



巨匠線上真人

# Java Web OCE JWCD元件系統 開發認證

[www.pcschoolonline.com.tw](http://www.pcschoolonline.com.tw)



巨匠線上真人

Java Web OCE JWCD元件系統開發認證

# 第十六堂： WebSocket 服務整合應用

# 本堂教學重點

- ◆ WebSocket通訊架構與原理介紹
- ◆ 實作WebSocket Server進行Session控制與發佈訂閱資訊
- ◆ 實作一個前端Page-設計一個私訊聊天室

# WebSocket通訊架構與原理介紹

- ◆ WebSocket是一種通訊協定，可在單個TCP連接上進行**全雙工通訊**(bi-directional)。WebSocket協定在2011年由IETF標準化為RFC 6455，後由RFC 7936補充規範。Web IDL中的WebSocket API由W3C標準化。
- ◆ WebSocket使得用戶端和伺服器之間的資料交換變得更加簡單，允許伺服器端主動向用戶端推播資料。在WebSocket API中，瀏覽器和伺服器只需要完成一次交握，兩者之間就可以建立永續性的連接，並進行雙向資料傳輸。

# WebSocket Java 定義

## ◆ WebSocket Endpoint

- ◆ WebSocket Endpoint是使對等端能夠發送的組件數據和從連接接收數據。在JSR 356中，Java Web套接字端點是表示序列的一側的組件連接對等體之間的交互 Web Socket端點有兩種狀態：連接或斷開連接。

## ◆ WebSocket Connection

- ◆ WebSocket連接是一個完整的通信路徑已事先就協議握手達成一致的同行。連接是作為兩個WebSocket端點之間的網絡維護，直到其中一個應用程序或Web瀏覽器運行時關閉或強制切斷

## ◆ WebSocket Peer

- ◆ WebSocket Peer表示正在參與的應用程序通過WebSocket端點進行通信。

## ◆ WebSocket Session

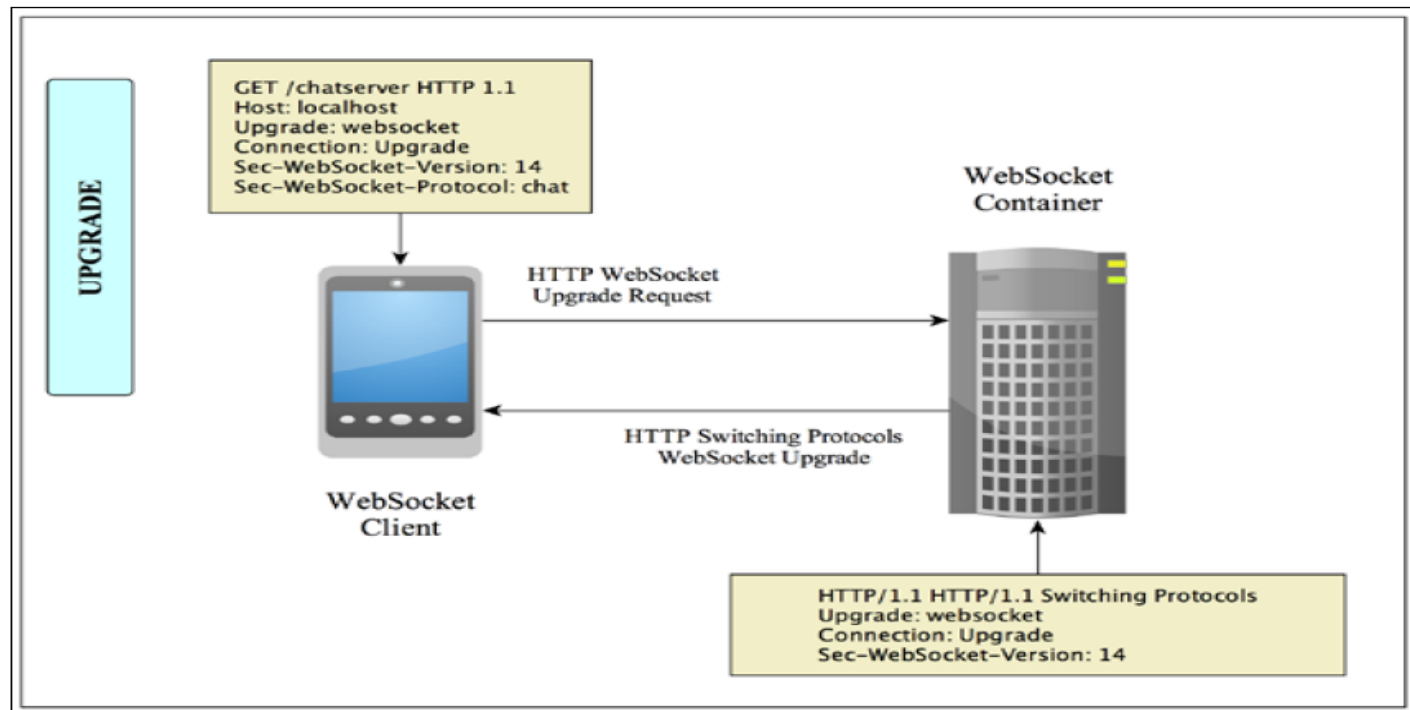
- ◆ WebSocket用語中的WebSocket會話是代表集隨著時間的推移，數據通信由兩個獨立的兩個對等方共享端點。Session是識別這些序列的渠道兩個同行之間的溝通

