

🏠 首页 (<https://www.zifangsky.cn/>) » Java (<https://www.zifangsky.cn/java/>) » Spring (<https://www.zifangsky.cn/java/spring/>) » 正文

Spring Boot中使用WebSocket总结（一）：几种实现方式详解

📅 2018/11/20 | 📁 Spring (<https://www.zifangsky.cn/java/spring/>) |
👤 admin (<https://www.zifangsky.cn/author/zifangskyw/>)

简介

所谓WebSocket，类似于Socket，它的作用是可以让Web应用中的客户端和服务端建立全双工通信。在基于Spring的应用中使用WebSocket一般可以有以下三种方式：

- 使用Java提供的@ServerEndpoint注解实现
- 使用Spring提供的低层级WebSocket API实现
- 使用STOMP消息实现

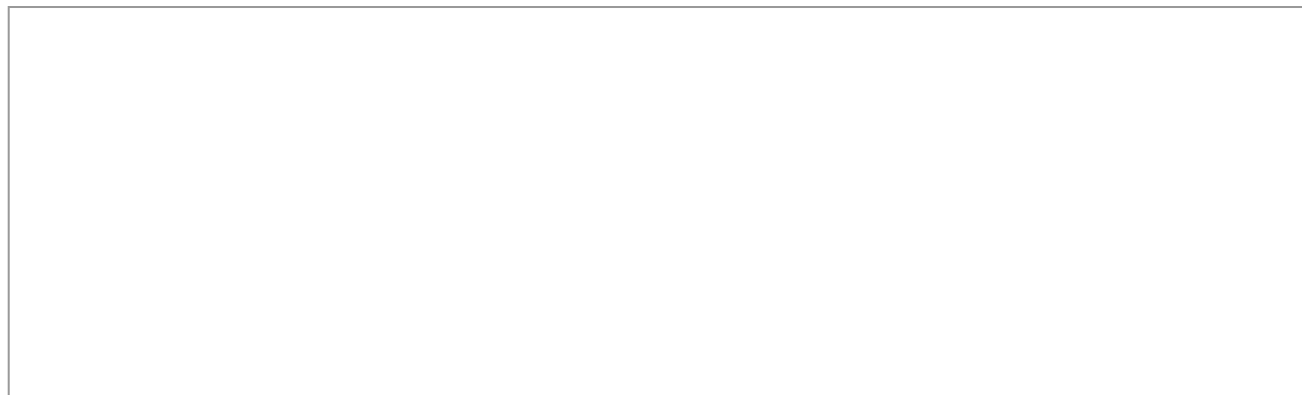
下面，我将对这三种实现方式做一个简单介绍，此外有关WebSocket性质的更多介绍可以参考以下这篇文章：[WebSocket探秘 \(https://juejin.im/post/5a1bdf676fb9a045055dd99d\)](https://juejin.im/post/5a1bdf676fb9a045055dd99d)

注：本篇文章的完整源码可以参考：<https://github.com/zifangsky/WebSocketDemo>
(<https://github.com/zifangsky/WebSocketDemo>)

使用Java提供的@ServerEndpoint注解实现

（1）使用@ServerEndpoint注解监听一个WebSocket请求路径：

这里监听了客户端的连接端口 /reverse，并定义了如何处理客户端发来的消息



```
1 package cn.zifangsky.samplewebsocket.websocket;
2
3 import javax.websocket.OnMessage;
4 import javax.websocket.Session;
5 import javax.websocket.server.ServerEndpoint;
6 import java.io.IOException;
7
8 /**
9  * ReverseWebSocketEndpoint
10  *
11  * @author zifangsky
12  * @date 2018/9/30
13  * @since 1.0.0
14  */
15 @ServerEndpoint("/reverse")
16 public class ReverseWebSocketEndpoint {
17
18     @OnMessage
19     public void handleMessage(Session session, String message) throws IOException {
20         session.getBasicRemote().sendText("Reversed: " + new StringBuilder(message).reverse());
21     }
22
23 }
```

