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

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №4**

по дисциплине «Архитектура клиент-серверных приложений»

**Тема практической работы: Разработка клиент-серверного приложения с использованием технологии Websocket и Spring Framework**

**Студент группы** ИКБО-10-19 Харитонов Дмитрий Михайлович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** преподаватель Степанов П. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Москва 2021

**Цель:**

Знакомство студентов с веб-сокетом.

**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.

**Ход работы:**

В ходе выполнения практической работы были созданы необходимые классы. Ниже, на рисунках, представлена реализация программы и результат её работы.

Исходный код программы:

Greeting.java

package com.pr4;  
  
public class Greeting {  
  
 private String content;  
  
 public Greeting(String content) {  
 this.content = content;  
 }  
  
 public String getContent() {  
 return content;  
 }  
  
}

GreetingController.java

package com.pr4;  
  
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 GreetingController {  
  
 @GetMapping("/webs")  
 public String home() {  
 return "webs";  
 }  
 @MessageMapping("/hello")  
 @SendTo("/topic/greetings")  
 public Greeting greeting(Message message) throws Exception {  
 Thread.*sleep*(500); // simulated delay  
 return new Greeting(HtmlUtils.*htmlEscape*(message.getName()));  
 }  
  
}

Message.java

package com.pr4;  
  
import lombok.Data;  
  
  
public class Message {  
  
 private String name;  
  
 public Message() {  
 }  
  
 public Message(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

Pr4Application.java

package com.pr4;  
  
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;  
import org.springframework.boot.SpringApplication;  
import org.springframework.context.annotation.ComponentScan;  
  
@ComponentScan  
@EnableAutoConfiguration  
public class Pr4Application {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Pr4Application.class, args);  
 }  
}

WebSocketConfig.java

package com.pr4;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.messaging.simp.config.MessageBrokerRegistry;  
import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;  
import org.springframework.web.socket.config.annotation.StompEndpointRegistry;  
import org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer;  
  
@Configuration  
@EnableWebSocketMessageBroker  
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {  
  
 @Override  
 public void configureMessageBroker(MessageBrokerRegistry config) {  
 config.enableSimpleBroker("/topic");  
 config.setApplicationDestinationPrefixes("/app");  
 }  
  
 @Override  
 public void registerStompEndpoints(StompEndpointRegistry registry) {  
 registry.addEndpoint("/gs-guide-websocket").withSockJS();  
 }  
  
  
}

app.js

var ***stompClient*** = null;  
  
function setConnected(connected) {  
 $("#connect").prop("disabled", connected);  
 $("#disconnect").prop("disabled", !connected);  
 if (connected) {  
 $("#conversation").show();  
 }  
 else {  
 $("#conversation").hide();  
 }  
 $("#greetings").html("");  
}  
  
function connect() {  
 var socket = new SockJS('/gs-guide-websocket');  
 ***stompClient*** = Stomp.over(socket);  
 ***stompClient***.connect({}, function (frame) {  
 setConnected(true);  
 ***console***.log('Connected: ' + frame);  
 ***stompClient***.subscribe('/topic/greetings', function (greeting) {  
 showGreeting(***JSON***.parse(greeting.body).content);  
 });  
 });  
}  
  
function disconnect() {  
 if (***stompClient*** !== null) {  
 ***stompClient***.disconnect();  
 }  
 setConnected(false);  
 ***console***.log("Disconnected");  
}  
  
function sendName() {  
 ***stompClient***.send("/app/hello", {}, ***JSON***.stringify({'name': $("#name").val()}));  
}  
  
function showGreeting(message) {  
 $("#greetings").append("<tr><td>" + message + "</td></tr>");  
}  
  
$(function () {  
 $("form").on('submit', function (e) {  
 e.preventDefault();  
 });  
 $( "#connect" ).click(function() { connect(); });  
 $( "#disconnect" ).click(function() { disconnect(); });  
 $( "#send" ).click(function() { sendName(); });  
});

webs.html

<!DOCTYPE html>  
<html>  
<head>  
 <title>WebSocket</title>  
 <link href="/webjars/bootstrap/css/bootstrap.min.css" rel="stylesheet">  
 <link href="/main.css" rel="stylesheet">  
 <script src="/webjars/jquery/jquery.min.js"></script>  
 <script src="/webjars/sockjs-client/sockjs.min.js"></script>  
 <script src="/webjars/stomp-websocket/stomp.min.js"></script>  
 <script src="/app.js"></script>  
</head>  
<body>  
<noscript><h2 style="color: #ff0000">Seems your browser doesn't support Javascript! Websocket relies on Javascript being  
 enabled. Please enable  
 Javascript and reload this page!</h2></noscript>  
<div id="main-content" class="container">  
 <div class="row">  
 <div class="col-md-6">  
 <form class="form-inline">  
 <div class="form-group">  
 <label for="connect">WebSocket connection:</label>  
 <button id="connect" class="btn btn-default" type="submit">Connect</button>  
 <button id="disconnect" class="btn btn-default" type="submit" disabled="disabled">Disconnect  
 </button>  
 </div>  
 </form>  
 </div>  
 <div class="col-md-6">  
 <form class="form-inline">  
 <div class="form-group">  
 <label for="name">Message</label>  
 <input type="text" id="name" class="form-control" placeholder="Message...">  
 </div>  
 <button id="send" class="btn btn-default" type="submit">Send</button>  
 </form>  
 </div>  
 </div>  
 <div class="row">  
 <div class="col-md-12">  
 <table id="conversation" class="table table-striped">  
 <thead>  
 <tr>  
 <th>Messages</th>  
 </tr>  
 </thead>  
 <tbody id="greetings">  
 </tbody>  
 </table>  
 </div>  
 </div>  
</div>  
</body>  
</html>

