# Spring之WebSocket网页聊天以及服务器推送



2015年6月12日 飞翔的拖鞋up Springframework 26 评论



**Websocket简介** 摘要自百度百科

**1. WebSocket protocol 是HTML5一种新的协议。它实现了浏览器与服务器全双工通信(full-duplex)。**

**2. 轮询是在特定的的时间间隔（如每1秒），由浏览器对服务器发出HTTP request，然后由服务器返回最新的数据给客服端的浏览器。这种传统的HTTP request 的模式带来很明显的缺点 – 浏览器需要不断的向服务器发出请求，然而HTTP request 的header是非常长的，里面包含的有用数据可能只是一个很小的值，这样会占用很多的带宽。**

**3. 比较新的技术去做轮询的效果是Comet – 用了AJAX。但这种技术虽然可达到全双工通信，但依然需要发出请求**

**4. 在 WebSocket API，浏览器和服务器只需要要做一个握手的动作，然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送**

**5. 在此WebSocket 协议中，为我们实现即时服务带来了两大好处：**

### 5.1. Header

**互相沟通的Header是很小的-大概只有 2 Bytes**

### 5.2. Server Push

**浏览器支持情况**

| | |
|---|---|
| Chrome | 4+ |
| Firefox | 4+ |
| Internet Explorer | 10+ |
| Opera | 10+ |
| Safari | 5+ |

**服务器支持**

| | |
|---|---|
| jetty | 7.0.1+ |
| tomcat | 7.0.27+ |
| Nginx | 1.3.13+ |
| resin | 4+ |

**API**

```
var ws = new WebSocket("ws://echo.websocket.org");

ws.onopen = function(){ws.send("Test!"); };
//当有消息时，会自动调用此方法
ws.onmessage = function(evt){console.log(evt.data);ws.close();};

ws.onclose = function(evt){console.log("WebSocketClosed!");};

ws.onerror = function(evt){console.log("WebSocketError!");};
```