(2) WebSocket相关配置:

```
1 package cn.zifangsky.samplewebsocket.config;
2
3 import cn.zifangsky.samplewebsocket.websocket.ReverseWebSocketEndpoint;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.web.socket.config.annotation.EnableWebSocket;
7 import org.springframework.web.socket.server.standard.ServerEndpointExporter;
8
9 /**
10  * WebSocket相关配置
11  *
12  * @author zifangsky
13  * @date 2018/9/30
14  * @since 1.0.0
15  */
16 @Configuration
17 @EnableWebSocket
18 public class WebSocketConfig{
19
20     @Bean
21     public ReverseWebSocketEndpoint reverseWebSocketEndpoint() {
22         return new ReverseWebSocketEndpoint();
23     }
24
25     @Bean
26     public ServerEndpointExporter serverEndpointExporter() {
27         return new ServerEndpointExporter();
28     }
29
30 }
```

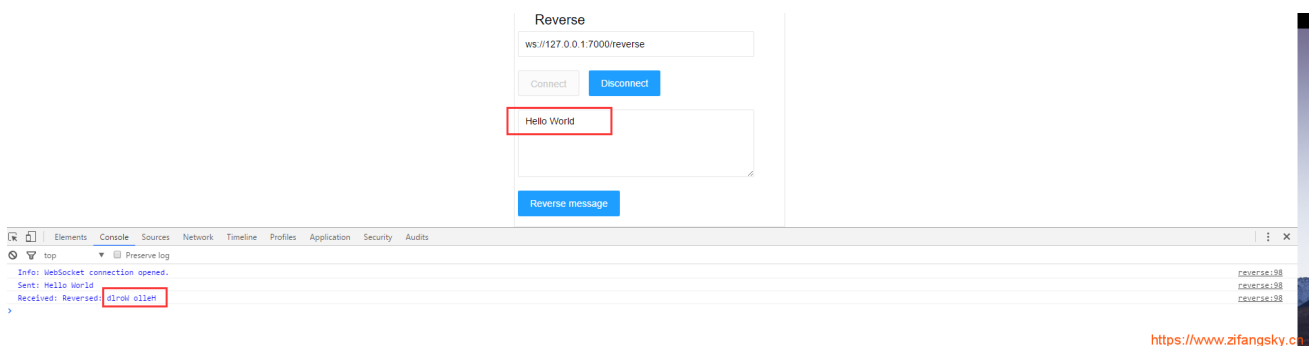
(3) 示例页面：

```
1 <html xmlns:th="http://www.thymeleaf.org">
2 <head>
3     <meta content="text/html; charset=UTF-8"/>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
5     <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
6     <meta name="viewport" content="width=device-width, initial-scale=1"/>
7     <title>WebSocket Examples: Reverse</title>
8     <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
9     <script th:src="@{/layui/layui.js}"></script>
10    <link th:href="@{/layui/css/layui.css}" rel="stylesheet">
11    <style type="text/css">
12        #connect-container {
13            margin: 0 auto;
14            width: 400px;
15        }
16
17        #connect-container div {
18            padding: 5px;
19            margin: 0 7px 10px 0;
20        }
21
22        .layui-btn {
23            display: inline-block;
24        }
25    </style>
26    <script type="text/javascript">
27        var ws = null;
28
29        $(function () {
30            var target = $("#target");
31            if (window.location.protocol === 'http:') {
32                target.val('ws://' + window.location.host + target.val());
33            } else {
34                target.val('wss://' + window.location.host + target.val());
35            }
36        });
37
38        function setConnected(connected) {
39            var connect = $("#connect");
40            var disconnect = $("#disconnect");
41            var reverse = $("#reverse");
42
43            if (connected) {
44                connect.addClass("layui-btn-disabled");
45                disconnect.removeClass("layui-btn-disabled");
46                reverse.removeClass("layui-btn-disabled");
47            } else {
48                connect.removeClass("layui-btn-disabled");
49                disconnect.addClass("layui-btn-disabled");
50                reverse.addClass("layui-btn-disabled");
51            }
52
53            connect.attr("disabled", connected);
54            disconnect.attr("disabled", !connected);
55            reverse.attr("disabled", !connected);
56        }
```

```
57
58 //连接
59 function connect() {
60     var target = $("#target").val();
61
62     ws = new WebSocket(target);
63     ws.onopen = function () {
64         setConnected(true);
65         log('Info: WebSocket connection opened.');
```

```
115         onclick="disconnect();">Disconnect
116     </button>
117
118 </div>
119 <div>
120     <textarea id="message" class="layui-textarea" placeholder="请输入需要反
121 </div>
122 <div>
123     <button id="reverse" class="layui-btn layui-btn-normal layui-btn-disc
124         onclick="reverse();">Reverse message
125     </button>
126 </div>
127 </div>
128 </div>
129 </body>
130 </html>
```

