



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.....	3
Цель работы.....	3
Задание.....	3
Выполнение практической работы.....	4
Выводы по работе.....	10
Используемая литература.....	11

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 вебсокетe данных, необходимо ответить их же содержимым.

Выполнение практической работы

На рисунке 1 представлена иерархия файлов проекта.

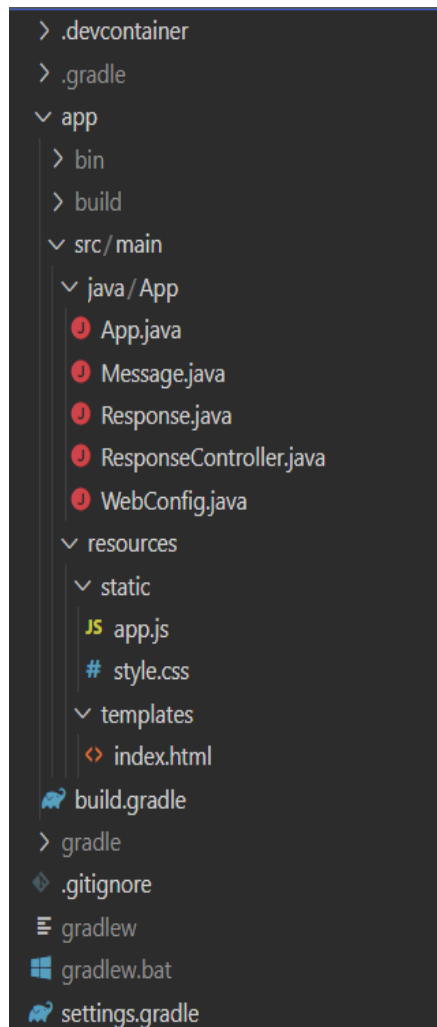


Рис. 1. Скриншот иерархии файлов проекта

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.

```
vscode → /workspaces/GitRep/Pr4/Code (master X) $ gradle run

> Task :app:run

=====
:: Spring Boot ::                (v2.5.2)

2021-11-02 20:36:16.202 INFO 17912 --- [main] App.App          : Starting App using Java 16.0.2 on abccadf1430 wit
h PID 17912 (/workspaces/GitRep/Pr4/Code/app/build/classes/java/main started by vscode in /workspaces/GitRep/Pr4/Code/app)
2021-11-02 20:36:16.215 INFO 17912 --- [main] App.App          : No active profile set, falling back to default pro
files: default
2021-11-02 20:36:22.030 INFO 17912 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-11-02 20:36:22.087 INFO 17912 --- [main] org.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-11-02 20:36:22.087 INFO 17912 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.48]
2021-11-02 20:36:22.431 INFO 17912 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/]     : Initializing Spring embedded WebApplicationContext
2021-11-02 20:36:22.431 INFO 17912 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization complet
ed in 6018 ms
2021-11-02 20:36:24.780 INFO 17912 --- [main] o.s.b.a.w.s.WelcomePageHandlerMapping   : Adding welcome page template: index
2021-11-02 20:36:26.022 INFO 17912 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with contex
t path ''
2021-11-02 20:36:26.027 INFO 17912 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler    : Starting...
2021-11-02 20:36:26.027 INFO 17912 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler    : BrokerAvailabilityEvent[available=true, SimpleBrok
erMessageHandler [org.springframework.messaging.simp.broker.DefaultSubscriptionRegistry@5854a18]]
2021-11-02 20:36:26.028 INFO 17912 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler    : Started.
2021-11-02 20:36:26.053 INFO 17912 --- [main] App.App          : Started App in 12.314 seconds (JVM running for 13.632)

<=====----> 75% EXECUTING [21s]
```

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.

Соединение с веб-сокетом:

Сообщение:

Сообщения

23:38:44: Вы подключились к сокету.

23:38:59: Привет

23:39:12: Это сообщение получено от сервера

23:39:33: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

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.