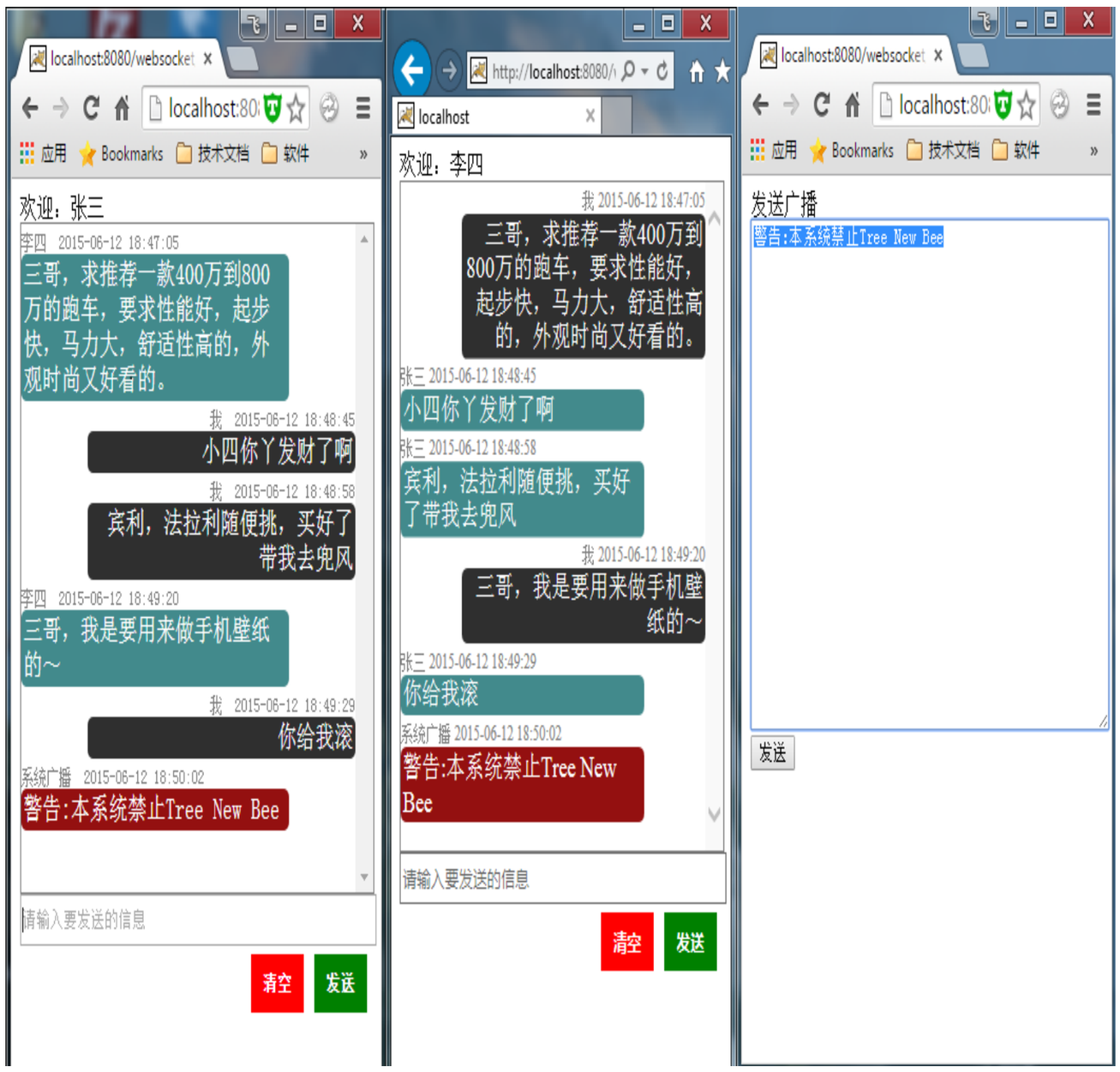
**Demo简介**

模拟了两个用户的对话，张三和李四，然后还有发送一个广播，即张三和李四都是可以接收到的，登录的时候分别选择张三和李四即可

近期Demo

- Java串口通信工具类 2017年6月20日
- JS加减乘除操作 2016年5月9日
- JS生成随机数Random.js 2016年5月9日
- [Calendar.js]模拟Java的Calendar日期类 2016年5月9日
- 零散知识积累 2016年4月7日
- JS跨域实例 2016年3月7日
- JS回调函数和匿名函数小例子 2016年3月7日
- POI自定义表头（合并列）导出Excel工具类 2015年9月17日
- Java监控文件夹或文件的变动 2015年7月6日
- Spring之WebSocket网页聊天以及服务器推送 2015年6月12日
- Flying-Saucer使用HTML或者FTL(Freemarker模板)生成PDF 2015年6月4日
- Java基于注解和反射导入导出Excel（Bean转Excel，Excel转Bean） 2015年6月2日
- SpringMVC利用AOP实现自定义注解记录日志 2015年2月27日
- Java导入导出MySQL 2015年2月26日
- Java发送邮件工具类 2015年2月25日

### Demo效果




### Maven依赖

```
<dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-annotations</artifactId>
            <version>2.3.0</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-core</artifactId>
            <version>2.3.1</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.3.3</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-messaging</artifactId>
            <version>4.0.5.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-websocket</artifactId>
            <version>4.0.5.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>4.0.5.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.3.1</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
```

### Web.xml，spring-mvc.xml，User.java请查看附件

### WebSocket相关的类

### WebSocketConfig，配置WebSocket的处理器（MyWebSocketHandler）和拦截器（HandShake）

```
package org.xdemo.example.websocket.websocket;

import javax.annotation.Resource;

import org.springframework.stereotype.Component;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import org.springframework.web.socket.config.annotation.EnableWebSocket;
import org.springframework.web.socket.config.annotation.WebSocketConfigurer;
import org.springframework.web.socket.config.annotation.WebSocketHandlerRegistry;

/**
 * WebScoket配置处理器
 * @author Goofy
 * @Date 2015年6月11日 下午1:15:09
 */
@Component
@EnableWebSocket
```

```
public class WebSocketConfig extends WebMvcConfigurerAdapter implements WebSocketConfig

    @Resource
    MyWebSocketHandler handler;

    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(handler, "/ws").addInterceptors(new HandShake());

        registry.addHandler(handler, "/ws/sockjs").addInterceptors(new HandShake()).wit
    }

}
```