启动项目后访问页面，效果如下：



使用Spring提供的低层级WebSocket API实现

Spring 4.0为WebSocket通信提供了支持，包括：

- 发送和接收消息的低层级API；
- 发送和接收消息的高级API；
- 用来发送消息的模板；
- 支持SockJS，用来解决浏览器端、服务器以及代理不支持WebSocket的问题。

使用Spring提供的低层级API实现WebSocket，主要需要以下几个步骤：

（1）添加一个WebSocketHandler：

定义一个继承了 `AbstractWebSocketHandler` 类的消息处理类，然后自定义对”建立连接“、”接收/发送消息“、”异常情况“等情况进行处理

```
1 package cn.zifangsky.samplewebsocket.websocket;
2
3 import cn.zifangsky.samplewebsocket.service.EchoService;
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6 import org.springframework.web.socket.CloseStatus;
7 import org.springframework.web.socket.TextMessage;
8 import org.springframework.web.socket.WebSocketSession;
9 import org.springframework.web.socket.handler.TextWebSocketHandler;
10
11 import javax.annotation.Resource;
12 import java.text.MessageFormat;
13
14 /**
15  * 通过继承 {@link org.springframework.web.socket.handler.AbstractWebSocketHandler} 的示
16  *
17  * @author zifangsky
18  * @date 2018/10/9
19  * @since 1.0.0
20  */
21 public class EchoWebSocketHandler extends TextWebSocketHandler{
22     private final Logger logger = LoggerFactory.getLogger(getClass());
23
24     @Resource(name = "echoServiceImpl")
25     private EchoService echoService;
26
27     @Override
28     public void afterConnectionEstablished(WebSocketSession session) throws Exception {
29         logger.debug("Opened new session in instance " + this);
30     }
31
32     @Override
33     protected void handleTextMessage(WebSocketSession session, TextMessage message) throws Exception {
34         //组装返回的Echo信息
35         String echoMessage = this.echoService.echo(message.getPayload());
36         logger.debug(MessageFormat.format("Echo message \"{0}\"", echoMessage));
37
38         session.sendMessage(new TextMessage(echoMessage));
39     }
40
41     @Override
42     public void handleTransportError(WebSocketSession session, Throwable exception) throws Exception {
43         session.close(CloseStatus.SERVER_ERROR);
44         logger.debug("Info: WebSocket connection closed.");
45     }
46 }
```

(2) WebSocket相关配置：

```
1 package cn.zifangsky.samplewebsocket.config;
2
3 import cn.zifangsky.samplewebsocket.websocket.EchoWebSocketHandler;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.web.socket.WebSocketHandler;
7 import org.springframework.web.socket.config.annotation.EnableWebSocket;
8 import org.springframework.web.socket.config.annotation.WebSocketConfigurer;
9 import org.springframework.web.socket.config.annotation.WebSocketHandlerRegistry;
10
11 /**
12  * WebSocket相关配置
13  *
14  * @author zifangsky
15  * @date 2018/9/30
16  * @since 1.0.0
17  */
18 @Configuration
19 @EnableWebSocket
20 public class WebSocketConfig implements WebSocketConfigurer{
21
22     @Override
23     public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
24         registry.addHandler(echoWebSocketHandler(), "/echoMessage");
25         registry.addHandler(echoWebSocketHandler(), "/echoMessage_SockJS").withSockJS();
26     }
27
28     /**
29      * 通过继承 {@link org.springframework.web.socket.handler.AbstractWebSocketHandler}
30      */
31     @Bean
32     public WebSocketHandler echoWebSocketHandler(){
33         return new EchoWebSocketHandler();
34     }
35
36 }
```

从上面代码可以看出，这里除了配置了基本的 WebSocket（也就是`/echoMessage`这个连接地址），还使用 SockJS 配置了浏览器不支持 WebSocket 技术时的替代方案（也就是`/echoMessage_SockJS`这个连接地址）。

