KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

AOP (심화1)

[KB] IT's Your Life

KB 국민은행

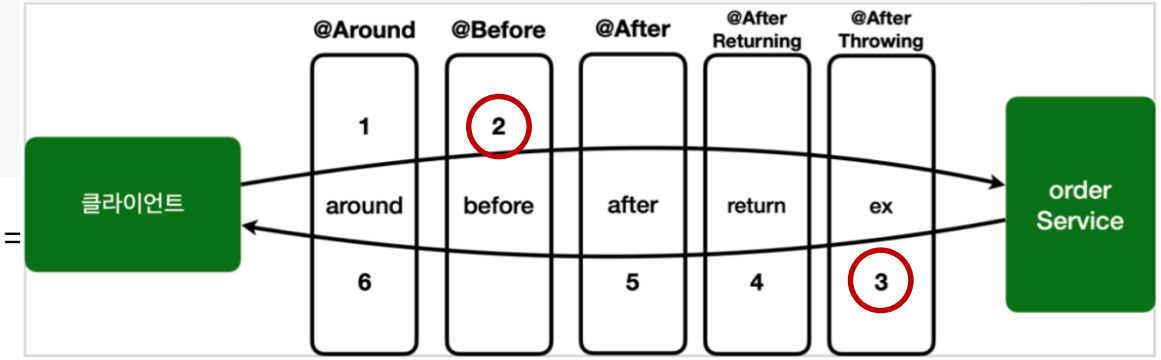
- **SampleService의 모든 메서드에 대해 예외 발생시 호출되는 매개변수를 체크하는 AfterThrowing Advice 추가하고, 단위 테스트로 확인하세요.**
  - 로그 내용:
    - Exception...!!!!

@AfterThrowing

```
@Aspect
@Log4j
@Component
public class LogAdvice {
  ...

  @AfterThrowing(pointcut = "execution(* org.scoula.sample.service.SampleService*.*(..))", throwing="exception")
  public void logException(Exception exception) {
    log.info("Exception...!!!!");
    log.info("exception: " + exception);
  }
}
```

test::SampleServiceTest.java

```
public class SampleServiceTest {
    ...

    @Test
    public void addError() throws Exception {
        log.info(service.doAdd("123", "ABC"));
    }
}
```




```
INFO : org.scoula.advice.LogAdvice - ==========================
INFO : org.scoula.advice.LogAdvice - str1:123
INFO : org.scoula.advice.LogAdvice - str2:ABC
INFO : org.scoula.advice.LogAdvice - Exception...!!!!
INFO : org.scoula.advice.LogAdvice - exception: java.lang.NumberFormatException: For input string: "ABC"
```

- **SampleService의 모든 메서드에 대해 Around Advice에서 매서드 실행시간을 측정하여 출력하도록 작성하고, 단위 테스트로 확인하세요.**
  - 로그 내용:
    - TIME: 실행시간

