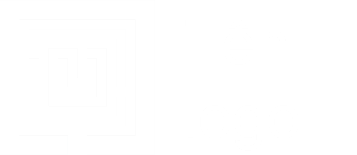


|  |
| --- |
| Springboot – Angular - Websocket |
|  |
| 10 Tháng Mười Hai  TÊN CÔNG TY  Tác giả: Tên bạn |



## WebSocket with Spring boot and Angular

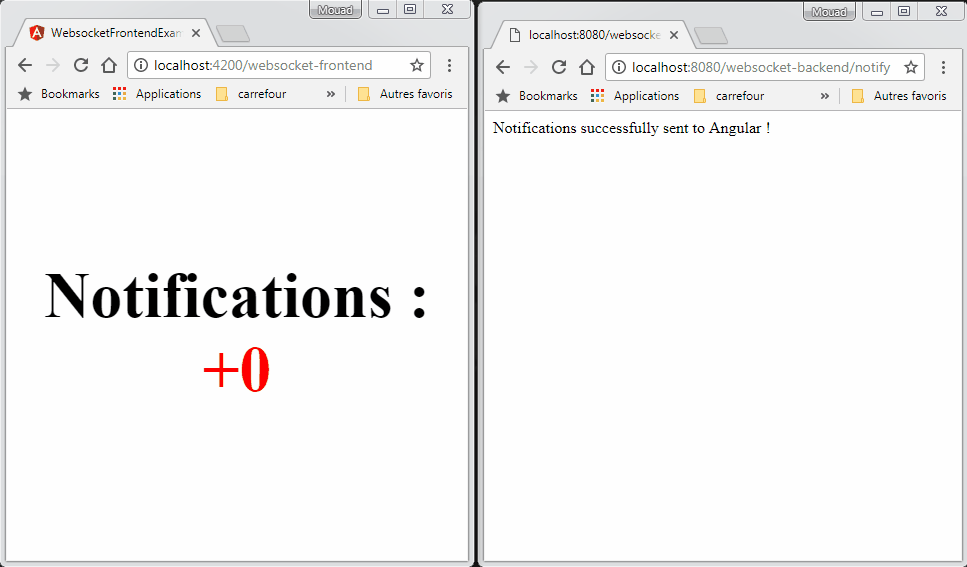
# 1. Overview

**WebSocket** makes it possible to open an interactive communication between a **browser** (front-end) and a **server** (back-end).

It’s a two-way communication protocol that allows not only communication from the front-end to the back-end, but also from the back-end to the front-end as well.

In this article, I will show you how to use **WebSockets** in both **Angular** and **Spring boot** using [SockJS](http://sockjs.org/), [StompJS](http://jmesnil.net/stomp-websocket/doc/) and [Spring WebSocket](https://docs.spring.io/spring/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/websocket.html).

This is an overview of how the final project will look like :



* A Spring boot application as the back-end that use Spring WebSocket to push **notifications** to a **topic**.
* An Angular application as the front-end that use SockJS and StompJS to subscribe to the **topic,**consume **notifications** from it and displaying them in a simple html page.

# 2. Create back-end

## 2.1. Create Spring boot app

Generate a Spring boot project using Spring [intializr](https://start.spring.io/), with **Web**and **Websocket** dependencies selected or add them manually to your **pom.xml**.

<!-- pom.xml -->

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-websocket</artifactId>

</dependency>

</dependencies>

## 2.2. Implement WebSocket

1. Create a configuration Java class annotated with **@EnableWebSocketMessageBroker** to enable WebSockets.

@Configuration

@EnableWebSocketMessageBroker

public class WebSocketConfiguration extends AbstractWebSocketMessageBrokerConfigurer {

@Override

public void registerStompEndpoints(StompEndpointRegistry stompEndpointRegistry) {

stompEndpointRegistry.addEndpoint("/socket")

.setAllowedOrigins("\*")

.withSockJS();

}

@Override

public void configureMessageBroker(MessageBrokerRegistry registry) {

registry.enableSimpleBroker("/topic");

registry.setApplicationDestinationPrefixes("/app");

}

}

2. Create a POJO class to hold the message to be shared between the back-end and the front-end.

public class Notifications {

private int count;

public Notifications(int count) {

this.count = count;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

public void increment() {

this.count++;

}

}

3. Create a web **Controller** and inject **SimpMessagingTemplate** bean in it.

Every time the Controller is called, the **Notifications.count** will be incremented by one andsent to the topic using **SimpMessagingTemplate.convertAndSend()** method.

@RestController

public class NotificationController {

@Autowired

private SimpMessagingTemplate template;

// Initialize Notifications

private Notifications notifications = new Notifications(0);

@GetMapping("/notify")

public String getNotification() {

// Increment Notification by one

notifications.increment();

// Push notifications to front-end

template.convertAndSend("/topic/notification", notifications);

return "Notifications successfully sent to Angular !";

}

}

# 3. Create front-end

## 3.1. Prepare Angular app

1. Create an Angular project using **ng new** command.

ng new websocket-front-end

2. Install StomJS and SockJS-client using **npm install** command.

npm install --save sockjs-client stompjs

## 3.2. Implement WebSocket

1. Create a service with name **WebSocketService**, using **ng generate service** command.

import {Injectable} from "@angular/core";

var SockJs = require("sockjs-client");

var Stomp = require("stompjs");

@Injectable()

export class WebSocketService {

// Open connection with the back-end socket

public connect() {

let socket = new SockJs(`http://localhost:8080/socket`);

let stompClient = Stomp.over(socket);

return stompClient;

}

}

Register the **WebSocketService** as a provider.

import {WebSocketService} from "./services/websocket.service";

@NgModule({

...

providers: [WebSocketService],

....

})

export class AppModule { }

2. Inside the**app.component.ts**, open a connection with the back-end socket and subscribe to the notification topic.

import {Component} from '@angular/core';

import {WebSocketService} from "./services/websocket.service";

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

public notifications = 0;

constructor(private webSocketService: WebSocketService) {

// Open connection with server socket

let stompClient = this.webSocketService.connect();

stompClient.connect({}, frame => {

// Subscribe to notification topic

stompClient.subscribe('/topic/notification', notifications => {

// Update notifications attribute with the recent messsage sent from the server

this.notifications = JSON.parse(notifications.body).count;

})

});

}

}

3. Display the notifications attribute inside the **app.component.html**.

<div style="text-align:center; margin-top: 150px;">

<h1 style="font-size: 4em;">

Notifications : <span style="color: red;">+{{ notifications }}</span>

</h1>

</div>