

xNPE 

2018年09月29日 阅读 2875

[关注](#)

# WebSocket的故事（六）—— Springboot中，实现更灵活的WebSocket

## 概述

WebSocket的故事系列计划分五大篇六章，旨在由浅入深的介绍WebSocket以及在Springboot中如何快速构建和使用WebSocket提供的能力。本系列计划包含如下几篇文章：

[第一篇，什么是WebSocket以及它的用途](#)

[第二篇，Spring中如何利用STOMP快速构建WebSocket广播式消息模式](#)

[第三篇，Springboot中，如何利用WebSocket和STOMP快速构建点对点的消息模式\(1\)](#)

[第四篇，Springboot中，如何利用WebSocket和STOMP快速构建点对点的消息模式\(2\)](#)

[第五篇，Springboot中，实现网页聊天室之自定义WebSocket消息代理](#)

**第六篇，Springboot中，实现更灵活的WebSocket**

## 本篇的主线

本篇是这个系列的最后一篇，将介绍另一种实现WebSocket的方式。仍然会以一个简单聊天室为例子进行讲述。至此我们也可以根据具体情况，选择不同的实现方式。

## 本篇适合的读者

想了解如何在Springboot上自定义实现更为复杂的WebSocket产品逻辑的各路有志青年。

## 使用Tomcat提供的WebSocket支持

早在 **Java EE 7** 时，就发布了 **JSR356** 规范。Tomcat7.0.47开始，也支持了统一的 **WebSocket** 接口。

[首页](#) ▾[登录](#)

## 1. 引入依赖

xml

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>
```

Springboot内置了tomcat，我们直接引入spring的这个高级组件即可。顺便多说一句，Springboot的高级组件会自动引用基础的组件，像 `spring-boot-starter-websocket` 就引入了 `spring-boot-starter-web` 和 `spring-boot-starter`，所以不要重复引入。

## 2. 使用@ServerEndpoint创建WebSocket Endpoint

首先要注入 `ServerEndpointExporter`，这个Bean会自动注册使用了 `@ServerEndpoint` 注解声明的WebSocket Endpoint。

java

```
package com.draw.wdraw.websocket;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.socket.server.standard.ServerEndpointExporter;

@Configuration
public class WebSocketConfig {
    @Bean
    public ServerEndpointExporter serverEndpointExporter() {
        return new ServerEndpointExporter();
    }
}
```

然后，我们动手实现WebSocket服务的实现类，这里是 `WebSocketServer`，注意别忘了用 `@ServerEndpoint` 和 `@Component` 声明下。虽然 `@Component` 默认是单例模式的，但Springboot还是会为每个WebSocket连接初始化一个 `Bean`，所以可以用一个静态Map保存起来。换句话说，每当有一个用户向服务器发起连接时，都会创建一个 `WebSocketServer` 对象，将此对象按 `roomId` 保存在 `HashMap` 中，方便后续使用。

创建ServerEndpoint时，需要对应实现其所需的几个功能性方法：`OnOpen`、`OnMessage`、`OnClose`、`OnError`。

- `@OnOpen`：客户端向服务端发起建立连接时，服务端调用，可传入的参数为Session(WebSocket的

[首页](#) ▼[登录](#)

是roomId，即在建立连接时，携带的请求地址上的参数，与我们上一篇中介绍的{INFO}是一样的作用。

- **@OnMessage**：客户端消息到来时调用，包含会话Session，根据消息的形式，如果是文本消息，传入String类型参数或者Reader，如果是二进制消息，传入byte[]类型参数或者InputStream。
- **@OnClose**：当断开连接，关闭WebSocket时调用。
- **@OnError**：当发生错误时调用，传入异常Session和错误信息。

重写上述方法，即可实现WebSocket的服务端业务逻辑。

java

```
@ServerEndpoint("/websocket/{roomId}")
@Component
public class WebSocketServer {
    private static ConcurrentHashMap<String, List<WebSocketServer>> websocketMap =
        new ConcurrentHashMap<>(3);

    //与某个客户端的连接会话，需要通过它来给客户端发送数据
    private Session session;

    //接收roomId
    private String roomId = "";

    /**
     * 连接建立成功调用的方法
     */
    @OnOpen
    public void onOpen(Session session, EndpointConfig config, @PathParam("roomId") String
        if (roomId == null || roomId.isEmpty()) return;
        this.session = session;
        this.roomId = roomId;
        addSocketServer2Map(this);
        try {
            sendMessage("连接成功", true);
        } catch (IOException e) {
        }
    }