@Around와 ProceedingJoinPoint

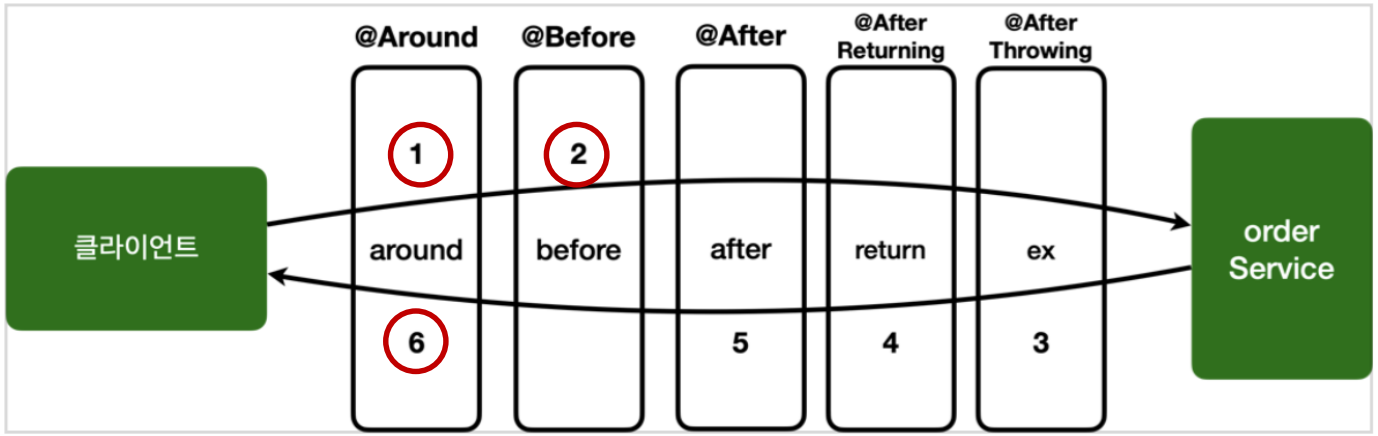
- 메서드의 실행 전과 실행 후에 처리가 가능

```
...
public class LogAdvice {

  @Around("execution(* org.scoula.sample.service.SampleService*.*(..))")
  public Object logTime(ProceedingJoinPoint pjp) {
    long start = System.currentTimeMillis();

    log.info("Target: " + pjp.getTarget());
    log.info("Param: " + Arrays.toString(pjp.getArgs()));

    Object result = null;
    try {
      result = pjp.proceed(); // 실제 메서드 호출
    } catch(Throwable e) {
      e.printStackTrace();
    }
```

@Around와 ProceedingJoinPoint

```
    long end = System.currentTimeMillis();

    log.info("TIME: " + (end - start));

    return result;
  }
}
```

INFO : org.scoula.advice.LogAdvice - Target: org.scoula.sample.service.SampleServiceImpl@74971ed9
INFO : org.scoula.advice.LogAdvice - Param: [123, 456]
INFO : org.scoula.advice.LogAdvice - ==========
INFO : org.scoula.advice.LogAdvice - str1:123
INFO : org.scoula.advice.LogAdvice - str2:456
INFO : org.scoula.advice.LogAdvice - TIME: 3
INFO : org.scoula.sample.service.SampleServiceTest - 579



KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

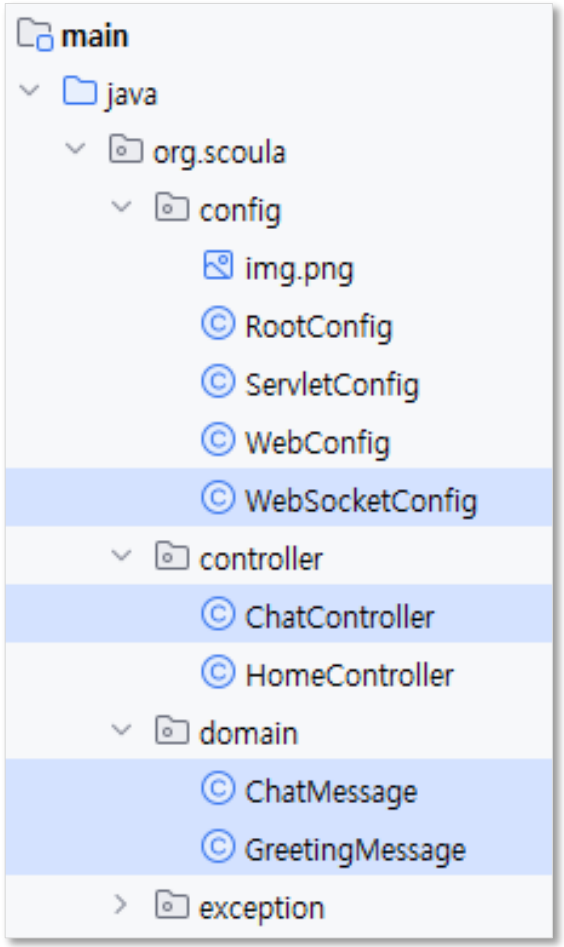
Spring Web Socket, STOMP (심화2)

