МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий

КАФЕДРА ИНСТРУМЕНТАЛЬНОГО И ПРИКЛАДНОГО
ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ИиППО)

Практическая работа №4

«Разработка клиент-серверного приложения с использованием технологии Websocket и Spring Framework»

По дисциплине: «Архитектура клиент-серверных приложений»

Выполнил студент группы ИКБО-10-19 Дараган Ф.А.

Принял преподаватель Степанов П.В.

Практическая работы выполнена «___»_____2021 г.

(подпись студента)

«Зачтено» «___»_____2021 г.

(подпись руководителя)

Москва 2021

Оглавление

Практическая работа № 4 Разработка клиент-серверного приложения с использованием технологии Websocket и Spring Framework

Цель работы

Знакомство с технологией Websocket и построение приложения с помощью этой технологии и Spring Framework.

Задание

Используя информацию из данной практической работы, необходимо реализовать клиент- серверное приложение с использованием Websocket. Суть приложения заключается в следующем. При обращении клиентской части по адресу /webs необходимо выполнять обработку Websocket. В случае, получения в вебсокете данных, необходимо ответить их же содержимым.

Выполнение практической работы

На рисунке 1 представлена иерархия файлов проекта.

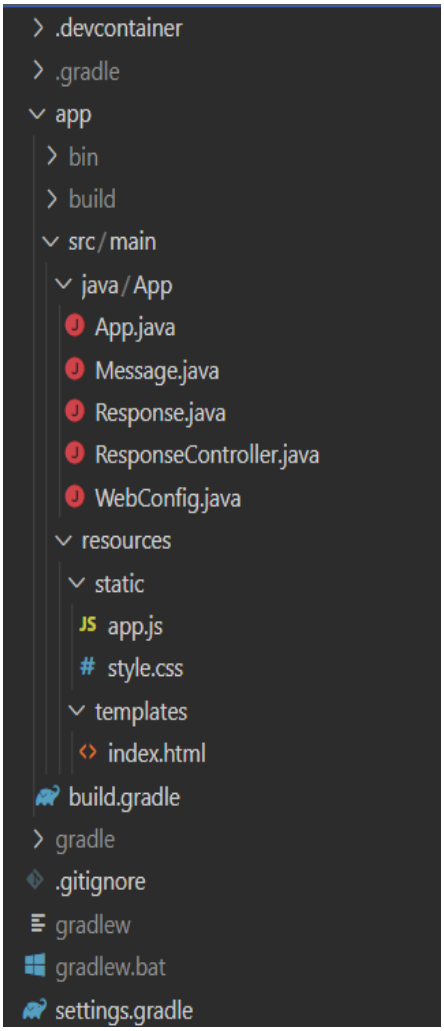

Рис. 1. Скриншот иерархии файлов проекта

На листингах 1 и 2 представлены POJO-классы для сообщений на сервере.

Листинг 1. Класс Message

```
package App;

public class Message {

    private String value;

    public Message() {
        value = "";
    }

    public Message(String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }
}
```

Листинг 2. Класс Response

```
package App;

public class Response {

    private String content;

    public Response(String content) {
        this.content = content;
    }

    public String getContent() {
        return content;
    }

}
```

На листинге 3 представлен класс с конфигурацией сокетов для сервера.

Листинг 3. Класс WebConfig

```
package App;

import org.springframework.context.annotation.Configuration;
import
org.springframework.messaging.simp.config.MessageBrokerRegistry;
import
org.springframework.web.socket.config.annotation.EnableWebSocketMessa
geBroker;
import
org.springframework.web.socket.config.annotation.StompEndpointRegistr
y;
import
org.springframework.web.socket.config.annotation.WebSocketMessageBrok
erConfigurer;

@Configuration
@EnableWebSocketMessageBroker
public class WebConfig implements WebSocketMessageBrokerConfigurer {

    @Override
    public void configureMessageBroker(MessageBrokerRegistry config)
{
        config.enableSimpleBroker("/topic");
        config.setApplicationDestinationPrefixes("/app");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry
registry) {
        registry.addEndpoint("/websocket").withSockJS();
    }

}
```

На листинге 4 представлен класс контроллера для сервера.

