
Java EE 7 Web WebSocket

鄭安翔

ansel_cheng@hotmail.com

課程大綱

1) **WebSocket**

WebSocket 需求

■ WebSocket 需求

- 市場需要能夠即時傳遞資料且永遠連線的技術
- 典型應用場景
 - 股市行情、通訊軟體、即時轉播
- HTTP 是無狀態、短連接、用完即關的通訊協定
- 網站推撥需採用輪詢技術
 - 由瀏覽器定時向伺服器發出請求查詢資料
 - 占用頻寬資源，效能差
 - 無法真正即時

WebSocket 通訊架構及標準

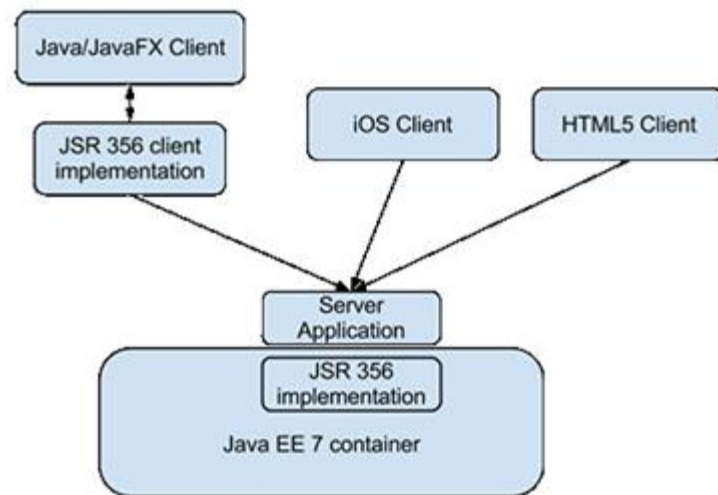
■ WebSocket 通訊架構及標準

- 2011年 IETF 訂定 RFC 6455 標準
- 2016年 RFC 7936 補充規範
- 在單一TCP連接上進行全雙工通訊(bi-directional)
 - 應用層通訊協定，可進行雙向資料傳輸
 - 伺服器端可以主動向客戶端推播資料
- 透過 HTTP 建立 WebSocket 連線
 - 瀏覽器和伺服器完成一次交握後，建立永續性的連接
 - 通訊通過TCP80或443埠完成
 - 不受防火牆阻止非網路連接的限制

WebSocket 通訊標準

■ 伺服器端支援

- JSR 356 WebSocket Java API
- JavaEE7標準的一部分
- Tomcat 8 開始支援



■ 瀏覽器端支援

- WebSocket JS API
- 支援的瀏覽器

	IE	Chrome	Safari	Opera	Firefox	Android Browser
支援	10.0	14.0~29.0	6.0	12.1~16.0	6.0~23.0	X
部分支援	X	4.0~13.0	5.0~5.1	11.0~12.0	4.0~5.0	X
不支援	5.0~9.0	X	3.1~4.0	9.0~10.6	2.0~3.6	2.1~4.2

