WebSocket

안 화 수

WebSocket

- WebSocket

- 서버와 클라이언트 간에 Socket Connection을 유지해서 언제든 양방향 통신 또는 데이터 전송이 가능하도록 하는 기술로써, sns, 화상 채팅, 증권 거래 등에서 널리 사용되고 있다.

WebSocket

WebSocket을 사용하는 이유

- 기존에는 pooling 방식으로 일정한 주기로 서버에 요청하고, 응답을 받는 방식이 사용 되기 때문에 서버에 부하가 많이 걸린다.

- 실시간 채팅 앱과 같은 요구사항들을 반영하기 위해 클라이언트에서 매번 HTTP 프로토콜을 통해 서버에 요청하는 것은 비효율적이다.

- 클라이언트에서 Ajax 통신 등으로 일부 보완할 수 있지만, Ajax도 결국 HTTP를 사용하기 때문에 여전히 문제가 되며, 웹 소켓을 통해 이를 해결할 수 있다.

WebSocket

WebSocket의 특징

- WebSocket은 HTTP를 통해 최초 연결되며, 이후 일정 시간이 지나면 HTTP 연결은 자동으로 끊어지고 webSocket connection은 유지된다.
- HTTP와 달리 WebSocket은 stateful 프로토콜이다.
- HTTP는 stateless하기 때문에 서버에 변경사항이 생겨도 클라이언트에서 요청을 하지 않으면 변경사항이 적용되지 않지만, WebSocket은 지속적으로 connection을 유지하기 때문에 실시간으로 변경사항이 적용된다.
- WebSocket은 HTTP와 동일한 port(80)를 사용한다.

WebSocket 환경구축

- WebSocket 환경 설정

1. pom.xml 파일에 WebSocket 의존 라이브러리를 추가한다.

2. servlet-context.xml 파일에 WebSocket handler mapping을 설정한다.

3. WebSocket handler 클래스를 생성한다.

WebSocket 환경구축

1. pom.xml 파일에 WebSocket 의존 라이브러리를 추가한다.

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-websocket</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
```

WebSocket 환경구축

2. servlet-context.xml 파일에 WebSocket handler mapping을 설정한다.

```
<default-servlet-handler/>
<websocket:handlers>
    <websocket:mapping handler="chatHandler" path="chat-ws.do"/>
</websocket:handlers>
<beans:bean id="chatHandler" class="com.ch.webSock.WebChatHandler"/>
```

WebSocket 환경구축

3. WebSocket handler 클래스를 생성한다. (1/2)

```
public class WebChatHandler extends TextWebSocketHandler {
        Map<String, WebSocketSession> users = new HashMap<String, WebSocketSession>();

        @Override        // 웹소켓 연결 직후에 처리
        public void afterConnectionEstablished(WebSocketSession session) throws Exception {
                users.put(session.getId(), session);
        }
        @Override    // 웹소켓 연결이 닫힌 직후에 처리
        public void afterConnectionClosed(WebSocketSession session, CloseStatus status) throws
                Exception {
                users.remove(session.getId());
        }
```

WebSocket 환경구축

3. WebSocket handler 클래스를 생성한다. (2/2)

```
@Override    // 웹소켓 연결후 세션과 메시지를 처리
protected void handleTextMessage(WebSocketSession session, TextMessage message)  throws
            Exception {
                String msg = message.getPayload();
                TextMessage tMsg = new TextMessage(msg.substring(4));
                Collection<WebSocketSession> list = users.values();
                for (WebSocketSession wss : list) {
                        wss.sendMessage(tMsg);
                }
        }
}
```

WebSocket Project

- websocket project를 import 해보자.

WebSocket Project

- websocket project 실행화면