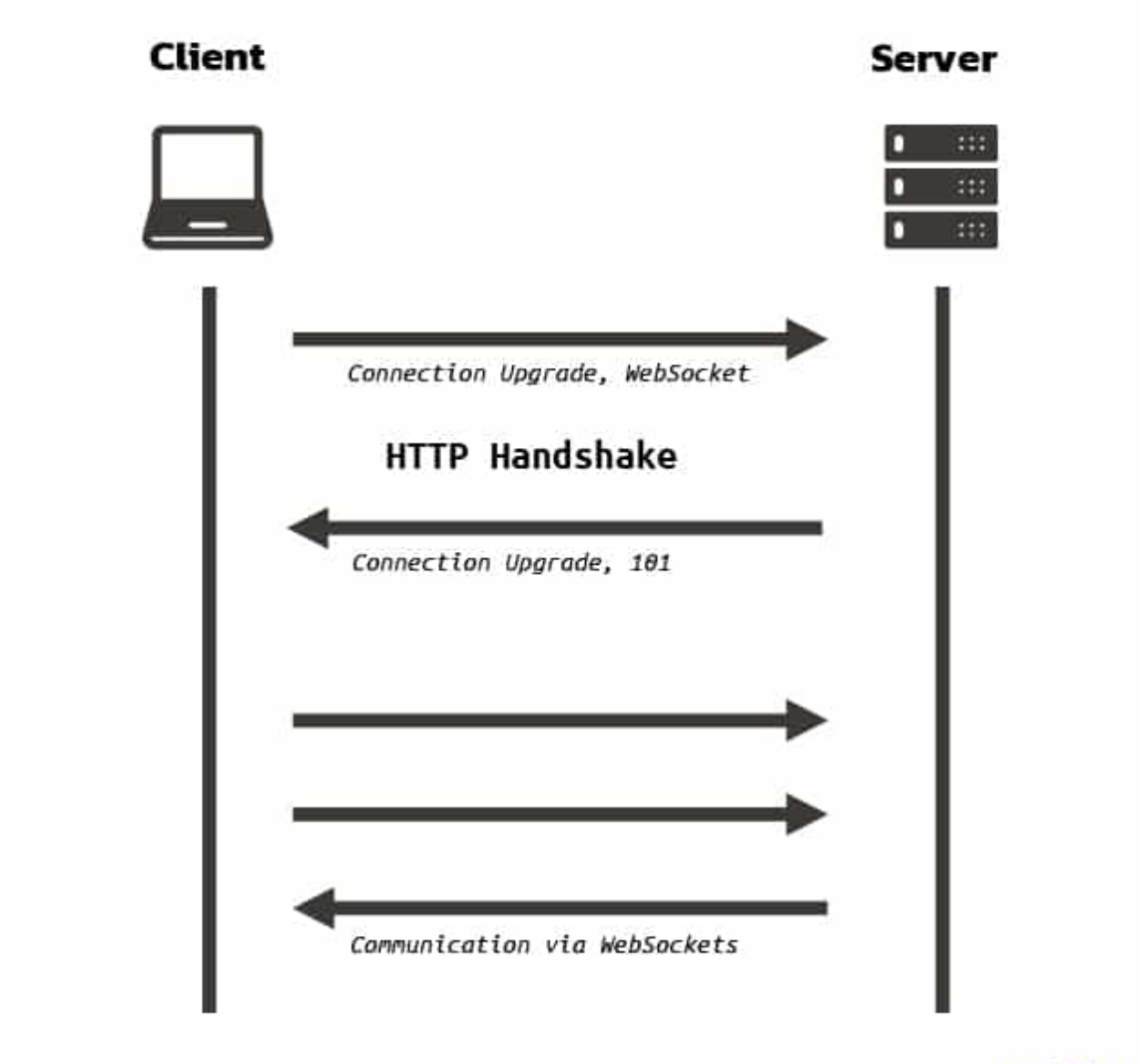
SpringBoot WebSocket (STOMP over WebSocket)

SpringBoot WebSocket



Configuration

SpringBoot WebSocket

org.springframework.web.socket.config.annotation.AbstractWebSocketMessageBrokerConfigurer

```
@Configuration
@EnableWebSocketMessageBroker
@Order(Ordered.HIGHEST_PRECEDENCE + 50)
public class WebSocketConfig extends AbstractWebSocketMessageBrokerConfigurer {

  @Bean
  @Primary
  public SimpUserRegistry userRegistry() {
    return userRegistry;
  }

  @Bean
  @Primary
  public UserDestinationResolver userDestinationResolver() {
    return resolver;
  }

  @Bean(autowire = Autowire.BY_TYPE)
  public BizChannelInterceptorAdapter bizChannelInterceptorAdapter() {
    return new BizChannelInterceptorAdapter();
  }

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

1

2

1) WebSocketMessageBroker 활성화
2) 웹소켓 Broker 설정 및 메시지 URI Prefix 설정

Interceptor

SpringBoot WebSocket

org.springframework.messaging.support.ChannelInterceptorAdapter

```
@Override
public Message<?> preSend(Message<?> message, MessageChannel channel) {
  StompHeaderAccessor accessor = MessageHeaderAccessor.getAccessor(message, StompHeaderAccessor.class);
  …
}
```

1

```
@Override
public void postSend(Message<?> message, MessageChannel channel, boolean sent) {
  if (StompCommand.CONNECT.equals(accessor.getCommand())) {

  } else if (StompCommand.SUBSCRIBE.equals(accessor.getCommand())) {

  } else if (StompCommand.UNSUBSCRIBE.equals(accessor.getCommand())) {

  } else if (StompCommand.DISCONNECT.equals(accessor.getCommand())) {

  }
  …
```

2

1) 웹소켓 채널 이벤트 preSend 처리
2) 웹소켓 채널 이벤트 postSend 처리

Controller

SpringBoot WebSocket

org.springframework.messaging.handler.annotation.MessageMapping

```
@Controller
public class ActivityWebSocketController {


@MessageMapping("/activities/add")  1
public void addActivity(SimpMessageHeaderAccessor accessor, ActivityStream activity) throws Exception {
  LOGGER.debug("User({}, {}) activity type : {}, activity recevied : {}",
              accessor.getUser().getName(), accessor.getSessionId(), activity.getType(), activity.getObject());

  activityStreamService.addActivity(activity, accessor.getUser());
}
```

1) 웹소켓을 통한 메시지 수신

SpringBoot WebSocket

Push

SpringBoot WebSocket

org.springframework.messaging.simp.SimpMessageHeaderAccessor
org.springframework.messaging.simp.SimpMessageSendingOperations

```
private SimpMessageSendingOperations messagingTemplate;


@Autowired
public void setMessagingTemplate(SimpMessageSendingOperations messagingTemplate) {
  this.messagingTemplate = messagingTemplate;
}
```

```
messagingTemplate.convertAndSend(topicUri, GlobalObjectMapper.writeValueAsString(serializableResponse));
```

1A

```
public class WebSocketUtils {

  public static void sendMessage(SimpMessageSendingOperations messagingTemplate, String user, String destination, Object payload) {
    messagingTemplate.convertAndSendToUser(user, destination, payload, createHeaders(user));
  }

  private static MessageHeaders createHeaders(String sessionId) {
    SimpMessageHeaderAccessor headerAccessor = SimpMessageHeaderAccessor.create(SimpMessageType.MESSAGE);
    headerAccessor.setSessionId(sessionId);
    headerAccessor.setLeaveMutable(true);
    return headerAccessor.getMessageHeaders();
  }

}
```

1B

1) 웹소켓을 통한 메시지 발송