<http://blog.csdn.net/linlzk/article/details/51086545>

版权声明：本文为博主原创文章，未经博主允许不得转载。

折腾了将近一天，终于搭建好websocket服务，中间遇到不少的坑，现在记录一下，也供他们参考少走弯路。

开发环境：<**[spring](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "_blank)**.version>4.2.5.RELEASE</spring.version>     Tomcat8   jdk8  maven

spring搭建websocket有两种方式：

1、利用javaee-api以注解的形式搭建，见上一篇博文[《spring+tomcat8实现websocket 》](http://blog.csdn.net/linlzk/article/details/51079291" \t "_blank)

2、利用spring-websocket包搭建，本篇介绍的内容。

注意：这种方式必须注释掉这个，不然会冲突

<!-- <dependency>  
            <groupId>javax</groupId>  
            <artifactId>javaee-api</artifactId>  
            <version>7.0</version>  
        </dependency> -->

需要用到的spring包：

<dependency>    
           <groupId>org.springframework</groupId>    
           <artifactId>spring-websocket</artifactId>    
           <version>${spring.version}</version>    
        </dependency>    
        <dependency>    
           <groupId>org.springframework</groupId>    
           <artifactId>spring-messaging</artifactId>    
           <version>${spring.version}</version>    
        </dependency>   
其他辅助包：

<dependency>  
            <groupId>com.fasterxml.jackson.core</groupId>  
            <artifactId>jackson-core</artifactId>  
            <version>${jackson-version}</version>  
        </dependency>  
        <dependency>  
            <groupId>com.fasterxml.jackson.core</groupId>  
            <artifactId>jackson-databind</artifactId>  
            <version>${jackson-version}</version>  
        </dependency>  
        <dependency>  
            <groupId>com.fasterxml.jackson.core</groupId>  
            <artifactId>jackson-annotations</artifactId>  
            <version>${jackson-version}</version>  
        </dependency>

[**Java**](http://lib.csdn.net/base/java)代码：两个类  WebSocketConfig 和 SystemWebSocketHandler

**[java]** [view plain](http://blog.csdn.net/linlzk/article/details/51086545) [copy](http://blog.csdn.net/linlzk/article/details/51086545)

[print?](http://blog.csdn.net/linlzk/article/details/51086545)

1. @Configuration
2. @EnableWebMvc
3. @EnableWebSocket
4. **public** **class** WebSocketConfig **extends** WebMvcConfigurerAdapter **implements** WebSocketConfigurer{
6. @Override
7. **public** **void** registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
8. registry.addHandler(systemWebSocketHandler(),"/webSocketServer");
9. registry.addHandler(systemWebSocketHandler(),"/webSocketServer/sockjs").setAllowedOrigins("\*").withSockJS();
10. //registry.addHandler(systemWebSocketHandler(),"/webSocketServer").addInterceptors(new WebSocketHandshakeInterceptor());
11. //registry.addHandler(systemWebSocketHandler(), "/sockjs/webSocketServer").addInterceptors(new WebSocketHandshakeInterceptor()).withSockJS();
12. //registry.addHandler(systemWebSocketHandler(), "/webSocketServer/sockjs").withSockJS();
13. /\*registry.addHandler(systemWebSocketHandler(),"/ws").addInterceptors(new WebSocketHandshakeInterceptor());
14. registry.addHandler(systemWebSocketHandler(), "/ws/sockjs").addInterceptors(new WebSocketHandshakeInterceptor())
15. .withSockJS();\*/
16. }
18. @Bean
19. **public** WebSocketHandler systemWebSocketHandler(){
20. **return** **new** SystemWebSocketHandler();
21. }

**[java]** [view plain](http://blog.csdn.net/linlzk/article/details/51086545) [copy](http://blog.csdn.net/linlzk/article/details/51086545)

[print?](http://blog.csdn.net/linlzk/article/details/51086545)

1. **import** java.io.IOException;
2. **import** java.util.ArrayList;
3. **import** java.util.Date;
5. **import** org.slf4j.Logger;
6. **import** org.slf4j.LoggerFactory;
7. **import** org.springframework.messaging.Message;
8. **import** org.springframework.web.socket.CloseStatus;
9. **import** org.springframework.web.socket.TextMessage;
10. **import** org.springframework.web.socket.WebSocketHandler;
11. **import** org.springframework.web.socket.WebSocketMessage;
12. **import** org.springframework.web.socket.WebSocketSession;
14. **import** com.mspj.frame.core.web.controller.BaseWebControllerImpl;
16. **public** **class** SystemWebSocketHandler **implements** WebSocketHandler {
18. **private** Logger log = LoggerFactory.getLogger(SystemWebSocketHandler.**class**);
20. **private** **static** **final** ArrayList<WebSocketSession> users = **new** ArrayList<WebSocketSession>();;