Листинг 4. Класс ResponseController

```
package App;

import
org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.messaging.handler.annotation.SendTo;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.util.HtmlUtils;

@Controller
public class ResponseController {

    @GetMapping(path = "/")
    public String home() {
        return "index";
    }

    @MessageMapping("/hello")
    @SendTo("/topic/messages")
    public Response greeting(Message message) throws Exception {
        return new
Response(HtmlUtils.htmlEscape(message.getValue()));
    }

}
```

На листинге 5 представлен клиентский скрипт для взаимодействия с сервером.

Листинг 5. Файл app.js

```
"use strict";

(() => {

    let stompClient = null;

    const showMessage = message =>
        $("#messages").append(`<hr><p class="msg"><span>${new
Date().toLocaleTimeString()}</span>:\t${message}</p>`);

    const setConnected = connected => {
        $("#connect").prop("disabled", connected);
        $("#disconnect").prop("disabled", !connected);
        if (connected)
```

```
            $("#conversation").show();
        else
            $("#conversation").hide();
        $("#messages").append(`<hr><p class="msg"><span>${new
Date().toLocaleTimeString()}</span>:
            Вы ${connected ? "подключились к сокету" : "отключились
от сокета"}.
        `);
    }

    const connect = () => {
        let socket = new SockJS("/websocket");
        stompClient = Stomp.over(socket);
        stompClient.connect({}, frame => {
            setConnected(true);
            console.log("Connected: " + frame);
            stompClient.subscribe("/topic/messages",
                message =>
showMessage(JSON.parse(message.body).content)
            );
        });
    }

    const disconnect = () => {
        if (stompClient !== null) {
            stompClient.disconnect();
        }
        setConnected(false);
        console.log("Disconnected");
    }

    const sendValue = () => {
        stompClient.send(
            "/app/hello",
            {},
            JSON.stringify({"value": $("#value").val()})
        );
    }

    $(() => {
        $("form").on("submit", e => e.preventDefault());
        $("#connect").click(connect);
        $("#disconnect").click(disconnect);
        $("#send").click(sendValue);
    });

})();
```

Остальные файлы не имеют ценности для рассмотрения в отчете.

Для запуска сервера достаточно ввести команду `gradle run`, логи выполнения которой показаны на рисунке 2.

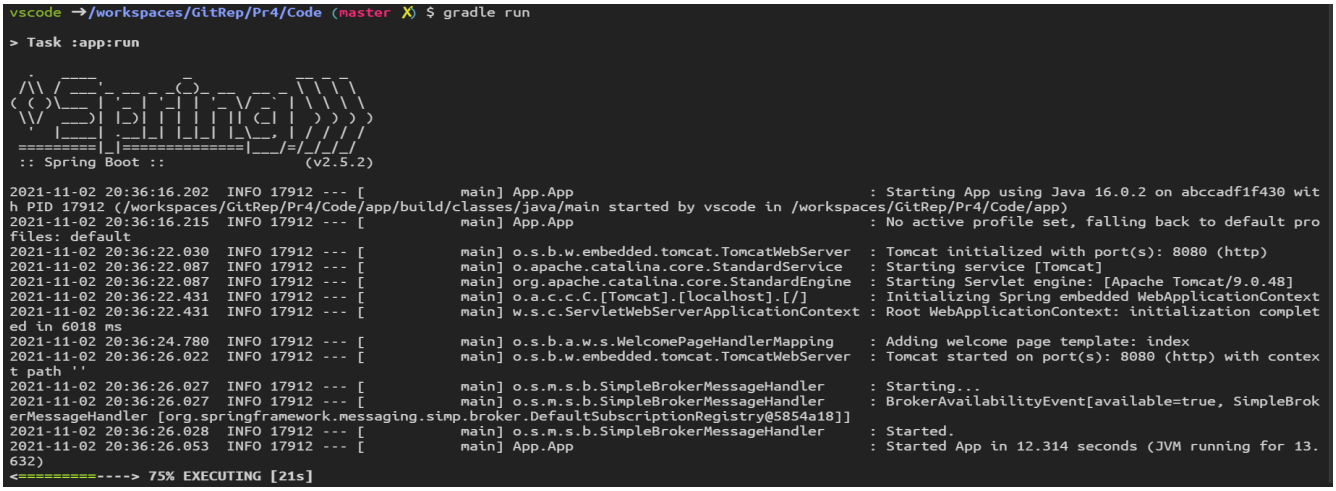

Рис. 2. Скриншот запуска сервера

Страница для взаимодействия с сокетом доступна по адресу http://localhost:8080/. Подключение к сокету и отправка сообщений продемонстрированы на рисунке 3.