## MyWebSocketHandler

```
package org.xdemo.example.websocket.websocket;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Map.Entry;

import org.springframework.stereotype.Component;
import org.springframework.web.socket.CloseStatus;
import org.springframework.web.socket.TextMessage;
import org.springframework.web.socket.WebSocketHandler;
import org.springframework.web.socket.WebSocketMessage;
import org.springframework.web.socket.WebSocketSession;
import org.xdemo.example.websocket.entity.Message;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

/**
 * Socket处理器
 *
 * @author Goofy
 * @Date 2015年6月11日 下午1:19:50
 */
@Component
public class MyWebSocketHandler implements WebSocketHandler {
    public static final Map<Long, WebSocketSession> userSocketSessionMap;

    static {
        userSocketSessionMap = new HashMap<Long, WebSocketSession>();
    }

    /**
     * 建立连接后
     */
    public void afterConnectionEstablished(WebSocketSession session)
            throws Exception {
        Long uid = (Long) session.getAttributes().get("uid");
        if (userSocketSessionMap.get(uid) == null) {
            userSocketSessionMap.put(uid, session);
        }
    }

    /**
     * 消息处理，在客户端通过Websocket API发送的消息会经过这里，然后进行相应的处理
     */
    public void handleMessage(WebSocketSession session, WebSocketMessage<?> message) th
            if(message.getPayloadLength()==0)return;
            Message msg=new Gson().fromJson(message.getPayload().toString(),Message.cla
            msg.setDate(new Date());
            sendMessageToUser(msg.getTo(), new TextMessage(new GsonBuilder().setDateFor
    }

    /**
     * 消息传输错误处理
     */
    public void handleTransportError(WebSocketSession session,
            Throwable exception) throws Exception {
        if (session.isOpen()) {
            session.close();
        }
        Iterator<Entry<Long, WebSocketSession>> it = userSocketSessionMap
                .entrySet().iterator();
        // 移除Socket会话
        while (it.hasNext()) {
            Entry<Long, WebSocketSession> entry = it.next();
            if (entry.getValue().getId().equals(session.getId())) {
                userSocketSessionMap.remove(entry.getKey());
                System.out.println("Socket会话已经移除:用户ID" + entry.getKey());
                break;
            }
        }
    }

    /**
     * 关闭连接后
     */
    public void afterConnectionClosed(WebSocketSession session,
            CloseStatus closeStatus) throws Exception {
        System.out.println("Websocket:" + session.getId() + "已经关闭");
        Iterator<Entry<Long, WebSocketSession>> it = userSocketSessionMap
                .entrySet().iterator();
        // 移除Socket会话
        while (it.hasNext()) {
            Entry<Long, WebSocketSession> entry = it.next();
            if (entry.getValue().getId().equals(session.getId())) {
                userSocketSessionMap.remove(entry.getKey());
                System.out.println("Socket会话已经移除:用户ID" + entry.getKey());
                break;
            }
        }
    }

    public boolean supportsPartialMessages() {
        return false;
    }

    /**
     * 给所有在线用户发送消息
     *
     * @param message
     * @throws IOException
     */
    public void broadcast(final TextMessage message) throws IOException {
        Iterator<Entry<Long, WebSocketSession>> it = userSocketSessionMap
                .entrySet().iterator();

        // 多线程群发
```

```
        while (it.hasNext()) {

            final Entry<Long, WebSocketSession> entry = it.next();

            if (entry.getValue().isOpen()) {
                // entry.getValue().sendMessage(message);
                new Thread(new Runnable() {

                    public void run() {
                        try {
                            if (entry.getValue().isOpen()) {
                                entry.getValue().sendMessage(message);
                            }
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }

                }).start();
            }

        }
    }

    /**
     * 给某个用户发送消息
     *
     * @param userName
     * @param message
     * @throws IOException
     */
    public void sendMessageToUser(Long uid, TextMessage message)
            throws IOException {
        WebSocketSession session = userSocketSessionMap.get(uid);
        if (session != null && session.isOpen()) {
            session.sendMessage(message);
        }
    }

}
```

