Anyframe STOMP Plugin

Version 1.0.0

I.Introduction

STOMP Plugin은 Java EE 7에 새롭게 추가된 메시지 푸시 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 푸시 처리에 대한 활용 방법을 가이드하기 위한 샘플 코드와 이 오픈 소스를 활용하는데 필요한 가이드라인으로 구성되어있다.

Installation

Command 창에서 다음과 같이 명령어를 입력하여 STOMP plugin을 설치한다.

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

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 STOMP 는 Spring 4.0부터 제공되는 WebSocket과 SockJS를 탑재하여 간단하게 메시징 서비스를 가능하게 한다. STOMP(Simple Text-Oriented Messaging Protol)는 텍스트기반의 메시지 프로토콜로써 스크립트언어에서도 메시지 브로커를 사용할 수 있다.

Spring STOMP를 이용하면 다음과 같은 장점을 얻을 수 있다.

- 양방향 streaming network protocol(WebSocket)에 적용가능하며 과도한 소켓 연결에도 지연현상이 적다.
- 클라이언트 사이드를 제공한다.(stomp.js)
- javascript단에서 Enterprise Message Broker와의 연결이 가능하다.(ActiveMQ, HornetQ, RabbitMQ, MorbidQ, Ruby Server …)

1.Configuration

Spring Framework 4.0 버전부터 Java EE 7 의 핵심기능인 WebSocket과 SockJS를 사용한 메시지 전송 프로토콜인 STOMP 기능을 지원한다. STOMP 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

- WebSocket : ws://, wss://
- SockJS : http://, https://

1.3.Anyframe Plugin Config

- pom.xml

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

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

- Xml Config

  Spring context의 namespace에 spring-websocket을 선언하고 websocket message-broker를 등록하여 메시지 처리를 위한 공간을 확보하며, stomp의 enpoint를 등록하여 handshake URL을 매핑시킨다.

  아래 등록으로 클라이언트와 서버 간의 websocket 연결 및 message-broker를 설정한다.

```
<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:message-broker application-destination-prefix="/app" >
 <websocket:stomp-endpoint path="/getMovieList" />
 <websocket:simple-broker prefix="/topic"/>
</websocket:message-broker>
```

- Java Config

  WebMvcConfigurerAdapter를 확장하여 servlet handler 및 Web MVC를 활성화 한다. (Servlet 3.0+ no web.xml)

```
@Configuration
@EnableWebMvc
public class WebConfig extends WebMvcConfigurerAdapter {

@Override
public void configureDefaultServletHandling(
 DefaultServletHandlerConfigurer configurer) {
  configurer.enable();
 }
}
```

  WebSocketMessageBrokerConfigurer 인터페이스를 구현하여 WebSocket의 handshake URL을 매핑시켜 주고 Message Broker의 destination을 지정한다.

```
@Configuration
@ComponentScan(
 basePackages="com.anyframe.sample.websocket",
 excludeFilters = @ComponentScan.Filter(
 type= FilterType.ANNOTATION, value = Configuration.class)
)

@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {
    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/getMovieList");
    }

@Override
public void configureMessageBroker(MessageBrokerRegistry registry) {
 registry.enableSimpleBroker("/topic/");
 registry.setApplicationDestinationPrefixes("/app");
}
}
```

- Annotation 목록

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

2.Samples

다음은 STOMP의 속성 설정 및 구현 코드에 대한 예제이다.

2.1.Configuration

Spring환경에서 STOMP 서비스를 사용하기 위해 정의한 stomp-servlet.xml의 일부이다.

websocket handler 집합을 정의 하고 handler와의 연결을 위한 path를 지정한다. Message Broker의 destination URL(prefix)을 선언한다.

```
<websocket:message-broker application-destination-prefix="/app">
  <websocket:stomp-endpoint path="/getMovieList">
   <!-- <websocket:sockjs/> -->
  </websocket:stomp-endpoint>
  <websocket:simple-broker prefix="/topic/"/>
 </websocket:message-broker>
```

2.2.client

클라이언트의 javascript로 WebSocket을 생성하고 STOMP 프로토콜에 탑재한다. Message Broker의 destination에 연결 후 메시지 전송에 대한 서버 응답을 연결된 모든 client에게 푸시 메시지로 전송한다.

```
<script type="text/javascript" src="<c:url value='/sample/javascript/stomp.js'/>" />
<script>
function connect() {
 var socket = new WebSocket("ws://" + window.location.host +
 "/anyframe-sample-stomp/_stomp_/getMovieList");
 client = Stomp.over(socket);

  client.connect({}, function(frame) {
 console.log('Connected: ' + frame);
 client.subscribe('/topic/movieList', function(message){
 showMessage(message.body);
 });
 });
}

function disconnect() {
 client.disconnect();
 console.log("Disconnected");
}

$('#btnSearch').on("click", function () {
 $('#movieGrid').clearGridData();

 client.send("/app/list", {},
 JSON.stringify({
 title : $("#title").val(),
 nowPlaying : $("#nowPlaying").val()
 }));
});
</script>
```

2.3.server

MessageMapping의 URL로 전달된 메시지를 처리하여 MessageBroker의 URL로 처리결과를 전송한다.

```
@Controller
public class MovieFinderStompController {
private static Logger log =

@Autowired
private MovieFinder movieFinder;

@MessageMapping(value="/list")
    @SendTo("/topic/movieList")
    public Page list(Movie movie) throws Exception {
int pageIndex = 1;

        Page page = movieFinder.getPagingList(movie, pageIndex);

        log.debug("The result will display after three seconds.!!");
        Thread.sleep(3000);

        return page;
    }
}
```

3.Resources

3.1.참고자료

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

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.