- Google Chrome、Firefox、Safari、Microsoft Edge、Internet Explorer、Opera

WebSocket 優點

■ WebSocket 優點

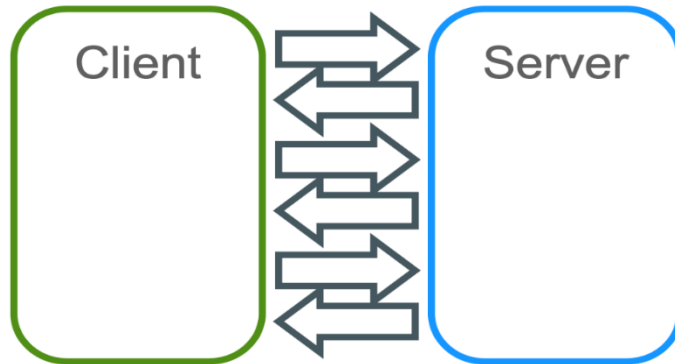
- 雙向通信
- 低延遲
- 通信成本小

■ 適用情境

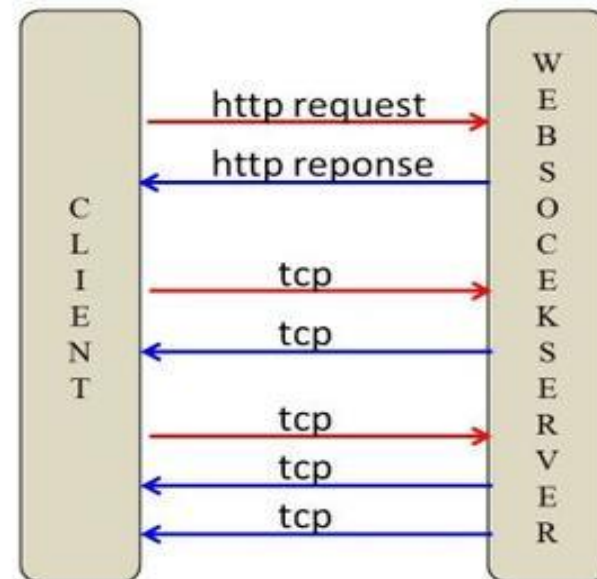
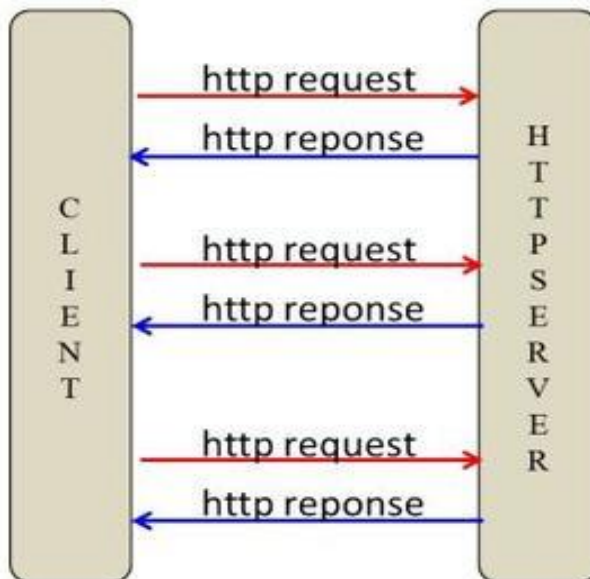
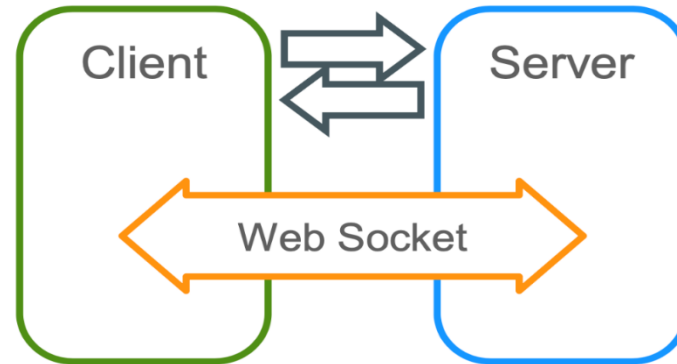
- 需要發送大量相對較小的訊息的網路應用程式
 - 線上遊戲或股票市場報價廣播
- WebSocket參考實現
 - Tyrus(泰魯斯) <https://tyrus-project.github.io/>

WebSocket 運作

HTTP Poll

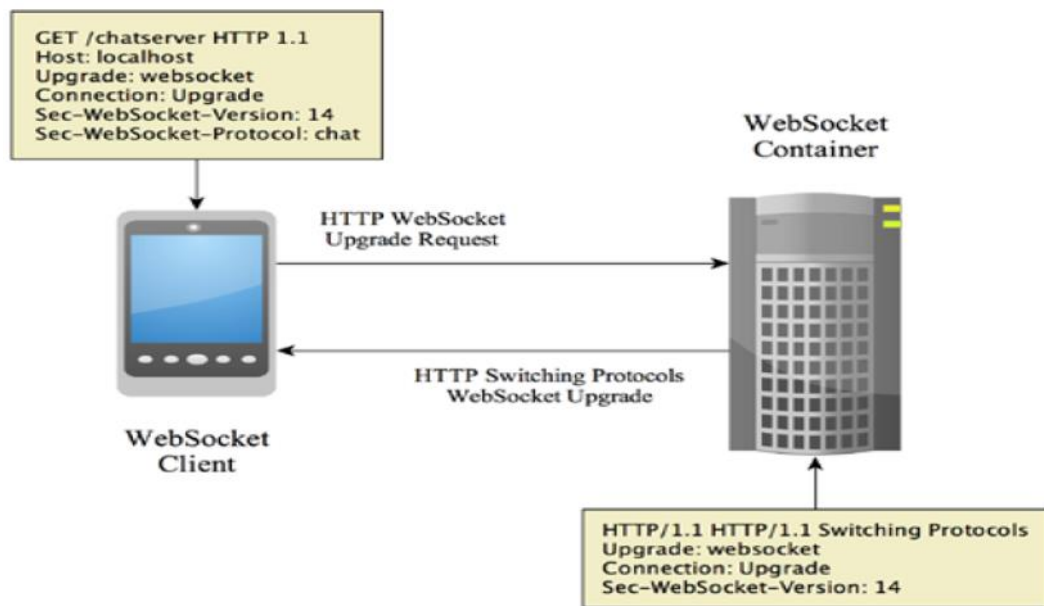


WebSocket Push



WebSocket 協定交握

- 透過HTTP協定建立TCP連接通道
- 在TCP連接上雙向傳輸資料



WebSocket 應用程式

■ WebSocket 應用程式

□ Server

- 發佈 WebSocket Endpoint 提供連線 URIs
- `ws://127.0.0.1:8080/WebSocket/websocket`
- `wss://127.0.0.1:8080/WebSocket/websocket`