HandShake(每次建立连接都会进行握手)

```
package org.xdemo.example.websocket.websocket;

import java.util.Map;

import javax.servlet.http.HttpSession;

import org.springframework.http.server.ServerHttpRequest;
import org.springframework.http.server.ServerHttpResponse;
import org.springframework.http.server.ServletServerHttpRequest;
import org.springframework.web.socket.WebSocketHandler;
import org.springframework.web.socket.server.HandshakeInterceptor;


/**
 * Socket建立连接（握手）和断开
 *
 * @author Goofy
 * @Date 2015年6月11日 下午2:23:09
 */
public class HandShake implements HandshakeInterceptor {

    public boolean beforeHandshake(ServerHttpRequest request, ServerHttpResponse respon
        System.out.println("Websocket:用户[ID:" + ((ServletServerHttpRequest) request).g
        if (request instanceof ServletServerHttpRequest) {
            ServletServerHttpRequest servletRequest = (ServletServerHttpRequest) reques
            HttpSession session = servletRequest.getServletRequest().getSession(false);
            // 标记用户
            Long uid = (Long) session.getAttribute("uid");
            if(uid!=null){
                attributes.put("uid", uid);
            }else{
                return false;
            }
        }
        return true;
    }

    public void afterHandshake(ServerHttpRequest request, ServerHttpResponse response,
    }

}
```

## 一个Controller

```
package org.xdemo.example.websocket.controller;

import java.io.IOException;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

import javax.annotation.Resource;
import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.socket.TextMessage;
import org.xdemo.example.websocket.entity.Message;
import org.xdemo.example.websocket.entity.User;
import org.xdemo.example.websocket.websocket.MyWebSocketHandler;

import com.google.gson.GsonBuilder;

@Controller
@RequestMapping("/msg")
public class MsgController {

    @Resource
    MyWebSocketHandler handler;

    Map<Long, User> users = new HashMap<Long, User>();

        //模拟一些数据
    @ModelAttribute
    public void setReqAndRes() {
        User u1 = new User();
```

```
        u1.setId(1L);
        u1.setName("张三");
        users.put(u1.getId(), u1);

        User u2 = new User();
        u2.setId(2L);
        u2.setName("李四");
        users.put(u2.getId(), u2);

    }

    //用户登录
    @RequestMapping(value="login",method=RequestMethod.POST)
    public ModelAndView doLogin(User user,HttpServletRequest request){
        request.getSession().setAttribute("uid", user.getId());
        request.getSession().setAttribute("name", users.get(user.getId()).getName());
        return new ModelAndView("redirect:talk");
    }

    //跳转到交谈聊天页面
    @RequestMapping(value="talk",method=RequestMethod.GET)
    public ModelAndView talk(){
        return new ModelAndView("talk");
    }
    //跳转到发布广播页面
    @RequestMapping(value="broadcast",method=RequestMethod.GET)
    public ModelAndView broadcast(){
        return new ModelAndView("broadcast");
    }

    //发布系统广播（群发）
    @ResponseBody
    @RequestMapping(value="broadcast",method=RequestMethod.POST)
    public void broadcast(String text) throws IOException{
        Message msg=new Message();
        msg.setDate(new Date());
        msg.setFrom(-1L);
        msg.setFromName("系统广播");
        msg.setTo(0L);
        msg.setText(text);
        handler.broadcast(new TextMessage(new GsonBuilder().setDateFormat("yyyy-MM-dd H
    }

}
```

一个消息的封装的类

```
package org.xdemo.example.websocket.entity;

import java.util.Date;
/**
 * 消息类
 * @author Goofy
 * @Date 2015年6月12日 下午7:32:39
 */
public class Message {

    //发送者
    public Long from;
    //发送者名称
    public String fromName;
    //接收者
    public Long to;
    //发送的文本
    public String text;
    //发送日期
    public Date date;

    public Long getFrom() {
        return from;
    }

    public void setFrom(Long from) {
        this.from = from;
    }

    public Long getTo() {
        return to;
    }

    public void setTo(Long to) {
        this.to = to;
    }

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

    public String getFromName() {
        return fromName;
    }

    public void setFromName(String fromName) {
        this.fromName = fromName;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

}
```

