<https://blog.csdn.net/sanchan/article/details/54846806>

本文转载自：[Spring Boot使用Netty SocketIO实现WebIM功能](https://my.oschina.net/u/3233864/blog/829748)

Netty SocketIO是一个 Java语言版本的[Socket.IO](http://www.oschina.net/p/socket-io)服务器的实现，基于[Netty](https://www.oschina.net/p/netty)框架开发，使用简单，功能强大。

在Spring Boot中包含了对Netty SocketIO的支持，只需要简单配置即可加入。

### 第一步：增加Netty SocketIO的依赖。

<dependency>

<groupId>com.corundumstudio.socketio</groupId>

<artifactId>netty-socketio</artifactId>

<version>1.7.11</version>

</dependency>

### 第二步：在Application中增加启动项

package com.ukefu;

import javax.servlet.MultipartConfigElement;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.context.embedded.ConfigurableEmbeddedServletContainer;

import org.springframework.boot.context.embedded.EmbeddedServletContainerCustomizer;

import org.springframework.boot.context.embedded.ErrorPage;

import org.springframework.boot.context.embedded.MultipartConfigFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import org.springframework.http.HttpStatus;

import com.corundumstudio.socketio.AuthorizationListener;

import com.corundumstudio.socketio.Configuration;

import com.corundumstudio.socketio.HandshakeData;

import com.corundumstudio.socketio.SocketConfig;

import com.corundumstudio.socketio.SocketIOServer;

import com.corundumstudio.socketio.annotation.SpringAnnotationScanner;

import com.ukefu.core.UKDataContext;

@EnableAutoConfiguration

@SpringBootApplication

@EnableJpaRepositories("com.ukefu.service.repository")

public class Application {

@Value("${uk.im.server.host}")

private String host;

@Value("${uk.im.server.port}")

private Integer port;

@Bean

public SocketIOServer socketIOServer()

{

Configuration config = new Configuration();

// config.setHostname("localhost");

config.setPort(port);

config.setSocketConfig(new SocketConfig());

// config.setOrigin("http://im.ukewo.com");

config.setWorkerThreads(100);

// config.setStoreFactory(new HazelcastStoreFactory());

config.setAuthorizationListener(new AuthorizationListener() {

public boolean isAuthorized(HandshakeData data) {

return true;

}

});

SocketIOServer server = new SocketIOServer(config);

server.addNamespace(UKDataContext.NameSpaceEnum.IM.toString()) ;

server.addNamespace(UKDataContext.NameSpaceEnum.AGENT.toString()) ;

return server;

}

@Bean

public MultipartConfigElement multipartConfigElement() {

MultipartConfigFactory factory = new MultipartConfigFactory();

factory.setMaxFileSize("50MB"); //KB,MB

factory.setMaxRequestSize("100MB");

return factory.createMultipartConfig();

}

@Bean

public EmbeddedServletContainerCustomizer containerCustomizer() {

return new EmbeddedServletContainerCustomizer() {

@Override

public void customize(ConfigurableEmbeddedServletContainer container) {

ErrorPage error = new ErrorPage("/error.html");

container.addErrorPages(error);

}

};

}

@Bean

public SpringAnnotationScanner springAnnotationScanner(SocketIOServer socketServer) {

return new SpringAnnotationScanner(socketServer);

}

public static void main(String[] args) {

UKDataContext.setApplicationContext(SpringApplication.run(Application.class, args));

}

}

### 第三步：配置Netty Server相关代码

package com.ukefu.util.server;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import com.corundumstudio.socketio.SocketIOServer;

import com.ukefu.core.UKDataContext;

import com.ukefu.util.server.handler.AgentEventHandler;

import com.ukefu.util.server.handler.IMEventHandler;

@Component

public class ServerRunner implements CommandLineRunner {

private final SocketIOServer server;

private IMEventHandler imEventHandler ;

private AgentEventHandler agentEventHandler ;

@Autowired

public ServerRunner(SocketIOServer server , IMEventHandler imEventHandler , AgentEventHandler agentEventHandler) {

this.server = server;

this.imEventHandler = imEventHandler ;

this.agentEventHandler = agentEventHandler ;