## ◆ Server WebSocket Endpoint

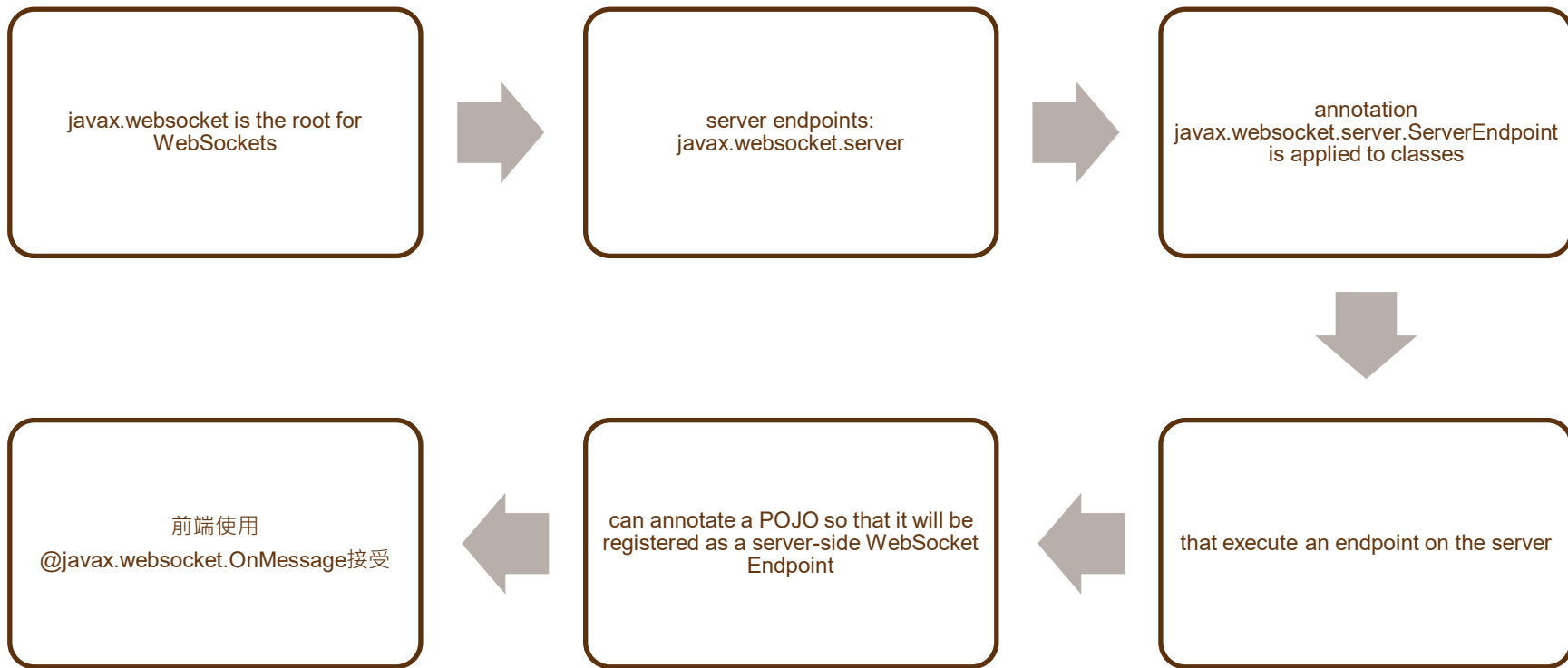
- ◆ 服務器WebSocket端點是接受來自的啟動請求的對等體客戶端 - 遠程端點 - 連接到其WebSocket。實施將提供WebSocket或在有連接請求時創建一個。服務器幫助握手協議並激活之間的Session客戶端和服務器端點，建立連接。服務器端點不會啟動其他連接或充當客戶端。

