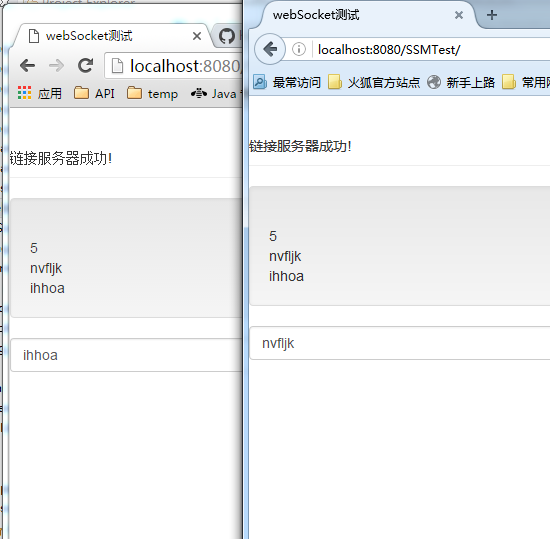
[**SpringMVC+HTML5+websocket即时聊天Demo**](http://hejiawangjava.iteye.com/blog/2289437)

**博客分类：**

* [Spring](http://hejiawangjava.iteye.com/category/343727)

[spring](http://www.iteye.com/blogs/tag/spring)[html](http://www.iteye.com/blogs/tag/html)[websocket](http://www.iteye.com/blogs/tag/websocket)[即时聊天](http://www.iteye.com/blogs/tag/%E5%8D%B3%E6%97%B6%E8%81%8A%E5%A4%A9)

websocket的通信形式今天刚学，好处就不说了，网上有很多，下面把写的即时通信的demo记录下来，运行出来像下面一样：

  
 一、spring必须4.0以上，maven的pom文件如下：

**Xml代码  [收藏代码](javascript:void())**

1. **<dependency>**
2. **<groupId>**org.springframework**</groupId>**
3. **<artifactId>**spring-orm**</artifactId>**
4. **<version>**4.2.4.RELEASE**</version>**
5. **</dependency>**
6. **<dependency>**
7. **<groupId>**org.springframework**</groupId>**
8. **<artifactId>**spring-webmvc**</artifactId>**
9. **<version>**4.2.4.RELEASE**</version>**
10. **</dependency>**
12. **<dependency>**
13. **<groupId>**org.springframework**</groupId>**
14. **<artifactId>**spring-websocket**</artifactId>**
15. **<version>**4.2.4.RELEASE**</version>**
16. **</dependency>**
17. **<dependency>**
18. **<groupId>**org.springframework**</groupId>**
19. **<artifactId>**spring-messaging**</artifactId>**
20. **<version>**4.2.4.RELEASE**</version>**
21. **</dependency>**

当然了，还有servlet等等了。。。那些就自己查了

二、spring中，关于websocket的配置：

**Xml代码  [收藏代码](javascript:void())**

1. **<bean** id="websocket" class="com.wang.test.websocket.demo.WebSocketHander" **/>**
3. **<websocket:handlers>**
4. **<websocket:mapping** path="/echo" handler="websocket" **/>**
5. **<websocket:handshake-interceptors>**
6. **<bean** class="com.wang.test.websocket.demo.HandshakeInterceptor" **/>**
7. **</websocket:handshake-interceptors>**
8. **</websocket:handlers>**

 不要忘记引入xsd文件，

**Xml代码  [收藏代码](javascript:void())**

1. http://www.springframework.org/schema/websocket
2. http://www.springframework.org/schema/websocket/spring-websocket-4.0.xsd

 整个关于websocket的配置文件如下：

**Xml代码  [收藏代码](javascript:void())**

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:websocket="http://www.springframework.org/schema/websocket"
5. xsi:schemaLocation="
6. http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
8. http://www.springframework.org/schema/websocket
9. http://www.springframework.org/schema/websocket/spring-websocket-4.0.xsd"**>**
11. **<bean** id="websocket" class="com.wang.test.websocket.demo.WebSocketHander" **/>**
13. **<websocket:handlers>**
14. **<websocket:mapping** path="/echo" handler="websocket" **/>**
15. **<websocket:handshake-interceptors>**
16. **<bean** class="com.wang.test.websocket.demo.HandshakeInterceptor" **/>**
17. **</websocket:handshake-interceptors>**
18. **</websocket:handlers>**
20. **</beans>**