聊天页面

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath = request.getServerName() + ":"
+ request.getServerPort() + path + "/";
String basePath2 = request.getScheme() + "://"
+ request.getServerName() + ":" + request.getServerPort()
+ path + "/";
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title></title>
<script type="text/javascript" src="<%=basePath2%>resources/jquery.js"></script>
<style>
textarea {
height: 300px;
width: 100%;
resize: none;
outline: none;
}

input[type=button] {
float: right;
margin: 5px;
width: 50px;
height: 35px;
border: none;
color: white;
font-weight: bold;
outline: none;
}

.clear {
background: red;
}

.send {
background: green;
}

.clear:active {
background: yellow;
}

.send:active {
background: yellow;
}

.msg {
width: 100%;
height: 25px;
outline: none;
}

#content {
border: 1px solid gray;
width: 100%;
height: 400px;
overflow-y: scroll;
}

.from {
background-color: green;
width: 80%;
border-radius: 10px;
height: 30px;
line-height: 30px;
margin: 5px;
float: left;
color: white;
padding: 5px;
font-size: 22px;
}

.to {
background-color: gray;
width: 80%;
border-radius: 10px;
height: 30px;
line-height: 30px;
margin: 5px;
float: right;
color: white;
padding: 5px;
font-size: 22px;
}

.name {
color: gray;
font-size: 12px;
}

.tmsg_text {
color: white;
background-color: rgb(47, 47, 47);
font-size: 18px;
border-radius: 5px;
padding: 2px;
}

.fmsg_text {
color: white;
background-color: rgb(66, 138, 140);
font-size: 18px;
border-radius: 5px;
padding: 2px;
}

.sfmsg_text {
color: white;
background-color: rgb(148, 16, 16);
font-size: 18px;
border-radius: 5px;
padding: 2px;
}

.tmsg {
clear: both;
float: right;
width: 80%;
text-align: right;
}

.fmsg {
clear: both;
float: left;
width: 80%;
}
</style>
<script>
var path = '<%=basePath%>';
var uid=${uid eq null?-1:uid};
if(uid==-1){
location.href="<%=basePath2%>";
}
var from=uid;
var fromName='${name}';
```

```
var to=uid==1?2:1;

var websocket;
if ('WebSocket' in window) {
websocket = new WebSocket("ws://" + path + "/ws?uid="+uid);
} else if ('MozWebSocket' in window) {
websocket = new MozWebSocket("ws://" + path + "/ws"+uid);
} else {
websocket = new SockJS("http://" + path + "/ws/sockjs"+uid);
}
websocket.onopen = function(event) {
console.log("WebSocket:已连接");
console.log(event);
};
websocket.onmessage = function(event) {
var data=JSON.parse(event.data);
console.log("WebSocket:收到一条消息",data);
var textCss=data.from==-1?"sfmsg_text":"fmsg_text";
$("#content").append("<div><label>"+data.fromName+" "+data.date+"</label><div clas
scrollToBottom();
};
websocket.onerror = function(event) {
console.log("WebSocket:发生错误 ");
console.log(event);
};
websocket.onclose = function(event) {
console.log("WebSocket:已关闭");
console.log(event);
}
function sendMsg(){
var v=$("#msg").val();
if(v==""){
return;
}else{
var data={};
data["from"]=from;
data["fromName"]=fromName;
data["to"]=to;
data["text"]=v;
websocket.send(JSON.stringify(data));
$("#content").append("<div><label>我 "+new Date().Format("yyyy-MM-dd hh:mm:ss")+"</
scrollToBottom();
$("#msg").val("");
}
}

function scrollToBottom(){
var div = document.getElementById('content');
div.scrollTop = div.scrollHeight;
}