    /**
     * 连接关闭调用的方法
     */
    @OnClose
    public void onClose() {
        List<WebSocketServer> wssList = websocketMap.get(roomId);
        if (wssList != null) {
            for (WebSocketServer item : wssList) {
                if (item.session.getId().equals(session.getId())) {
                    item.session.close();
                }
            }
        }
    }
}
```



首页 ▾

搜索掘金



```

        websocketMap.remove(roomId);
    }
    break;
}
}
}

/**
 * 收到客户端消息后调用的方法
 */
@OnMessage
public void onMessage(String message, Session session) {
    //群发消息
    String msg = filterMessage(message);
    if (msg != null) {
        sendInfo(msg, roomId, session);
    }
}

/**
 * 发生错误时，调用的方法
 */
@OnError
public void onError(Session session, Throwable error) {
}

```

这样，服务端的代码就实现完了，这里仅贴出来部分源码，文后会给出项目源码地址。

### 3. 实现客户端页面

```

<script type="text/javascript">
    var ws;

    function setConnected(connected){
        document.getElementById('connect').disabled = connected;
        document.getElementById('disconnect').disabled = !connected;
        document.getElementById('conversationDiv').style.visibility = connected ? 'visible'
        $('#response').html();
    }

    function connect(){
        var roomId = $('#roomId').val();

```

html


[首页](#) ▼



```
ws.onmessage = WSonMessage;
ws.onclose = WSonClose;
ws.onerror = WSonError;
}

function WSonOpen() {
  var message = {
    name: 'Server',
    chatContent: '成功连接'
  }
  setConnected(true);
  showResponse(message)
};

function WSonMessage(event) {
  var message = {
    name: 'Server',
    chatContent: event.data
  }
  showResponse(message)
};

function WSonClose() {
  var message = {
    name: 'Server',
    chatContent: '已断开'
  }
  showResponse(message)
};

function WSonError() {
  var message = {
    name: 'Server',
    chatContent: '连接错误!'
  }
  showResponse(message)
};

function disconnect(){
  ws.close()
  setConnected(false);
  console.log("Disconnected");
}

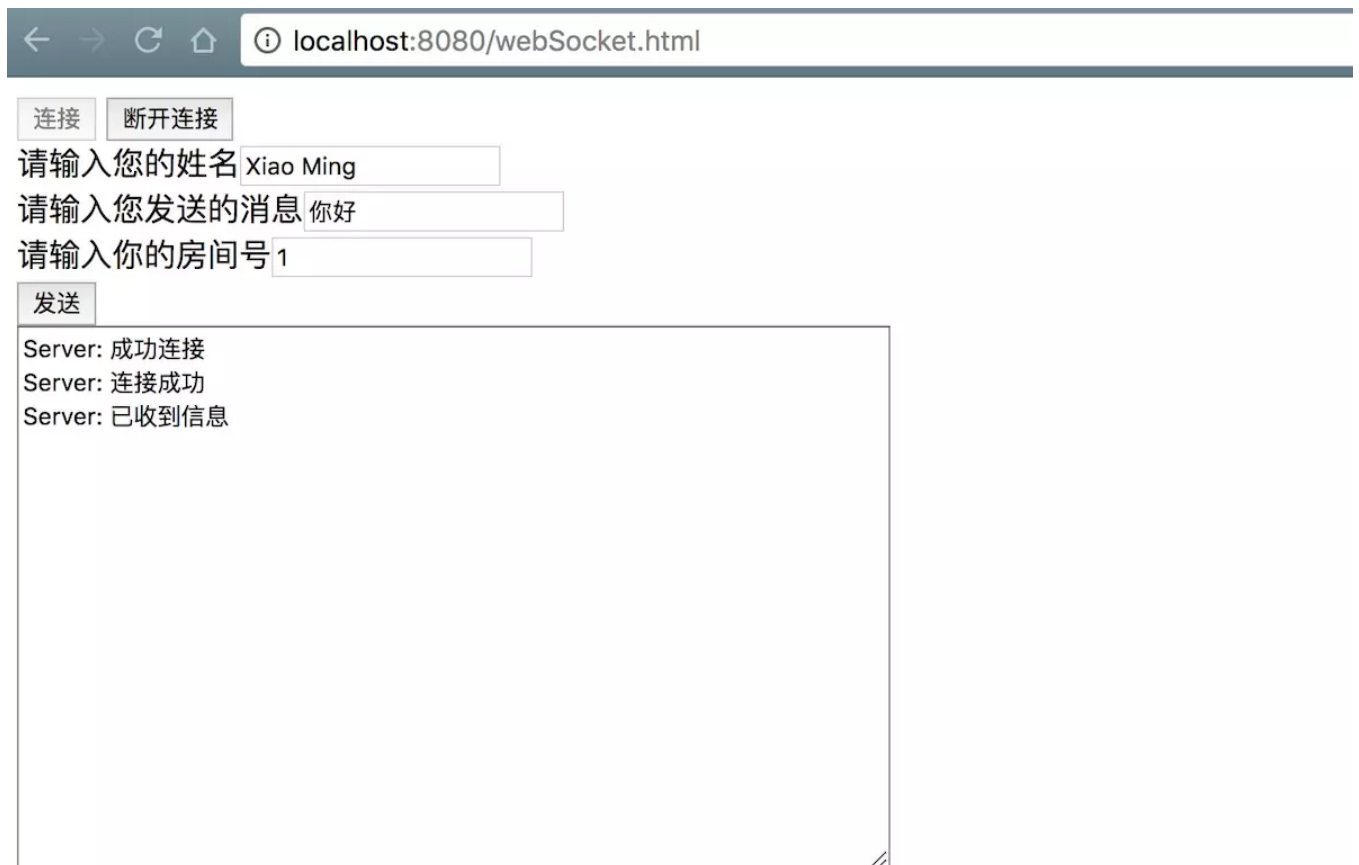
function sendMessage(){
  var chatContent = $('#chatContent').val();
  var roomId = $('#roomId').val();
```



