|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра … |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ЛАБОРАТОРНОЙ РАБОТЕ №4** | |
| **по дисциплине** |  |
| **«Программирование на языке Java»**  **Тема: Программирование JAVA сокетов** | |
| Выполнил студент группы ИКБО-20-19 Анваржонов Ж.Т |  |
| Принял ассистент кафедры ИиППО | Степанов П.В |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « 09 »ноября 2020г. |  |
| «Зачтено» | « » 2020 г. |  |

Москва 2020

**Теоретическое введение**

**Цель работы:** знакомство студентов с Разработкой клиент-серверного приложения с использованием технологии Websocket и Spring Framework.

**Постановка задачи**

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

**Программный код**

**Wsconfig.java:**

*package* ru.anvarzhonov.springbootwebsocket.config;  
  
  
*import* org.springframework.context.annotation.*Configuration*;  
*import* org.springframework.messaging.simp.config.MessageBrokerRegistry;  
*import* org.springframework.web.socket.WebSocketMessage;  
*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* WsConfig *implements* WebSocketMessageBrokerConfigurer {  
 *@Override  
 public void* registerStompEndpoints(StompEndpointRegistry registry) {  
 registry.addEndpoint("/javatechie").withSockJS();  
 }  
  
 *@Override  
 public void* configureMessageBroker(MessageBrokerRegistry registry) {  
 registry.enableSimpleBroker("/topic");  
 registry.setApplicationDestinationPrefixes("/app");  
 }  
}

В этой конфигурации у нас WebSocketConfig аннотирован @Configuration, означая, что это конфигурационный Spring класс. Он также аннотирован @EnableWebSocketMessageBroker. Как и предполагает его название @EnableWebSocketMessageBroker включает обработке сообщений по WebSocket, возвращаемый брокером сообщений.

Метод configureMessageBroker() переопределяет поведение по умолчанию в WebSocketMessageBrokerConfigurer для настройки брокера сообщений. Он вызывает enableSimpleBroker() для включения простого брокера сообщений в памятичтобы возвращать обратно сообщения клиенту по направлениям с префиксом /topic. Он также объявляет префикс /app для сообщений, привязанных к методам, аннотированными @MessageMapping.

Метод registerStompEndpoints() регистрирует /javatechie, включая дополнительно SockJS как альтернативный вариант обмена сообщениями, когда WebSocket не доступен.

**chatMessage:**

*package* ru.anvarzhonov.springbootwebsocket.models;  
  
*public class* ChatMessage {  
 *private* String content;  
 *private* String sender;  
 *private* MessageType messageType;  
  
 *public* String getContent() {  
 *return* content;  
 }  
  
 *public void* setContent(String content) {  
 *this*.content = content;  
 }  
  
 *public* String getSender() {  
 *return* sender;  
 }  
  
 *public void* setSender(String sender) {  
 *this*.sender = sender;  
 }  
  
 *public* MessageType getMessageType() {  
 *return* messageType;  
 }  
  
 *public void* setMessageType(MessageType messageType) {  
 *this*.messageType = messageType;  
 }  
  