[KB] IT's Your Life

KB 국민은행

Spring Web Socket, STOMP

- [x] **채팅을 위한 백엔드 Stomp 프로그램을 작성하세요.**

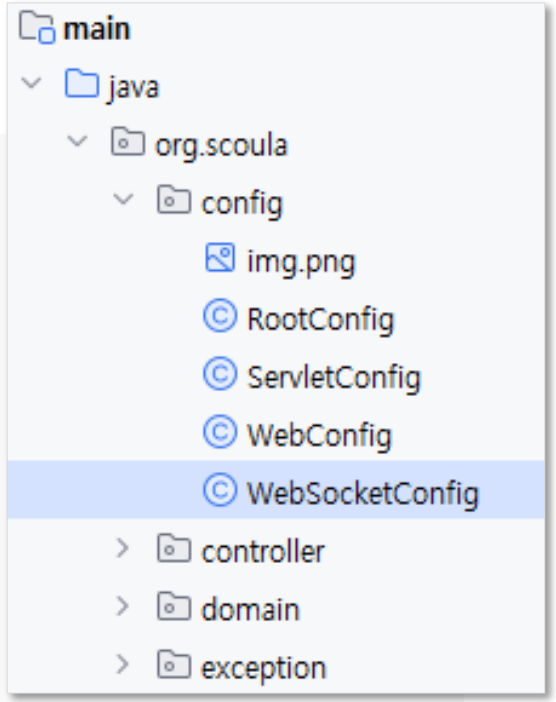


build.gradle

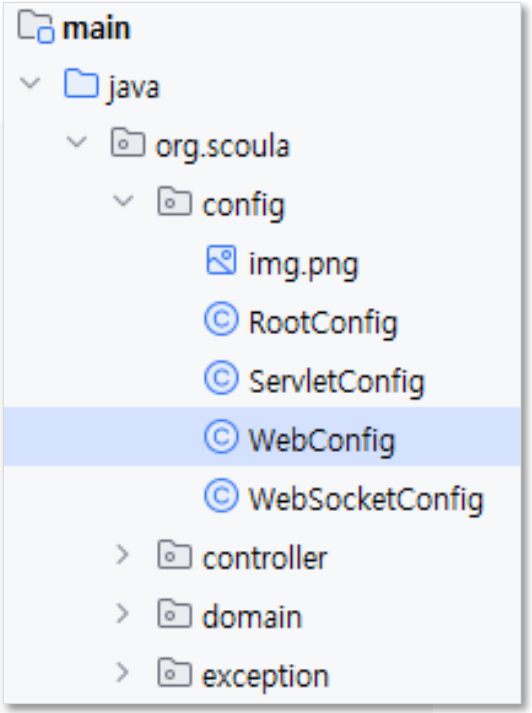
```
implementation "org.springframework:spring-websocket:${springVersion}"
implementation "org.springframework:spring-messaging:${springVersion}"
implementation "org.springframework.integration:spring-integration-stomp:5.5.20"
```

config.WebSocketConfig

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

  @Override
  public void configureMessageBroker(MessageBrokerRegistry config) {
    config.enableSimpleBroker("/topic");                    // 구독시 사용할 토픽 접두어
    // 클라이언트가 발행 시 사용해야하는 접두어
    config.setApplicationDestinationPrefixes("/app");
  }

  @Override
  public void registerStompEndpoints(StompEndpointRegistry registry) {
    registry.addEndpoint("/chat-app")  // 접속 엔드포인트, ws://localhost:8080/chat-app
            .setAllowedOrigins("*");                    // CORS 허용
  }

}
```



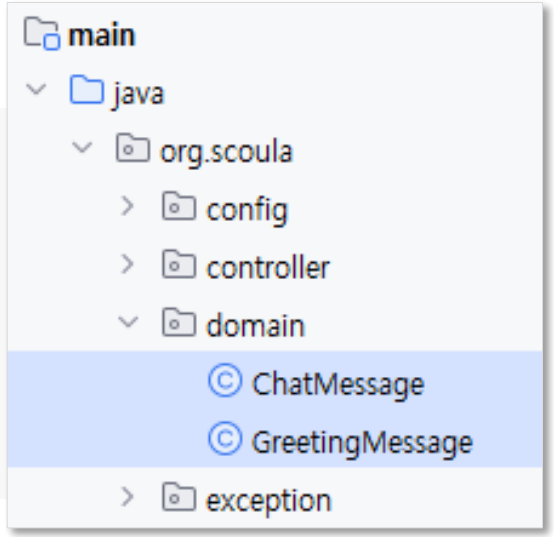
config.WebConfig :: WebSocketConfig 등록




```
public class WebConfig extends AbstractAnnotationConfigDispatcherServletInitializer {

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class[] { RootConfig.class };
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[] { ServletConfig.class, WebSocketConfig.class };
    }
  ...