(3) 两个示例页面：

i) echo.html:

```
1 <html xmlns:th="http://www.thymeleaf.org">
2 <head>
3     <meta content="text/html; charset=UTF-8"/>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
5     <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
6     <meta name="viewport" content="width=device-width, initial-scale=1"/>
7     <title>WebSocket Examples: Reverse</title>
8     <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
9     <script th:src="@{/layui/layui.js}"></script>
10    <link th:href="@{/layui/css/layui.css}" rel="stylesheet">
11    <style type="text/css">
```

```
12     #connect-container {
13         margin: 0 auto;
14         width: 400px;
15     }
16
17     #connect-container div {
18         padding: 5px;
19         margin: 0 7px 10px 0;
20     }
21
22     .layui-btn {
23         display: inline-block;
24     }
25 </style>
26 <script type="text/javascript">
27     var ws = null;
28
29     $(function () {
30         var target = $("#target");
31         if (window.location.protocol === 'http:') {
32             target.val('ws://' + window.location.host + target.val());
33         } else {
34             target.val('wss://' + window.location.host + target.val());
35         }
36     });
37
38     function setConnected(connected) {
39         var connect = $("#connect");
40         var disconnect = $("#disconnect");
41         var echo = $("#echo");
42
43         if (connected) {
44             connect.addClass("layui-btn-disabled");
45             disconnect.removeClass("layui-btn-disabled");
46             echo.removeClass("layui-btn-disabled");
47         } else {
48             connect.removeClass("layui-btn-disabled");
49             disconnect.addClass("layui-btn-disabled");
50             echo.addClass("layui-btn-disabled");
51         }
52
53         connect.attr("disabled", connected);
54         disconnect.attr("disabled", !connected);
55         echo.attr("disabled", !connected);
56     }
57
58     //连接
59     function connect() {
60         var target = $("#target").val();
61
62         ws = new WebSocket(target);
63
64         ws.onopen = function () {
65             setConnected(true);
66             log('Info: WebSocket connection opened.');
```



```
70     ws.onclose = function () {
71         setConnected(false);
72         log('Info: WebSocket connection closed.');
```

```
73     };
74 }
75
76 //断开连接
77 function disconnect() {
78     if (ws != null) {
79         ws.close();
80         ws = null;
81     }
82     setConnected(false);
83 }
84
85 //Echo
86 function echo() {
87     if (ws != null) {
88         var message = $("#message").val();
89         log('Sent: ' + message);
90         ws.send(message);
91     } else {
92         alert('WebSocket connection not established, please connect.');
```

```
93     }
94 }
95
96 //日志输出
97 function log(message) {
98     console.debug(message);
99 }
100 </script>
101 </head>
102 <body>
103     <noscript><h2 style="color: #ff0000">Seems your browser doesn't support Javascript
104     enabled. Please enable
105     Javascript and reload this page!</h2></noscript>
106     <div>
107         <div id="connect-container" class="layui-elem-field">
108             <legend>Echo</legend>
109             <div>
110                 <input id="target" type="text" class="layui-input" size="40" style="width: 100%; border: 1px solid #ccc;" />
111             </div>
112             <div>
113                 <button id="connect" class="layui-btn layui-btn-normal" onclick="connect();" >Connect
114                 <button id="disconnect" class="layui-btn layui-btn-normal layui-btn-disabled"
115                     onclick="disconnect();" >Disconnect
116                 </button>
117             </div>
118         </div>
119         <div>
120             <textarea id="message" class="layui-textarea" placeholder="请输入请求的消息" />
121         </div>
122         <div>
123             <button id="echo" class="layui-btn layui-btn-normal layui-btn-disabled"
124                 onclick="echo();" >Echo message
125             </button>
126         </div>
127     </div>
```

```
128     </div>
129 </body>
130 </html>
```