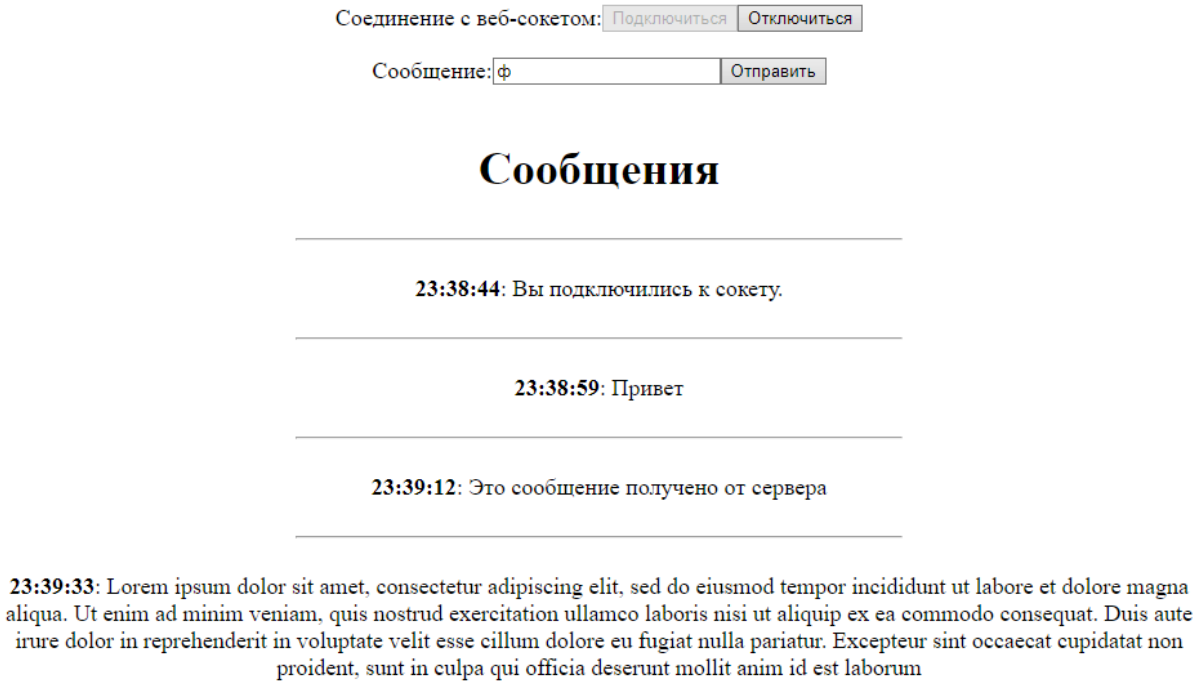

Рис. 3. Скриншот страницы для взаимодействия с сокетом

Выводы по работе

Мы научились работать с Java websocket. Сделали клиент-серверное приложение с их помощью, а так же пользуясь фреймворком Spring и библиотеками webjars.

Используемая литература

1. Вязовик, Н. А. Программирование на Java : учебное пособие / Н. А. Вязовик. — 2-е изд. — Москва : ИНТУИТ, 2016. — 603 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/100405 (дата обращения: 13.09.2021). — Режим доступа: для авториз. пользователей.
2. Наир, В. Предметно-ориентированное проектирование в Enterprise Java : руководство / В. Наир ; перевод с английского А. В. Снастина. — Москва : ДМК Пресс, 2020. — 306 с. — ISBN 978-5-97060-872-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/179503 (дата обращения: 13.09.2021). — Режим доступа: для авториз. пользователей.
3. Васильев, А. Н. Самоучитель Java с примерами и программами : учебное пособие / А. Н. Васильев. — 4-е, изд. — Санкт-Петербург : Наука и Техника, 2017. — 368 с. — ISBN 978-5-94387-745-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/101548 (дата обращения: 13.09.2021). — Режим доступа: для авториз. пользователей.