}
```

domain.GreetingMessage

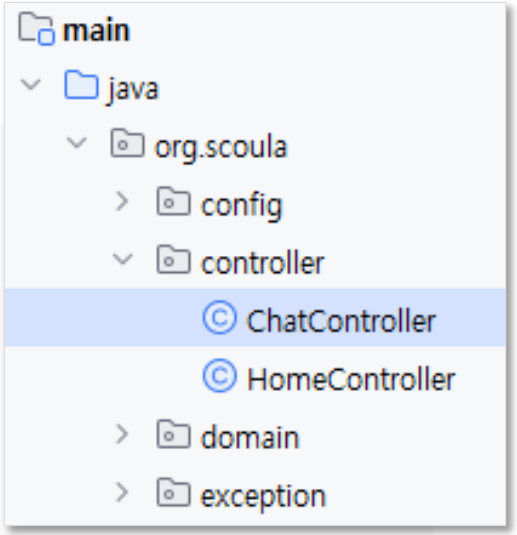
```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class GreetingMessage {
    private String name;
}
```



domain.ChatMessage

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class ChatMessage {
    private String name;
    private String content;
}
```

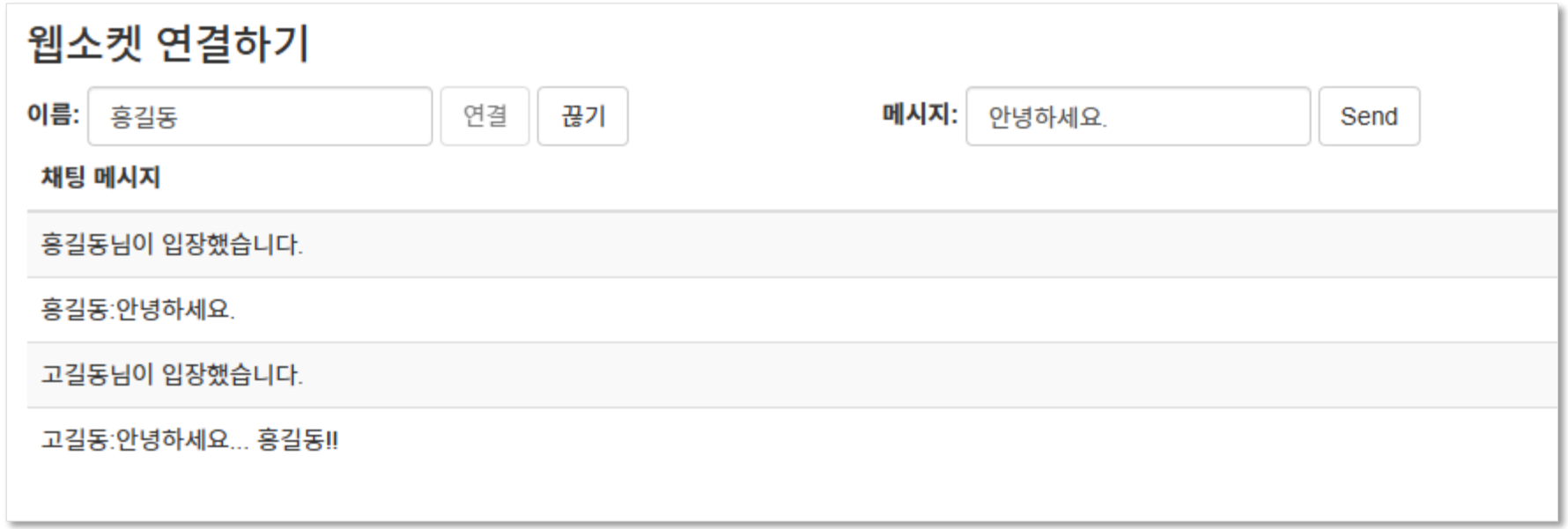
controller.ChatController.java

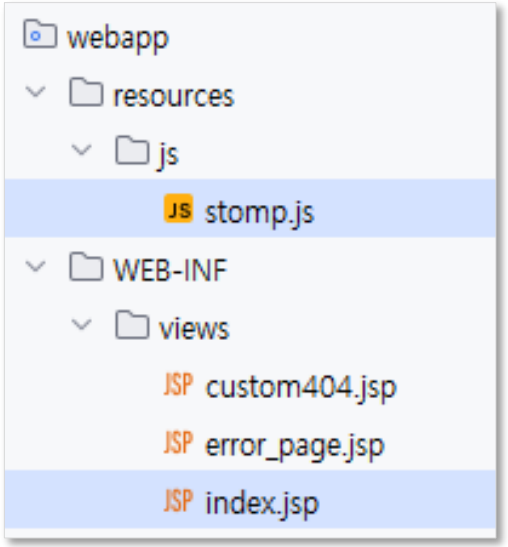



```
@Controller
@Log4j2
public class ChatController {

    @MessageMapping("/hello")