```
function showResponse(message){  
    var response = $("#response").val();  
    $("#response").val(response+message.name+': '+message.chatContent+'\n');  
}  
</script>
```

客户端页面实现了简单的连接、断开和消息发送功能。这部分就不详细介绍了。

## 4. 演示截图



## 源码地址

本篇的源码地址：

## 总结

本篇直接使用了Tomcat提供的WebSocket，也是一种相对灵活的实现方式，只需要按照上述步骤来实现即可。集中精力编写业务逻辑代码。

整个一个系列下来，我们介绍了几种实现WebSocket的方式，有的集成度高，有些相对灵活。大家可以按实际业务需求来选取合适的方式。至此，这个系列就结束了。感谢大家阅读。

小铭出品，必属精品

欢迎关注xNPE技术论坛，更多原创干货每日推送。



微信搜一搜

Q xNPE技术论坛

关注下面的标签，发现更多相似文章

WebSocket

后端

Tomcat

Spring

xNPE Lv2

非典型程序员 @ 自由开发者  
获得点赞 452 · 获得阅读 23,323

关注

### 安装掘金浏览器插件

打开新标签页发现好内容，掘金、GitHub、Dribbble、ProductHunt 等站点内容轻松获取。快来安装掘金浏览器插件获取高质量内容吧！

### 评论

输入评论...



首页 ▾

搜索掘金

登录



请问这种方案怎么做鉴权比较好，在open方法里面做？

1年前



 回复

相关推荐

专栏 · 叁公子KCN · 4分钟前 · 后端 / Python

图像搜索：给你爬的美女图建一个搜索引擎



专栏 · 江五渣 · 12小时前 · Go / 后端

聊聊 Go 语言中的字符表示与字符串遍历

 7  7

专栏 · 腾讯IVWEB团队 · 2天前 · 后端 / Serverless

「NGW」前端新技术赛场：Serverless SSR 技术内幕

 43  3


专栏 · CoderZS · 2天前 · 后端

美丽的一致性Hash算法

 30  1

专栏 · 码农小胖哥 · 2天前 · Docker / 后端

开源模式面临的生存危机

 12  3

专栏 · shanyue · 4天前 · Node.js / WebSocket

关于一个 websocket 多节点分布式问题的头条前端面试题

 47  4

专栏 · 漫话编程 · 27天前 · 后端 / Java

业务复杂=if else? 刚来的大神竟然用策略+工厂彻底干掉了他们！

 511  84


专栏 · 何甜甜在吗 · 10天前 · 后端 / Java


为什么阿里巴巴要禁用Executors创建线程池？

 83  19



# 运营商劫持狠起来，连json都改

 145

 32

