edit";
    }
    @PostMapping("/edit")
    public String editFormHandler(
            @ModelAttribute @Valid Stationery stationery,
            BindingResult bindingResult,
            @RequestParam("id") Integer id) {
        System.out.println(bindingResult);
        if (bindingResult.hasErrors())
            return "redirect:/edit";
```

Окончание листинга 8

```
        stationeryService.update(id, stationery);
        return "redirect:/";
    }

    @GetMapping("/delete")
    public String deletePage() {
        return "delete";
    }
    @PostMapping("/delete")
    public String deleteFormHandler( @RequestParam("id") Integer id ) {
        stationeryService.delete(id);
        return "redirect:/";
    }

    @GetMapping("/show_criteria")
    public String criteriaPage() {
        return "criteria";
    }
    @PostMapping("/show_criteria")
    public String criteriaFormHandler( @RequestParam String manufacturer,
ModelMap model ) {
        model.addAttribute("stationery",
stationeryService.getByManufacturer(manufacturer));
        return "criteria_result";
    }
}
```