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

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №7

Обмен сообщениями в Spring
Тема

Преподаватель _____ А.С. Черниговский
подпись, дата — инициалы, фамилия

Студент КИ19-16/1б 031939175 _____ А.Д. Непомнящий
номер группы, зачетной книжки — подпись, дата — инициалы, фамилия

Красноярск 2021

1 Цель работы

Цель состоит в ознакомлении с механизмом JMS в Spring.

2 Задачи

Взять практическое задание №6 и выполнить следующее.

1. Настроить очередь приема сообщений для администратора.
2. При выполнении операций добавления, удаления или редактирования ресурса через REST API / форму создавать соответствующие уведомления и отправлять их в очередь.
3. Любым удобным способом (можно через консоль) продемонстрировать извлечение административных сообщений о выполненных операциях.

Вариант 10. Одежда.

3 Ход работы

3.1 Отправка сообщение

В ходе работы было изменено в соответствии с заданием приложение, написанное в ходе шестой работы. В качестве брокера сообщений используется RabbitMQ. На листингах 1 и 2 приведены код класса JmsConfiguration и код класса ApparelController.

Листинг 1 – код класса JmsConfiguration

```
package com.github.durakin.isdlabs.lab7.configuration;

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 JmsConfiguration {
    static final String queueName = "apparel-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;
    }
}
```

Листинг 2 – код класса ApparelController

```
package com.github.durakin.isdlabs.lab7.controller;

import com.github.durakin.isdlabs.lab7.model.ApaprelUpdate;
import com.github.durakin.isdlabs.lab7.entity.Apparel;
import com.github.durakin.isdlabs.lab7.model.Message;
import com.github.durakin.isdlabs.lab7.service.ApparelService;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.server.ResponseStatusException;
```

Продолжение листинга 2

```
import javax.servlet.http.HttpServletResponse;
import java.util.List;

import static org.springframework.http.HttpStatus.NOT_FOUND;

@Controller
@RequestMapping("/apparels")
public class ApparelController {
    @Autowired
    private ApparelService apparelService;

    @Autowired
    private RabbitTemplate rabbitTemplate;

    @ResponseBody
    @GetMapping(value = "/", headers = {"Accept=application/json"})
    public List<Apparel> getApparels() {
        return apparelService.findAll();
    }

    @PutMapping(value = "/{id}", consumes = MediaType.APPLICATION_JSON_VALUE)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void putApparel(@PathVariable int id, @RequestBody ApaprelUpdate
apaprelUpdate) {
        apparelService.update(id, apaprelUpdate.getNewInStock(),
apaprelUpdate.getNewPrice());
        var edited = apparelService.findById(id);
        if (edited != null) {
            rabbitTemplate.convertAndSend("apparel-queue",
                    new Message("EDIT",
java.time.LocalDateTime.now().toString(), edited));
        }
    }

    @ResponseBody
    @GetMapping(value = "/{id}", headers = {"Accept=application/json"})
    public Apparel getApparel(@PathVariable("id") int id) {
        return apparelService.findById(id);
    }

    @GetMapping(value = "/{id}", headers = {"Accept=text/html"})
    public String getStudent(@PathVariable("id") int id, Model model) {
        var apparel = apparelService.findById(id);
        if (apparel != null) {
            model.addAttribute(apparelService.findById(id));
            return "show";
        }
        throw new ResponseStatusException(NOT_FOUND, "Unable to find resource");
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteApparel(@PathVariable int id) {
        var deleted = apparelService.delete(id);
        if (deleted != null) {
            rabbitTemplate.convertAndSend("apparel-queue",
                    new Message("DELETE",
java.time.LocalDateTime.now().toString(), deleted));
```

Окончание листинга 2

```
        }
    }

    @PostMapping(consumes = MediaType.APPLICATION_JSON_VALUE)
    @ResponseStatus(HttpStatus.CREATED)
    public Apparel create(@RequestBody final Apparel newApparel,
HttpServletResponse response) {
        var createdId = apparelService.add(newApparel);
        response.setHeader("Location", "/apparels/" + createdId);
        var created = apparelService.findById(createdId);
        if (created != null) {
            rabbitTemplate.convertAndSend("apparel-queue",
                    new Message("ADD", java.time.LocalDateTime.now().toString(),
created));
        }
        return created;
    }
}
```

Листинг 3 – Код класса Message

```
package com.github.durakin.isdlabs.lab7.model;

import com.github.durakin.isdlabs.lab7.entity.Apparel;

import java.io.Serializable;
public class Message implements Serializable {
    private String action;
    private String time;
    private Apparel apparel;

    // Get- и Set- методы

    public Message() {
    }

    public Message(String action, String time, Apparel apparel) {
        this.action = action;
        this.time = time;
        this.apparel = apparel;
    }
}
```

3.2 Ресивер

Было реализовано простое веб-приложение для извлечения и вывода сообщений.

Листинг 4 – Код класса Reciever

```
package com.github.durakin.receiver;

import com.github.durakin.model.Message;
```

Окончание листинга 4

```
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.stereotype.Component;

@Component
public class Receiver {
    @RabbitListener(queues = "apparel-queue", containerFactory =
            "rabbitListenerContainerFactory")

    public void listen(Message message) {
        System.out.println('\n' + message.getAction() + "\nOn apparel\n" +
message.getApparel().toString() + "On time\n" + message.getTime());
    }
}
```

4 Вывод

Было произведено ознакомление с механизмом JMS в Spring.