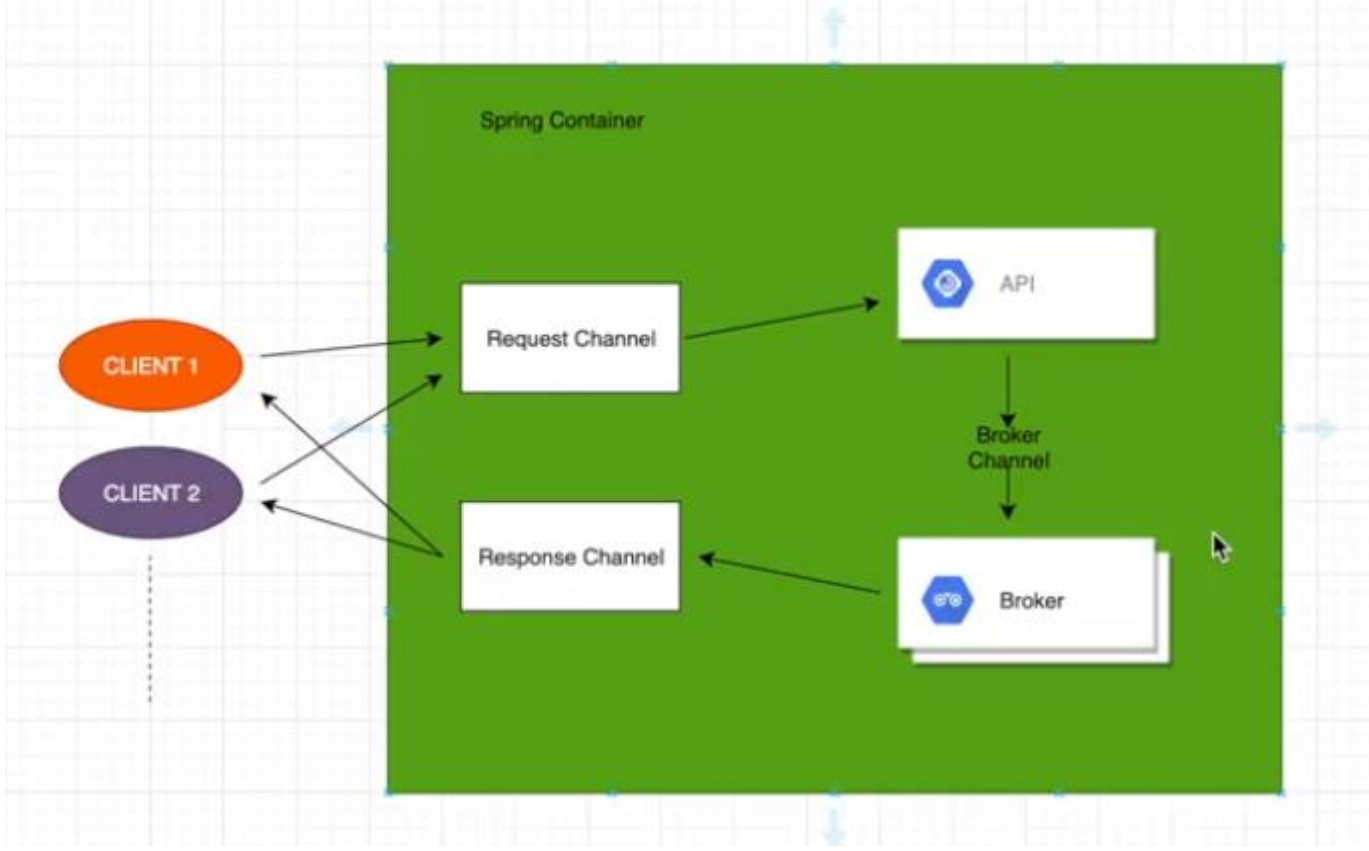


WebSocket yapısının nasıl çalıştığını inceleyeceğiz.  
Basit bir html sayfası ile websocket üzerinden birden fazla client'ı bağlayıp chat uygulaması yapacağız.

Request Channel, websocket'e client olduğumuz yerdir. İsteklerimizi request channel'a yaparız.  
Response Channel'a ise subscribe oluyor yani burayı dinliyoruz. Server'dan gelen mesajları aldığımız channel'dır.  
Broker üzerinden bütün kanallara mesaj göndereceğiz veya sadece client2'ye mesaj gönderebilsin gibi işlemleri yapabiliriz.