ii) echo_sockjs.html:

```
1 <html xmlns:th="http://www.thymeleaf.org">
2 <head>
3     <meta content="text/html; charset=UTF-8"/>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
5     <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
6     <meta name="viewport" content="width=device-width, initial-scale=1"/>
7     <title>WebSocket Examples: Reverse</title>
8     <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"><
9     <script src="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.1.4/sockjs.mi
10    <script th:src="@{/layui/layui.js}"></script>
11    <link th:href="@{/layui/css/layui.css}" rel="stylesheet">
12    <style type="text/css">
13        #connect-container {
14            margin: 0 auto;
15            width: 400px;
16        }
17
18        #connect-container div {
19            padding: 5px;
20            margin: 0 7px 10px 0;
21        }
22
23        .layui-btn {
24            display: inline-block;
25        }
26    </style>
27    <script type="text/javascript">
28        var ws = null;
29
30        $(function () {
31            var target = $("#target");
32            if (window.location.protocol === 'http:') {
33                target.val('http://' + window.location.host + target.val());
34            } else {
35                target.val('https://' + window.location.host + target.val());
36            }
37        });
38
39        function setConnected(connected) {
40            var connect = $("#connect");
41            var disconnect = $("#disconnect");
42            var echo = $("#echo");
43
44            if (connected) {
45                connect.addClass("layui-btn-disabled");
46                disconnect.removeClass("layui-btn-disabled");
47                echo.removeClass("layui-btn-disabled");
48            } else {
49                connect.removeClass("layui-btn-disabled");
50                disconnect.addClass("layui-btn-disabled");
51                echo.addClass("layui-btn-disabled");
```

```
52     }
53
54     connect.attr("disabled", connected);
55     disconnect.attr("disabled", !connected);
56     echo.attr("disabled", !connected);
57 }
58
59 //连接
60 function connect() {
61     var target = $("#target").val();
62
63     ws = new SockJS(target);
64     ws.onopen = function () {
65         setConnected(true);
66         log('Info: WebSocket connection opened.');
```

```
110     <div>
111         <input id="target" type="text" class="layui-input" size="40" style="width: 100%;"/>
112     </div>
113     <div>
114         <button id="connect" class="layui-btn layui-btn-normal" onclick="connect();">Connect
115         <button id="disconnect" class="layui-btn layui-btn-normal layui-btn-disabled"
116             onclick="disconnect();">Disconnect
117     </button>
118
119     </div>
120     <div>
121         <textarea id="message" class="layui-textarea" placeholder="请输入请求的消息"/>
122     </div>
123     <div>
124         <button id="echo" class="layui-btn layui-btn-normal layui-btn-disabled"
125             onclick="echo();">Echo message
126     </button>
127 </div>
128 </div>
129 </div>
130 </body>
131 </html>
```

具体效果省略，可自行运行源码查看。

使用STOMP消息实现

所谓STOMP(Simple Text Oriented Messaging Protocol)，就是在WebSocket基础之上提供了一个基于帧的线路格式（frame-based wire format）层。它对发送简单文本消息定义了一套规范格式（STOMP消息基于Text，当然也支持传输二进制数据），目前很多服务端消息队列都已经支持STOMP，比如：RabbitMQ、ActiveMQ等。

(1) WebSocket相关配置：

```
1 package cn.zifangsky.stompwebsocket.config;
2
3 import cn.zifangsky.stompwebsocket.interceptor.websocket.MyChannelInterceptor;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.messaging.simp.config.ChannelRegistration;
7 import org.springframework.messaging.simp.config.MessageBrokerRegistry;
8 import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;
9 import org.springframework.web.socket.config.annotation.StompEndpointRegistry;
10 import org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer;
11
12 /**
13  * WebSocket相关配置
14  *
15  * @author zifangsky
16  * @date 2018/9/30
17  * @since 1.0.0
18  */
19 @Configuration
20 @EnableWebSocketMessageBroker
21 public class WebSocketConfig implements WebSocketMessageBrokerConfigurer{
22     @Autowired
23     private MyChannelInterceptor myChannelInterceptor;
24
25     @Override
26     public void registerStompEndpoints(StompEndpointRegistry registry) {
27         registry.addEndpoint("/stomp-websocket").withSockJS();
28     }
29
30     @Override
31     public void configureMessageBroker(MessageBrokerRegistry registry) {
32         //客户端需要把消息发送到/message/xxx地址
33         registry.setApplicationDestinationPrefixes("/message");
34         //服务端广播消息的路径前缀，客户端需要相应订阅/topic/yyy这个地址的消息
35         registry.enableSimpleBroker("/topic");
36     }
37
38     @Override
39     public void configureClientInboundChannel(ChannelRegistration registration) {
40         registration.interceptors(myChannelInterceptor);
41     }
42
43 }
```

