# **[websocket之二：WebSocket编程入门](https://www.cnblogs.com/duanxz/p/4822811.html)**

# **一、WebSocket客户端**

websocket允许通过JavaScript建立与远程服务器的连接，从而实现客户端与服务器间双向的通信。在websocket中有两个方法：　　  
　　　　1、send() 向远程服务器发送数据  
　　　　2、close() 关闭该websocket链接  
　　websocket同时还定义了几个监听函数　　　　  
　　　　1、onopen 当网络连接建立时触发该事件  
　　　　2、onerror 当网络发生错误时触发该事件  
　　　　3、onclose 当websocket被关闭时触发该事件  
　　　　4、onmessage 当websocket接收到服务器发来的消息的时触发的事件，也是通信中最重要的一个监听事件。msg.data  
　　websocket还定义了一个readyState属性，这个属性可以返回websocket所处的状态：  
　　　　1、CONNECTING(0) websocket正尝试与服务器建立连接  
　　　　2、OPEN(1) websocket与服务器已经建立连接  
　　　　3、CLOSING(2) websocket正在关闭与服务器的连接  
　　　　4、CLOSED(3) websocket已经关闭了与服务器的连接

　　websocket的url开头是ws，如果需要ssl加密可以使用wss，当我们调用websocket的构造方法构建一个websocket对象（new WebSocket(url)）的之后，就可以进行即时通信了。

[IMG_256](https://www.cnblogs.com/duanxz/p/javascript:void(0);)

<!DOCTYPE html><html>

<head>

<meta name="viewport" content="width=device-width" />

<title>WebSocket 客户端</title>

</head>

<body>

<div>

<input type="button" id="btnConnection" value="连接" />

<input type="button" id="btnClose" value="关闭" />

<input type="button" id="btnSend" value="发送" />

</div>

<script src="js/jquery-1.11.1.min.js" type="text/javascript" charset="utf-8"></script>

<script type="text/javascript">

var socket;

if(typeof(WebSocket) == "undefined") {

alert("您的浏览器不支持WebSocket");

//return; }

$("#btnConnection").click(function() {

//实现化WebSocket对象，指定要连接的服务器地址与端口 socket = new WebSocket("ws://127.0.0.1:8080/spring-websocket-test/demo2/duanxz");

//打开事件 socket.onopen = function() {

alert("Socket 已打开");

//socket.send("这是来自客户端的消息" + location.href + new Date()); };

//获得消息事件 socket.onmessage = function(msg) {

alert(msg.data);

};

//关闭事件 socket.onclose = function() {

alert("Socket已关闭");

};

//发生了错误事件 socket.onerror = function() {

alert("发生了错误");

}

});

//发送消息 $("#btnSend").click(function() {

socket.send("这是来自客户端的消息" + location.href + new Date());

});

//关闭 $("#btnClose").click(function() {

socket.close();

});

</script>

</body>

</html>

[IMG_257](https://www.cnblogs.com/duanxz/p/javascript:void(0);)

# **二、WebSocket服务器端**

JSR356定义了WebSocket的规范，Tomcat7中实现了该标准。JSR356 的 WebSocket 规范使用 javax.websocket.\*的 API，可以将一个普通 Java 对象（POJO）使用 @ServerEndpoint 注释作为 WebSocket 服务器的端点。

[IMG_258](https://www.cnblogs.com/duanxz/p/javascript:void(0);)

package org.springframework.samples.websocket.echo2;

import java.io.IOException;

import javax.websocket.CloseReason;import javax.websocket.OnClose;import javax.websocket.OnError;import javax.websocket.OnMessage;import javax.websocket.OnOpen;import javax.websocket.Session;import javax.websocket.server.ServerEndpoint;

@ServerEndpoint("/push")public class EchoEndpoint {

@OnOpen

public void onOpen(Session session) throws IOException {

// 以下代码省略... }

@OnMessage

public String onMessage(String message) {

// 以下代码省略...

return null;

}

@Message(maxMessageSize = 6)

public void receiveMessage(String s) {

// 以下代码省略... }

@OnError

public void onError(Throwable t) {

// 以下代码省略... }

@OnClose

public void onClose(Session session, CloseReason reason) {

// 以下代码省略... }

}

[IMG_259](https://www.cnblogs.com/duanxz/p/javascript:void(0);)

