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

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №6

Поддержка архитектуры REST 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 механизмами поддержки архитектуры REST в Spring.

2 Задачи

Взять практическое задание №4 и добавить функционал, перечисленный ниже.

1. Преобразовать веб-приложение таким образом, чтобы оно поддерживало архитектуру REST. Должны поддерживаться следующие типы запросов: GET (показ (html) и извлечение (json) всех/одной записей/сущностей), POST (добавление), PUT (редактирование), DELETE (удаление).

2. Разработать REST-клиент для вашего приложения, который, используя RestTemplate позволяет выполнять базовые операции по извлечению (GET), добавлению (POST), редактированию (PUT), удалению (DELETE) ресурсов. REST-клиент не обязан иметь пользовательский интерфейс, необходим тестовый пример, который можно запускать из консоли.

3. Обязательным условием является сохранение всего предшествующего функционала приложения. Для удовлетворения всем характеристикам RESTархитектуры приложение может быть реорганизовано (убраны GET-запросы с параметрами) или добавлен новый функционал.

4. PUT и DELETE запросы не обязательно делать через запросы из браузера. Достаточно реализации для клиентов-приложений.

Вариант 10. Одежда.

3 Ход работы

3.1 Поддержка REST веб-приложением

В ходе работы было изменено в соответствии с заданием приложение, написанное в ходе четвертой работы. В листинге приведен контроллер, ответственный за поддержку REST.

Листинг 1 – код класса ApparelController

```
package com.github.durakin.isdlabs.lab6.controller;

import com.github.durakin.isdlabs.lab6.entity.ApaprelUpdate;
import com.github.durakin.isdlabs.lab6.entity.Apparel;
import com.github.durakin.isdlabs.lab6.service.ApparelService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.server.ResponseStatusException;

import javax.servlet.http.HttpServletResponse;

import java.math.BigDecimal;
import java.util.List;

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

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

    @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());
    }

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

Окончание листинга 1

```
    }

    @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");
    }

    /*
    @PutMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void putApparel(@PathVariable int id, @RequestBody Apparel apparel) {
    }

    */

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteFurniture(@PathVariable int id) {
        apparelService.delete(id);
    }

    @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);
        return apparelService.findById(createdId);
    }
}
```

3.2 Клиент

В ходе работы был разработан REST-клиент для приложения, который, используя RestTemplate, позволяет выполнять базовые операции по извлечению (GET), добавлению (POST), редактированию (PUT), удалению (DELETE) ресурсов.

Листинг 2 – Код основного класса клиента

```
package com.github.durakin;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.json.JSONObject;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
```

Продолжение листинга 2

```
import org.springframework.http.MediaType;
import org.springframework.web.client.RestTemplate;

import java.math.BigDecimal;
import java.net.URI;

public class App {
    private static final String url = "http://localhost:8080/Lab6_war/apparels";
    private static final ObjectMapper objectMapper = new ObjectMapper();

    public static void main(String[] args) throws JsonProcessingException {
        System.out.println(retrieveApparel(13));
        var apparels = retrieveAll();
        for (var i : apparels) { System.out.println(i); }
        var apparel = new Apparel();
        apparel.setPrice(BigDecimal.valueOf(120.91));
        apparel.setInStock(120);
        apparel.setApparelType("Added via REST API");

        updateApparel(13, 60, BigDecimal.valueOf(50000));

        //System.out.println(create(apparel));

        //deleteApparel(20);
    }

    public static Apparel retrieveApparel(int id) {
        return new RestTemplate().getForObject(
                url + "/{id}",
                Apparel.class, id);
    }

    public static Apparel[] retrieveAll() {
        return new RestTemplate().getForObject(url + '/', Apparel[].class);
    }

    public static void deleteApparel(int id) {
        new RestTemplate().delete(url + "/{id}", id);
    }

    public static void updateApparel(Integer id, int newInStock, BigDecimal
newPrice) {

        var restTemplate = new RestTemplate();
        var apparelJsonObject = new JSONObject();

        apparelJsonObject.put("newInStock", newInStock);
        apparelJsonObject.put("newPrice", newPrice);

        var headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        HttpEntity<String> request =
                new HttpEntity<String>(apparelJsonObject.toString(), headers);

        restTemplate.put(url+'/'+id.toString(), request);
    }

    public static URI create(Apparel apparel) throws JsonProcessingException {
```

Окончание листинга 2

```
        var restTemplate = new RestTemplate();
        var headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        var apparelJsonObject = new JSONObject();

        apparelJsonObject.put("price", apparel.getPrice());
        apparelJsonObject.put("inStock", apparel.getInStock());
        apparelJsonObject.put("apparelType", apparel.getApparelType());

        if (apparel.getSize() != null) {
            apparelJsonObject.put("size", apparel.getSize());
        }
        if (apparel.getSex() != null) {
            apparelJsonObject.put("sex", apparel.getApparelType());
        }

        HttpEntity<String> request =
                new HttpEntity<String>(apparelJsonObject.toString(), headers);

        return restTemplate.postForLocation(url, request);
    }
}
```

4 Вывод

Было произведено ознакомление с механизмами поддержки архитектуры REST в Spring.