網頁實時聊天之PHP實現websocket

2018-11-26 254

分類導航

HTML / CSS

JavaScript

服務端

資料庫

行動端

原文地址：https://www.cnblogs.com/zhenbianshu/p/6111257.html

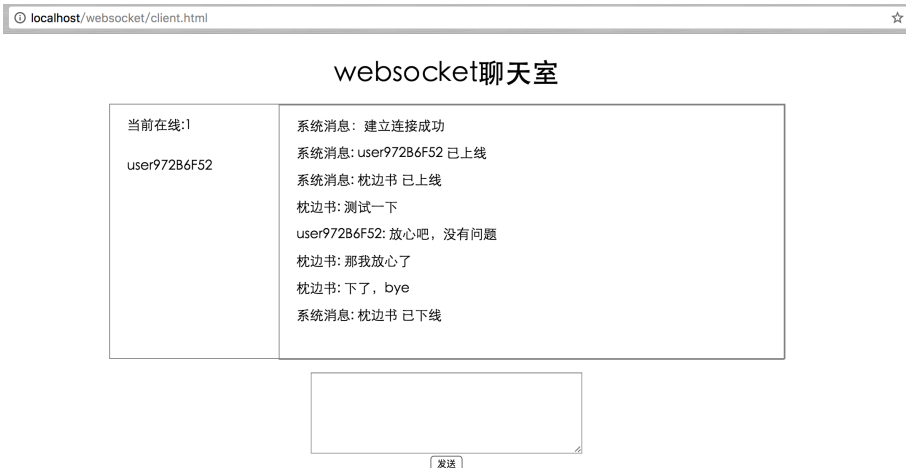
前言

websocket 作為 HTML5 裡一個新的特性一直很受人關注，因為它真的非常酷，打破了 http “請求-響應”的常規思維，實現了伺服器向客戶端主動推送訊息，本文介紹如何使用 PHP 和 JS 應用 websocket 實現一個網頁實時聊天室；

以前寫過一篇文章講述如何使用ajax長輪詢實現網頁實時聊天，見連結： 網頁實時聊天之js和jQuery實現ajax長輪詢 ，但是輪詢和伺服器的 pending 都是無謂的消耗，websocket 才是新的趨勢。

最近艱難地“擠”出了一點時間，完善了很早之前做的 websocket “請求-原樣返回”伺服器，用js完善了下客戶端功能，把過程和思路分享給大家，順便也普及一下 websocket 相關的知識，當然現在討論 websocket 的文章也特別多，有些理論性的東西我也就略過了，給出參考文章供大家選擇閱讀。

正文開始前，先貼一張聊天室的效果圖（請不要在意CSS渣的頁面）：



然後當然是原始碼： 我是原始碼連結 - github - 枕邊書

---

websocket

簡介

WebSocket 不是一門技術，而是一種全新的協議。它應用 TCP 的 Socket（套接字），為網路應用定義了一個新的重要的能力：客戶端和伺服器端的雙全工傳輸和雙向通訊。是繼 Java applets、 XMLHttpRequest、 Adobe Flash,、ActiveXObject、 各類 Comet 技術之後，伺服器推送客戶端訊息的新趨勢。

與http的關係

在網路分層上，websocket 與 http 協議都是應用層的協議，它們都是基於 tcp 傳輸層的，但是 websocket 在建立連線時，是借用 http 的 101 switch protocol 來達到協議轉換（Upgrade）的，從 HTTP 協議切換成 WebSocket 通訊協議,這個動作協議中稱“握手”；

握手成功後，websocket 就使用自己的協議規定的方式進行通訊，跟 http 就沒有關係了。

握手

以下是一個我自己的瀏覽器傳送的典型的握手 http 頭：

```
handshake buffer:
Cache-Control: no-cache
Upgrade: websocket
Origin: http://localhost
Sec-WebSocket-Version: 13
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.98 Safari/537.36
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Sec-WebSocket-Key: V5CbPOtZalZYat9mE+YHow==
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
```

伺服器收到握手請求後，提取出請求頭中的 “Sec-WebSocket-Key” 欄位，追回一個固定的字串 ‘258EAFA5-E914-47DA-95CA-C5AB0DC85B11’， 然後進行 sha1 加密，最後轉換為 base64 編碼，作為 key 以 “Sec-WebSocket-Accept” 欄位返回給客戶端，客戶端匹配此 key 後，便建立了連線，完成了握手；