□ Client

- 透過 Endpoint URIs 向 Server 建立連線需求

□ WebSocket 在完成建立連線後雙方為對稱狀態

- 任何一方皆可自由接收與傳遞訊息，並可以任意關閉連線。

Java WebSocket 應用程式

■ WebSocket Endpoint

- 伺服器與瀏覽器兩端發送及接收資料的物件
- 連接或斷開兩種狀態

■ WebSocket Connection

- 經協議交握兩個端點之間的通信網路
- 直到其中伺服器端或瀏覽器端關閉或強制切斷連線

■ WebSocket Session

- 用來識別相互溝通管道的集合
- 獨立的對等共享端點交換資訊對話

WebSocket Server 端開發

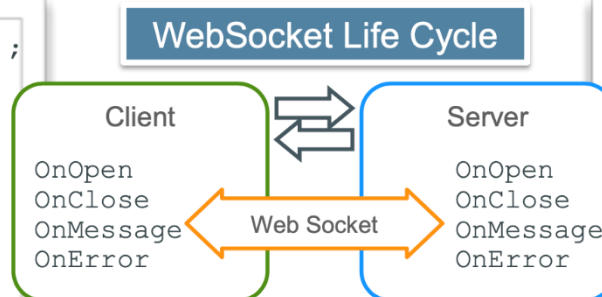
- WebSocket Server 端開發兩種方法
 - 類別使用 @ Annotation
 - @ServerEndpoint("/websocket")
 - @ServerEndpoint("/customer/{id}")
 - 類別繼承 Endpoint
 - ServerEndpointConfig.Builder.create(xxx.class, "/websocket").build();
 - 發佈 WebSocket Endpoint URIs
 - ws://127.0.0.1:8080/xxx/websocket
 - ws://127.0.0.1:8080/xxx/customer/sean

WebSocket Server 端開發

■ Annotation 模式

註解	描述
@ServerEndpoint	如果使用修飾 @ServerEndpoint，則該容器可確保該類作為偵聽特定URI空間的WebSocket服務器的可用性
@ClientEndpoint	用此註釋裝飾的類被視為WebSocket客戶端
@OnOpen	@OnOpen 啟動新的WebSocket連接時，對Java方法進行註釋，以供容器調用該方法
@OnMessage	將 @OnMessage 消息發送到端點時，用註釋的Java方法從WebSocket容器接收
@OnError	裝飾一個方法，@OnError 以便在WebSocket通信出現問題時調用該方法
@OnClose	用於裝飾您要在WebSocket連接關閉時由容器調用的Java方法

```
var socket = new WebSocket("uri");
socket.onopen
    = function(event){...}
socket.onclose
    = function(event){...}
socket.onmessage
    = function(event){...}
socket.onerror
    = function(event){...}
```



```
@ServerEndpoint("uri")
public class ServerHandler {
    @OnOpen
    public void openSocket...
    @OnClose
    public void closeSocket...
    @OnMessage
    public void handleMessage...
    @OnError
    public void handleError...
}
```

WebSocket Server 連線建立

■ 標註連線建立

□ ws://127.0.0.1:8080/xxx/websocket

@Open

```
public void onOpen(Session session) {  
    ....  
}
```

□ ws://127.0.0.1:8080/xxx/customer/Sean

@Open

```
public void onOpen(Session session,  
    @PathParam("id") String id) {  
    ....  
}
```

WebSocket Server 訊息處理

■ 標註訊息處理

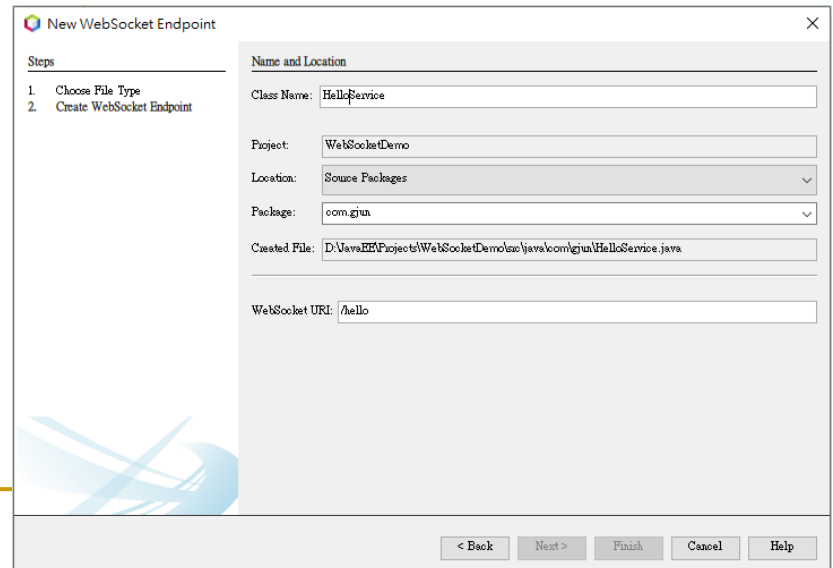
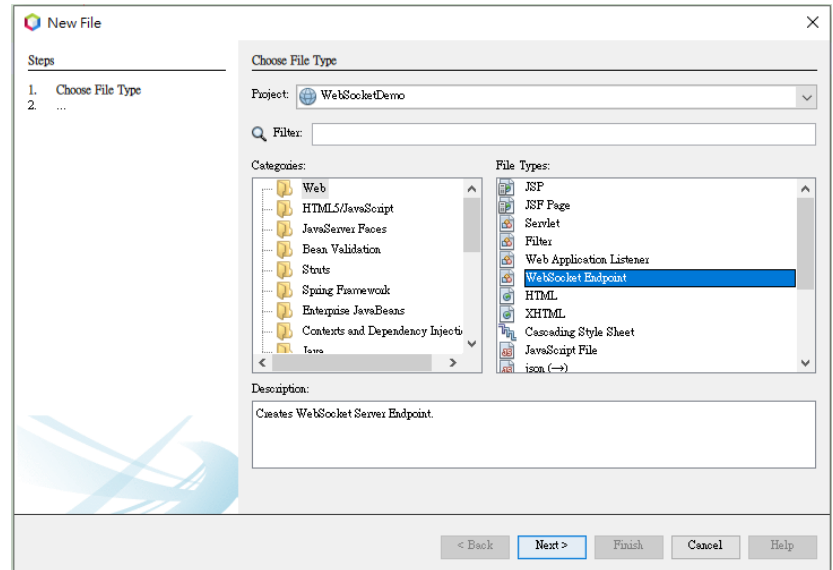
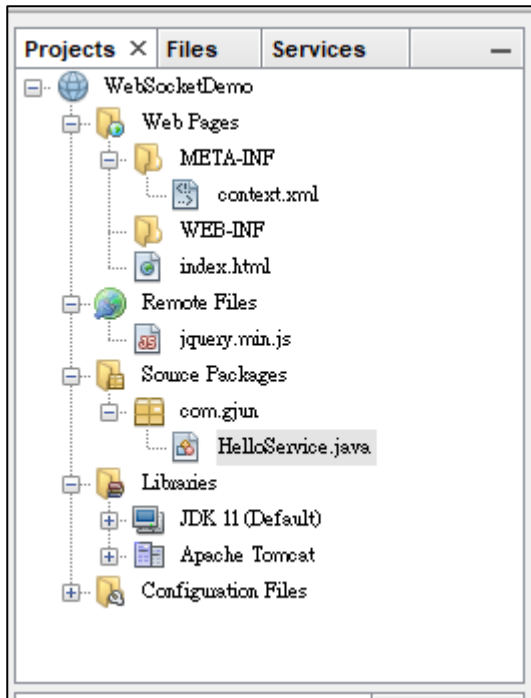
ws://127.0.0.1:8080/xxx/websocket

```
@OnMessage
public void onOpen(String message) {
    ....
}
```