从上面代码可以看出，这里设置了好几个地址，简单解释如下：

- 首先注册了一个名为**/stomp-websocket**的端点，也就是STOMP客户端连接的地址。
- 此外，定义了服务端处理WebSocket消息的前缀是**/message**，这个地址用于客户端向服务端发送消息（比如客户端向**/message/hello**这个地址发送消息，那么服务端通过**@MessageMapping("/hello")**这个注解来接收并处理消息）

- 最后，定义了一个简单消息代理，也就是服务端广播消息的路径前缀（比如客户端监听/**topic/greeting**这个地址，那么服务端就可以通过**@SendTo("/topic/greeting")**这个注解向客户端发送STOMP消息）。

需要注意的是，上面代码中还添加了一个名为**MyChannelInterceptor**的拦截器，目的是为了在客户端断开连接后打印一下日志。相关代码如下：

```
1 package cn.zifangsky.stompwebsocket.interceptor.websocket;
2
3 import org.apache.commons.lang3.StringUtils;
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6 import org.springframework.messaging.Message;
7 import org.springframework.messaging.MessageChannel;
8 import org.springframework.messaging.simp.stomp.StompCommand;
9 import org.springframework.messaging.simp.stomp.StompHeaderAccessor;
10 import org.springframework.messaging.support.ChannelInterceptor;
11 import org.springframework.stereotype.Component;
12
13 import java.security.Principal;
14 import java.text.MessageFormat;
15
16 /**
17  * 自定义{@link org.springframework.messaging.support.ChannelInterceptor}，实现断开连接的
18  *
19  * @author zifangsky
20  * @date 2018/10/10
21  * @since 1.0.0
22  */
23 @Component
24 public class MyChannelInterceptor implements ChannelInterceptor{
25     private final Logger logger = LoggerFactory.getLogger(getClass());
26
27     @Override
28     public void afterSendCompletion(Message<?> message, MessageChannel channel, boolean
29         StompHeaderAccessor accessor = StompHeaderAccessor.wrap(message);
30         StompCommand command = accessor.getCommand();
31
32         //用户已经断开连接
33         if(StompCommand.DISCONNECT.equals(command)){
34             String user = "";
35             Principal principal = accessor.getUser();
36             if(principal != null && StringUtils.isNotBlank(principal.getName())){
37                 user = principal.getName();
38             }else{
39                 user = accessor.getSessionId();
40             }
41
42             logger.debug(MessageFormat.format("用户{0}的WebSocket连接已经断开", user));
43         }
44     }
45 }
46 }
```

(2) 使用@MessageMapping和@SendTo注解处理消息：

@MessageMapping 注解用于监听指定路径的客户端消息，而 @SendTo 注解则用于将服务端的消息发送给监听了该路径的客户端。

```
1 package cn.zifangsky.stompwebsocket.controller;
2
3 import cn.zifangsky.stompwebsocket.model.websocket.Greeting;
4 import cn.zifangsky.stompwebsocket.model.websocket.HelloMessage;
5 import org.springframework.messaging.handler.annotation.MessageMapping;
6 import org.springframework.messaging.handler.annotation.SendTo;
7 import org.springframework.stereotype.Controller;
8
9 /**
10  * Greeting
11  * @author zifangsky
12  * @date 2018/9/30
13  * @since 1.0.0
14  */
15 @Controller
16 public class GreetingController {
17
18     @MessageMapping("/hello")
19     @SendTo("/topic/greeting")
20     public HelloMessage greeting(Greeting greeting) {
21         return new HelloMessage("Hello," + greeting.getName() + "!");
22     }
23 }
```