23. @Override
24. **public** **void** afterConnectionEstablished(WebSocketSession session) **throws** Exception {
25. System.out.println("ConnectionEstablished");
26. log.debug("ConnectionEstablished");
27. users.add(session);
29. session.sendMessage(**new** TextMessage("connect"));
30. session.sendMessage(**new** TextMessage("new\_msg"));
32. }
34. @Override
35. **public** **void** handleMessage(WebSocketSession session, WebSocketMessage<?> message) **throws** Exception {
36. System.out.println("handleMessage" + message.toString());
37. log.debug("handleMessage" + message.toString());
38. //sendMessageToUsers();
39. session.sendMessage(**new** TextMessage(**new** Date() + ""));
40. }
42. @Override
43. **public** **void** handleTransportError(WebSocketSession session, Throwable exception) **throws** Exception {
44. **if**(session.isOpen()){
45. session.close();
46. }
47. users.remove(session);
49. log.debug("handleTransportError" + exception.getMessage());
50. }
52. @Override
53. **public** **void** afterConnectionClosed(WebSocketSession session, CloseStatus closeStatus) **throws** Exception {
54. users.remove(session);
55. log.debug("afterConnectionClosed" + closeStatus.getReason());
57. }
59. @Override
60. **public** **boolean** supportsPartialMessages() {
61. **return** **false**;
62. }
64. /\*\*
65. \* 给所有在线用户发送消息
66. \*
67. \* @param message
68. \*/
69. **public** **void** sendMessageToUsers(TextMessage message) {
70. **for** (WebSocketSession user : users) {
71. **try** {
72. **if** (user.isOpen()) {
73. user.sendMessage(message);
74. }
75. } **catch** (IOException e) {
76. e.printStackTrace();
77. }
78. }
79. }
81. }

前端调式页面代码

**[html]** [view plain](http://blog.csdn.net/linlzk/article/details/51086545) [copy](http://blog.csdn.net/linlzk/article/details/51086545)

[print?](http://blog.csdn.net/linlzk/article/details/51086545)

1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"**>**
2. **<html>**
3. **<head>**
4. **<meta** http-equiv="Content-Type" content="text/html; charset=UTF-8"**>**
5. **<title>**WebSocket/SockJS Echo Sample (Adapted from Tomcat's echo sample)**</title>**
6. **<style** type="text/css"**>**
7. #connect-container {
8. float: left;
9. width: 400px
10. }
12. #connect-container div {
13. padding: 5px;
14. }
16. #console-container {
17. float: left;
18. margin-left: 15px;
19. width: 400px;
20. }
22. #console {
23. border: 1px solid #CCCCCC;
24. border-right-color: #999999;
25. border-bottom-color: #999999;
26. height: 170px;
27. overflow-y: scroll;
28. padding: 5px;
29. width: 100%;
30. }
32. #console p {
33. padding: 0;
34. margin: 0;
35. }
36. **</style>**
38. **<script** src="http://cdn.sockjs.org/sockjs-0.3.min.js"**></script>**
40. **<script** type="text/javascript"**>**
41. var ws = null;
42. var url = null;
43. var transports = [];
45. function setConnected(connected) {
46. document.getElementById('connect').disabled = connected;
47. document.getElementById('disconnect').disabled = !connected;
48. document.getElementById('echo').disabled = !connected;
49. }
51. function connect() {
52. if (!url) {
53. alert('Select whether to use W3C WebSocket or SockJS');
54. return;
55. }

