**[[置顶] WebSocket spring示例demo （已使用sockJs库）](http://blog.csdn.net/u011991249/article/details/51472020)**

标签： [java](http://www.csdn.net/tag/java)[maven](http://www.csdn.net/tag/maven)[博客](http://www.csdn.net/tag/%e5%8d%9a%e5%ae%a2)[websocket](http://www.csdn.net/tag/websocket)[spring](http://www.csdn.net/tag/spring)

2016-05-21 19:56 5198人阅读 [评论](http://blog.csdn.net/u011991249/article/details/51472020#comments)(15) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/u011991249/article/details/51472020#report)

 分类：

java基础（22）

版权声明：本文为博主原创文章，未经博主允许不得转载。

1、简介

作为下一代的Web标准，[**HTML5**](http://lib.csdn.net/base/html5)拥有许多引人注目的新特性，如 Canvas、本地存储、多媒体编程接口、WebSocket等等。这其中有“Web的 TCP”之称的 WebSocket格外吸引开发人员的注意。WebSocket的出现使得浏览器提供对 Socket的支持成为可能，从而在浏览器和服务器之间提供了一个基于TCP连接的双向通道。Web开发人员可以非常方便地使用WebSocket构建实时 web 应用，开发人员的手中从此又多了一柄神兵利器。

            Web 应用的信息交互过程通常是客户端通过浏览器发出一个请求，服务器端接收和审核完请求后进行处理并返回结果给客户端，然后客户端浏览器将信息呈现出来，这种机制对于信息变化不是特别频繁的应用尚能相安无事，但是对于那些实时要求比较高的应用来说，比如说在线游戏、在线证券、设备监控、新闻在线播报、RSS订阅推送等等，当客户端浏览器准备呈现这些信息的时候，这些信息在服务器端可能已经过时了。所以保持客户端和服务器端的信息同步是实时 Web应用的关键要素，对 Web开发人员来说也是一个难题。在 WebSocket规范出来之前，开发人员想实现这些实时的Web应用，不得不采用一些折衷的方案，其中最常用的就是轮询(Polling)和 Comet 技术，而Comet技术实际上是轮询技术的改进，又可细分为两种实现方式，一种是长轮询机制，一种称为流技术。

[**html5**](http://lib.csdn.net/base/html5) WebSocket 设计出来的目的就是要取代轮询和 Comet技术，使客户端浏览器具备像 C/S[**架构**](http://lib.csdn.net/base/architecture)下桌面系统的实时通讯能力。浏览器通过[**JavaScript**](http://lib.csdn.net/base/javascript)向服务器发出建立 WebSocket连接的请求，连接建立以后，客户端和服务器端就可以通过 TCP连接直接交换数据。因为WebSocket连接本质上就是一个TCP连接，所以在数据传输的稳定性和数据传输量的大小方面，和轮询以及Comet技术比较，具有很大的性能优势。

但是鉴于web socket 对浏览器要求比较高，为了解决这个问题，推出了sockJS，SockJS是一个[Java](http://www.07net01.com/tags-Java-0.html)Script库，提供跨[浏览器](http://www.07net01.com/tags-%E6%B5%8F%E8%A7%88%E5%99%A8-0.html)[javascript](http://www.07net01.com/tags-javascript-0.html)的API，创建了一个低延迟、全双工的浏览器和web[服务器](http://www.07net01.com/tags-%E6%9C%8D%E5%8A%A1%E5%99%A8-0.html)之间通信通道。

2、相关环境要求

Spring4.0.6(要选择4.0+),tomcat7.0.55版本，JDK1.7。

3、具体代码

（以下代码亲测可用！）

（1）web.xml:

**[html]** [view plain](http://blog.csdn.net/u011991249/article/details/51472020) [copy](http://blog.csdn.net/u011991249/article/details/51472020)

[print?](http://blog.csdn.net/u011991249/article/details/51472020)