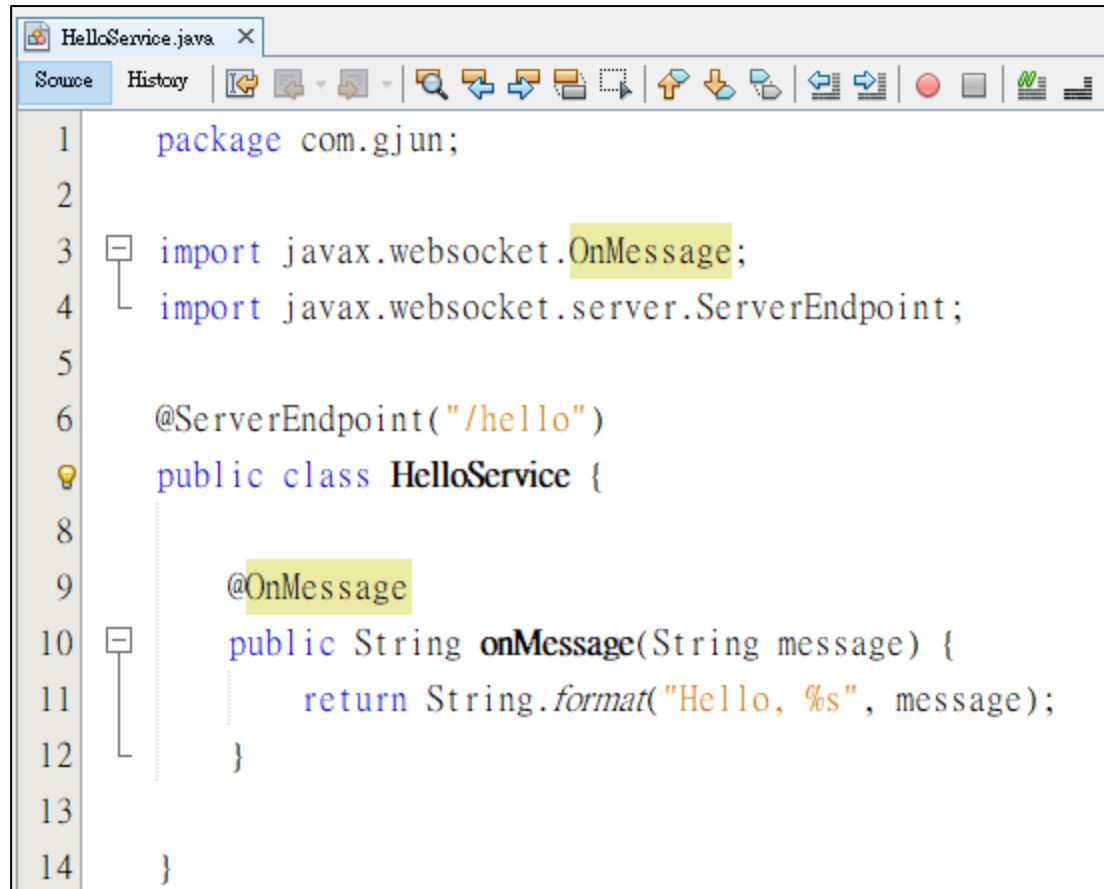
ws://127.0.0.1:8080/xxx/customer/Sean

```
@OnMessage
public void onMessage(@PathParam("id") String id,
                      String message) {
    ....
}
```

WebSocket End Point



WebSocket End Point



The screenshot shows a code editor window titled "HelloService.java". The code is as follows:

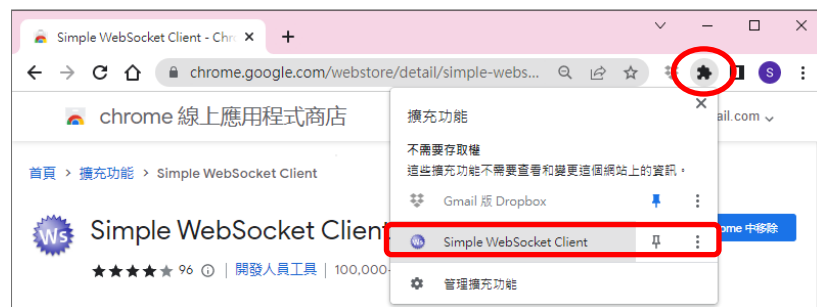
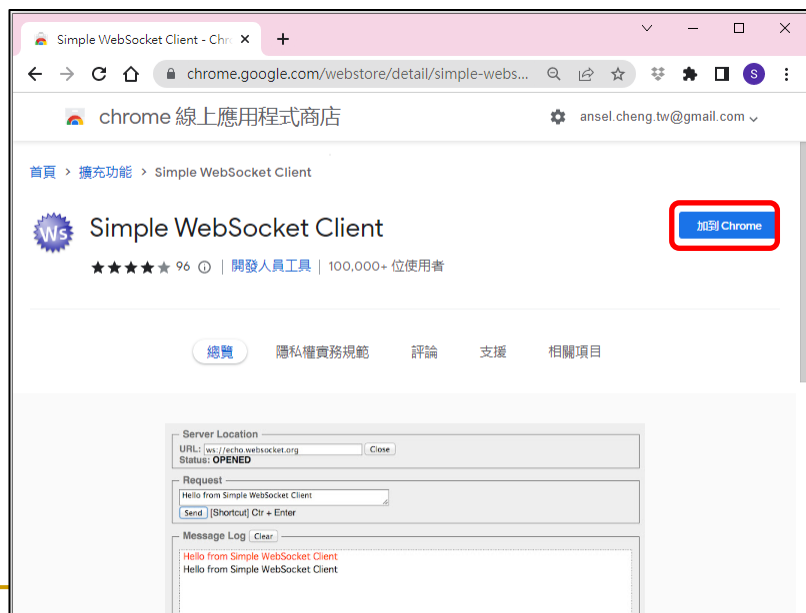
```
1 package com.gjun;
2
3 import javax.websocket.OnMessage;
4 import javax.websocket.server.ServerEndpoint;
5
6 @ServerEndpoint("/hello")
7 public class HelloService {
8
9     @OnMessage
10    public String onMessage(String message) {
11        return String.format("Hello, %s", message);
12    }
13
14 }
```

The code defines a `ServerEndpoint` for the path `/hello`. The `onMessage` method is annotated with `@OnMessage` and returns a formatted string "Hello, %s" where `%s` is replaced by the received `message`.

測試 WebSocket Server

■ 測試 WebSocket Server

- ❑ 安裝 Google Chrome Simple WS Client 工具
- ❑ <https://chrome.google.com/webstore/detail/simple-websocket-client/pfdhoblngboilpfeibdedpjgfnlcodoo?hl=zh-TW>



Simple WS Client 測試

■ Simple WS Client

□ ws://localhost:8080/WebSocketDemo/hello

```
@ServerEndpoint("/hello")  
public class HelloService {  
  
    @OnMessage  
    public String onMessage(String message) {  
        return String.format("Hello, %s", message);  
    }  
}
```

Simple Web Socket Client

Server Location
URL:
Status: **CLOSED**

Request

 [Shortcut] Ctr + Enter

Message Log

Simple Web Socket Client

Server Location
URL:
Status: **OPENED**

Request

 [Shortcut] Ctr + Enter

Message Log

Simple Web Socket Client

Server Location
URL:
Status: **OPENED**

Request

 [Shortcut] Ctr + Enter

Message Log

Sean
Hello, Sean

WebSocket Client 端開發

■ WebSocket JS API

- ❑ <https://developer.mozilla.org/en-US/docs/Web/API/WebSocket/>

- ❑ 建立 WebSocket 連線

- ❑ 偵測 WebSocket 的各種事件

- ❑ onopen：連線成功時觸發

- ❑ onmessage：接收到來自 Server 的訊息時觸發

- ❑ onerror：連線錯誤時觸發

- ❑ onclose：中斷連線時觸發

- ❑ send()：發送訊息

```
var websocket = new WebSocket(url);
```

```
websocket.onopen = function (event) {  
    ...  
    websocket.send(yyy);  
};
```

```
websocket.onmessage = function (event) {  
    ...  
};
```