Date.prototype.Format = function (fmt) { //author: meizz
   var o = {
       "M+": this.getMonth() + 1, //月份
       "d+": this.getDate(), //日
       "h+": this.getHours(), //小时
       "m+": this.getMinutes(), //分
       "s+": this.getSeconds(), //秒
       "q+": Math.floor((this.getMonth() + 3) / 3), //季度
       "S": this.getMilliseconds() //毫秒
   };
   if (/(y+)/.test(fmt)) fmt = fmt.replace(RegExp.$1, (this.getFullYear() + "").substr(
   for (var k in o)
   if (new RegExp("(" + k + ")").test(fmt)) fmt = fmt.replace(RegExp.$1, (RegExp.$1.len
   return fmt;
}

function send(event){
var code;
if(window.event){
code = window.event.keyCode; // IE
}else{
code = e.which; // Firefox
}
if(code==13){
sendMsg();
}
}

function clearAll(){
$("#content").empty();
}
</script>
</head>
<body>
欢迎：${sessionScope.name }
<div id="content"></div>
<input type="text" placeholder="请输入要发送的信息" id="msg" onkeydown="send(event)">
<input type="button" value="发送" onclick="sendMsg()" >
<input type="button" value="清空" onclick="clearAll()">
</body>
</html>
```

发布广播的页面

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath= request.getScheme() + "://" + request.getServerName() + ":" + request.
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title></title>
<script type="text/javascript" src="<%=basePath%>resources/jquery.js"></script>
<script type="text/javascript">
var path='<%=basePath%>';
function broadcast(){
$.ajax({
url:path+'msg/broadcast',
type:"post",
data:{text:$("#msg").val()},
dataType:"json",
success:function(data){
alert("发送成功");
}
});
}
</script>
</head>
<body>
发送广播
<textarea style="width:100%;height:300px;" id="msg" ></textarea>
```

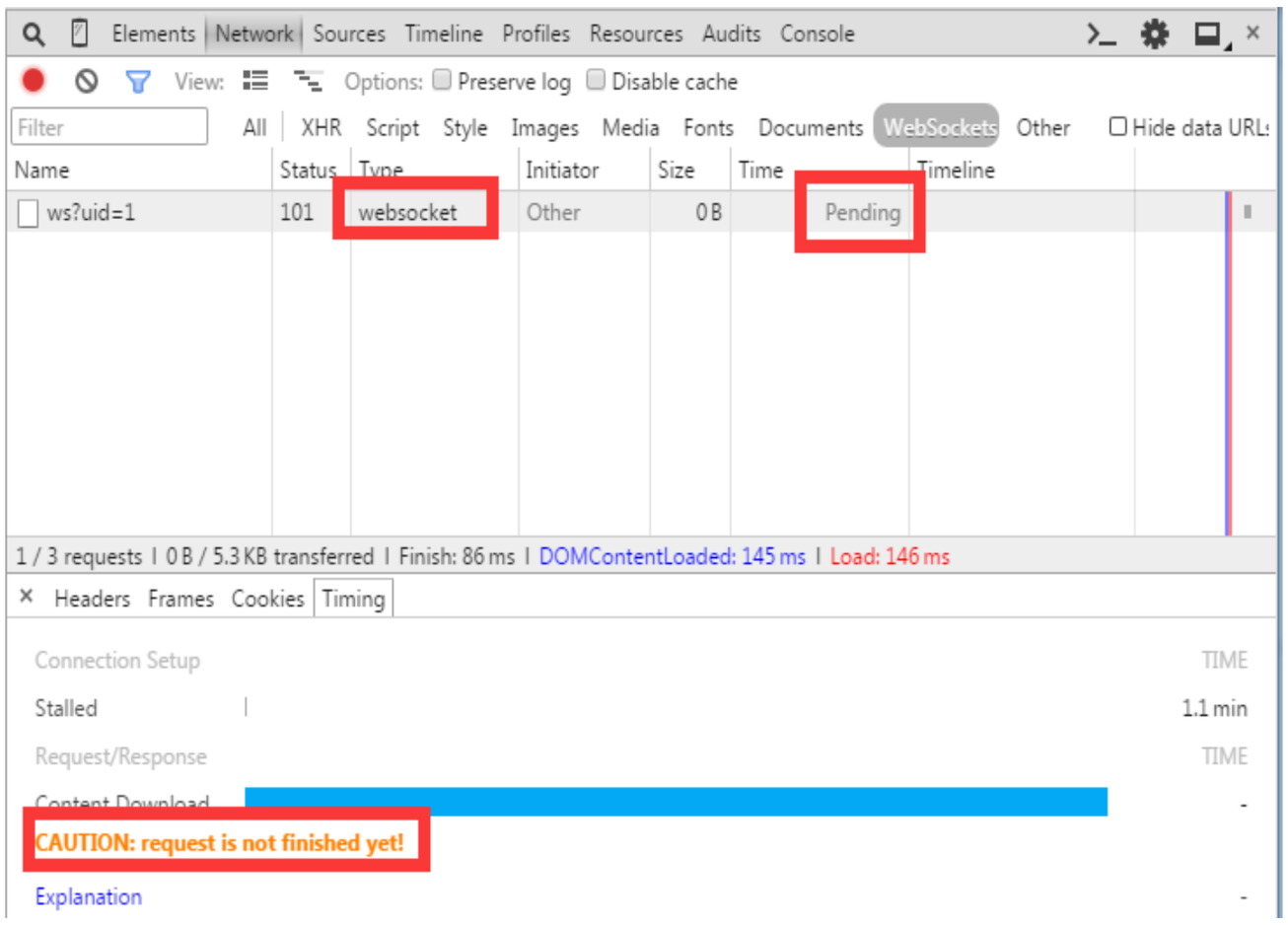
```
<input type="button" value="发送" onclick="broadcast()">
</body>
</html>
```