## WebSocket

WebSocket protokolü, tek bir TCP bağlantısı üzerinden client ve server arasında full-duplex, two-way communication kanalı kurmak için standart bir yol sağlar.

HTTP'den farklı bir TCP protokolüdür ancak 80 ve 443 numaralı port'ları kullanarak ve mevcut firewall kurallarının yeniden kullanılmasına izin vererek HTTP üzerinden çalışmak üzere tasarlanmıştır.

## STOMP Nedir?

STOMP, **Simple Text Oriented Messaging Protocol** (Basit Metin Yönelimli Mesajlaşma Protokolü) anlamına gelir.

Veri alışverişi için formatı ve kuralları tanımlayan bir mesajlaşma protokolüdür.

*Neden STOMP'a ihtiyacımız var?*

WebSocket sadece bir iletişim protokolüdür. Yalnızca belirli bir topic'e subscribe olan kullanıcılara nasıl mesaj gönderilir veya belirli bir kullanıcıya nasıl mesaj gönderilir gibi şeyleri tanımlamaz.  
Bu işlevler için STOMP'a ihtiyacımız var.

## SockJS Nedir?

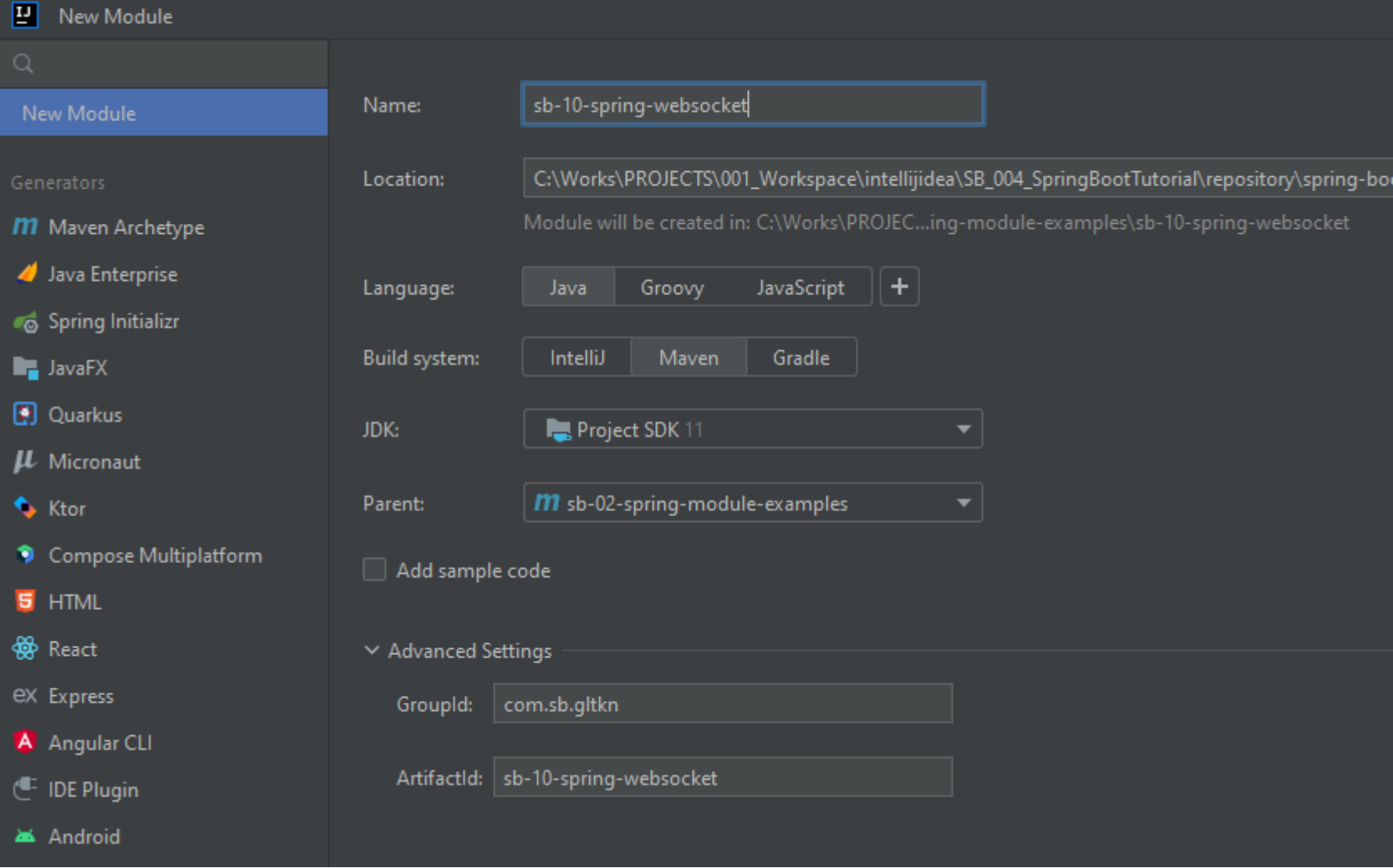
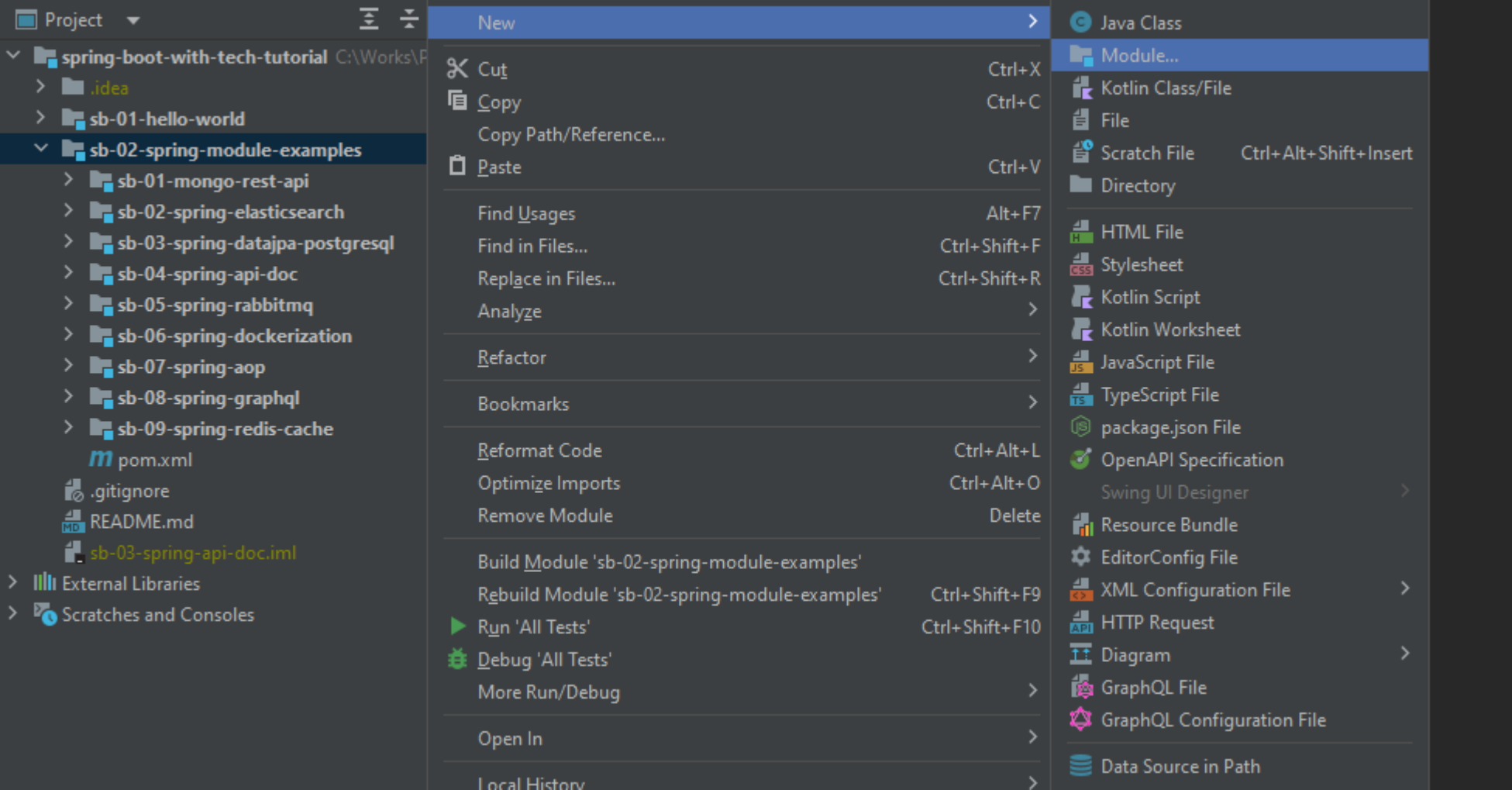
**SockJS**, WebSocket benzeri bir nesne sağlayan bir browser JavaScript library'sidir.

