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

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 7

Spring JMS
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 механизмом JMS в Spring.

2 Задачи

Взять практическую работу №4 или 5 и добавить следующие возможности.

1) Добавить кнопку-ссылку «купить» на форме. После чего в брокер сообщений отправляется сообщение о том какой «товар»/сущность хочет купить пользователь.

2) В хорошем варианте товар помечается как купленный и не будет показан в общем списке товаров (необходимо добавить соответствующий столбец). Или же просто удалить запись о купленном товаре из БД, но перед этим не забыть отправить информацию о товаре в брокер сообщений.

3) Реализовать приложение-сервис приемки сообщений, которое принимает сообщение и на основе содержимого сообщения отправляет e-mail администратору по некоторому адресу (можно использовать константную строку вашего почтового ящика) о том что у него хотят купить товар.

3 Описание варианта

Класс «Канцтовары».

4 Листинги программ

4.1 Основной сервер

На листингах 1-4 представлен программный код классов-конфигураторов.

Листинг 1 – Конфигурация БД

```
package ru.nikitin.configs;

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

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

dataSource.setDriverClassName(Objects.requireNonNull(env.getProperty("spring.dat
asource.driverClassName")));
```

Окончание листинга 1

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

}
```

Листинг 2 – Конфигурация сервера

```
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;
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();
```

Окончание листинга 2

```
        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 3 – Конфигурация 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 4 – Конфигурация JMS

```
package ru.nikitin.configs;

import org.springframework.amqp.core.AmqpAdmin;
import org.springframework.amqp.core.Queue;
import org.springframework.amqp.rabbit.connection.CachingConnectionFactory;
import org.springframework.amqp.rabbit.core.RabbitAdmin;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.amqp.support.converter.Jackson2JsonMessageConverter;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class JmsConfig {
    static final String queueName = "stationery-queue";

    @Bean
    RabbitTemplate rabbitTemplate(){
        CachingConnectionFactory connectionFactory =
                new CachingConnectionFactory("localhost", 5672);
        AmqpAdmin admin = new RabbitAdmin(connectionFactory);
        admin.declareQueue(new Queue(queueName));
        RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(new Jackson2JsonMessageConverter());
        return rabbitTemplate;
    }
}
```

На листингах 5-9 представлен код с логикой сервера, реализующий задание.

Листинг 5 – Класс-контроллер

```
package ru.nikitin.controllers;

import lombok.Data;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.ApplicationContext;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
```

Продолжение листинга 5

```
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import ru.nikitin.entities.Stationery;
import ru.nikitin.model.Message;
import ru.nikitin.services.impl.StationeryServiceImpl;

import javax.validation.Valid;

@Controller
public @Data
class MainController {
    private final RabbitTemplate rabbitTemplate;
    private final StationeryServiceImpl stationeryService;

    @Autowired
    public                    MainController(@Qualifier("stationeryServiceImpl")
StationeryServiceImpl stationeryService,
                          @Qualifier("rabbitTemplate")            RabbitTemplate
rabbitTemplate) {
        this.stationeryService = stationeryService;
        this.rabbitTemplate = rabbitTemplate;
    }

    @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:/";
    }
```

Продолжение листинга 5

```
@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";

    if (!stationeryService.checkId(id))
        return "redirect:/edit";

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

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

    stationeryService.delete(id);
    return "redirect:/";
}

@GetMapping("/show_criteria")
public String criteriaPage() {
    return "criteria";
}
@PostMapping("/show_criteria")
```

Окончание листинга 5

```
    public String criteriaFormHandler( @RequestParam String manufacturer,
ModelMap model ) {
        model.addAttribute("stationery",
stationeryService.getByManufacturer(manufacturer));
        return "criteria_result";
    }

    @GetMapping("/buy")
    public String buyPage() {
        return "purchase";
    }
    @PostMapping("/buy")
    public String buyFormHandler( @RequestParam("id") Integer id ) {
        if (!stationeryService.checkId(id))
            return "redirect:/buy";

        rabbitTemplate.convertAndSend("stationery-queue",
                new Message("New purchase", stationeryService.get(id).get()));

        stationeryService.delete(id);
        return "redirect:/";
    }
}
```

Листинг 6 – Класс-сущность канцелярского предмета

```
package ru.nikitin.entities;

import lombok.AccessLevel;
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;
```

Продолжение листинга 6

```
    @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;
    }

    public Integer getId() {
        return id;
    }

    public String getType() {
        return type;
    }

    public Double getPrice() {
        return price;
    }
```

Окончание листинга 6

```
    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;
    }

    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;
    }
```

```
}
```

Листинг 7 – Класс со структурой сообщения

```
package ru.nikitin.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import ru.nikitin.entities.Stationery;

import java.io.Serializable;

public @Data
class Message implements Serializable {
    private String message;
    private Stationery stationery;

    public Message(String message, Stationery stationery) {
        this.message = message;
        this.stationery = stationery;
    }

    public String getMessage() {
        return message;
    }

    public Stationery getStationery() {
        return stationery;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public void setStationery(Stationery stationery) {
        this.stationery = stationery;
    }
}
```

Листинг 8 – Класс-репозиторий БД

```
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 9 – Класс, реализующий обращения к БД

```
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;
import java.util.Optional;

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

    @Override
    public boolean add(Stationery stationery) {
        this.stationeryRepository.save(stationery);
        return true;
    }

    @Override
    public boolean delete(Integer id) {
        if (checkId(id)) {
```

Окончание листинга 9

```
            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"));
    }

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

    public Optional<Stationery> get(Integer id) {
        return this.stationeryRepository.findById(id);
    }
}
```

4.2 Сервер-обработчик сообщений с основного сервера

На листингах 10-12 представлен программный код классов-конфигураторов.

Листинг 10 – Конфигурация отправки электронных писем