## ◆ Client WebSocket Endpoint

- ◆ 客戶端WebSocket端點是創建WebSocket（或從實現中檢索一個池化的WebSocket並調用一個遠程端點上的連接請求以啟動WebSocket會話。當遠程端點接受連接請求時客戶端綁定到WebSocket會話。客戶不接受來自其他遠程端點的連接或充當服務器。



# Java WebSockets Server API



# 實作WebSocket Server 進行Session控制與發佈訂閱資訊



- ◆ 建立一個Hello World WebSocket Server
- ◆ 使用@ServerEndpoint Annotation 定義端點
- ◆ 使用@OnMessage Annotation 撰寫訊息處理

```
package com.gjun.service;

import javax.websocket.OnMessage;

@ServerEndpoint("/gjun")
public class HelloService {

    @OnMessage
    public String helloWorld(String clientRequestMessage)
    {
        return String.format("%s Hello World!! 世界和平!!", clientRequestMessage);
    }
}
```



# 使用Simple WS Client測試

- ◆ 使用Google Chrome安裝Simple WS Client工具。
- ◆ 進行WebSocket Server測試。
- ◆ <https://chrome.google.com/webstore/detail/simple-websocket-client/pfdhoblngboilpfeibdedpjgfnlcodoo?hl=zh-TW>

The screenshot displays the Simple WS Client interface with three main sections:

- Server Location**: Contains a text input field for the URL, an "Open" button, and a status indicator showing "Status: CLOSED".
- Request**: Features a large text area for entering the request, a "Send" button, and a shortcut key indicator "[Shortcut] Ctr + Enter".
- Message Log**: Includes a "Clear" button and a large area for displaying the message log.

# 測試WebSocket End Point

◆ ws://localhost:8080/mod16/gjun

服務端點

```
@ServerEndpoint("/gjun")  
public class HelloService {  
  
    @OnMessage  
    public String helloWorld(String clientRequestMessage)  
    {  
        return String.format("%s Hello World!! 世界和平!!", clientRequestMessage);  
    }  
}
```

The image shows two screenshots of a WebSocket client interface. The top screenshot shows the initial state: the URL is 'ws://localhost:8080/mod16/gjun', the status is 'OPENED', and the request field is empty. The bottom screenshot shows the state after sending a message: the request field contains '張三丰', and the message log displays '張三丰 Hello World!! 世界和平!!'.

**Server Location**  
URL: ws://localhost:8080/mod16/gjun  
Status: OPENED  
Close

**Request**  
張三丰  
Send [Shortcut] Ctr + Enter

**Message Log** Clear

**Server Location**  
URL: ws://localhost:8080/mod16/gjun  
Status: OPENED  
Close

**Request**  
張三丰  
Send [Shortcut] Ctr + Enter

**Message Log** Clear  
張三丰  
張三丰 Hello World!! 世界和平!!

# 使用前端網站網頁-JavaScript測試

socketclient.html

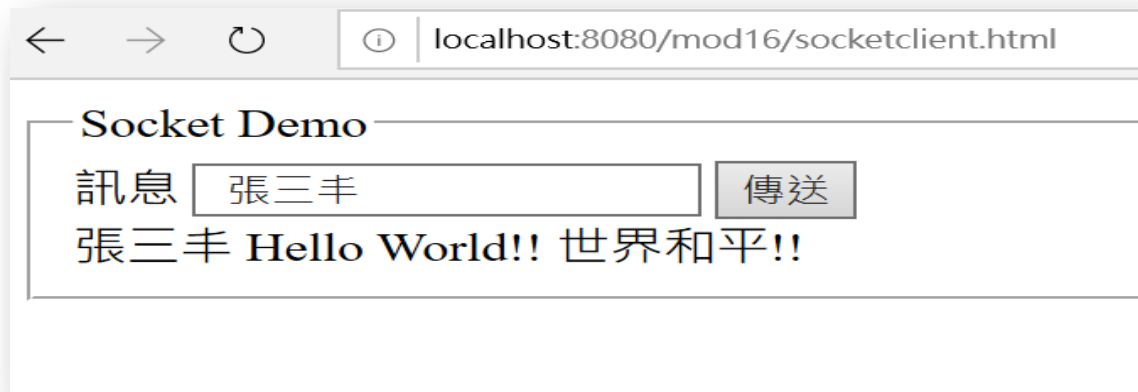
- ◆ 撰寫靜態頁面
- ◆ 使用CDN連接網路上的jQuery Framework。
- ◆ 使用jQuery選擇器綁定事件程序與DOM，進行訂閱WebSocket應用

```
<body>
  <fieldset>
    <legend>Socket Demo</legend>
    <label>訊息</label>
    <input type="text" id="message"/>
    <input type="button" value="傳送" id="btnGo"/>

    <div id="result"></div>
  </fieldset>
</body>
```

```
<title>WebSocket Server簡易測試</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>
<script>
  $(document).ready(
    function(){
      uri="ws://localhost:8080/mod16/gjun";
      //選擇按鈕與文字輸入方塊
      $('#btnGo').click(
        function()
        {
          //建構WebSocket Client
          var ws=new WebSocket(uri);
          //埋事件進行callback
          ws.onopen=function(evt)
          {
            //alert('Opened');
            //送出訊息
            var msg=$('#message').val();
            ws.send(msg);
          }
          ws.onmessage=function(evt){
            //alert(evt.data);
            //UI
            $('#result').text(evt.data);
          }
        }
      )
    }
  )
}
```