 三、Interceptor类

**Java代码  [收藏代码](javascript:void())**

1. **package** com.wang.test.websocket.demo;
3. **import** org.springframework.http.server.ServerHttpRequest;
4. **import** org.springframework.http.server.ServerHttpResponse;
5. **import** org.springframework.http.server.ServletServerHttpRequest;
6. **import** org.springframework.web.socket.WebSocketHandler;
8. **import** javax.servlet.http.HttpServletRequest;
9. **import** java.util.Map;
11. **public** **class** HandshakeInterceptor **implements** org.springframework.web.socket.server.HandshakeInterceptor{
12. // 初次握手访问前
13. @Override
14. **public** **boolean** beforeHandshake(ServerHttpRequest request, ServerHttpResponse serverHttpResponse,
15. WebSocketHandler webSocketHandler, Map<String, Object> map) **throws** Exception {
16. **if** (request **instanceof** ServletServerHttpRequest) {
17. HttpServletRequest servletRequest = ((ServletServerHttpRequest) request).getServletRequest();
18. String userName = "xq";
19. // 使用userName区分WebSocketHandler，以便定向发送消息
20. // String userName = (String)
21. // session.getAttribute("WEBSOCKET\_USERNAME");
22. map.put("WEBSOCKET\_USERNAME", userName);
23. servletRequest.getSession().setAttribute("WEBSOCKET\_USERNAME", userName);
24. }
25. **return** **true**;
26. }
28. // 初次握手访问后
29. @Override
30. **public** **void** afterHandshake(ServerHttpRequest serverHttpRequest, ServerHttpResponse serverHttpResponse,
31. WebSocketHandler webSocketHandler, Exception e) {
33. }
34. }

 四、WebSocketHander类

**Java代码  [收藏代码](javascript:void())**

1. **package** com.wang.test.websocket.demo;
3. **import** org.apache.log4j.Logger;
4. **import** org.springframework.web.socket.\*;
6. **import** java.io.IOException;
7. **import** java.util.ArrayList;
9. **public** **class** WebSocketHander **implements** WebSocketHandler {
11. **private** **static** **final** Logger logger = Logger.getLogger(WebSocketHander.**class**);
13. **private** **static** **final** ArrayList<WebSocketSession> users = **new** ArrayList<>();
15. // 初次链接成功执行
16. @Override
17. **public** **void** afterConnectionEstablished(WebSocketSession session) **throws** Exception {
18. logger.debug("链接成功......");
19. users.add(session);
20. String userName = (String) session.getAttributes().get("WEBSOCKET\_USERNAME");
21. **if** (userName != **null**) {
22. // 查询未读消息
23. **int** count = 5;
24. session.sendMessage(**new** TextMessage(count + ""));
25. }
26. }
28. // 接受消息处理消息
29. @Override
30. **public** **void** handleMessage(WebSocketSession webSocketSession, WebSocketMessage<?> webSocketMessage)
31. **throws** Exception {
32. sendMessageToUsers(**new** TextMessage(webSocketMessage.getPayload() + ""));
33. }
35. @Override
36. **public** **void** handleTransportError(WebSocketSession webSocketSession, Throwable throwable) **throws** Exception {
37. **if** (webSocketSession.isOpen()) {
38. webSocketSession.close();
39. }
40. logger.debug("链接出错，关闭链接......");
41. users.remove(webSocketSession);
42. }
44. @Override
45. **public** **void** afterConnectionClosed(WebSocketSession webSocketSession, CloseStatus closeStatus) **throws** Exception {
46. logger.debug("链接关闭......" + closeStatus.toString());
47. users.remove(webSocketSession);
48. }
50. @Override
51. **public** **boolean** supportsPartialMessages() {
52. **return** **false**;
53. }
55. /\*\*
56. \* 给所有在线用户发送消息
57. \*
58. \* @param message
59. \*/
60. **public** **void** sendMessageToUsers(TextMessage message) {
61. **for** (WebSocketSession user : users) {
62. **try** {
63. **if** (user.isOpen()) {
64. user.sendMessage(message);
65. }
66. } **catch** (IOException e) {
67. e.printStackTrace();
68. }
69. }
70. }
72. /\*\*
73. \* 给某个用户发送消息
74. \*
75. \* @param userName
76. \* @param message
77. \*/
78. **public** **void** sendMessageToUser(String userName, TextMessage message) {
79. **for** (WebSocketSession user : users) {
80. **if** (user.getAttributes().get("WEBSOCKET\_USERNAME").equals(userName)) {
81. **try** {
82. **if** (user.isOpen()) {
83. user.sendMessage(message);
84. }
85. } **catch** (IOException e) {
86. e.printStackTrace();
87. }
88. **break**;
89. }
90. }
91. }
92. }