 *public enum* MessageType{  
 CHAT, LEAVE, JOIN  
 }  
  
  
}

**ChatController:**

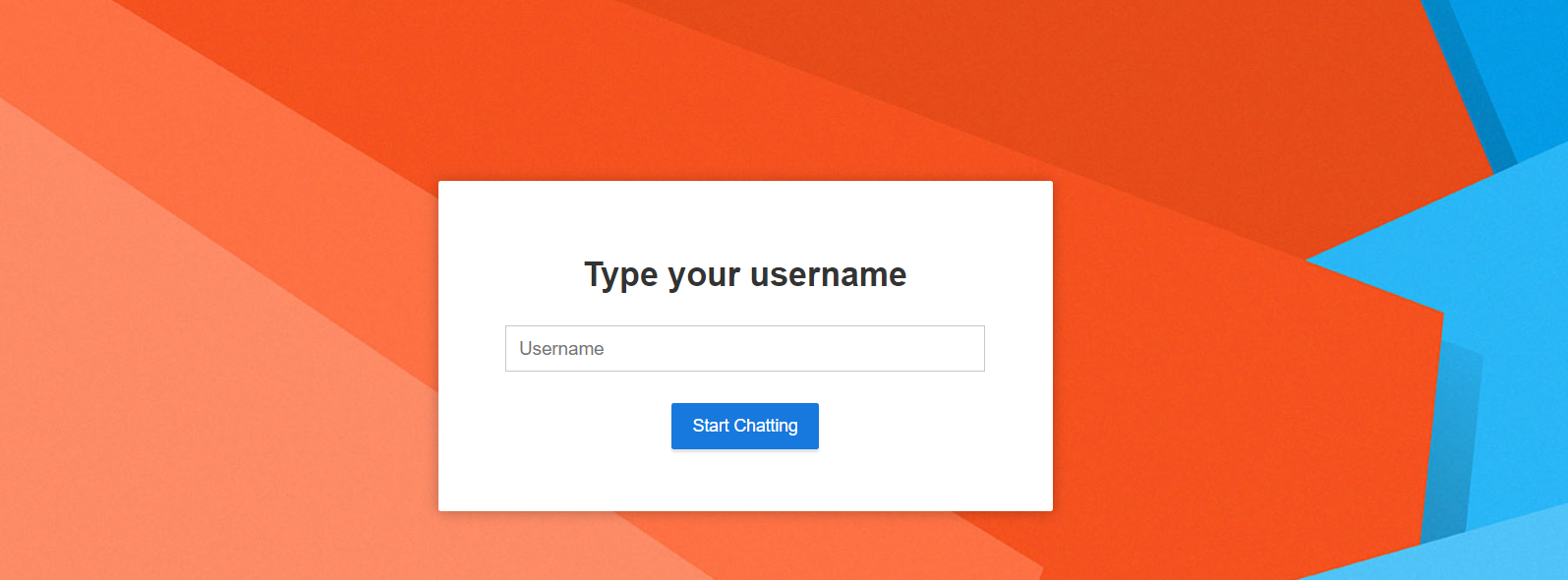
*package* ru.anvarzhonov.springbootwebsocket.controller;  
  
  
*import* org.springframework.messaging.handler.annotation.*MessageMapping*;  
*import* org.springframework.messaging.handler.annotation.*Payload*;  
*import* org.springframework.messaging.handler.annotation.*SendTo*;  
*import* org.springframework.messaging.simp.SimpMessageHeaderAccessor;  
*import* org.springframework.stereotype.*Controller*;  
*import* ru.anvarzhonov.springbootwebsocket.models.ChatMessage;  
  
*@Controller  
public class* ChatController {  
  
 *@MessageMapping*("/chat.register")  
 *@SendTo*("/topic/public")  
 *public* ChatMessage register(ChatMessage chatMessage, SimpMessageHeaderAccessor headerAccessor) {  
 headerAccessor.getSessionAttributes().put("username", chatMessage.getSender());  
 *return* chatMessage;  
 }  
  
 *@MessageMapping*("/chat.send")  
 *@SendTo*("/topic/public")  
 *public* ChatMessage sendMessage (ChatMessage chatMessage) {  
  