    @SendTo("/topic/greetings")
    public GreetingMessage greeting(GreetingMessage message) throws Exception {
        log.info("greeting: " + message);
        return message;
    }

    @MessageMapping("/chat")
    @SendTo("/topic/chat")
    public ChatMessage chat(ChatMessage message) throws Exception {
        log.info("chat received: " + message);
        return message;
    }
}
```

Spring Web Socket, STOMP

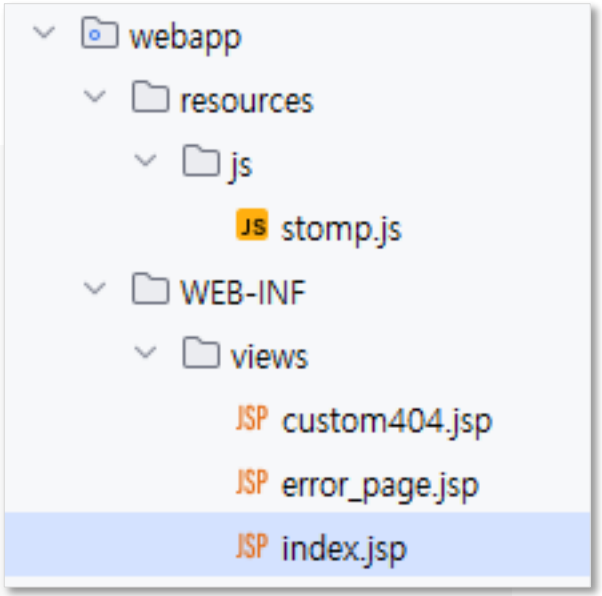
- **다음과 같이 클라이언트 화면을 구성하고, 채팅 프로그램을 작성하세요.**
  - 화면





