WebSocket

❖ 웹 소켓

- 클라이언트와 서버간 실시간 양방향 통신 지원
- HTML5 표준
- ws 프로토콜 사용
- 서버에서의 웹 소켓 지원
  - Servlet 버전 3 이상 컨테이너에서 지원
  - **디폴트 스프링 MVC 프로젝트는 Servlet 2.5임 (업그레이드 필요)**

❖ Servlet 버전 3 지정

- web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
  <context-param>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://java.sun.com/xml/ns/javaee"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0">
```

❖ Servlet 버전 3 지정

- pom.xml

```
<!-- Servlet -->
<dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>servlet-api</artifactId>
   <version>2.5</version>
   <scope>provided</scope>
</dependency>

<dependency>
   <groupId>javax.servlet.jsp</groupId>
   <artifactId>jsp-api</artifactId>
   <version>2.1</version>
   <scope>provided</scope>
</dependency>
```

❖ Servlet 버전 3 지정

- pom.xml 수정

```
<!-- Servlet -->
<dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>javax.servlet-api</artifactId>
   <version>3.1.0</version>
   <scope>provided</scope>
</dependency>

<dependency>
   <groupId>javax.servlet.jsp</groupId>
   <artifactId>javax.servlet.jsp-api</artifactId>
   <version>2.3.1</version>
</dependency>
```

- **프로젝트 clean 필요 → rebuild**

❖ WebSocket 의존성 설정

- 스프링 웹 소켓 모듈

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

❖ WebSocket 의존성 설정

- Jackson2 모듈

```
<dependency>
   <groupId>com.fasterxml.jackson.core</groupId>
   <artifactId>jackson-core</artifactId>
   <version>2.8.11</version>
</dependency>

<dependency>
   <groupId>com.fasterxml.jackson.core</groupId>
   <artifactId>jackson-databind</artifactId>
   <version>2.5.3</version>
   <scope>runtime</scope>
</dependency>
```

❖ 웹 소켓 핸들러

- WebSocketHandler 인터페이스

```
public interface WebSocketHandler {
   // 접속 성공시 호출
   void afterConnectionEstablished(WebSocketSession session)
           throws Exception;
   // 메시지 수신시 호출
   void handleMessage(WebSocketSession session,
           WebSocketMessage<?> message) throws Exception;
   void handleTransportError(WebSocketSession session,
           Throwable exception) throws Exception;
   // 접속 해제시 호출
   void afterConnectionClosed(WebSocketSession session,
           CloseStatus closeStatus) throws Exception;

   boolean supportsPartialMessages();
}
```

❖ 웹 소켓 핸들러

- WebSocketHandler 인터페이스 구현객체
  - 텍스트 기반 통신인 경우 : TextWebSocketHandler

- TextWebSocketHandler를 상속하여 정의

❖ 웹 소켓 핸들러 : edu.iot.butter.handler 패키지

```
@Component
public class EchoHandler extends TextWebSocketHandler {

   @Override
   public void afterConnectionClosed(WebSocketSession session,
           CloseStatus status) throws Exception {
      Member member = (Member) session.getAttributes().get("USER");
      if(member != null)
         System.out.println(member.getUserId() + " 해제");
      else
         System.out.println(" 해제");

      super.afterConnectionClosed(session, status);
   }
```

❖ 웹 소켓 핸들러 : edu.iot.butter.handler 패키지

```
@Override
public void afterConnectionEstablished(WebSocketSession session)
throws Exception {
   Member member = (Member) session.getAttributes().get("USER");
   if(member != null)
      System.out.println(member.getUserId() + " 접속");
   else
      System.out.println(" 접속");