1. **<?xmlversion**xmlversion="1.0"encoding="UTF-8"**?>**
2. **<web-app** version="3.1"
3. xmlns="http://xmlns.jcp.org/xml/ns/javaee"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"**>**
6. **<description>**用来测试WebSocket基础上推送的功能**</description>**
7. **<distributable/>**
8. **<filter>**
9. **<filter-name>**encodingFilter**</filter-name>**
10. **<filter-class>**org.springframework.web.filter.CharacterEncodingFilter**</filter-class>**
11. **<async-supported>**true**</async-supported>**
12. **<init-param>**
13. **<param-name>**encoding**</param-name>**
14. **<param-value>**UTF-8**</param-value>**
15. **</init-param>**
16. **<init-param>**
17. **<param-name>**forceEncoding**</param-name>**
18. **<param-value>**true**</param-value>**
19. **</init-param>**
20. **</filter>**
21. **<filter-mapping>**
22. **<filter-name>**encodingFilter**</filter-name>**
23. **<url-pattern>**/\***</url-pattern>**
24. **</filter-mapping>**
25. **<listener>**
26. **<listener-class>**org.springframework.web.context.ContextLoaderListener**</listener-class>**
27. **</listener>**
28. <!-- Spring 刷新Introspector防止内存泄露 -->
29. **<listener>**
30. **<listener-class>**org.springframework.web.util.IntrospectorCleanupListener**</listener-class>**
31. **</listener>**
32. **<servlet>**
33. **<servlet-name>**dispatcher**</servlet-name>**
34. **<servlet-class>**org.springframework.web.servlet.DispatcherServlet**</servlet-class>**
35. **<init-param>**
36. **<param-name>**contextConfigLocation**</param-name>**
37. **<param-value>**classpath:dispatcher-servlet.xml**</param-value>**
38. **</init-param>**
39. **<load-on-startup>**1**</load-on-startup>**
40. **<async-supported>**true**</async-supported>**
41. **</servlet>**
42. **<servlet-mapping>**
43. **<servlet-name>**dispatcher**</servlet-name>**
44. **<url-pattern>**/**</url-pattern>**
45. **</servlet-mapping>**
46. **<session-config>**
47. **<session-timeout>**
48. 30
49. **</session-timeout>**
50. **</session-config>**
51. **<welcome-file-list>**
52. **<welcome-file>**testSocket.jsp**</welcome-file>**
53. **</welcome-file-list>**
54. **</web-app>**

(2) dispatcher-servlet.xml

**[html]** [view plain](http://blog.csdn.net/u011991249/article/details/51472020) [copy](http://blog.csdn.net/u011991249/article/details/51472020)

[print?](http://blog.csdn.net/u011991249/article/details/51472020)

1. **<?xmlversion**xmlversion="1.0"encoding="UTF-8"**?>**
2. **<**beans:beansxmlns**beans:beansxmlns**="http://www.springframework.org/schema/mvc"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:beans="http://www.springframework.org/schema/beans"
5. xmlns:context="http://www.springframework.org/schema/context"
6. xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd
7. http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8. http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"**>**
9. **<annotation-driven/>**
10. <!-- 自动扫描的包名 -->
11. **<**context:component-scanbase-package**context:component-scanbase-package**="zyy.sockjs.config"**></context:component-scan>**
12. **</beans:beans>**

(3)HandshakeInterceptor.java

**[java]** [view plain](http://blog.csdn.net/u011991249/article/details/51472020) [copy](http://blog.csdn.net/u011991249/article/details/51472020)

[print?](http://blog.csdn.net/u011991249/article/details/51472020)

1. **package** zyy.sockjs.config;
2. **import** java.util.Map;
3. **import** org.springframework.http.server.ServerHttpRequest;
4. **import** org.springframework.http.server.ServerHttpResponse;
5. **import** org.springframework.stereotype.Component;
6. **import** org.springframework.web.socket.WebSocketHandler;
7. **import** org.springframework.web.socket.server.support.HttpSessionHandshakeInterceptor;
9. @Component
10. **public** **class** HandshakeInterceptor **extends** HttpSessionHandshakeInterceptor {
11. @Override
12. **public** **boolean** beforeHandshake(ServerHttpRequest request,
13. ServerHttpResponse response, WebSocketHandler wsHandler,
14. Map<String, Object> attributes) **throws** Exception {
15. System.out.println("Before Handshake");
16. **return** **super**.beforeHandshake(request,response,wsHandler,attributes);
17. }
19. @Override
20. **public** **void** afterHandshake(ServerHttpRequest request,
21. ServerHttpResponse response, WebSocketHandler wsHandler,
22. Exception ex) {
23. System.out.println("After Handshake");
24. **super**.afterHandshake(request,response,wsHandler,ex);
25. }
26. }

(4)InfoSocketEndPoint.[**Java**](http://lib.csdn.net/base/java)

**[java]** [view plain](http://blog.csdn.net/u011991249/article/details/51472020) [copy](http://blog.csdn.net/u011991249/article/details/51472020)

[print?](http://blog.csdn.net/u011991249/article/details/51472020)

1. **package** zyy.sockjs.config;
2. **import** org.springframework.stereotype.Component;
3. **import** org.springframework.web.socket.TextMessage;
4. **import** org.springframework.web.socket.WebSocketSession;
5. **import** org.springframework.web.socket.handler.TextWebSocketHandler;
7. @Component
8. **public** **class** InfoSocketEndPoint **extends** TextWebSocketHandler {
9. **public** InfoSocketEndPoint() {
10. }
12. @Override
13. **protected** **void** handleTextMessage(WebSocketSession session,
14. TextMessage message) **throws** Exception {
15. **super**.handleTextMessage(session, message);
16. TextMessage returnMessage = **new** TextMessage(message.getPayload()
17. + " received at server");
18. session.sendMessage(returnMessage);
19. }
20. }