```
package ru.nikitin.configs;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.JavaMailSenderImpl;

import java.util.Properties;

@Configuration
public class MailConfig {

    @Bean
    public JavaMailSender getJavaMailSender() {
        JavaMailSenderImpl mailSender = new JavaMailSenderImpl();
        mailSender.setHost("smtp.gmail.com");
        mailSender.setPort(587);

        mailSender.setUsername("alekc080901@gmail.com");
        mailSender.setPassword("vyozltbrrcxsmokc");

        Properties props = mailSender.getJavaMailProperties();
        props.put("mail.transport.protocol", "smtp");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.debug", "true");
        props.put("mail.smtp.ssl.trust", "smtp.gmail.com");

        return mailSender;
    }
}
```

Листинг 11 – Конфигурация сервлета

```
package ru.nikitin.configs;
import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServle
tInitializer;

public class DispatcherConfig extends
AbstractAnnotationConfigDispatcherServletInitializer {
```

Окончание листинга 11

```
    @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 12 – Конфигурация JMS

```
package ru.nikitin.configs;

import org.springframework.amqp.rabbit.annotation.EnableRabbit;
import org.springframework.amqp.rabbit.annotation.RabbitListenerConfigurer;
import
org.springframework.amqp.rabbit.config.SimpleRabbitListenerContainerFactory;
import org.springframework.amqp.rabbit.connection.CachingConnectionFactory;
import org.springframework.amqp.rabbit.connection.ConnectionFactory;
import org.springframework.amqp.rabbit.listener.RabbitListenerEndpointRegistrar;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.messaging.converter.MappingJackson2MessageConverter;
import
org.springframework.messaging.handler.annotation.support.DefaultMessageHandlerMe
thodFactory;

@Configuration
@ComponentScan("ru.nikitin")
@EnableRabbit
public class JmsConfig implements RabbitListenerConfigurer {
    @Override
    public     void     configureRabbitListeners(RabbitListenerEndpointRegistrar
registrar) {
        registrar.setMessageHandlerMethodFactory(myHandlerMethodFactory());
```

Окончание листинга 12

```
    }
    @Bean
    ConnectionFactory connectionFactory(){
        return new CachingConnectionFactory("localhost", 5672);
    }
    @Bean
    public SimpleRabbitListenerContainerFactory rabbitListenerContainerFactory()
{
        SimpleRabbitListenerContainerFactory factory =
                new SimpleRabbitListenerContainerFactory();
        factory.setConnectionFactory(connectionFactory());
        factory.setBatchListener(true);
        return factory;
    }
    @Bean
    public DefaultMessageHandlerMethodFactory myHandlerMethodFactory() {
        DefaultMessageHandlerMethodFactory factory =
                new DefaultMessageHandlerMethodFactory();
        factory.setMessageConverter(new MappingJackson2MessageConverter());
        return factory;
    }
}
```

На листингах 13-16 представлен код с логикой сервера, реализующий задание.

Листинг 13 – Класс-обработчик электронных писем

```
package ru.nikitin.handlers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Service;

@Service
public class EmailHandler {

    @Autowired
    private JavaMailSender emailSender;

    public void sendEmail(String toAddress, String subject, String message) {
```

Окончание листинга 13

```
        SimpleMailMessage simpleMailMessage = new SimpleMailMessage();
        simpleMailMessage.setTo(toAddress);
        simpleMailMessage.setSubject(subject);
        simpleMailMessage.setText(message);
        emailSender.send(simpleMailMessage);
    }

}
```

Листинг 14 – Класс-сущность канцелярского предмета

```
package ru.nikitin.model;

import org.springframework.stereotype.Component;

@Component
public class Stationery {
    private Integer id;

    private String type;

    private Double price;

    private Integer amount;

    private String subtype;

    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 14

```
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 14

```
    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;
    }
}
```

Листинг 15 – Класс со структурой сообщения

```
package ru.nikitin.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import ru.nikitin.entities.Stationery;

import java.io.Serializable;

public @Data
class Message implements Serializable {
    private String message;
    private Stationery stationery;

    public Message(String message, Stationery stationery) {
        this.message = message;
        this.stationery = stationery;
    }

    public String getMessage() {
        return message;
    }

    public Stationery getStationery() {
        return stationery;
    }
```

Окончание листинга 15

```
    public void setMessage(String message) {
        this.message = message;
    }

    public void setStationery(Stationery stationery) {
        this.stationery = stationery;
    }
}
```

Листинг 16 – Класс-обработчик сообщений

```
package ru.nikitin.receivers;

import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import ru.nikitin.handlers.EmailHandler;
import ru.nikitin.model.Message;
import ru.nikitin.model.Stationery;

@Component
public class Receiver {
    @Autowired
    private EmailHandler emailHandler;

    private final String ADMIN_EMAIL = "rei.nikitin@ya.ru";
    private final String SUBJECT = "Новая покупа";

    @RabbitListener(queues = "stationery-queue", containerFactory =
            "rabbitListenerContainerFactory")
    public void listen(Message message) {
        Stationery stationery = message.getStationery();
        String messageText =
                "Stationery " + stationery.getType() +
                ", " + stationery.getSubtype() +
                " (id " + stationery.getId() +
                ") by \"" + stationery.getManufacturer() + "\" has been bought.";

        emailHandler.sendEmail(ADMIN_EMAIL, SUBJECT, messageText);
    }
}
```

5 Вывод

В результате работы были закреплены на практике знания JMS и асинхронной обработки сообщений и реализована программа по заданию, состоящая из сервера и обработчика email-писем.