58. //ws = new WebSocket('ws://192.168.10.107:8080/mspjapi/webSocketServer');/\* (url.indexOf('sockjs') != -1) ?
59. //new SockJS(url, undefined, {protocols\_whitelist: transports}) :  \*/
60. ws = new SockJS("http://192.168.10.107:8080/mspjapi/webSocketServer/sockjs");
61. //console.log("http://192.168.10.107:8080/mspjapi/webSocketServer/sockjs");
63. ws.onopen = function () {
64. setConnected(true);
65. log('Info: connection opened.');
66. };
68. ws.onmessage = function (event) {
69. log('Received: ' + event.data);
70. };
72. ws.onclose = function (event) {
73. setConnected(false);
74. log('Info: connection closed.');
75. log(event);
76. };
77. }
79. function disconnect() {
80. if (ws != null) {
81. ws.close();
82. ws = null;
83. }
84. setConnected(false);
85. }
87. function echo() {
88. if (ws != null) {
89. var message = document.getElementById('message').value;
90. log('Sent: ' + message);
91. ws.send(message);
92. } else {
93. alert('connection not established, please connect.');
94. }
95. }
97. function updateUrl(urlPath) {
98. if (urlPath.indexOf('sockjs') != -1) {
99. url = urlPath;
100. document.getElementById('sockJsTransportSelect').style.visibility = 'visible';
101. }
102. else {
103. if (window.location.protocol == 'http:') {
104. url = 'ws://' + window.location.host + urlPath;
105. } else {
106. url = 'wss://' + window.location.host + urlPath;
107. }
108. document.getElementById('sockJsTransportSelect').style.visibility = 'hidden';
109. }
110. }
112. function updateTransport(transport) {
113. transports = (transport == 'all') ?  [] : [transport];
114. }
116. function log(message) {
117. var console = document.getElementById('console');
118. var p = document.createElement('p');
119. p.style.wordWrap = 'break-word';
120. p.appendChild(document.createTextNode(message));
121. console.appendChild(p);
122. while (console.childNodes.length **>** 25) {
123. console.removeChild(console.firstChild);
124. }
125. console.scrollTop = console.scrollHeight;
126. }
127. **</script>**
128. **</head>**
129. **<body>**
130. **<noscript><h2** style="color: #ff0000"**>**Seems your browser doesn't support Javascript! Websockets
131. rely on Javascript being enabled. Please enable
132. Javascript and reload this page!**</h2></noscript>**
133. **<div>**
134. **<div** id="connect-container"**>**
135. **<input** id="radio1" type="radio" name="group1" onclick="updateUrl('/mspjapi');"**>**
136. **<label** for="radio1"**>**W3C WebSocket**</label>**
137. **<br>**
138. **<input** id="radio2" type="radio" name="group1" onclick="updateUrl('/spring-websocket-test/sockjs/echo');"**>**
139. **<label** for="radio2"**>**SockJS**</label>**
140. **<div** id="sockJsTransportSelect" style="visibility:hidden;"**>**
141. **<span>**SockJS transport:**</span>**
142. **<select** onchange="updateTransport(this.value)"**>**
143. **<option** value="all"**>**all**</option>**
144. **<option** value="websocket"**>**websocket**</option>**
145. **<option** value="xhr-polling"**>**xhr-polling**</option>**
146. **<option** value="jsonp-polling"**>**jsonp-polling**</option>**
147. **<option** value="xhr-streaming"**>**xhr-streaming**</option>**
148. **<option** value="iframe-eventsource"**>**iframe-eventsource**</option>**
149. **<option** value="iframe-htmlfile"**>**iframe-htmlfile**</option>**
150. **</select>**
151. **</div>**
152. **<div>**
153. **<button** id="connect" onclick="connect();"**>**Connect**</button>**
154. **<button** id="disconnect" disabled="disabled" onclick="disconnect();"**>**Disconnect**</button>**
155. **</div>**
156. **<div>**
157. **<textarea** id="message" style="width: 350px"**>**Here is a message!**</textarea>**
158. **</div>**
159. **<div>**
160. **<button** id="echo" onclick="echo();" disabled="disabled"**>**Echo message**</button>**
161. **</div>**
162. **</div>**
163. **<div** id="console-container"**>**
164. **<div** id="console"**></div>**
165. **</div>**
166. **</div>**
168. **<a** href="echoendpoint.jsp"**>**echoendpoint test**</a>**
169. **<a** href="websocket2.jsp"**>**echoendpoint test**</a>**
171. **</body>**
172. **</html>**

最后说下几个坑点：

1、html页面访问限制的问题的：403 Forbidden

**[java]** [view plain](http://blog.csdn.net/linlzk/article/details/51086545) [copy](http://blog.csdn.net/linlzk/article/details/51086545)

[print?](http://blog.csdn.net/linlzk/article/details/51086545)

1. registry.addHandler(systemWebSocketHandler(),"/webSocketServer/sockjs").setAllowedOrigins("\*").withSockJS();

2、web.xml配置需要升级成3.1

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_1.xsd"  
    version="3.1">

3、web.xml里面所有过滤器需要加上

<async-supported>true</async-supported>

如：

<servlet>  
        <servlet-name>SpringMVC</servlet-name>  
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
        <init-param>  
            <param-name>contextConfigLocation</param-name>  
            <param-value>classpath:spring-mvc.xml</param-value>  
        </init-param>  
        <load-on-startup>1</load-on-startup>  
        <async-supported>true</async-supported>  
    </servlet>