(3) 示例页面：

```
1 <html xmlns:th="http://www.thymeleaf.org">
2 <head>
3     <meta content="text/html; charset=UTF-8"/>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
5     <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
6     <meta name="viewport" content="width=device-width, initial-scale=1"/>
7     <title>WebSocket Examples: Reverse</title>
8     <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"><
9     <script src="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.1.4/sockjs.min.js"><
10    <script src="https://cdnjs.cloudflare.com/ajax/libs/stomp.js/2.3.3/stomp.min.js"><
11    <script th:src="@{/layui/layui.js}"></script>
12    <link th:href="@{/layui/css/layui.css}" rel="stylesheet">
13    <style type="text/css">
14        #connect-container {
15            margin: 0 auto;
16            width: 400px;
17        }
18
19        #connect-container div {
20            padding: 5px;
21            margin: 0 7px 10px 0;
22        }
23
24        .layui-btn {
25            display: inline-block;
26        }
27    </style>
28    <script type="text/javascript">
```

```
29 var stompClient = null;
30
31 $(function () {
32     var target = $("#target");
33     if (window.location.protocol === 'http:') {
34         target.val('http://' + window.location.host + target.val());
35     } else {
36         target.val('https://' + window.location.host + target.val());
37     }
38 });
39
40 function setConnected(connected) {
41     var connect = $("#connect");
42     var disconnect = $("#disconnect");
43     var echo = $("#echo");
44
45     if (connected) {
46         connect.addClass("layui-btn-disabled");
47         disconnect.removeClass("layui-btn-disabled");
48         echo.removeClass("layui-btn-disabled");
49     } else {
50         connect.removeClass("layui-btn-disabled");
51         disconnect.addClass("layui-btn-disabled");
52         echo.addClass("layui-btn-disabled");
53     }
54
55     connect.attr("disabled", connected);
56     disconnect.attr("disabled", !connected);
57     echo.attr("disabled", !connected);
58 }
59
60 //连接
61 function connect() {
62     var target = $("#target").val();
63
64     var ws = new SockJS(target);
65     stompClient = Stomp.over(ws);
66
67     stompClient.connect({}, function () {
68         setConnected(true);
69         log('Info: STOMP connection opened.');
```



```
87     }
88     setConnected(false);
89     log('Info: STOMP connection closed.');
```

```
90 }
91
92 //向服务端发送姓名
93 function sendName() {
94     if (stompClient != null) {
95         var username = $("#username").val();
96         log('Sent: ' + username);
97         stompClient.send("/message/hello", {}, JSON.stringify({'name': username}));
98     } else {
99         alert('STOMP connection not established, please connect.');
```

启动项目后访问页面，效果如下：

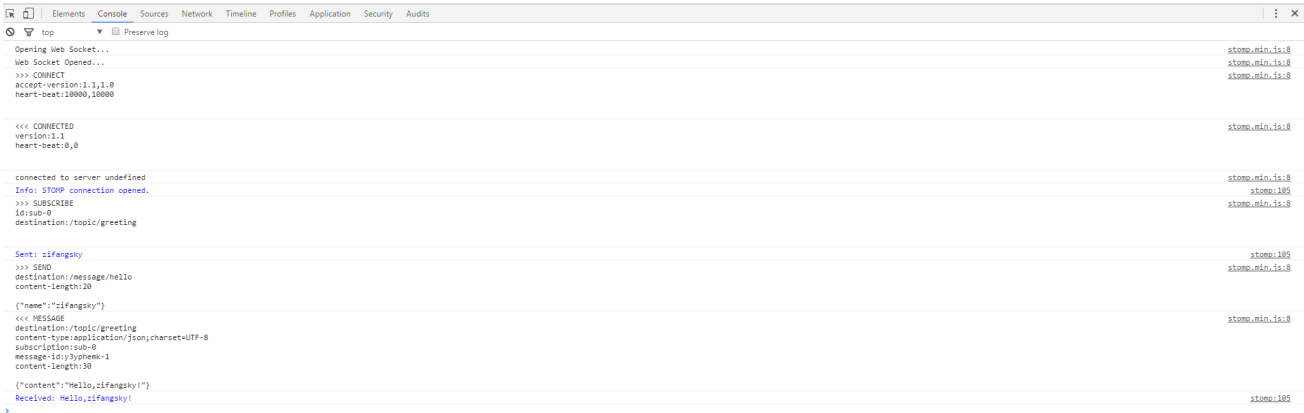
STOMP Message With SockJS

http://127.0.0.1:7001/stomp-websocket

ConnectDisconnect

zifangsky

Say hello



<https://www.zifangsky.cn>

参考：

- spring-boot-sample-websocket-tomcat (<https://github.com/spring-projects/spring-boot/tree/master/spring-boot-samples/spring-boot-sample-websocket-tomcat>)
- messaging-stomp-websocket (<https://spring.io/guides/gs/messaging-stomp-websocket/>)
- WebSocket 的故事（二）—— Spring 中如何利用 STOMP 快速构建 WebSocket 广播式消息模式 (<https://juejin.im/post/5b7071ade51d45665816f8c0>)