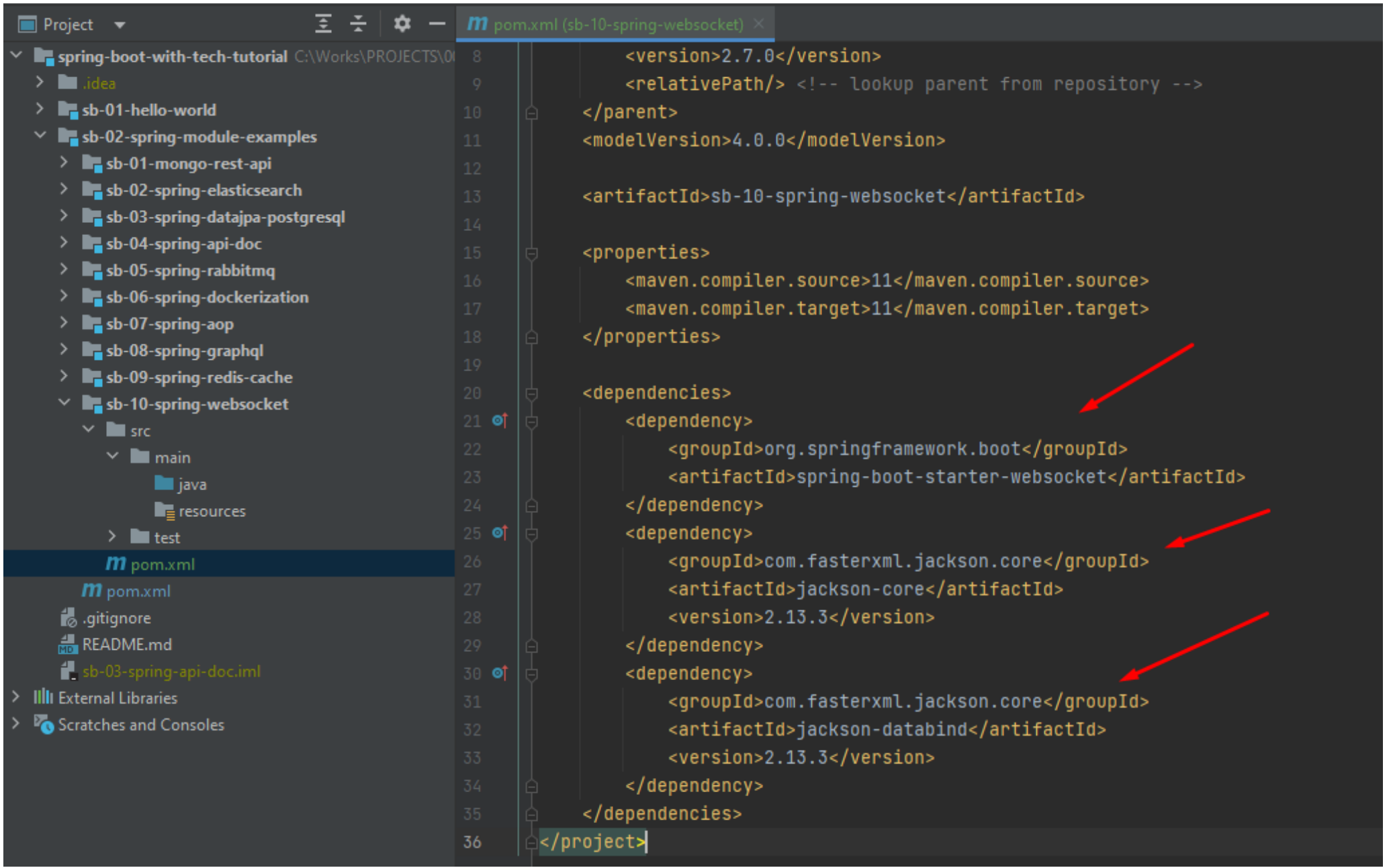
**SockJS**, browser ve web server arasında düşük gecikmeli(low latency), tam çift yönlü(full duplex), cross-domain iletişim kanalı oluşturan tutarlı, cross-browser, Javascript API'si sunar.