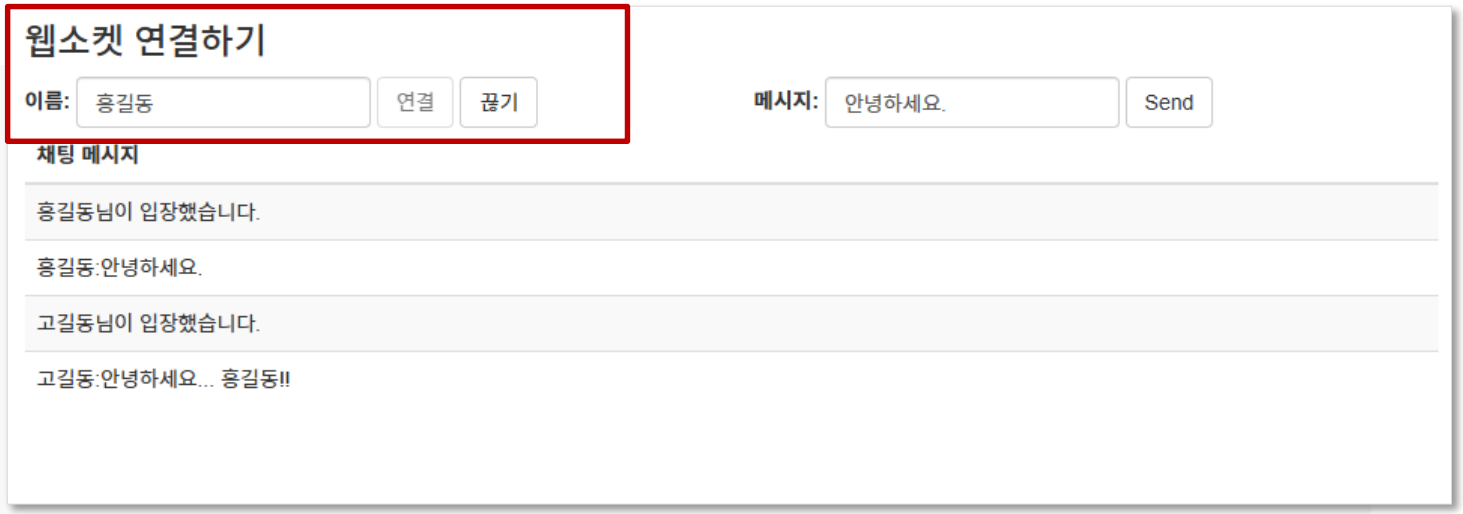
  - Stomp 라이브러리 CDN
    - https://cdn.jsdelivr.net/npm/@stomp/stompjs@7.0.0/bundles/stomp.umd.min.js

index.jsp 기본 골격




```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>
<html>
<head>
  <title>Hello WebSocket</title>
  <link rel="stylesheet"
     href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
     integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
     crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/@stomp/stompjs@7.0.0/bundles/stomp.umd.min.js"></script>
</head>
<body>

  <script src="/resources/js/stomp.js"></script>
</body>
</html>
```

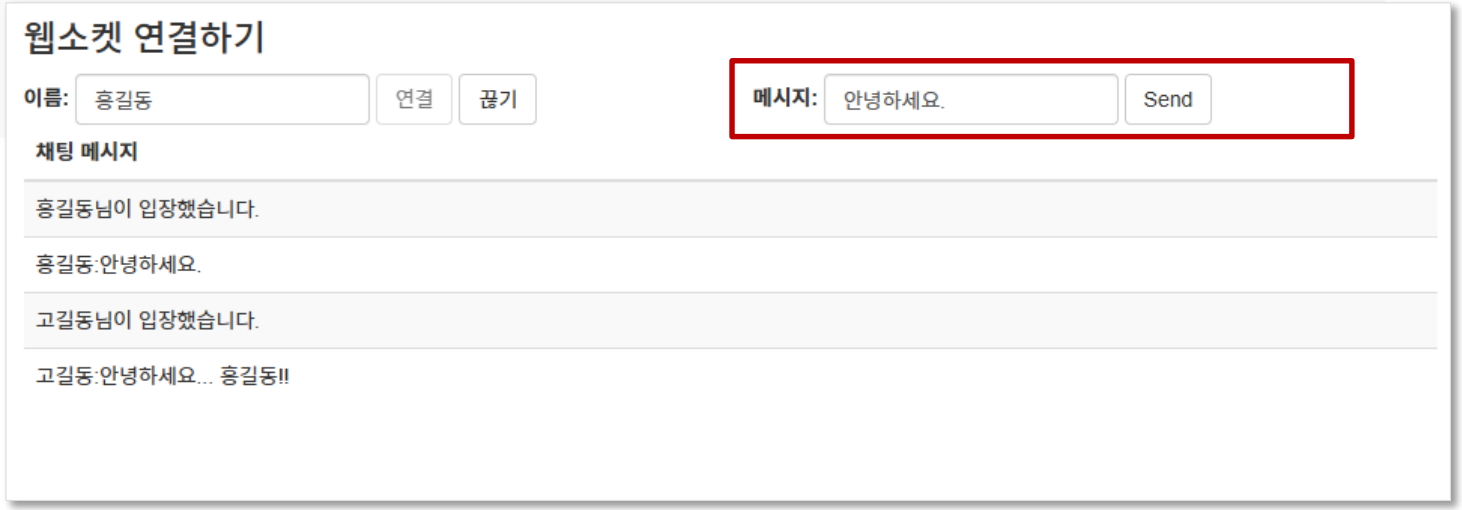
index.jsp




```
<body>
<div id="main-content" class="container">
  <h3>웹소켓 연결하기</h3>
  <div class="row">
    <div class="col-md-6">
      <form class="form-inline">
        <div class="form-group">
          <div class="form-group">
            <label for="name">이름: </label>
            <input type="text" id="name" class="form-control" placeholder="이름을 이력하세요.">
          </div>
          <button id="connect" class="btn btn-default" type="submit">연결</button>
          <button id="disconnect" class="btn btn-default" type="submit" disabled="disabled">끊기
          </button>
        </div>
      </form>
    </div>
```