}

public void run(String... args) throws Exception {

server.getNamespace(UKDataContext.NameSpaceEnum.IM.toString()).addListeners(imEventHandler);

server.getNamespace(UKDataContext.NameSpaceEnum.AGENT.toString()).addListeners(agentEventHandler);

server.start();

}

}

### 第四步：配置消息处理的代码

package com.ukefu.util.server.handler;

import java.net.InetSocketAddress;

import org.apache.commons.lang.StringUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import com.corundumstudio.socketio.AckRequest;

import com.corundumstudio.socketio.SocketIOClient;

import com.corundumstudio.socketio.SocketIOServer;

import com.corundumstudio.socketio.annotation.OnConnect;

import com.corundumstudio.socketio.annotation.OnDisconnect;

import com.corundumstudio.socketio.annotation.OnEvent;

import com.ukefu.util.server.message.AgentStatusMessage;

import com.ukefu.util.server.message.ChatMessage;

import com.ukefu.util.server.message.NewRequestMessage;

@Component

public class IMEventHandler extends EventHandler

{

@Autowired

public IMEventHandler(SocketIOServer server)

{

super(server) ;

}

@OnConnect

public void onConnect(SocketIOClient client)

{

System.out.println(client.getSessionId());

}

//添加@OnDisconnect事件，客户端断开连接时调用，刷新客户端信息

@OnDisconnect

public void onDisconnect(SocketIOClient client)

{

System.out.println(client.getSessionId());

}

//消息接收入口，网站有新用户接入对话

@OnEvent(value = "new")

public void onEvent(SocketIOClient client, AckRequest request, NewRequestMessage data)

{

try {

String user = client.getHandshakeData().getSingleUrlParam("userid") ;

String orgi = client.getHandshakeData().getSingleUrlParam("orgi") ;

String session = client.getHandshakeData().getSingleUrlParam("session") ;

String appid = client.getHandshakeData().getSingleUrlParam("appid") ;

if(!StringUtils.isBlank(session)){

session = session.replaceAll("-", "") ;

}

if(!StringUtils.isBlank(user)){

/\*\*

\* 用户进入到对话连接 ， 排队用户请求 , 如果返回失败，表示当前坐席全忙，用户进入排队状态，当前提示信息 显示 当前排队的队列位置，不可进行对话，用户发送的消息作为留言处理

\*/

InetSocketAddress address = (InetSocketAddress) client.getRemoteAddress() ;

// NewRequestMessage newRequestMessage = OnlineUserUtils.newRequestMessage(user, orgi , session , appid , address.getHostString() , client.getHandshakeData().getSingleUrlParam("osname") , client.getHandshakeData().getSingleUrlParam("browser")) ;

// /\*\*

// \* 加入到 缓存列表

// \*/

// NettyClients.getInstance().putIMEventClient(user, client);

//

// if(newRequestMessage!=null && !StringUtils.isBlank(newRequestMessage.getMessage())){

// MessageOutContent outMessage = new MessageOutContent() ;

// outMessage.setMessage(newRequestMessage.getMessage());

// outMessage.setMessageType(UKDataContext.MessageTypeEnum.MESSAGE.toString());

// outMessage.setCalltype(UKDataContext.CallTypeEnum.IN.toString());

//

// client.sendEvent(UKDataContext.MessageTypeEnum.STATUS.toString(), outMessage);

// }

}

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

//消息接收入口，坐席状态更新

@OnEvent(value = "agentstatus")

public void onEvent(SocketIOClient client, AckRequest request, AgentStatusMessage data)

{

System.out.println(data.getMessage());

}

//消息接收入口，收发消息，用户向坐席发送消息和 坐席向用户发送消息

@OnEvent(value = "message")

public void onEvent(SocketIOClient client, AckRequest request, ChatMessage data)

{

System.out.println(data.getMessage());

}

}

第五步：完成配置。

以上所有代码在[优客服](https://git.oschina.net/ukewo/ukefu)中找到。[码云地址](https://git.oschina.net/ukewo/ukefu)