   super.afterConnectionEstablished(session);
}
```

❖ 웹 소켓 핸들러

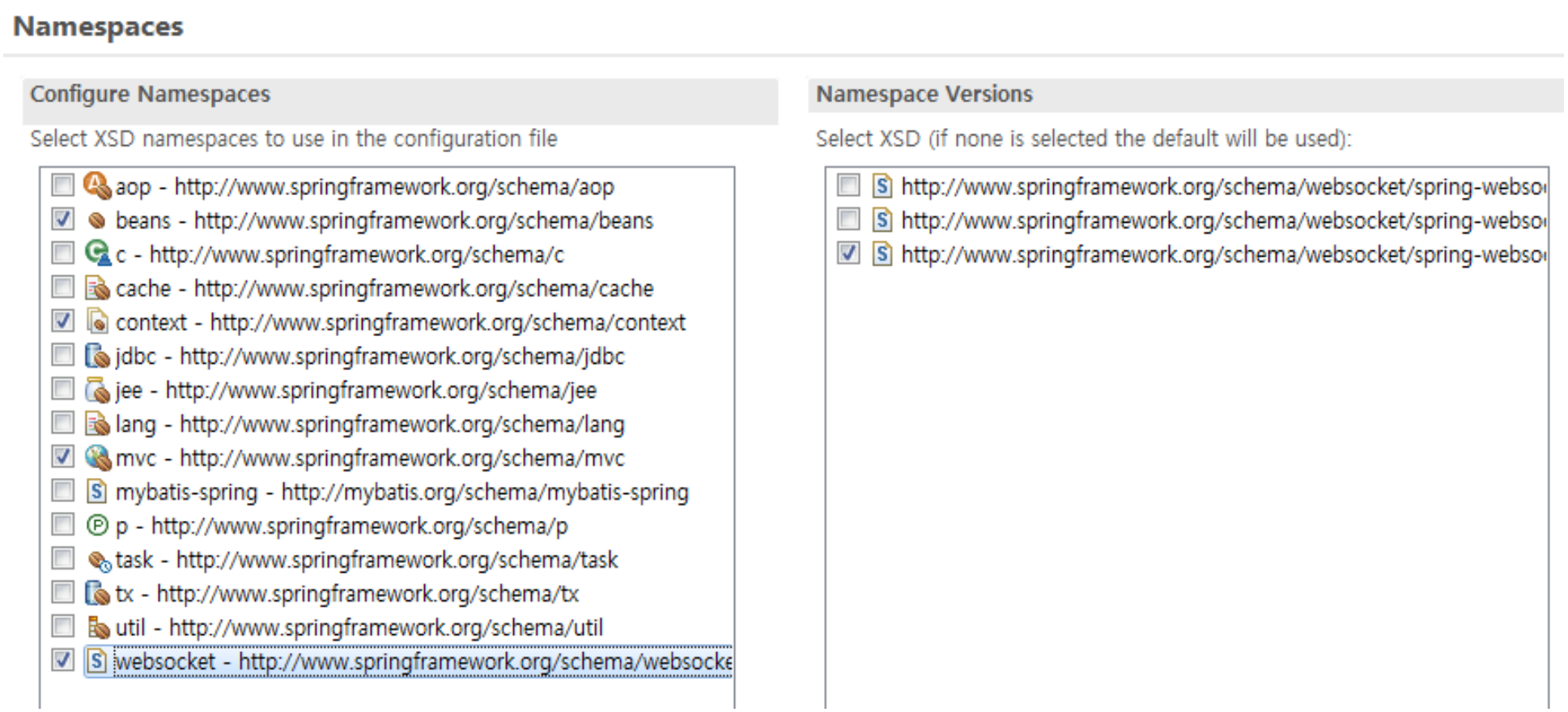
```
    @Override
    protected void handleTextMessage(WebSocketSession session,
                                     TextMessage message) throws Exception {
        System.out.println(message);

        // 메시지 전송 : echo
        session.sendMessage(message);

        super.handleTextMessage(session, message);
    }
}
```

❖ WebSocket 핸들러 설정

- servlet-context.xml
  - Websocket 네임스페이스 추가



❖ WebSocket 핸들러 설정

- servlet-context.xml
  - 핸들러 등록

```
   <websocket:handlers>
      <websocket:mapping handler="echoHandler" path="/talk/echo"/>
      <websocket:handshake-interceptors>
         <beans:bean class=
"org.springframework.web.socket.server.support.HttpSessionHandshake
Interceptor" />
      </websocket:handshake-interceptors>
      <websocket:sockjs />
   </websocket:handlers>
```

❖ 컨트롤러 : TalkController

```
@Controller
@RequestMapping("/talk")
public class TalkController {

   @RequestMapping(value="/echo", method=RequestMethod.GET)
   public void talk()  throws Exception {
   }
}
```

WebSocket
- 브라우저 -

WebSocket - 브라우저

❖ WebSocket 자바 스크립트

- WebSocket 객체(HTML5 표준 객체

```
var socket = new WebSocket("ws://localhost:8080/butter/talk/echo");
```

  - ws 프로토콜 사용
    - 서버 주소, 포트, 전대 경로 지정 불편함 있음.

- SockJS 라이브러리
  - HTML5 미지원 브라우저에서도 동작 지원
  - 스프링 WebSocket 연동 지원
  - 동일 서버에 대해서는 상대경로, 절대경로 지정 가능
  - CDN : *https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.1.4/sockjs.js*

WebSocket - 브라우저

❖ 뷰 : talk/test.jsp

```
<script
   src="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.1.4/sockjs.js">
</script>
<script>
$(function(){
   var socket = new SockJS("echo");   // "/butter/talk/echo"
});
</script>
<h1>웹 소켓 테스트</h1>
```

- 생성자 호출시 접속 시도
- 이벤트 핸들러
  - onopen : 접속 성공 시 호출
  - onmessage : 메시지 수신 시 호출
  - onclose : 접속 해제 시 호출
  - onerror : 에러 발생 시 호출
- 메시지 전송 메서드
  - send('전송 문자열')

WebSocket - 브라우저

❖ 뷰 : talk/home.jsp

```
$(function(){
   var socket = new SockJS("echo");
   socket.onopen = function() {
      console.log('접속 성공');
      // 접속후 바로 데이터 전송
      socket.send('Hello');
   }

   socket.onclose = function() {
      console.log('접속 해제');
   }

   socket.onmessage = function(msg) {
      console.log('데이터 수신 : ', msg.data);
   }

   socket.onerror = function(err) {
      console.log('에러 ', err);
   }
});
```

WebSocket - 브라우저

❖ 테스트

- 로그인
- Localhost:8080/butter/talk/echo 접속
- 브라우저/서버 콘솔 확인