**Вывод программы:**

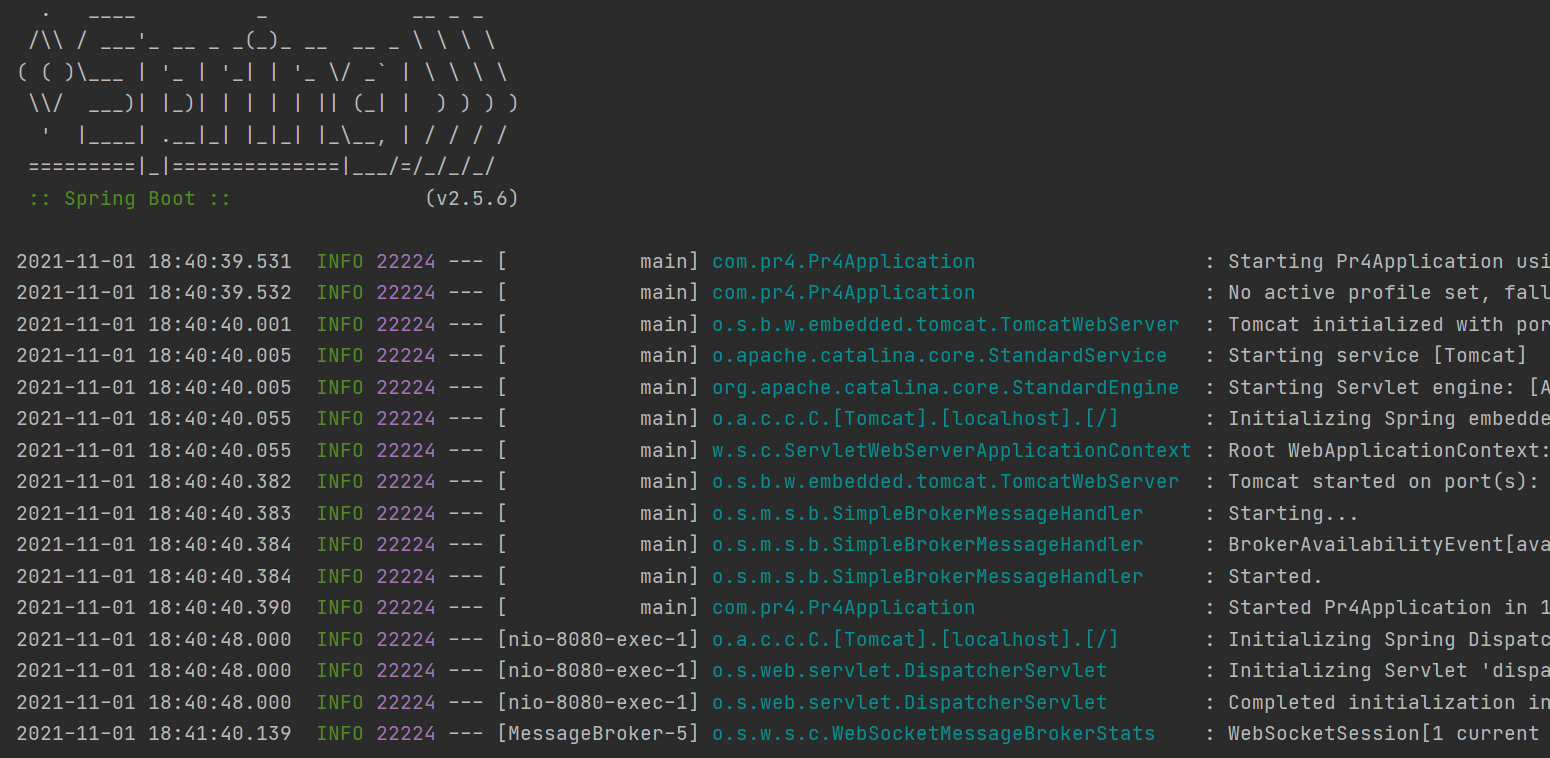


Рисунок 1 – результат выполнения программы

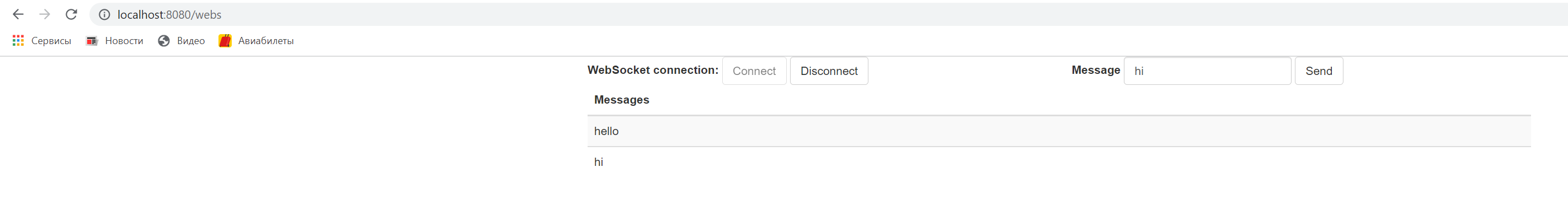


Рисунок 2 – результат выполнения программы

**Вывод:**

В результате выполнения практической работы были получены знания и навыки работы c websocket в JAVA.

**GitHub:**

Github - https://github.com/Unripper1/Architecture