# Anyframe SockJS Plugin

**Version 1.0.0**

# I.Introduction

SockJS Plugin은 Java EE 7에 새롭게 추가된 WebSocket 처리의 특징과 사용법을 설명하고 그에 기반해 스프링 4.0.0.RELEASE - - Spring Framework 4.0.0.RELEASE API [http://static.springsource.org/spring/docs/4.0.x/javadoc-api/] 에서 추가된 WebSocket 처리에 대한 활용 방법을 가이드하기 위한 샘플 코드와 이 오픈 소스를 활용하는데 필요한 가이드라인으로 구성되어있다.

**Installation**

Command 창에서 다음과 같이 명령어를 입력하여 SockJS plugin을 설치한다.

```
mvn anyframe:install -Dname=sockjs
```

본 플러그인은 JDK 7 이상, TOMCAT 7.0.47 이상, SPRING 4.0.0-RELEASE에서 동작한다. 아래의 설정을 통해서 어플리케이션을 실행하여 설치 확인을 하도록한다.

Dependent Plugins

| Plugin Name | Version Range |
|---|---|
| Core [http://dev.anyframejava.org/docs/anyframe/plugin/core/1.6.0/reference/htmlsingle/core.html] | 2.0.0 > * > 1.5.1 |

Spring SockJS 는 Spring 4.0부터 제공되는 Java EE 7의 핵심 기능인 WebSocket과 유사한 형태로 구현이 가능하며 다소 유연한 환경에서 클라이언트와 서버간의 비동기 처리가 가능한 서비스이다.

Spring SockJS를 이용하면 다음과 같은 장점을 얻을 수 있다.

- WebSocket과 친밀한 형태의 구현이 가능하다.
- javascript로 WebSocket형태의 비동기 송수신이 가능하다.(sockjs-client 제공)
- servlet 3.0의 async에 의존한다.
- 클라이언트 Web Browser의 제약이 적다.
- Fall back option을 지원하며 적절한 전송방식을 자동으로 설정할 수 있다.

# 1.Configuration

Spring Framework 4.0 버전부터 Java EE 7의 핵심기능인 WebSocket 기능을 지원한다. WebSocket 기능을 사용하기 위해서는 다음과 같은 환경이 제공되어야 한다.

## 1.1.사용환경

- JVM : Java 7+
- Servlet Container : Apache Tomcat 7.0.47+, Eclipse Jetty 9.0+, GlassFish 4.0+

## 1.2.Protocol

- SockJS : http://
- Secure SockJS : https://

## 1.3.데이터 전송방식

- WebSocket : 웹소켓
- xhr-polling : XmlHttpRequest 폴링
- xhr-streaming : XmlHttpRequest 스트리밍
- iframe-eventsource, iframe-htmlfile
- jsonp-polling : jsonp 폴링

## 1.4.Anyframe Plugin Config

- pom.xml

  Anyframe 기반으로 프로젝트를 생성하고 spring-websocket 및 Java EE API 라이브러리와 JSON 처리 시 사용된 Jackson 라이브러리를 추가한다.

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-websocket</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-web-api</artifactId>
    <version>7.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.2.3</version>
</dependency>
```

- web.xml

SockJS는 Servlet 3.0 이상의 스펙에서 동작하며 반드시 async-supported 가 지원되어야 한다.

```
<?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"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
id="WebApp_ID"
version="3.0">

<filter>
    <filter-name>encodingFilter</filter-name>
      <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
      <async-supported>true</async-supported>
      <init-param>
        <param-name>encoding</param-name>
        <param-value>utf-8</param-value>
      </init-param>
  </filter>
```

- Xml Config

  Spring context의 namespace에 spring-websocket을 선언하고 websocket handler를 등록하여 handshake url을 매핑시킨다.

  아래 등록으로 클라이언트와 서버 간의 websocket 연결을 한다.

```
<beans xmlns:websocket=http://www.springframework.org/schema/websocket
xsi:schemaLocation="http://www.springframework.org/schema/websocket
http://www.springframework.org/schema/websocket/spring-websocket-4.0.xsd">

<websocket:handlers>
    <websocket:mapping path="/getSockJSMovieList.do" handler="myHandler"/>
 <websocket:sockjs />
</websocket:handlers>
```

- Java Config

  WebSocketConfigurer 인터페이스를 구현하여 WebSocket의 handshake URL을 매핑시켜 주고, WebMvcConfigurerAdapter를 확장하여 servlet handler 및 Web MVC를 활성화 시킨다.

