

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

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №7

Поддержка архитектуры REST в Spring

тема

Преподаватель

подпись, дата

А.С. Черниговский

инициалы, фамилия

Студент КИ18-17/16 031830504

номер группы, зачетной книжки

подпись, дата

Е.В. Железкин

инициалы, фамилия

Красноярск 2020

1 Цель работы

Познакомиться с механизмами поддержки архитектуры REST в Spring.

2 Задача работы

Взять практическое задание №6 (или №5, на усмотрение студента, при работе с защищенным приложением могут возникнуть трудности) и добавить следующий функционал:

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), PUT (добавление), DELETE (удаление).

2) Разработать REST-клиент для вашего приложения, который, используя RestTemplate позволяет выполнять базовые операции по извлечению (GET), добавлению (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.

3 Ход работы

1) Веб-приложение доделано до функционала, необходимого в 7 работе (рисунки 1-):

```

// @PreAuthorize("hasAnyAuthority('USER', 'ADMIN')")
@RequestMapping(value="/{id}", method=RequestMethod.GET, headers =
    {"Accept=application/json"})
public @ResponseBody Book getBook(@PathVariable("id") int id)
{
    return bookService.findById(id).orElse(new Book());
}

// @PreAuthorize("hasAnyAuthority('USER', 'ADMIN')")
@RequestMapping(value="/{id}", method=RequestMethod.GET, headers = {"Accept=text/html"})
public String getBook(@PathVariable("id") int id, Model model)
{
    model.addAttribute(bookService.findById(id).orElse(new Book()));
    return "books/single";
}

```

Рисунок 1 – Обновлённый маппинг контроллера BooksController.java

```

package ru.xorsiphus;

import org.springframework.web.client.RestTemplate;
|
public class RestClient
{
    public static void main( String[] args )
    {
        var param = "0";
        if (args.length != 0)
            param = args[0];
        String ret = retrieve(Integer.parseInt(param));
        System.out.println(ret);
    }

    public static String retrieve(int id)
    {
        if (id == 0)
            return new RestTemplate().getForObject(
                url: "http://localhost:8081/books",
                String.class);
        else
            return new RestTemplate().getForObject(
                url: "http://localhost:8081/books/{id}",
                String.class, id);
    }
}

```

Рисунок 2 – Использование RestTemplate

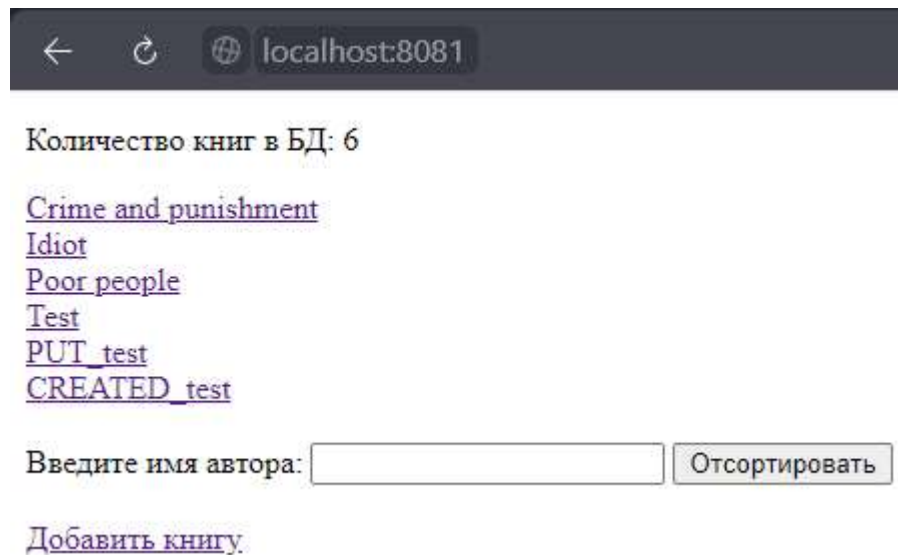


Рисунок 3 – Получение данных с помощью браузера

```
2:\Dev\University\Labs_Java\Lab_7\RestClient\out\artifacts\RestClient.jar>java -jar RestClient.jar 1
15:27:45.379 [main] DEBUG org.springframework.web.client.RestTemplate - HTTP GET http://localhost:8081/books/1
15:27:45.384 [main] DEBUG org.springframework.web.client.RestTemplate - Accept=[text/plain, application/json, application/*+json, */*]
15:27:45.430 [main] DEBUG org.springframework.web.client.RestTemplate - Response 200 OK
15:27:45.431 [main] DEBUG org.springframework.web.client.RestTemplate - Reading to [java.lang.String] as "application/json"
{"id":1,"name":"Crime and punishment","author":"Dost","print_edition":"Printer1","size_in_pages":500,"published_on":"1-1-1987"}

2:\Dev\University\Labs_Java\Lab_7\RestClient\out\artifacts\RestClient.jar>java -jar RestClient.jar
15:27:52.927 [main] DEBUG org.springframework.web.client.RestTemplate - HTTP GET http://localhost:8081/books
15:27:52.932 [main] DEBUG org.springframework.web.client.RestTemplate - Accept=[text/plain, application/*+json, */*]
15:27:52.970 [main] DEBUG org.springframework.web.client.RestTemplate - Response 200 OK
15:27:52.971 [main] DEBUG org.springframework.web.client.RestTemplate - Reading to [java.lang.String] as "application/json"
[{"id":1,"name":"Crime and punishment","author":"Dost","print_edition":"Printer1","size_in_pages":500,"published_on":"1-1-1987"}, {"id":2,"name":"Idiot","author":"Dost","print_edition":"Printer1","size_in_pages":250,"published_on":"10-10-1886"}, {"id":3,"name":"Poor people","author":"Dost","print_edition":"Printer1","size_in_pages":123,"published_on":"1-1-1"}, {"id":4,"name":"Test","author":"123","print_edition":"123","size_in_pages":123,"published_on":"1-1-1000"}, {"id":7,"name":"PUT_test","author":"123","print_edition":"123","size_in_pages":123,"published_on":"1-1-1988"}, {"id":8,"name":"CREATED_test","author":"123","print_edition":"123","size_in_pages":123,"published_on":"1-1-1987"}]
```

Рисунок 4 – Получение данных с помощью RestTemplate

```
Z:\Dev\University\Labs_Java\Lab_7\node>node script.js
[
  {
    id: 1,
    name: 'Crime and punishment',
    author: 'Dost',
    print_edition: 'Printer1',
    size_in_pages: 500,
    published_on: '1-1-1987'
  },
  {
    id: 2,
    name: 'Idiot',
    author: 'Dost',
    print_edition: 'Printer2',
    size_in_pages: 250,
    published_on: '10-10-1886'
  },
  {
    id: 3,
    name: 'Poor people',
    author: 'Dost',
    print_edition: 'Printer1',
    size_in_pages: 123,
    published_on: '1-1-1'
  },
  {
    id: 4,
    name: 'Test',
    author: '123',
    print_edition: '123',
    size_in_pages: 123,
    published_on: '1-1-1000'
  },
  {
    id: 7,
    name: 'PUT_test',
    author: '123',
    print_edition: '123',
    size_in_pages: 123,
    published_on: '1-1-1988'
  },
  {
    id: 8,
    name: 'CREATED_test',
    author: '123',
    print_edition: '123',
    size_in_pages: 123,
    published_on: '1-1-1987'
  }
]
```

Рисунок 5 – Получение данных с помощью кастомного Get-запроса

4 Вывод

В ходе данной лабораторной работы было проведено ознакомление с механизмами поддержки архитектуры REST в Spring.