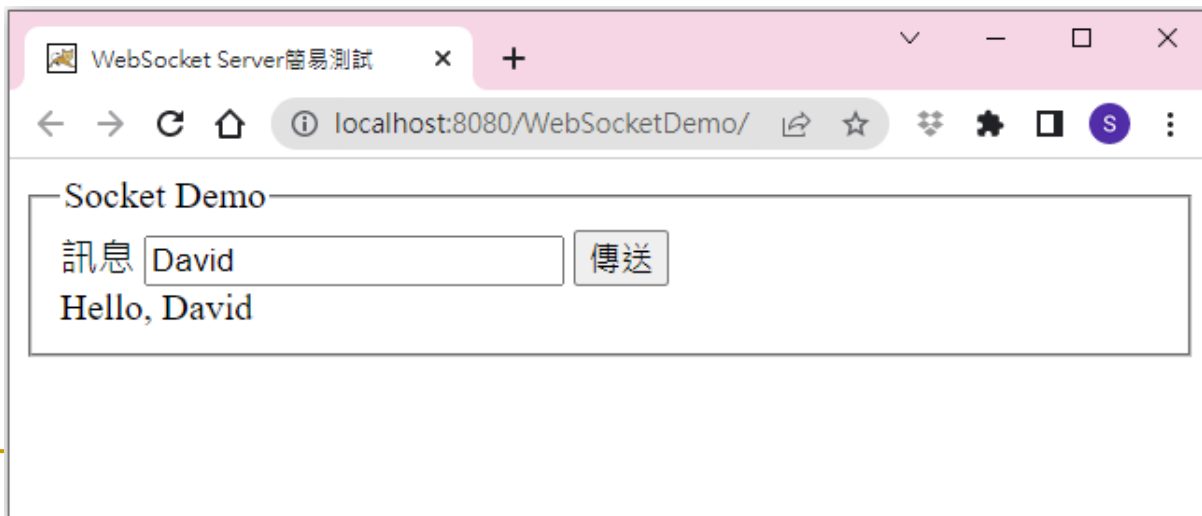
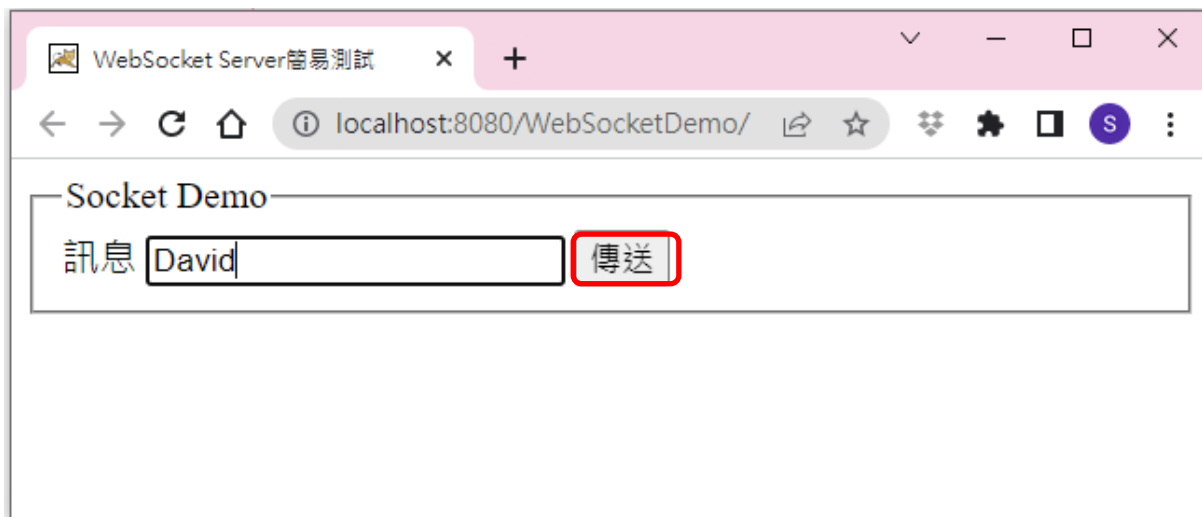
```
websocket.onerror = function (event) {  
    ....  
};
```

```
websocket.onclose = function (event) {  
    ....  
};
```

使用網站網頁-JavaScript測試

```
index.html x
Source History
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <title>WebSocket Server簡易測試</title>
6     <meta charset="UTF-8">
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>
9     <script>
10       $(document).ready(
11         function(){
12           uri="ws://localhost:8080/WebSocketDemo/hello";
13           //選擇按鈕與文字輸入方塊
14           $('#btn').click(
15             function() {
16               //建構WebSocket Client
17               var ws=new WebSocket(uri);
18               //處理事件進行callback
19               ws.onopen=function(evt) {
20                 //alert('Opened'); //送出訊息
21                 var msg=$('#message').val();
22                 ws.send(msg);
23               }
24               ws.onmessage=function(evt){
25                 //alert(evt.data); //UI
26                 $('#result').text(evt.data);
27               } //選擇文字輸入方塊
28             }
29           );
30         }
31       );
32     </script>
33
34   </head>
35   <body>
36     <fieldset>
37       <legend>Socket Demo</legend>
38       <label>訊息</label>
39       <input type="text" id="message" />
40       <input type="button" value="傳送" id="btn" />
41       <div id="result"></div>
42     </fieldset>
43   </body>
44 </html>
45
46
```

使用網站網頁-JavaScript測試

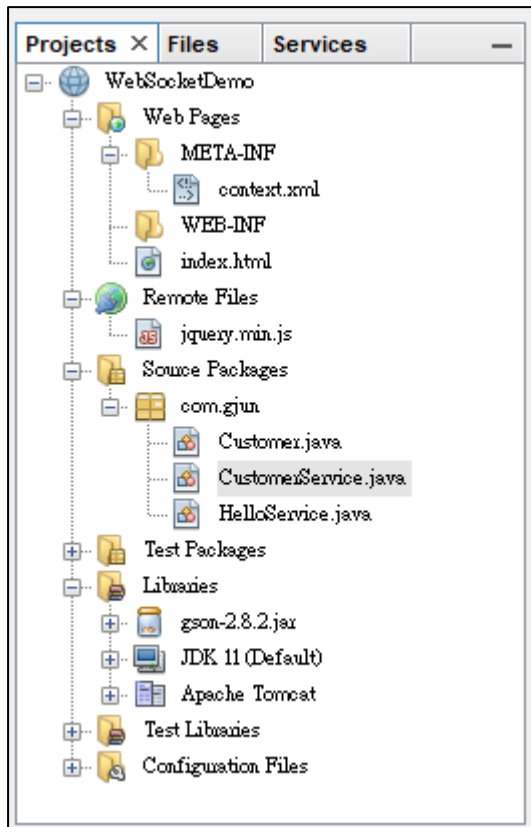


WebSocket Server 進階

- 使用End Point Path識別使用者
 - `ws://127.0.0.1:8080/xxx/customer/sean`
- WebSocket 物件傳送
 - 後端使用 Java Bean 實體
 - 前端以 JSON 文字格式表達
 - `com.google.gson.Gson` 套件序列化與反序列