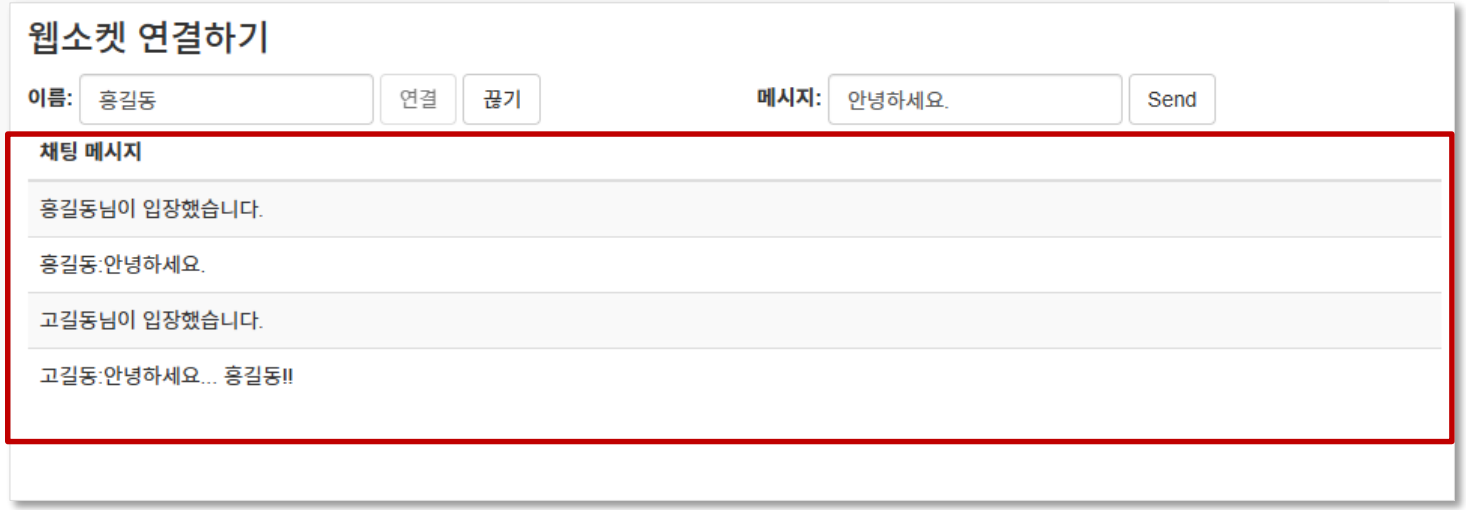
index.jsp

```
<div class="col-md-6">
  <form class="form-inline">
    <div class="form-group">
      <label for="content">메시지:</label>
      <input type="text" id="content" class="form-control" placeholder="메시지를 입력하세요...">
    </div>
    <button id="send" class="btn btn-default" type="submit">Send</button>
  </form>
</div>
</div>
```