SockJS önce native WebSocket'leri kullanmaya çalışır.  
Bu başarısız olursa, tarayıcıya özgü çeşitli aktarım protokollerini kullanabilir ve bunları WebSocket benzeri abstraction'lar aracılığıyla sunar.

SockJS, WebSocket'i desteklemeyen browser'lar için geri dönüş seçeneklerini etkinleştirmek için kullanılır.  
Spring, SockJS'in server-side implementation'unu sağlar.

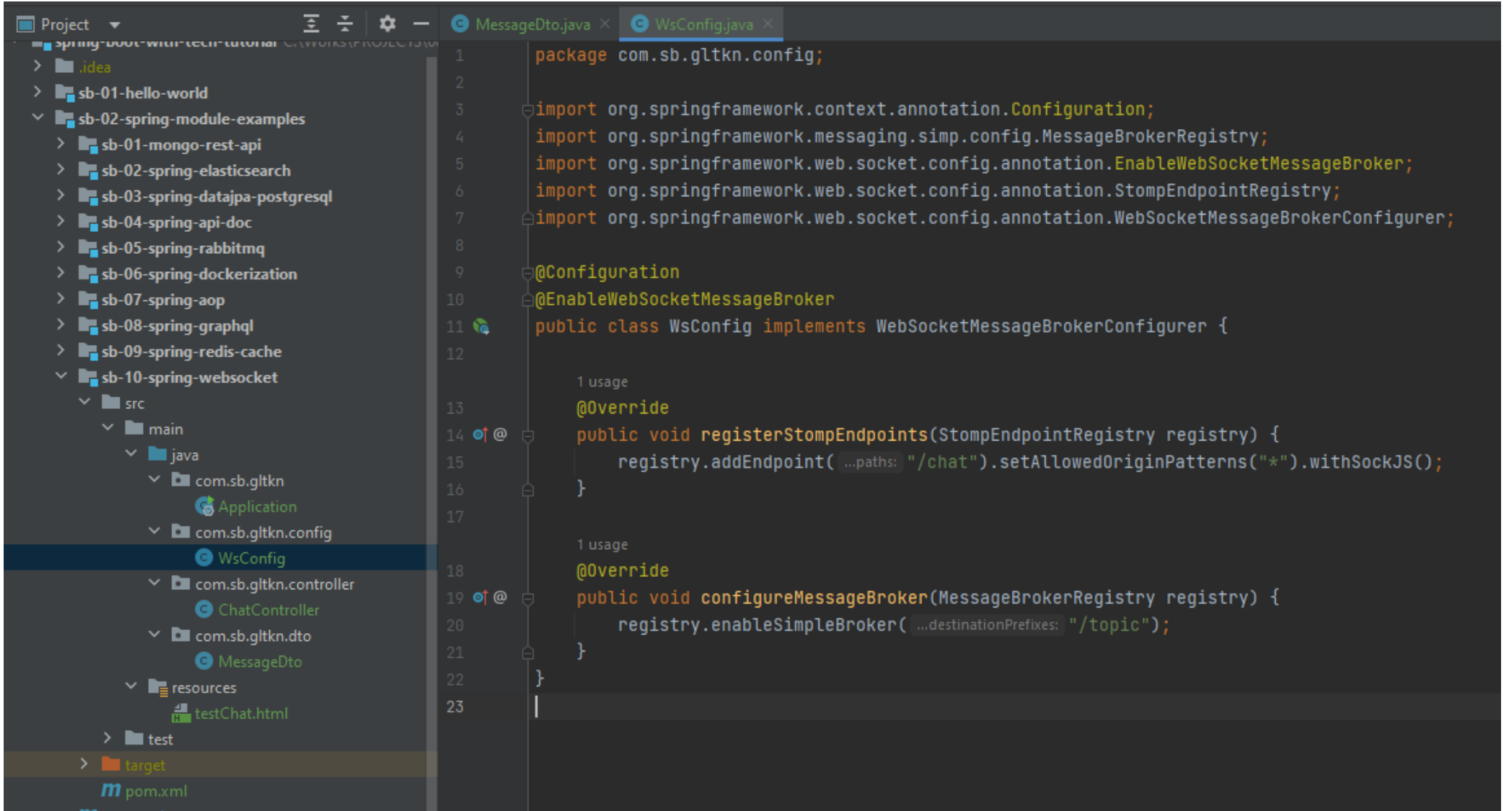
**Spring WebSocket**, WebSocket tarzı mesajlaşma desteği sağlayan Spring modülüdür.  
Spring WebSocket'in dokümantasyonunda belirtildiği gibi, WebSocket protokolü web uygulamaları için önemli bir yeni yetenek tanımlar: full-duplex, client ve server arasında two-way communication (çift yönlü iletişim).





```
8      <version>2.7.0</version>
9      <relativePath/> <!-- lookup parent from repository -->
10    </parent>
11    <modelVersion>4.0.0</modelVersion>
12
13    <artifactId>sb-10-spring-websocket</artifactId>
14
15    <properties>
16      <maven.compiler.source>11</maven.compiler.source>
17      <maven.compiler.target>11</maven.compiler.target>
18    </properties>
19
20    <dependencies>
21      <dependency>
22        <groupId>org.springframework.boot</groupId>
23        <artifactId>spring-boot-starter-websocket</artifactId>
24      </dependency>
25      <dependency>
26        <groupId>com.fasterxml.jackson.core</groupId>
27        <artifactId>jackson-core</artifactId>
28        <version>2.13.3</version>
29      </dependency>
30      <dependency>
31        <groupId>com.fasterxml.jackson.core</groupId>
32        <artifactId>jackson-databind</artifactId>
33        <version>2.13.3</version>
34      </dependency>
35    </dependencies>
36  </project>
```