常用方法	傳回值	說明
<code>fromJson(String json, Type typeOfT)</code>	<code><T> T</code>	將JSON 字串的資料注入至對應的POJO物件 Type為POJO的類別型態
<code>toJson(Object src)</code>	String	將POJO物件內容產生對應的JSON 字串

Java Bean



```
Customex.java
Source History
package com.gjun;

public class Customer implements java.io.Serializable{
    private String id;
    private String name;
    private String address;

    public String getId() {
        return id;
    }

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

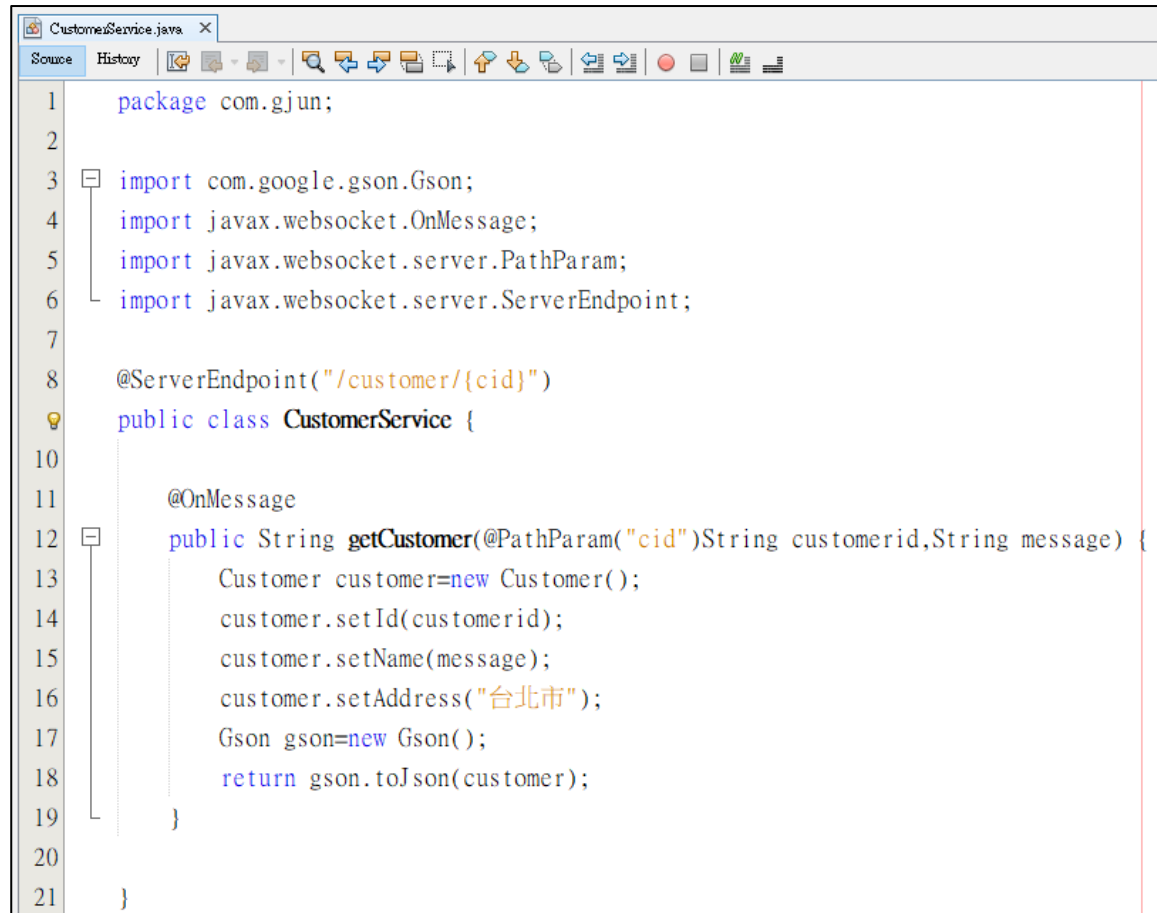
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
}
```