 *return* chatMessage;  
 }  
}

**View (index.html):**

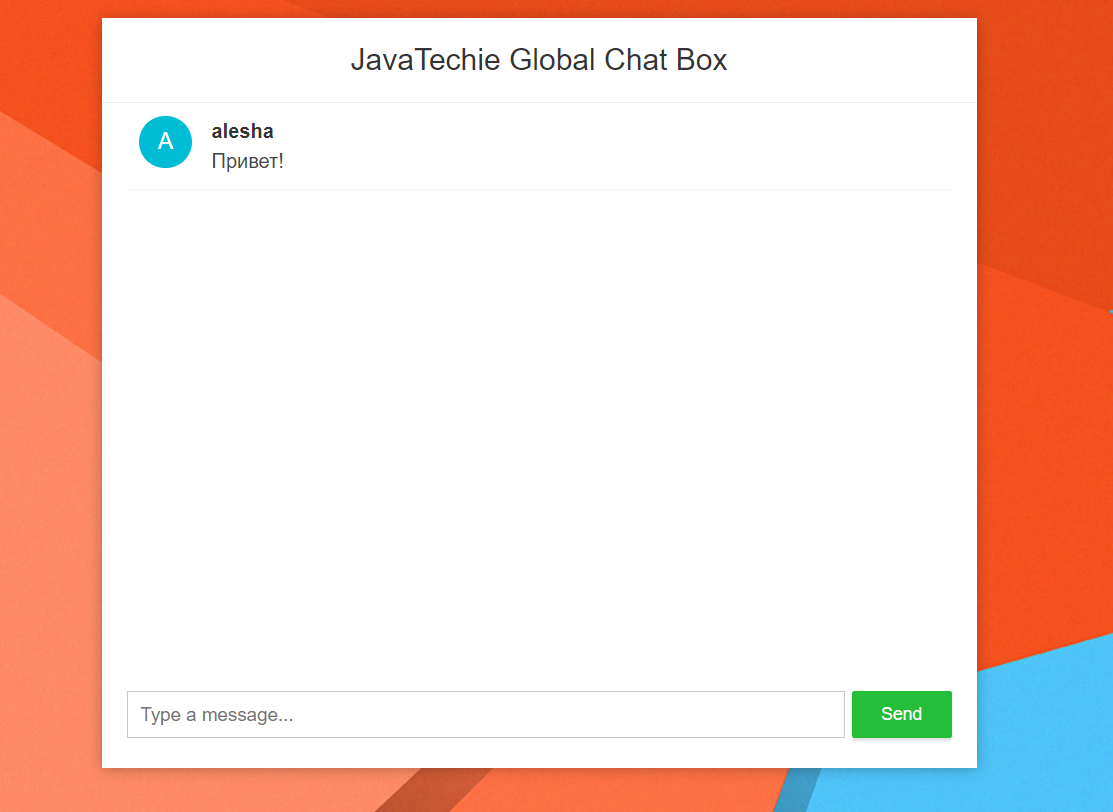
<!DOCTYPE *html*>  
<html>  
<head>  
 <meta *name*="viewport"  
 *content*="width=device-width, initial-scale=1.0, minimum-scale=1.0">  
 <title>Spring Boot WebSocket Chat Application | CalliCoder</title>  
 <link *rel*="stylesheet" *href*="main.css" />  
</head>  
<body *background*="18.jpg"  
 *style*="background-position: center; background-repeat: no-repeat; background-size: cover;">  
<noscript>  
 <h2>Sorry! Your browser doesn't support Javascript</h2>  
</noscript>  
  
<div *id*="username-page">  
 <div *class*="username-page-container">  
 <h1 *class*="title">Type your username</h1>  
 <form *id*="usernameForm" *name*="usernameForm">  
 <div *class*="form-group">  
 <input *type*="text" *id*="name" *placeholder*="Username"  
 *autocomplete*="off" *class*="form-control" />  
 </div>  
 <div *class*="form-group">  
 <button *type*="submit" *class*="accent username-submit">Start  
 Chatting</button>  
 </div>  
 </form>  
 </div>  
</div>  
  
<div *id*="chat-page" *class*="hidden">  
 <div *class*="chat-container">  
 <div *class*="chat-header">  
 <h2>Anvarzhonov Chat</h2>  
 </div>  
 <div *class*="connecting">Connecting...</div>  
 <ul *id*="messageArea">  
  
 </ul>  
 <form *id*="messageForm" *name*="messageForm" *nameForm*="messageForm">  
 <div *class*="form-group">  
 <div *class*="input-group clearfix">  
 <input *type*="text" *id*="message" *placeholder*="Type a message..."  
 *autocomplete*="off" *class*="form-control" />  
 <button *type*="submit" *class*="primary">Send</button>  
 </div>  
 </div>  
 </form>  
 </div>  
</div>  
  
<script  
 *src*="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.1.4/sockjs.min.js"></script>  
<script  
 *src*="https://cdnjs.cloudflare.com/ajax/libs/stomp.js/2.3.3/stomp.min.js"></script>  
<script *src*="main.js"></script>  
</body>  
</html>

Покажем результат работы:

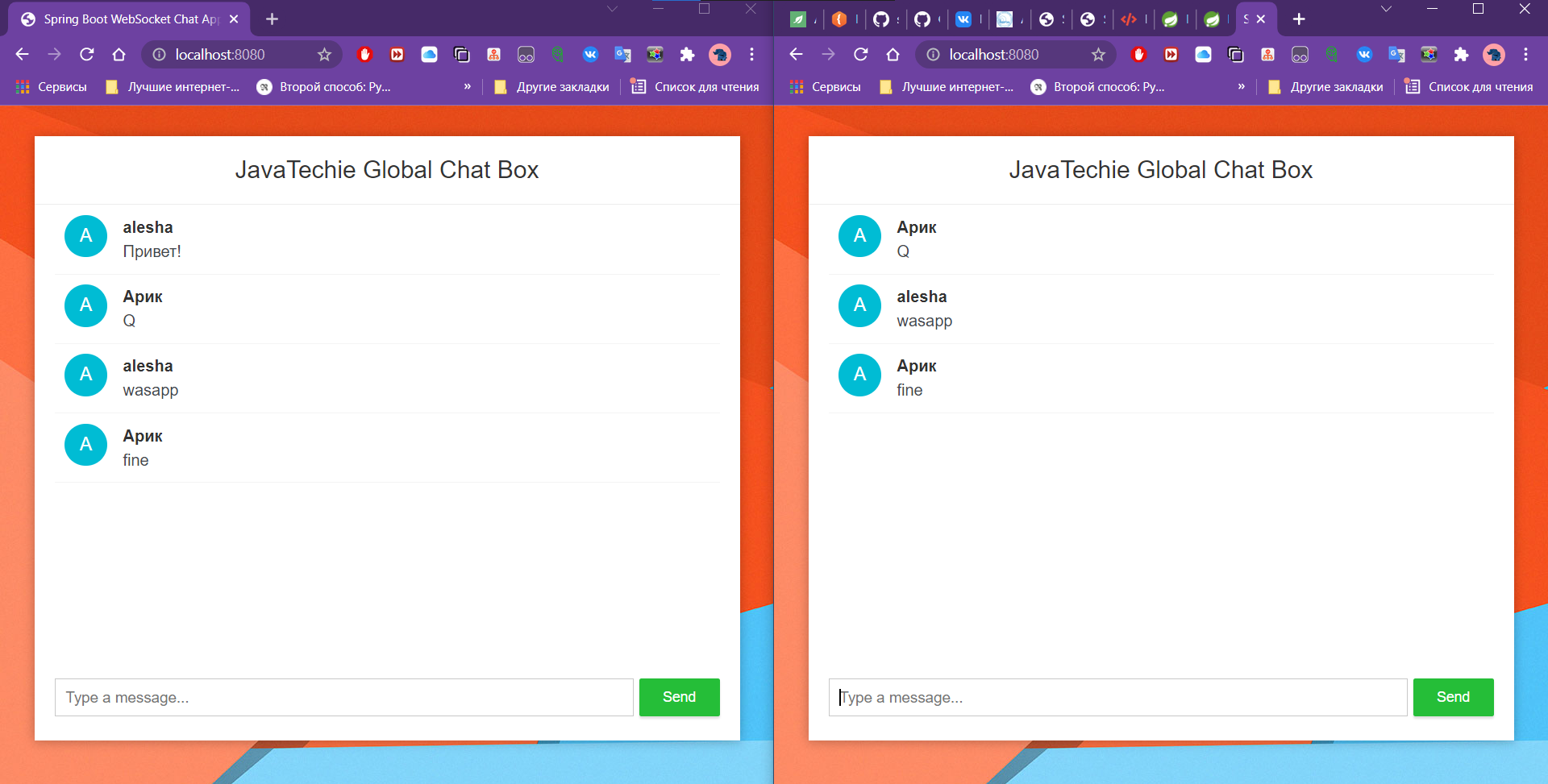
В начале работы приложения требуется ввести имя:



Попробуем зарегистрировать пользователя и написать сообщение:



Добавим еще одного клиента и покажем результат обмена сообщениями:



**Вывод**

В результате выполнения ознакомились с базовой разработкой клиент-серверного приложения с использованием технологии Websocket и Spring Framework.