Aşağıdaki **/chat** endpoint'ini yayınladık ve bu endpoint'i server dinlerken, **/topic** endpoint'ini client dinlemektedir.



```
1 package com.sb.gltkn.config;
2
3 import org.springframework.context.annotation.Configuration;
4 import org.springframework.messaging.simp.config.MessageBrokerRegistry;
5 import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;
6 import org.springframework.web.socket.config.annotation.StompEndpointRegistry;
7 import org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer;
8
9 @Configuration
10 @EnableWebSocketMessageBroker
11 public class WsConfig implements WebSocketMessageBrokerConfigurer {
12
13     1 usage
14     @Override
15     public void registerStompEndpoints(StompEndpointRegistry registry) {
16         registry.addEndpoint( ...paths: "/chat").setAllowedOriginPatterns("*").withSockJS();
17     }
18
19     1 usage
20     @Override
21     public void configureMessageBroker(MessageBrokerRegistry registry) {
22         registry.enableSimpleBroker( ...destinationPrefixes: "/topic");
23     }
24 }
```





Client aşağıdaki ChatController'da **/chat** path'ine istekte bulunacak. Bu endpoint'i server dinlemekteydi. Sonrasından **/topic** endpoint'ine **messageDto**'yu gönderiyoruz.

Buradaki yapıda **/topic** endpoint'ini dinleyen tüm client'lara **messageDto** gönderilecektir.

Project

▼ spring-boot-with-tech-tutorial C:\Works\PROJECTS\01

> .idea

> sb-01-hello-world

▼ sb-02-spring-module-examples

> sb-01-mongo-rest-api

> sb-02-spring-elasticsearch

> sb-03-spring-datajpa-postgresql

> sb-04-spring-api-doc

> sb-05-spring-rabbitmq

> sb-06-spring-dockerization

> sb-07-spring-aop

> sb-08-spring-graphql

> sb-09-spring-redis-cache

▼ sb-10-spring-websocket

▼ src

▼ main

▼ java

▼ com.sb.gltkn

Application

▼ com.sb.gltkn.config

WsConfig

▼ com.sb.gltkn.controller

ChatController

▼ com.sb.gltkn.dto

MessageDto

resources

> test

m pom.xml

m pom.xml

.gitignore

MD README.md

sb-03-spring-api-doc.iml

ChatController.java

1 package com.sb.gltkn.controller;