```
@Configuration
@EnableWebMvc
@EnableWebSocket
public class WebConfig extends WebMvcConfigurerAdapter implements WebSocketConfigurer {
    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(myHandler(), "getSockJSMovieList.do").withSockJS();
    }
    @Bean
    public WebSocketHandler myHandler() {
        return new MovieFinderWebSocketHandler();
    }
    @Override
    public void configureDefaultServletHandling(
DefaultServletHandlerConfigurer configurer) {
        configurer.enable();
    }
}
```

- Annotation 목록

| annotation | Description | | |
|---|---|---|---|
| @Configuration | Java Configuration 설정 | | |
| @EnableWebMvc | Spring MVC 사용 가능 | | |
| @EnableWebSocket | WebSocket 사용 가능 | | |

# 2.Samples

다음은 SockJS의 속성 설정 및 구현 코드에 대한 예제이다.

# 2.1.Configuration

Spring환경에서 SockJS 서비스를 사용하기 위해 정의한 sockjs-servlet.xml의 일부이다.

websocket handler를 집합을 정의 하고 handler와의 연결을 위한 path를 지정한다.

```
<websocket:handlers>
        <websocket:mapping path="/getSockJSMovieList.do" handler="sockjsMyHandler"/>
        <websocket:sockjs/>
    </websocket:handlers>

    <bean id="sockjsMyHandler"
 class="org.anyframe.sockjs.handler.MovieFinderWebSocketHandler"/>
```

# 2.2.client

클라이언트의 javascript로 WebSocket을 생성하고 서버와 메시지를 송수신한다. 생성된 WebSocket에 onopen, onmessage, onclose, onerror 등의 이벤트를 등록한다.

```
<script>
var sock = new SockJS('http://mydomain.com/_sockjs_/getSockJSMovieList.do');
  sock.onopen = function() {
    console.log('open');
};
sock.onmessage = function(e) {
    console.log('message', e.data);
  };
  sock.onclose = function() {
    console.log('close');
  };
</script>
```

# 2.3.server

WebSocketHandler를 구현하여 Text 메시지 전송 모듈인 TextWebSocketHandler와 바이너리 메시지 전송 모듈인 BinaryWebSocketHandler를 제공하며 handlerMessage 메소드에서 WebSocket메시지를 처리한다.(WebSocket과 동일)

```
public class MovieFinderWebSocketHandler extends TextWebSocketHandler {
    @Autowired
    private MovieFinder movieFinder;

    @Override
    public void handleMessage(
WebSocketSession session, WebSocketMessage<?> message)
throws Exception {

        String payloadMessage = (String) message.getPayload();
        try {
            ObjectMapper mapper = new ObjectMapper();
```

```
            Movie movie = mapper.readValue(payloadMessage, Movie.class);

            Page page = movieFinder.getPagingList(movie, pageIndex);
            StringWriter movieList = new StringWriter();
            mapper.writeValue(movieList, page);

            session.sendMessage(new TextMessage(movieList.toString()));

        } catch (JsonParseException jpe) {
            jpe.printStackTrace();
        } catch (JsonMappingException jme) {
            jme.printStackTrace();
        }
    }
}
```

# 3.Resources

## 3.1.참고자료

http://docs.spring.io/spring/docs/4.0.3.RELEASE/spring-framework-reference/htmlsingle/#websocket-fallback

## 3.2.주의 사항

### 플러그인으로 설치 후 웹어플리케이션 실행 시 주의사항

sockjs/stomp/websocket 3개의 플러그인은 최소 JDK 및 Servlet version의 기준이 다르다. (JDK : 1.6 -> 1.7, Servlet : 2.5 -> 3.0) 따라서, 이 세 가지 플러그인을 설치 후 실행 시에는, 그 전에 수작업으로 관련 내용을 변경해 주어야 한다. (물론 WAS는 해당 스펙을 지원하는 버전이어야 한다.)

Servlet 3.0 version 사용을 위해 web.xml 파일을 다음과 같이 수정한다. (타 플러그인을 위해 원래는 2.5로 지정되어 있다.)

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

sockjs/stomp/websocket 3개의 플러그인을 다시 uninstall하고 version 기준을 낮추고 싶다면, 마찬가지로 수작업을 통해 원복하도록 한다.