**Chrome的控制台网络信息**

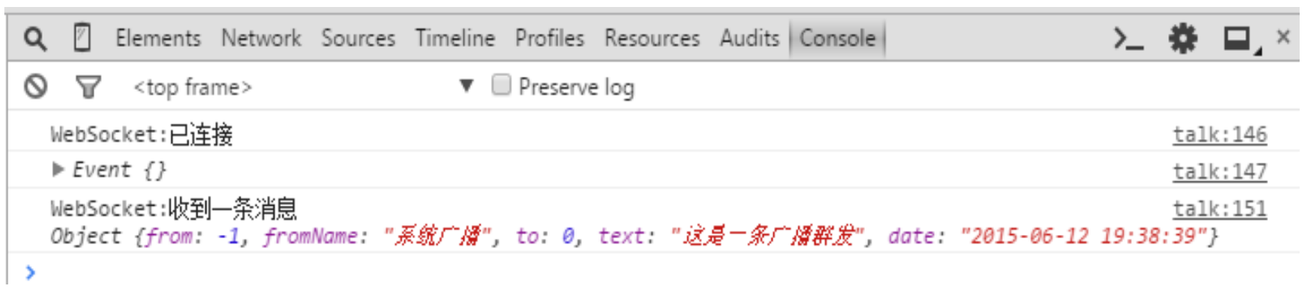
Type:websocket

Time:Pending

表示这是一个websocket请求，请求一直没有结束，可以通过此通道进行双向通信，即双工，实现了服务器推送的效果，也减少了网络流量。




**Chrome控制台信息**




**Demo下载**

> 百度网盘: http://pan.baidu.com/s/1dD0b15Z

**转载请注明来源: http://www.xdemo.org/spring-websocket-comet/**

## 相关文章

- SpringMVC利用AOP实现自定义注解记录日志
- SpringMVC数据绑定全面示例（复杂对象，数组等）
- 四步完成Spring国际化动态配置
- Spring不重启服务器重新加载上下文
- SpringMVC解决@ResponseBody乱码
- Spring获取已映射的URL列表

## 评论

**把酒临风** **2015年11月2日 在 下午2:20** 回复

点赞！

**张申曦** **2016年10月10日 在 下午3:25** 回复

你好，我遇到了一些问题，可以咨询一下吗？

**xxx** 2015年12月18日 在 上午9:35 回复

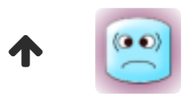
[赞]

**terryli** 2016年2月18日 在 下午4:54 回复

很不错的demo

**小寸** 2016年3月22日 在 上午9:25 回复

java.lang.ClassNotFoundException:
org.springframework.web.servlet.DispatcherServlet
配置到底是哪里出了问题