2

3 import com.sb.gltkn.dto.MessageDto;

4 import org.springframework.beans.factory.annotation.Autowired;

5 import org.springframework.messaging.handler.annotation.MessageMapping;

6 import org.springframework.messaging.handler.annotation.Payload;

7 import org.springframework.messaging.handler.annotation.SendTo;

8 import org.springframework.messaging.simp.SimpMessagingTemplate;

9 import org.springframework.messaging.simp.annotation.SendToUser;

10 import org.springframework.stereotype.Controller;

11 import org.springframework.web.bind.annotation.CrossOrigin;

12

13 @Controller

14 @CrossOrigin

15 public class ChatController {

16

17 @Autowired

18 private SimpMessagingTemplate messagingTemplate;

19

20 @MessageMapping("/chat")

21 //@SendTo("/topic")

22 //@SendToUser

23 public void chatEndpoint(@Payload MessageDto messageDto){

24 System.out.println(messageDto);

25 messagingTemplate.convertAndSend(destination: "/topic", messageDto);

26 }

27 }

28

Bir html sayfası üzerinden client'ların **/chat** endpoint'ini çağırdığı basit bir arayüz yazalım.

MessageDto.java × WsConfig.java × ChatController.java × testChat.html ×

1 <!DOCTYPE html>

2 <html>

3 <head>

4 <title>Chat WebSocket</title>

5 <script src="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.4.0/sockjs.js"></script>

6 <script src="https://cdnjs.cloudflare.com/ajax/libs/stomp.js/2.3.3/stomp.js"></script>

7

8 <script type="text/javascript">

9 var stompClient = null;

10 var baseAddress = 'http://localhost:8080';

11

12 function setConnected(connected) {

13 document.getElementById('connect').disabled = connected;

14 document.getElementById('disconnect').disabled = !connected;

15 document.getElementById('conversationDiv').style.visibility = connected ? 'visible' : 'hidden';

16 document.getElementById('response').innerHTML = '';

17 }

18

19 function connect() {

20 var socket = new SockJS(baseAddress + '/chat');

21 stompClient = Stomp.over(socket);

22 stompClient.connect({}, function (frame) {

23 setConnected(true);

24 console.log('Connected: ' + frame);

25 stompClient.subscribe('/topic', function (message) {

26 handleReceivedMessage(JSON.parse(message.body));

27 });

28 });