 赞 (13)

#Spring Boot (<https://www.zifangsky.cn/tag/spring-boot>) #WebSocket (<https://www.zifangsky.cn/tag/websocket>)

©版权声明：原创作品，允许转载，转载时请务必以超链接形式标明文章 [原始出处](https://www.zifangsky.cn/1355.html) (<https://www.zifangsky.cn/1355.html>)、作者信息和本声明。否则将追究法律责任。

转载请注明来源：Spring Boot中使用WebSocket总结（一）：几种实现方式详解 (<https://www.zifangsky.cn/1355.html>) - zifangsky的个人博客 (<https://www.zifangsky.cn>)

上一篇 (<https://www.zifangsky.cn/1347.html>)

下一篇 (<https://www.zifangsky.cn/1359.html>)

你可能也喜欢：.....

- 如何在普通Spring项目中手动实现类似Spring Boot中有条件生成Bean? (<https://www.zifangsky.cn/1416.html>)
- 基于Spring的项目中Redis存储对象使用Jackson序列化方式 (<https://www.zifangsky.cn/1366.html>)
- Spring Boot中使用WebSocket总结（三）：使用消息队列实现分布式WebSocket (<https://www.zifangsky.cn/1364.html>)
- Spring Boot中使用WebSocket总结（二）：向指定用户发送WebSocket消息并处理对方不在线的情况 (<https://www.zifangsky.cn/1359.html>)
- 在Spring Boot中使用Spring Data Redis实现基于“发布/订阅”模型的消息队列 (<https://www.zifangsky.cn/1347.html>)

本文共 4 个回复



jht385 2019/09/01 15:26

老哥啊，写demo不要参杂那么多无关的东西啊，什么工具类，连数据库，认证授权 都来了，完全都没用上啊。然后页面里打印用的是，`console.debug` 我还是这次去查的，之前都是用`console.log`，居然连chrome都不支持 `console.debug`，你用它打印干嘛啊。本来想看看demo，删得我好心累啊

(<https://www.zifangsky.cn/1355.html?replytocom=6084#respond>)

回复



admin (<http://www.zifangsky.cn>) 博主 2019/09/01 15:45

@ jht385 源码中的部分代码是后面示例需要用的，所以并不是无关紧要的东西。再说了，给出示例只是用来参考的，又不是让你直接抄代码，你都在写websocket了还不会基本的增删改查吗？这就有点说不过去了吧。最后，你觉得心累你可以关了不看，我既没有从你那儿赚一分钱，又没有逼着你看，既然觉得累又何必自讨苦吃呢！

(<https://www.zifangsky.cn/1355.html?replytocom=6085#respond>)

回复

发表评论 取消 (/1355.html#respond)

来都来了，何不留个足迹~



昵称

*



邮箱

*



网址



验证码 *

发表评论



不悔 2018/12/17 14:47

可以给个sql脚本么，老哥

(<https://www.zifangsky.cn/1355.html?replytocom=3460#respond>)

回复



admin (<http://www.zifangsky.cn>) **博主** 2018/12/18 09:38

@ 不悔 这几篇文章介绍的内容跟数据库没太大关系，你用你自己的SQL测试就行，或者不连数据库模拟登录吧。

(<https://www.zifangsky.cn/1355.html?replytocom=3463#respond>)

回复


友情链接

iceH's Blog (<http://www.secice.cn>) 业余草 (<http://www.xttblog.com/>) 仲威的博客 (<https://www.blogme.top>)

俄罗斯方块 (<https://sale.hacker.bid/>) 六阿哥博客 (<https://blog.6ag.cn>)

太空船博客 (<https://www.boatsky.com/>) 掘金专栏 (<https://juejin.im/user/5819f202d203090055df470e>)

朴实的追梦者 (<http://www.zmzblog.com>) 青木（简书） (<https://www.jianshu.com/u/cedd62e70951>)

Copyright © 2018 zifangsky的个人博客 (<https://www.zifangsky.cn>) | 

(https://www.cnzz.com/stat/website.php?web_id=1256860929) | Theme By Specs (<http://9iphp.com>)