# Client Demo



A screenshot of a web browser window. The address bar shows 'localhost:8080/mod16/socketclient.html'. The page title is 'Socket Demo'. Below the title, there is a label '訊息' followed by a text input field containing '張三丰'. To the right of the input field is a button labeled '傳送'. Below the input field, the text '張三丰 Hello World!! 世界和平!!' is displayed.

← → ↺ ⓘ localhost:8080/mod16/socketclient.html

Socket Demo

訊息

張三丰 Hello World!! 世界和平!!

# 實作一個前端Page-設計一個私訊聊天室

規劃聊天內容的  
JavaBean



設計WebSocket

## ◆ JavaBean訊息類別設定

聊天對象的識別碼  
(私訓對象)

聊天訊息

```
package com.gjun.entity;

//JavaBean聊天訊息
public class ChatMessage implements java.io.Serializable {
    private String id;
    private String message;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

# 設計WebSocket

- ◆ 使用End Point Path傳遞使用者識別碼。
- ◆ 設定Map集合收集連接進來的Session。
- ◆ 使用@OnOpen/@OnClose/@OnMessage Annotation規劃一個聊天室處理程序

ChatRoomService.java

```
import com.gjun.entity.ChatMessage;

@ServerEndpoint("/room/{id}/rawdata")
public class ChatRoomService {

    //使用static共用的集合物件 收集名稱對應一個WebSocket Session
    //收集前端Open建立起的Session
    private static Map<String,Session> clients=new HashMap<String,Session>();
    public ChatRoomService(){}

    }

    //聆聽使用端建立起WebSocket連接
    @OnOpen
    public void connected(@PathParam("id")String id,Session session){
        clients.put(id,session); //收集Session

        System.out.println(clients.size());
    }

    @OnClose
    public void connectionClose(@PathParam("id")String id,Session session){
        clients.remove(id, session);
        System.out.println("前端:"+clients.size());
    }
}
```

# 專屬ID/Session訊息回饋

@OnMessage

```
public void messageReceived(@PathParam("id")String id,String jsonMessage,Session session){
    //建構Gson序列化與反序列Json物件
    Gson gson=new Gson();
    //反序列化前端傳遞進來的Json字串
    ChatMessage message=gson.fromJson(jsonMessage, ChatMessage.class);
    //取出聊天對象識別碼
    String who=message.getId();
    //走訪集合中相對的識別碼
    for(String key:clients.keySet()){
        if(key.equals(who))
        {

            //取出應對前端的Session
            Session curSession=clients.get(key);
            try {
                //透過Session送出聊天文字串到前端
                curSession.getBasicRemote().sendText(id+"#"+message.getMessage());
            } catch (IOException ex) {
                Logger.getLogger(ChatRoomService.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

# 前端測試聊天室

**Server Location**

URL:

Status: **OPENED**

**Request**

[Shortcut] Ctr + Enter

**Message Log**

Linda#Eric 您好

**Server Location**

URL:

Status: **OPENED**

**Request**

```
{"id":"Eric","message":"Eric 您好"}
```

[Shortcut] Ctr + Enter

**Message Log**

```
{"id":"Eric","message":"Eric 您好"}
```



# Lab

- ◆ WebSocket通訊架構與原理為何
- ◆ 設計一個私訊聊天室