29 }

30

```

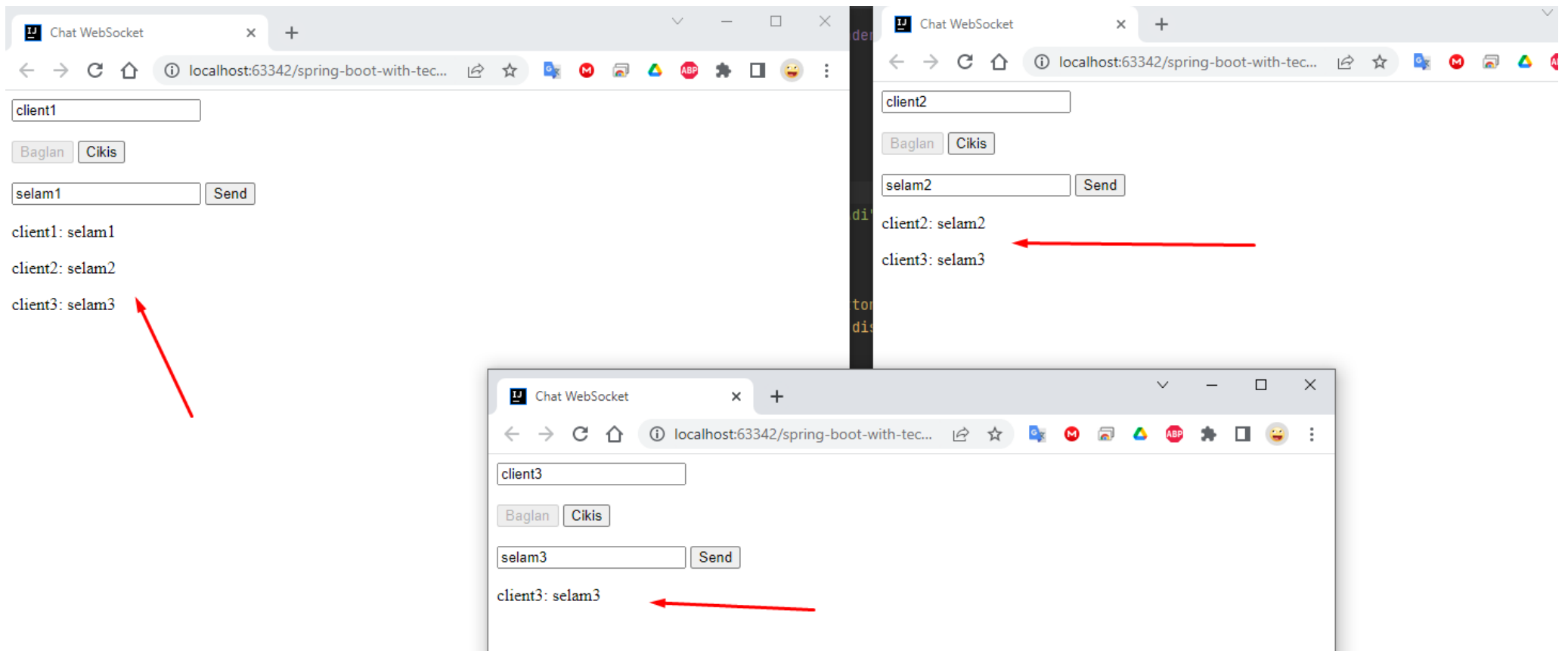
30
31     function disconnect() {
32         if (stompClient != null) {
33             stompClient.disconnect();
34         }
35         setConnected(false);
36         console.log("Disconnected");
37     }
38
39     function sendMessage() {
40         var from = document.getElementById('from').value;
41         var text = document.getElementById('text').value;
42         stompClient.send("/chat", {},
43             JSON.stringify({'sender': from, 'message': text}));
44     }
45
46     function handleReceivedMessage(message) {
47         var response = document.getElementById('response');
48         var p = document.createElement('p');
49         p.style.wordWrap = 'break-word';
50         p.appendChild(document.createTextNode(message.sender + ": " + message.message));
51         response.appendChild(p);
52     }
53 </script>
54 </head>

```

```

55 <body onload="disconnect()">
56 <div>
57     <div>
58         <input type="text" id="from" placeholder="Kullanici Adi"/>
59     </div>
60     <br/>
61     <div>
62         <button id="connect" onclick="connect();">Baglan</button>
63         <button id="disconnect" disabled="disabled" onclick="disconnect();">
64             Cikis
65         </button>
66     </div>
67     <br/>
68     <div id="conversationDiv">
69         <input type="text" id="text" placeholder="Mesaj.."/>
70         <button id="sendMessage" onclick="sendMessage();">Send</button>
71         <p id="response"></p>
72     </div>
73 </div>
74
75 </body>
76 </html>
77

```



```
Console  Actuator
2022-06-27 22:31:34.838 INFO 11396 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-06-27 22:31:34.838 INFO 11396 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.40]
2022-06-27 22:31:34.944 INFO 11396 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplication
2022-06-27 22:31:34.945 INFO 11396 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2022-06-27 22:31:35.395 INFO 11396 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with
2022-06-27 22:31:35.397 INFO 11396 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler : Starting...
2022-06-27 22:31:35.397 INFO 11396 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler : BrokerAvailabilityEvent[available=true, Sim
2022-06-27 22:31:35.397 INFO 11396 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler : Started.
2022-06-27 22:31:35.414 INFO 11396 --- [main] com.sb.gltkn.Application : Started Application in 2.226 seconds (JVM r
2022-06-27 22:31:45.219 INFO 11396 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'disp
2022-06-27 22:31:45.219 INFO 11396 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-06-27 22:31:45.220 INFO 11396 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
MessageDto{sender='client1', message='selam1'}
MessageDto{sender='client2', message='selam2'}
MessageDto{sender='client3', message='selam3'}
2022-06-27 22:32:35.112 INFO 11396 --- [MessageBroker-6] o.s.w.s.c.WebSocketMessageBrokerStats : WebSocketSession[3 current WS(3)-HttpStream
```