

用 Spring boot 实现 web socket 应用

作者: byte2018

- 原文链接: https://ld246.com/article/1557219954726
- 来源网站: 链滴
- 许可协议:署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

1、 首先通过idea创建一个Springboot项目,具体操作步骤如下:



点击next

HAR SHE SHE U. 1993	
Project Metad	lata
Artifact:	webscoket
Iype:	Maven Project (Generate a Maven based project archive)
Language:	
Packaging:	
(ava Version:	
Version	
	webscoket
Description:	web scoket test
Package:	com.example.webscoket

点击next

12		
Dependencies 🔍	Spring Boot 2.1.4 ¥	Selected Dependencies
	Spring Integration	Messaging
Web	RabbitMQ	WebSocket
Template Engines	T Yeller	
	Katka Streams	
NoSQL	JMS (ActiveMQ)	
	JMS (Artemis)	
Cloud Core	WebSocket	
Cloud Support	RSocket	
Cloud Config		
Cloud Routing		
Cloud Circuit Breaker		
Cloud Tracing		
Cloud Messaging		
	WebSocket	
Pivotal Cloud Foundry	WebSocket applications with Sock35 and STOMP	
Amazon Web Services		
Azure		
Google Cloud Platform	Using WebSocket to build an interactive web applica	
1/0	Q1 Reference doc	

按照上图操作



点击finish完成项目

2、 创建好的项目的目录结构



3、 创建如下的目录结构



4、 先定义一个常量类备用,代码如下:

package com.example.webscoket.consts;

```
/**
* @program: easy
* @description: WebScoket常量配置
* @author: 王震
* @create: 2019-05-06 10:11
**/
public class GlobalConsts {
  /**
  * @Description Stomp的端点
  **/
  public static final String ENDPOINT = "/gs-guide-websocket";
  /**
  * @Description 收发消息url前缀
  **/
  public static final String APP_PREFIX = "/app";
  /**
  * @Description 接收消息地址
  **/
  public static final String RECEIVE MAPPING = "/receive";
  /**
  * @Description 点对点消息推送地址前缀
  **/
  public static final String P2PPUSHBASEPATH = "/user";
  /**
  * @Description 点对点消息推送地址后缀,最后的地址为/user/用户识别码/msg
  **/
```



代码结构如下:

y 📰 Project 👻	😯 😤 🌣 🦟 🎢 webscoket 👘 🖪 WebscoketApplication java 👘 🚷 Global Cents java
🖉 🔻 💵 webscoket Cr(Users/wangshen/Jearn/webstaket	<pre>package com.example.webscoket.consts;</pre>
🖌 🕨 lides	
 Im seen 	
T III ST	* Anternation: webScoker#BRE
▼ Itt main	function: 1.
🐨 🖿 java	Creates 2019-05-06 10:21
TE com example webscoket	
Ra config	public static Final String TOPIC + "/topic/greetings";
▼ Bit consts	
GlobalConsts	12 Description Storperiko
🖿 controller	multic static final steins (ADDING a -Jas enite option bet
Dit service	
Di vo	1) Brescelation OthinBurling
G WebscoketApplication	
Tag resources	popule static times sering over sering a room t
eg application properties	>i * Rescention BOX8951
 Bit test 	
🛱 giágnore	22 public static final String <u>RECEIVE_MAPPING</u> = "/receive";
A HELP.md	
a mene	35 · · · · · · · · · · · · · · · · · · ·
il mmw.cmd	
m ponum!	public static tinal String <u>P2PPUSMUSEPATH</u> - //ster_1
eebscoket.im/	 IDescentation 点对点击原理关闭过后提,最后的放起为/asser/用户记用词
 Im External Libraries 	
 Scratches and Consoles 	



package com.example.webscoket.config;

import com.example.webscoket.consts.GlobalConsts; import org.springframework.context.annotation.Configuration; import org.springframework.messaging.simp.config.MessageBrokerRegistry; import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker; import org.springframework.web.socket.config.annotation.StompEndpointRegistry; import org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigu er;

```
/**
* @Author 王震
* @Description Webscoket配置类
* @Date 21:33 2019/5/6
**/
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {
 @Override
 public void configureMessageBroker(MessageBrokerRegistry config) {
   /**
   * 配置消息代理
   * 启动简单Broker, 消息的发送的地址符合配置的前缀来的消息才发送到这个broker
   */
   config.enableSimpleBroker(GlobalConsts.P2PPUSHBASEPATH);
   config.setUserDestinationPrefix(GlobalConsts.P2PPUSHBASEPATH);
   config.setApplicationDestinationPrefixes(GlobalConsts.APP PREFIX);
 }
 @Override
 public void registerStompEndpoints(StompEndpointRegistry registry) {
```

