Websocket

Các bước cài websocket với spring boot

1. Các dependency cần:
    a. spring-boot-starter-thymeleaf, spring-boot-starter-web, spring-boot-starter-websocket, spring-websocket, spring-messaging, jackson-core, jackson-databind, spring-boot-starter-test trong đó:

- spring-boot-starter-websocket: là một trong những dependency cơ bản của Spring Boot, cung cấp cho bạn khả năng sử dụng WebSocket trong ứng dụng Spring Boot. Nó bao gồm các thư viện chính của Spring Boot, bao gồm Spring MVC, Spring Web, và Netty.
- spring-websocket: cung cấp các thành phần và chức năng để xây dựng các ứng dụng WebSocket trong Spring Framework.
- spring-messaging: cung cấp các lớp và giao diện để hỗ trợ việc xây dựng các ứng dụng messaging trong Spring Framework.
- jackson-core và jackson-databind: là các thư viện của Jackson JSON, được sử dụng để chuyển đổi dữ liệu từ JSON sang các đối tượng Java và ngược lại.
- spring-boot-starter-test: là một dependency để xây dựng các test case trong Spring Boot. Nó bao gồm các thư viện phổ biến như JUnit, Mockito, và AssertJ

    b. File maven:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-parent</artifactId>
       <version>2.7.11</version>
       <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>websoket-demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>websoket-demo</name>
    <description>Demo project for Spring Boot</description>
    <properties>
       <java.version>1.8</java.version>
    </properties>
    <dependencies>
       <dependency>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-thymeleaf</artifactId>
       </dependency>
       <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>
       <dependency>
          <groupId>org.springframework</groupId>
          <artifactId>spring-websocket</artifactId>
```

```
            <version>5.2.2.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-messaging</artifactId>
            <version>5.2.2.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-core</artifactId>
            <version>2.10.2</version>
        </dependency>

        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.10.2</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

2. Tạo các dto
    a. Ví dụ: tạo với getter setter, constructer

```
public class ChatMessage {
    private String from;
    private String text;
    private String recipient;
    private String time;
}
```

3. Tạo file config
    a. WebSocketConfig

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig extends
AbstractWebSocketMessageBrokerConfigurer {

    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
```

```
        config.enableSimpleBroker("/topic");
        config.setApplicationDestinationPrefixes("/app");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/broadcast");
    }
}
```

Trong đó:

- **configureMessageBroker** là một method của interface WebSocketMessageBroker Configurer trong Spring Framework. Method này được sử dụng để cấu hình message broker cho ứng dụng WebSocket. Message broker là một phần mềm hoặc một service giúp định tuyến các tin nhắn giữa các client WebSocket và giữa các client và server. Có hai message broker phổ biến trong ứng dụng WebSocket là Simple Broker và RabbitMQ.
- **registerStompEndpoints**() là một method trong WebSocketMessageBrokerConfigurer interface trong Spring framework, nó được sử dụng để đăng ký một hoặc nhiều WebSocket "endpoints" mà các client có thể kết nối đến. Mỗi endpoint có thể được cấu hình để sử dụng một hoặc nhiều khung (frame) của WebSockets, ví dụ như SockJS.

4. Tạo các controller

```
@Controller
public class SocketController {
    @GetMapping("home")
    public String home(){
        return "index";
    }
    @MessageMapping("/broadcast")
    @SendTo("/topic/messages")
    public ChatMessage send(ChatMessage chatMessage) throws Exception {
        System.out.println("client sent: "+chatMessage.getText());
        return new ChatMessage(chatMessage.getFrom(),
chatMessage.getText(),"ALL");
    }
}
```

- **MessageMapping** @MessageMapping là một annotation trong Spring Framework được sử dụng để ánh xạ một phương thức xử lý tin nhắn (message) đến một đường dẫn cụ thể trên server.
- **@SendTo** là một annotation trong Spring Framework được sử dụng để chỉ định đường đi của tin nhắn trả về từ một phương thức xử lý tin nhắn (@MessageMapping).
- Khi một tin nhắn được xử lý bởi một phương thức được chú thích bằng @MessageMapping, phương thức này có thể trả về kết quả (hoặc giá trị). @SendTo được sử dụng để chỉ định đường dẫn đến một topic hoặc một queue trên broker message, trong đó kết quả trả về sẽ được gửi đến.

5. Phía client:
   Dùng stom js để connect tới websocket.
   Ví dụ:

```
// Kết nối đến endpoint "/ws"
var socket = new SockJS('/ws');

// Tạo client Stomp
var stompClient = Stomp.over(socket);

// Kết nối tới message broker
stompClient.connect({}, function(frame) {
    console.log('Connected: ' + frame);

    // Subscribe vào topic "/topic/greetings"
    stompClient.subscribe('/topic/greetings', function(greeting) {
        console.log(JSON.parse(greeting.body).content);
    });
});

// Gửi message tới "/hello" endpoint
stompClient.send("/app/hello", {}, JSON.stringify({'name': 'John'}));
```