(5)SystemWebSocketHandler.java

**[java]** [view plain](http://blog.csdn.net/u011991249/article/details/51472020) [copy](http://blog.csdn.net/u011991249/article/details/51472020)

[print?](http://blog.csdn.net/u011991249/article/details/51472020)

1. **package** zyy.sockjs.config;
2. **import** org.springframework.stereotype.Component;
3. **import** org.springframework.web.socket.CloseStatus;
4. **import** org.springframework.web.socket.TextMessage;
5. **import** org.springframework.web.socket.WebSocketHandler;
6. **import** org.springframework.web.socket.WebSocketMessage;
7. **import** org.springframework.web.socket.WebSocketSession;
8. /\*\*
9. \*
10. \* @author dayu
11. \*/
13. @Component
14. **public** **class** SystemWebSocketHandler **implements** WebSocketHandler {
15. @Override
16. **public** **void** afterConnectionEstablished(WebSocketSession session) **throws** Exception {
17. System.out.println("connect to the websocket success......");
18. session.sendMessage(**new** TextMessage("Server:connected OK!"));
19. }
21. @Override
22. **public** **void** handleMessage(WebSocketSession wss, WebSocketMessage<?> wsm) **throws** Exception {
23. TextMessage returnMessage = **new** TextMessage(wsm.getPayload()
24. + " received at server");
25. wss.sendMessage(returnMessage);
26. }
28. @Override
29. **public** **void** handleTransportError(WebSocketSession wss, Throwable thrwbl) **throws** Exception {
30. **if**(wss.isOpen()){
31. wss.close();
32. }
33. System.out.println("websocket connection closed......");
34. }
36. @Override
37. **public** **void** afterConnectionClosed(WebSocketSession wss, CloseStatus cs) **throws** Exception {
38. System.out.println("websocket connection closed......");
39. }
41. @Override
42. **public** **boolean** supportsPartialMessages() {
43. **return** **false**;
44. }
45. }

(6)WebSocketConfig.java

**[java]** [view plain](http://blog.csdn.net/u011991249/article/details/51472020) [copy](http://blog.csdn.net/u011991249/article/details/51472020)

[print?](http://blog.csdn.net/u011991249/article/details/51472020)

1. **package** zyy.sockjs.config;
2. **import** org.springframework.context.annotation.Bean;
3. **import** org.springframework.context.annotation.Configuration;
4. **import** org.springframework.web.servlet.config.annotation.EnableWebMvc;
5. **import** org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
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;
11. @Configuration
12. @EnableWebMvc
13. @EnableWebSocket
14. **public** **class** WebSocketConfig **extends** WebMvcConfigurerAdapter **implements**
15. WebSocketConfigurer {
16. **public** WebSocketConfig() {
17. }
19. @Override
20. **public** **void** registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
21. registry.addHandler(systemWebSocketHandler(), "/websck").addInterceptors(**new** HandshakeInterceptor());
22. System.out.println("registed!");
23. registry.addHandler(systemWebSocketHandler(), "/sockjs/websck/info").addInterceptors(**new** HandshakeInterceptor())
24. .withSockJS();
25. }
27. @Bean
28. **public** WebSocketHandler systemWebSocketHandler() {
29. **return** **new** SystemWebSocketHandler();
30. }
31. }

(7)testSocket.jsp

**[java]** [view plain](http://blog.csdn.net/u011991249/article/details/51472020) [copy](http://blog.csdn.net/u011991249/article/details/51472020)

[print?](http://blog.csdn.net/u011991249/article/details/51472020)