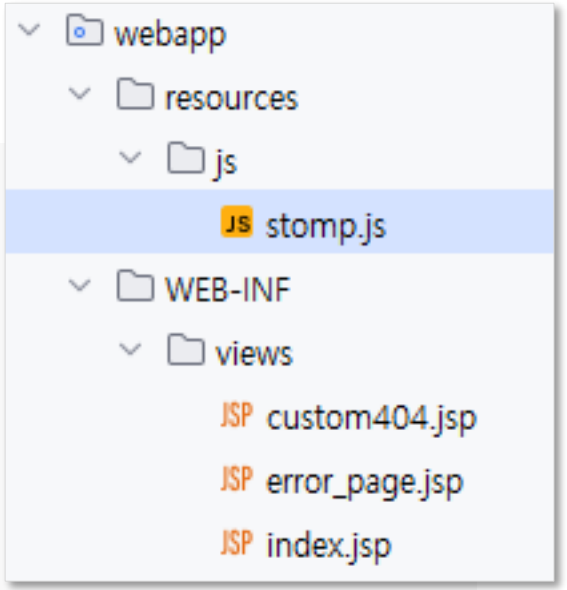


index.jsp

```
<div class="row">
  <div class="col-md-12">
    <table class="table table-striped">
      <thead>
      <tr>
        <th>채팅 메시지</th>
      </tr>
      </thead>
      <tbody id="chat-messages">
      </tbody>
    </table>
  </div>
</div>
</div>
```



resources/js/stomp.js




```
// StompJs.Client 객체 생성
const stompClient = new StompJs.Client({
    brokerURL: 'ws://localhost:8080/chat-app'
});


// 웹 소켓 에러 발생시 콜백
stompClient.onWebSocketError = (error) => {
    console.error('Error with websocket', error);
};

// Stomp 에러 발생시 콜백
stompClient.onStompError = (frame) => {
    console.error('Broker reported error: ' + frame.headers['message']);
    console.error('Additional details: ' + frame.body);
};
```

resources/js/stomp.js

```
// 연결 성공시 콜백
// 구독 토픽 등록
stompClient.onConnect = (frame) => {
    console.log(frame)
    setConnected(true);
    // 구독 토픽 등록 및 수신 처리 핸들러 등록
    // 토픽 문자열: '/topic/greetings' - 입장 메시지
    stompClient.subscribe('/topic/greetings', (greeting) => {
        console.log('/topic/greetings', greeting.body)
        showMessage(JSON.parse(greeting.body).name + '님이 입장했습니다.');
    });
    // 토픽 문자열: '/topic/chat' - chat 메시지
    stompClient.subscribe('/topic/chat', (chat) => {
        console.log('/topic/chat', chat.body)
        const message = JSON.parse(chat.body);
        showMessage(`${message.name}:${message.content}`);
    });
```

resources/js/stomp.js

```
    // 연결 성공시 입장 메시지 보내기
    const name =  document.getElementById('name').value;
    stompClient.publish({
        destination: '/app/hello',
        body: JSON.stringify({name})                    // GreetingMessage에 대응
    });

};

// 연결됐을 때 엘리먼트 프로퍼티 변경
function setConnected(connected) {
    const connectBtn = document.getElementById('connect');
    const disconnectBtn = document.getElementById('disconnect');
    const messages = document.getElementById('chat-messages');

    connectBtn.disabled = connected;
    disconnectBtn.disabled = !connected;
    messages.innerHTML = '';
}
```

resources/js/stomp.js

```
// 연결하기
function connect() {
    stompClient.activate();
}

// 연결 끊기
function disconnect() {
    stompClient.deactivate();
    setConnected(false);
    console.log('Disconnected');
}

// 메시지 전송하기
function sendMessage() {
    const name =  document.getElementById('name').value;
    const content =  document.getElementById('content').value;
    console.log({name, content})
    stompClient.publish({
        destination: '/app/chat',
        body: JSON.stringify({name, content})   // ChatMessage에 대응
    });
}
```

resources/js/stomp.js

```
// 수신 메시지 출력하기
function showMessage(message) {
    const messages = document.getElementById('chat-messages');
    messages.innerHTML += '<tr><td>' + message + '</td></tr>'
}

// 이벤트 핸들러 설정
window.addEventListener("DOMContentLoaded", (event) => {
    const forms = document.querySelectorAll('.form-inline');
    const connectBtn = document.getElementById('connect');
    const disconnectBtn = document.getElementById('disconnect');
    const sendBtn = document.getElementById('send');

    connectBtn.addEventListener('click', () => connect());
    disconnectBtn.addEventListener('click',  () => disconnect());
    sendBtn.addEventListener('click', () => sendMessage());
    for(const form of forms) {
        console.log(form)
        form.addEventListener('submit', (e) => e.preventDefault());
    }
});
```