/**

```
* addEndpoint: 添加STOMP协议的端点。这个HTTP URL是供WebSocket或SockJS客户端访的地址
```

```
* setAllowedOrigins("*") 允许跨域
* withSockJS: 指定端点使用SockJS协议
*/
registry.addEndpoint(GlobalConsts.ENDPOINT)
.setAllowedOrigins("*")
.withSockJS();
}
```

}

程序结构如下:



6、 定义POJO类

接收客户端发过来的消息

package com.example.webscoket.vo;

import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;

/**

* @program: easy

- * @description: 客户端发过来的消息
- * @author: 王震

* @create: 2019-05-06 10:22 **/ @Data @AllArgsConstructor @NoArgsConstructor public class ClientMessage { private int id; private String name;

}

服务端返回的消息

package com.example.webscoket.vo;

import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;

```
/**
* @program: easy
* @description: 服务端返回消息
* @author: 王震
* @create: 2019-05-06 10:25
**/
@Data
@AllArgsConstructor
@NoArgsConstructor
public class ServerMessage {
    private String content;
}
```

注意,项目采用了lombok和fastjson,需要在pom文件中加载相关依赖

<dependency>

```
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<optional>true</optional>
</dependency>
<dependency>
<groupId>com.alibaba</groupId>
<artifactId>fastjson</artifactId>
<version>1.2.54</version>
</dependency>
```

7、 定义service接口和service实现类

package com.example.webscoket.service;

/**

```
* @Author 王震
* @Description 消息服务接口实现
* @Date 21:46 2019/5/6
**/
public interface SimpMessagingService {
    /**
    * @Author 王震
    * @Description 定义消息发送模板,点对点发送
    * @Date 21:48 2019/5/6
    * @Param [user, destination, payload]
    * @return void
    **/
    void sendP2P(String user, String destination, Object payload);
}
```

package com.example.webscoket.service.impl;

import com.example.webscoket.service.SimpMessagingService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.messaging.simp.SimpMessagingTemplate; import org.springframework.stereotype.Service;

/**

* @program: webscoket

- * @description: 消息服务接口实现
- * @author: 王震
- * @create: 2019-05-06 21:48

**/

@Service

public class SimpMessagingServiceImpl implements SimpMessagingService {

```
@Autowired
private SimpMessagingTemplate template;
/**
* @Author 王震
* @Description 定义消息发送模板,点对点发送
* @Date 21:48 2019/5/6
* @Param [user, destination, payload]
* @return void
**/
@Override
public void sendP2P(String user, String destination, Object payload) {
    template.convertAndSendToUser(user, destination, payload);
  }
}
```

8、 创建控制类,用于接收和响应消息

package com.example.webscoket.controller;

import com.alibaba.fastjson.JSON; import com.example.webscoket.consts.GlobalConsts;

import com.example.webscoket.service.SimpMessagingService; import com.example.webscoket.vo.ClientMessage; import com.example.webscoket.vo.ServerMessage; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.messaging.handler.annotation.MessageMapping; import org.springframework.stereotype.Controller; import org.springframework.web.util.HtmlUtils;

/**

- * @program: easy
- * @description: webscoket测试controller
- * @author: 王震
- * @create: 2019-05-06 10:26

**/

@Controller
public class ScoketController {

@Autowired

private SimpMessagingService simpMessagingService;

```
/**
 * @Author 王震
 * @Description 接收和相应消息
 * @Date 16:42 2019/5/7
 * @Param [message]
 * @return void
 **/
 @MessageMapping(GlobalConsts.RECEIVE_MAPPING)
 public void scoket(ClientMessage message){
    simpMessagingService.sendP2P(message.getId() + "", GlobalConsts.P2PPUSHPATH, JSON
toJSON(new ServerMessage("Hello, " + HtmlUtils.htmlEscape(message.getName()) + "!")));
  }
}
```

9、 这样整个服务端的程序就搭建完成了,整体的代码结构如下图所示:



10、 现在编写测试程序, 先下载相关的js库支持, 如下图所示:

这台电脑 > 桌面 > test > test					~ Ü	搜索"
* ^	名称	修改日期	運行	大小		
* ^ vema	👔 jquery.min.js	2019/5/7 16:13	JavaScript 文件	85 KB		
	📓 sockjs.min.js	2019/5/7 16:13	JavaScript 文件	58 KB		
vemc	📓 stomp.min.js	2019/5/7 16:13	JavaScript 文件	8 KB		
		-				

11、 编写测试页面,代码如下:

<!DOCTYPE html>

<html lang="en"><head><meta http-equiv="Content-Type" content="text/html; charset=U F-8">

```
<title>WebScoket测试</title>
 <script src="./test/jquery.min.js"></script>
 <script src="./test/sockjs.min.js"></script>
 <script src="./test/stomp.min.js"></script>
</head>
<body onload="disconnect()">
\langle div \rangle
 < div >
   <button id="connect" onclick="connect();" disabled="">连接</button>
   <button id="disconnect" onclick="disconnect();">断开连接</button>
 </div>
 <div id="conversationDiv" style="visibility: visible;">
   <label>输入你的名字</label><input type="text" id="name">
   <button id="sendName" onclick="sendName();">发送</button>
   </div>
</div>
<script type="text/javascript">
 var stompClient = null;
 function setConnected(connected) {
   document.getElementById('connect').disabled = connected;
   document.getElementById('disconnect').disabled = !connected;
   document.getElementById('conversationDiv').style.visibility = connected ? 'visible' : 'hidde
';
   $('#response').html();
 }
 function connect() {
   // websocket的连接地址,此值等于WebSocketMessageBrokerConfigurer中registry.addEnd
oint("/websocket-simple").withSockJS()配置的地址
   var socket = new SockJS('http://localhost:8080/gs-guide-websocket');
   stompClient = Stomp.over(socket);
   stompClient.connect({}, function(frame) {
     setConnected(true);
     console.log('Connected: ' + frame);
     // 客户端订阅消息的目的地址:此值BroadcastCtl中被@SendTo("/topic/getResponse")注
的里配置的值
     stompClient.subscribe('/user/' + 123 + '/msg', function(respnose){
        showResponse(JSON.parse(respnose.body).content);
     });
   });
 }
 function disconnect() {
   if (stompClient != null) {
     stompClient.disconnect();
   }
   setConnected(false);
   console.log("Disconnected");
 }
```

```
function sendName() {
    var name = $('#name').val();
    // 客户端消息发送的目的: 服务端使用BroadcastCtl中@MessageMapping("/receive")注解的
法来处理发送过来的消息
    stompClient.send("/app/receive", {}, JSON.stringify({ 'id': 123, 'name': name }));
    function showResponse(message) {
        var response = $("#response");
        response.html(message + "\r\n" + response.html());
    }
    </script>
```

</body></html>

12、 启动java程序,页面用浏览器打开,出现如下效果,证明搭建成功。

SH HILLS	TR D Daman Canada Instant Performance	Merce
	E 8 m 1 8 m	0-0
	0.000	
and street	(2 C) Invest Levels Survey Marcel Schwarz Marcel Automore Servery Automore Servery Automore Servery	11 K
\$1.5.0008/P	S G to + O to Distant	0
	Record Market Market	-00
	Teaching bein Teachert	4.4
	And Model generation and a second sec	5
	and an operation of the second s	
	and the second	
	100 Distriction 1100-000 1100-000 1100-000 1100-000 1100-000 1100-000 1100-000 1100-000 1100-000 1100-000 1100-	
	mercland LA based to another a	
	constants and an and and	10
	invester (Merine Internet)	10
	And a second	
	an ear d	
	and the second	
as stan	The later base have been been been been been been been be	8 B
W1. OTOR Plant NO.	II 0 m 1 0 m Instant	0.1
Lines	Recorded Internet	
Hellin, Tevil	Georg and Revel	
	The second s	
	accept carrying (-1,), if	
	and the second sec	
	147 PRAFT/01	
	evice 1.1	
	manufactory application and a second s	
	Deschaf (1982) Int. Stat. Stat	
	National 6.0 (new 2014).1	
	11 100-000 100-000 100-00000 100-0000 100-0000 100-0000 100-0000 100-0000000 100-00000000	
	President Designed	
	merilation (mp. sinita)	
	transfer tables to	
	["AF THE SHOP ("SHOP")	
	and the second s	
	Contraction of the second	
	Neveral Linkshold Control of Cont	
	Province Control of Co	

13、 项目下载