 五、页面显示文件：

**Java代码  [收藏代码](javascript:void())**

1. <%@ page language="java" **import**="java.util.\*" pageEncoding="utf-8"%>
2. <%
3. String path = request.getContextPath();
4. String basePath = request.getScheme() + "://" + request.getServerName() + ":" + request.getServerPort()
5. + path + "/";
6. %>
8. <!DOCTYPE HTML>
9. <html>
10. <head>
11. <meta charset="UTF-8">
12. <script src="http://cdn.sockjs.org/sockjs-0.3.min.js"></script>
13. <!-- 新 Bootstrap 核心 CSS 文件 -->
14. <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css/bootstrap.min.css">
15. <!-- 可选的Bootstrap主题文件（一般不用引入） -->
16. <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css/bootstrap-theme.min.css">
17. <!-- jQuery文件。务必在bootstrap.min.js 之前引入 -->
18. <script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></script>
19. <!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
20. <script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
21. <title>webSocket测试</title>
22. <script type="text/javascript">
23. $(function() {
24. var websocket;
25. **if** ('WebSocket' in window) {
26. alert("WebSocket");
27. var url = 'ws://' + window.location.host + '/SSMTest/echo';
28. websocket = **new** WebSocket(url);
29. //websocket = new WebSocket("ws://127.0.0.1:8080/SSMTest/echo");
30. } **else** **if** ('MozWebSocket' in window) {
31. alert("MozWebSocket");
32. websocket = **new** MozWebSocket("ws://echo");
33. } **else** {
34. alert("SockJS");
35. websocket = **new** SockJS("http://127.0.0.1:8080/SSMTest/sockjs/echo");
36. }
37. websocket.onopen = function(evnt) {
38. $("#tou").html("链接服务器成功!")
39. };
40. websocket.onmessage = function(evnt) {
41. $("#msg").html($("#msg").html() + "<br/>" + evnt.data);
42. };
43. websocket.onerror = function(evnt) {
44. };
45. websocket.onclose = function(evnt) {
46. $("#tou").html("与服务器断开了链接!")
47. }
48. $('#send').bind('click', function() {
49. send();
50. });
51. function send() {
52. **if** (websocket != **null**) {
53. var message = document.getElementById('message').value;
54. websocket.send(message);
55. } **else** {
56. alert('未与服务器链接.');
57. }
58. }
59. });
60. </script>
62. </head>
63. <body>
65. <div **class**="page-header" id="tou">webSocket及时聊天Demo程序</div>
66. <div **class**="well" id="msg"></div>
67. <div **class**="col-lg">
68. <div **class**="input-group">
69. <input type="text" **class**="form-control" placeholder="发送信息..."
70. id="message"> <span **class**="input-group-btn">
71. <button **class**="btn btn-default" type="button" id="send">发送</button>
72. </span>
73. </div>
74. <!-- /input-group -->
75. </div>
76. <!-- /.col-lg-6 -->
77. </div>
78. <!-- /.row -->
79. </body>
80. </html>

 六、参考：<http://blog.csdn.net/mr_zhuqiang/article/details/46618197>  把这里面配置形式改了

 七、代码地址：<https://github.com/hejiawang/SSMTest>