上面简洁代码即建立了一个WebSocket的服务端，@ServerEndpoint("/push")的annotation注释端点表示将WebSocket服务端运行在ws://[Server端IP或域名]:[Server端口]/项目/push的访问端点，客户端浏览器已经可以对WebSocket客户端API发起HTTP长连接了。  
使用ServerEndpoint注释的类必须有一个公共的无参数构造函数，

@OnMessage注解的Java方法用于接收传入的WebSocket信息，这个信息可以是文本格式，也可以是二进制格式。  
@OnOpen注解的java方法用于在这个端点一个新的连接建立时被调用。参数提供了连接的另一端的更多细节。Session表明两个WebSocket端点对话连接的另一端，可以理解为类似HTTPSession的概念。  
@OnClose注解的java方法用于在连接被终止时调用。参数closeReason可封装更多细节，如为什么一个WebSocket连接关闭。  
更高级的定制如@Message注释，MaxMessageSize属性可以被用来定义消息字节最大限制，在示例程序中，如果超过6个字节的信息被接收，就报告错误和连接关闭。

### **三、常见状态码**

任何一端发送关闭帧给对方，即可关闭连接。关闭连接时通常都带有关闭连接的状态码(status code)。常见**状态码**的含义如下：

* 1000 连接正常关闭
* 1001 端点离线，例如服务器down，或者浏览器已经离开此页面
* 1002 端点因为协议错误而中断连接
* 1003 端点因为受到不能接受的数据类型而中断连接
* 1004 保留
* 1005 保留, 用于提示应用未收到连接关闭的状态码
* 1006 端点异常关闭
* 1007 端点收到的数据帧类型不一致而导致连接关闭
* 1008 数据违例而关闭连接
* 1009 收到的消息数据太大而关闭连接
* 1010 客户端因为服务器未协商扩展而关闭
* 1011 服务器因为遭遇异常而关闭连接
* 1015 TLS握手失败关闭连接

### **四、部署**

如果在与Web服务器集成的情况下使用使用WebSocket，通常需要Web服务器进行额外配置，具体可以参见各种Web服务器的配置方案，此处不再赘述：

Apache：[http://httpd.apache.org/docs/2.4/mod/mod\_proxy\_wstunnel.html](http://httpd.apache.org/docs/2.4/mod/mod_proxy_wstunnel.html" \t "https://www.cnblogs.com/duanxz/p/_blank)。   
Nginx：[https://www.nginx.com/blog/websocket-nginx/](https://www.nginx.com/blog/websocket-nginx/" \t "https://www.cnblogs.com/duanxz/p/_blank)。

既然WebSocket是HTML5新增的特性，那么在使用时我们就要考虑浏览器旧版本兼容的问题，这也是Comet方案尽管存在各种问题，但仍旧被采用的原因。

### **五、示例**

[IMG_260](https://www.cnblogs.com/duanxz/p/javascript:void(0);)

package org.springframework.samples.websocket.echo2;

import javax.websocket.CloseReason;import javax.websocket.OnClose;import javax.websocket.OnError;import javax.websocket.OnMessage;import javax.websocket.OnOpen;import javax.websocket.Session;import javax.websocket.server.PathParam;import javax.websocket.server.ServerEndpoint;

import org.springframework.web.socket.server.standard.SpringConfigurator;

//ws://127.0.0.1:8080/project/ws/张三

@ServerEndpoint(value = "/demo2/{user}", configurator = SpringConfigurator.class)public class WSServer {

private String currentUser;

// 连接打开时执行 @OnOpen

public void onOpen(@PathParam("user") String user, Session session) {

currentUser = user;

System.out.println("Connected ... " + session.getId());

}

// 收到消息时执行 @OnMessage

public String onMessage(String message, Session session) {

System.out.println(currentUser + "：" + message);

return currentUser + "：" + message;

}

// 连接关闭时执行 @OnClose

public void onClose(Session session, CloseReason closeReason) {

System.out.println(String.format("Session %s closed because of %s", session.getId(), closeReason));

}

// 连接错误时执行 @OnError

public void onError(Throwable t) {

t.printStackTrace();

}

}

[IMG_261](https://www.cnblogs.com/duanxz/p/javascript:void(0);)

url中的字符张三是的路径参数，响应请求的方法将自动映射。

# **六、测试运行**