1. <%@ page language="java"contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
2. <!DOCTYPE htmlPUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
3. <html>
4. <head>
5. <meta http-equiv="Content-Type"content="text/html; charset=ISO-8859-1">
6. <title>WebSocket/SockJS Echo Sample (Adapted from Tomcat's echo sample)</title>
7. <style type="text/css">
8. #connect-container {
9. **float**: left;
10. width: 400px
11. }
12. #connect-container div {
13. padding: 5px;
14. }
15. #console-container {
16. **float**: left;
17. margin-left: 15px;
18. width: 400px;
19. }
20. #console {
21. border:1px solid #CCCCCC;
22. border-right-color:#33333333;
23. border-bottom-color:#999999;
24. height: 170px;
25. overflow-y: scroll;
26. padding: 5px;
27. width: 100%;
28. }
29. #console p {
30. padding: 0;
31. margin: 0;
32. }
33. </style>
34. <script src="http://cdn.sockjs.org/sockjs-0.3.min.js"></script>
35. <script type="text/javascript">
36. var ws = **null**;
37. var url = **null**;
38. var transports = [];
39. function setConnected(connected) {
40. document.getElementById('connect').disabled = connected;
41. document.getElementById('disconnect').disabled = !connected;
42. document.getElementById('echo').disabled = !connected;
43. }
44. function connect() {
45. **if** (!url) {
46. log('Select whether to use W3C WebSocket or SockJS');
47. **return**;
48. }
49. //ws = (url.indexOf('sockjs') != -1) ?new SockJS(url, undefined, {protocols\_whitelist: transports}) : new WebSocket(url);
50. **if** ('WebSocket'in window) {
51. ws= **new** WebSocket("ws://localhost:8080/SpringSocketJs/websck");
52. }**else** {
53. ws = **new** SockJS("http://localhost:8080/SpringSocketJs/sockjs/websck/info");
54. }
55. //websocket = new SockJS("http://localhost:8084/SpringWebSocketPush/sockjs/websck");
56. ws.onopen = function () {
57. setConnected(**true**);
58. //log('Info: connection opened.');
59. };
60. ws.onmessage = function (event) {
61. log('Received: ' + event.data);
62. };
63. ws.onclose = function (event) {
64. setConnected(**false**);
65. log('Info: connection closed.');
66. log(event);
67. };
68. }
69. function disconnect() {
70. **if** (ws != **null**) {
71. ws.close();
72. ws = **null**;
73. }
74. setConnected(**false**);
75. }
76. function echo() {
77. **if** (ws != **null**) {
78. var message = document.getElementById('message').value;
79. log('Sent: ' + message);
80. ws.send(message);
81. } **else** {
82. alert('connection not established, please connect.');
83. }
84. }
85. function updateUrl(urlPath) {
86. **if** (urlPath.indexOf('sockjs') != -1) {
87. url = urlPath;
88. document.getElementById('sockJsTransportSelect').style.visibility ='visible';
89. }
90. **else** {
91. **if** (window.location.protocol =='http:') {
92. url = 'ws://' + window.location.host + urlPath;
93. } **else** {
94. url = 'wss://' + window.location.host + urlPath;
95. }
96. document.getElementById('sockJsTransportSelect').style.visibility ='hidden';
97. }
98. }
99. function updateTransport(transport) {
100. transports = (transport == 'all') ?  [] : [transport];
101. }
102. function log(message) {
103. var console = document.getElementById('console');
104. var p = document.createElement('p');
105. p.style.wordWrap = 'break-word';
106. p.appendChild(document.createTextNode(message));
107. console.appendChild(p);
108. **while** (console.childNodes.length > 25) {
109. console.removeChild(console.firstChild);
110. }
111. console.scrollTop = console.scrollHeight;
112. }
113. </script>
114. </head>
115. <body>
116. <noscript><h2 style="color:#ff0000">Seems your browser doesn't supportJavascript!Websockets
117. rely on Javascript being enabled. Please enable
118. Javascript and reload **this** page!</h2></noscript>
119. <div>
120. <div id="connect-container">
121. <input id="radio1"type="radio"name="group1"onclick="updateUrl('/SpringSocketJs/websocket');">
122. <label **for**="radio1">W3C WebSocket</label>
123. <br>
124. <input id="radio2"type="radio"name="group1"onclick="updateUrl('/SpringSocketJs/sockjs/websocket');">
125. <label **for**="radio2">SockJS</label>
126. <div id="sockJsTransportSelect" style="visibility:hidden;">
127. <span>SockJS transport:</span>
128. <select onchange="updateTransport(this.value)">
129. <option value="all">all</option>
130. <option value="websocket">websocket</option>
131. <option value="xhr-polling">xhr-polling</option>
132. <option value="jsonp-polling">jsonp-polling</option>
133. <option value="xhr-streaming">xhr-streaming</option>
134. <option value="iframe-eventsource">iframe-eventsource</option>
135. <option value="iframe-htmlfile">iframe-htmlfile</option>
136. </select>
137. </div>
138. <div>
139. <button id="connect"onclick="connect();">Connect</button>
140. <button id="disconnect"disabled="disabled"onclick="disconnect();">Disconnect</button>
141. </div>
142. <div>
143. <textarea id="message"style="width:350px">Here is a message!</textarea>
144. </div>
145. <div>
146. <button id="echo"onclick="echo();"disabled="disabled">Echo message</button>
147. </div>
148. </div>
149. <div id="console-container">
150. <div id="console"></div>
151. </div>
152. </div>
153. </body>
154. </html>