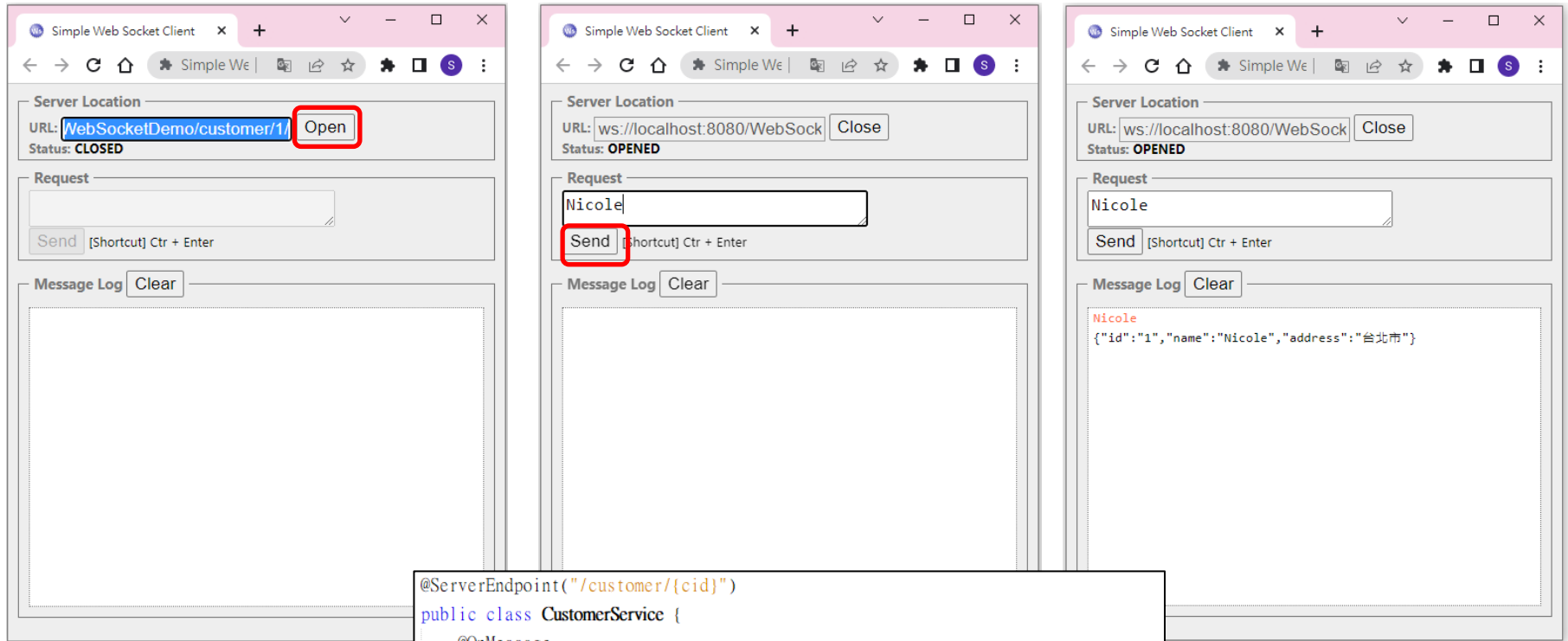
CustomerService



```
1 package com.gjun;
2
3 import com.google.gson.Gson;
4 import javax.websocket.OnMessage;
5 import javax.websocket.server.PathParam;
6 import javax.websocket.server.ServerEndpoint;
7
8 @ServerEndpoint("/customer/{cid}")
9 public class CustomerService {
10
11     @OnMessage
12     public String getCustomer(@PathParam("cid")String customerid,String message) {
13         Customer customer=new Customer();
14         customer.setId(customerid);
15         customer.setName(message);
16         customer.setAddress("台北市");
17         Gson gson=new Gson();
18         return gson.toJson(customer);
19     }
20
21 }
```


WebSocket 物件傳送測試

ws://localhost:8080/WebSocketDemo/customer/1/



```
@ServerEndpoint("/customer/{cid}")  
public class CustomerService {  
    @OnMessage  
    public String getCustomer(@PathParam("cid")String customerid,String message){  
        Customer customer=new Customer();  
        customer.setId(customerid);  
        customer.setName(message);  
        customer.setAddress("台北市");  
        Gson gson=new Gson();  
        return gson.toJson(customer);  
    }  
}
```

WebSocket Server 進階

- 使用End Point Path識別使用者
 - ws://127.0.0.1:8080/xxx/customer/sean
- javax.websocket.Session
 - 容器儲存
 - CopyOnWriteArraySet<Session> 進行並行讀取時不需加鎖
 - Map<key,Session>

Session 方法	傳回值	說明
getBasicRemote()	RemoteEndpoint.Basic	同步WebSocket端點，blocking現象
getAsyncRemote()	RemoteEndpoint.Async	非同步WebSocket端點，non-blocking
getId()	String	傳回Session ID 字串
getOpenSessions()	Set<Session>	取得WebSocket相同端點的開啟連線

RemoteEndpoint.Basic/Async	傳回值	說明
sendText(String text)	Future<Void>	同步/非同步傳送文字訊息

Lab

■ Chat 網路應用專案

- 使用 WebSocket API 製作線上聊天室
- 建立 `com.gjun.ChatMessage` Java Bean
 - 記錄聊天內容
 - `id / message` 屬性
 - `getter / setter` 方法
- 建立 `com.gjun.ChatService`
 - `WebSocket` 伺服器端服務
 - `@ServerEndpoint("/chatRoom/{id}")`
 - `CopyOnWriteArraySet<Session> sessions` 儲存 `Session`

Lab

- @OnOpen方法
 - 將session 儲存至 sessions
 - 對每個連接的Client傳送訊息：*系統 : <id> 加入聊天室*
- @OnClose方法
 - 將session 從 sessions中移除
 - 對每個連接的Client傳送訊息：*系統 : <id> 離開聊天室*
- @OnMessage方法
 - 建構Gson序列化與反序列Json物件
 - 反序列化前端傳遞進來的Json字串
 - 對每個連接的Client傳送訊息：*<id> : <message>*

Lab

- ❑ 複製 char.js 至專案
 - 檢視 char.js
- ❑ 修改 index.html 如右圖
 - 引入java script 檔案 char.js
- 測試、執行