**飞翔的拖鞋up** 文章作者 回复

2016年3月25日 在 上午10:09

少包

**飞翔的拖鞋up** 文章作者 回复

2016年5月26日 在 下午12:02

缺少包

**779550095** 2016年5月27日 在 上午11:12 回复

楼主你好 我想找你做一套在线聊天系统 如果你看到我的评论 请联系我 q q 779 55 00 95

**RoidChao** 2016年6月2日 在 下午12:49 回复

能部署在jboss下吗

**Tony-启明** 2016年8月7日 在 下午11:18 回复

不错。。。

**Hfg** 2016年8月16日 在 下午12:09 回复

楼主，得益于你的实例我本地跑起来了，那怎么实现发送聊天图片呢？

**Hfg** 2016年8月16日 在 下午12:11 回复

楼主啊，得益于你的实例我在本地跑起来了，那怎么实现发送聊天图片呢？

**最爱xiaoyu77** 2016年9月4日 在 上午8:41 回复

用户管理需要数据库吗？

**小超哥** 2016年9月6日 在 上午10:34 回复

感谢楼主的公开

---

**666** 2016年9月6日 在 下午4:33 回复

这个网站怎么登陆

---

**灰白** 2016年9月6日 在 下午5:39 回复

handleMessage方法里面有错

---

**qiu** 2016年9月8日 在 下午5:29 回复

ie9好用吗，我试了一下不好用呀？

---

**张申曦** 2016年10月8日 在 下午4:03 回复

请问附件在哪里啊

---

**张申曦** 2016年10月10日 在 下午3:23 回复

楼主你好，我正在做这个demo，遇到了一些问题，可以咨询下尼玛，谢谢！

---

**nihao** 2016年10月19日 在 下午6:41 回复

public void afterConnectionEstablished(WebSocketSession session) throws Exception { Long uid = (Long) session.getAttributes().get(“uid”); if (userSocketSessionMap.get(uid) == null) { userSocketSessionMap.put(uid, session); } }这个方法中 Long uid = (Long) session.getAttributes().get(“uid”);没有获取到值 返回了null改成 final String uid = session.getId(); 就拿到id了

---

**quange** 2016年12月27日 在 上午10:51 回复

WebSocket connection to ‘ws://localhost:8080/ssm/ws?uid=1’ failed: Error during WebSocket handshake: Unexpected response code: 404

---

**12** 2017年1月16日 在 上午9:51 回复

好东西

---

**漩涡** 2017年2月7日 在 上午11:49 回复

一个不错的学习范例，诚意满满的，我在补充几个 还原时候的细节，让下载的人少走弯路该代码所需要的jar包如下，请自行下载1.gson

**http://mvnrepository.com/artifact/com.google.code.gson/gson/2.2.4**

2.Spring **http://repo.spring.io/release/org/springframework/spring/**

3.commons-logging-1.2

**http://mvnrepository.com/artifact/commons-logging/commons-logging**

---

**漩涡** 2017年2月7日 在 上午11:49 回复

一个不错的学习范例，诚意满满的，我在补充几个 还原时候的细节，让下载的人少走弯路该代码所需要的jar包如下，请自行下载1.gson

**http://mvnrepository.com/artifact/com.google.code.gson/gson/2.2.4**

2.Spring **http://repo.spring.io/release/org/springframework/spring/**

3.commons-logging-1.2

**http://mvnrepository.com/artifact/commons-logging/commons-logging**

---

**No** **2017年2月23日 在 下午7:55** 回复

毕业设计有这个模块，但是出了问题，希望加个qq 940594851

---

**葫芦娃** **2017年10月5日 在 下午10:02** 回复

作者，我遇到张三发的信息李四不会共享，李四发的张三也不会共享，还有只要发送一条左右两边都显示出来了，请指教呢

## 发表评论

电子邮件地址不会被公开。 必填项已用*标注

姓名 *

电子邮件 *

站点

验证码*

您可以使用这些HTML标签和属性： <a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime="">  <i> <q cite=""> <strike> 

发表评论