資料傳輸

websocket 有自己規定的資料傳輸格式，稱為 幀（Frame），下圖是一個數據幀的結構，其中單位為bit：

```
  0                   1                   2
3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
0 1
 +-+-+-+-+-------+-+-------------+-------------------------
---+
 |F|R|R|R| opcode|M| Payload len |    Extended payload length
|
 |I|S|S|S|  (4)  |A|     (7)     |             (16/64)
|
 |N|V|V|V|       |S|             |   (if payload len==126/127)
|
 | |1|2|3|       |K|             |
|
 +-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - -
- +
 |     Extended payload length continued, if payload len == 12
7  |
 + - - - - - - - - - - - - - - - +-------------------------
---+
 |                               |Masking-key, if MASK set to
1  |
 +-------------------------------+-------------------------
---+
 | Masking-key (continued)       |          Payload Data
|
 +-------------------------------- - - - - - - - - - - - - -
- +
 :                     Payload Data continued ...
:
 + - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- +
 |                     Payload Data continued ...
|
 +---------------------------------------------------------
---+
```

具體每個欄位是什麼意思，有興趣的可以看一下這篇文章 The WebSocket Protocol 5.資料幀 感覺自己對二進位制的操作還不是很靈活，也就沒有挑戰自己寫演算法解析資料了，下面的資料幀解析和封裝都是使用的網上的演算法。

不過，我工作中寫支付閘道器中還是會經常用到資料的進位制操作的，這個一定是要仔細研究總結一下的，嗯，先記下。

PHP 實現 websocket 伺服器

PHP 實現 websocket 的話，主要是應用 PHP 的 socket 函式庫：

PHP 的 socket 函式庫跟 C 語言的 socket 函式非常類似，以前翻過一遍 APUE, 所以覺得還挺好理解。在 PHP 手冊中看一遍 socket 函式，我想大家也能對 php 的 socket 程式設計有一定的認識。

下面會在程式碼中對所用函式進行簡單的註釋。

檔案描述符

忽然提及'檔案描述符'，大家可能會有些奇怪。

但作為伺服器，是必須要對已經連線的 socket 進行儲存和識別的。每一個 socket 代表一個使用者，如何關聯和查詢使用者資訊與 socket 的對應就是一個問題了，這裡便應用了關於檔案描述符的一點小技巧。

我們知道 linux 是'萬物皆檔案'的，C 語言的 socket 的實現便是一個個的'檔案描述符‘，這個檔案描述符一般是開啟檔案的順序遞增的 int 數值，從 0 一直遞增（當然系統是有限制的）。每一個 socket 都對應一個檔案，讀寫 socket 都是操作對應的檔案，所以也能像檔案系統一樣應用 read 和 write 函式。

tips: linux 中， 標準輸入對應的是檔案描述符 0；標準輸出對應的檔案描述符是 1； 標準錯誤對應的檔案描述符是 2；所以我們可以使用 0 1 2對輸入輸出重定向。

那麼類似於 C socket 的 PHP socket 自然也繼承了這一點，它建立的 socket 也是型別於 int 值為 4 5 之類的資源型別。 我們可以使用 (int) 或 intval() 函式把 socket 轉換為一個唯一的ID，從而可以實現用一個 '類索引陣列‘ 來儲存 socket 資源和對應的使用者資訊；

結果類似：

```
$connected_sockets = array(
    (int)$socket => array(
        'resource' => $socket,
        'name' => $name,
        'ip' => $ip,
        'port' => $port,
        ...
    )
)
```

建立伺服器socket

下面是一段建立伺服器 socket 的程式碼：

```
// 建立一個 TCP socket, 此函式的可選值在官方文件中寫得十分詳細，這裡不再提了
$this->master = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
// 設定IP和埠重用,在重啟伺服器後能重新使用此埠;
socket_set_option($this->master, SOL_SOCKET, SO_REUSEADDR, 1);
// 將IP和埠繫結在伺服器socket上;
socket_bind($this->master, $host, $port);
// listen函式使主動連線套介面變為被連線套介面，使得此 socket 能被其他 socket 訪問，從而實現伺服器功能。後面的引數則是自定義的待處理socket的最大數目，併發高的情況下，這個值可以設定大一點，雖然它也受系統環境的約束。
socket_listen($this->master, self::LISTEN_SOCKET_NUM);
```

這樣，我們就得到一個伺服器 socket，當有客戶端連線到此 socket 上時，它將改變狀態為可讀，那就看接下來伺服器的處理邏輯了。

伺服器邏輯

這裡著重講一下 socket_select($read, $write, $except, $tv_sec [, $tv_usec]):

select 函式使用傳統的 select 模型，可讀、寫、異常的 socket 會被分別放入 $socket, $write, $except 陣列中,然後返回 狀態改變的 socket 的數目，如果發生了錯誤，函式將會返回 false.

需要注意的是最後兩個時間引數，它們只有單位不同，可以搭配使用，用來表示 socket_select 阻塞的時長，為0時此函式立即返回，可以用於輪詢機制。 為 NULL 時，函式會一直阻塞下去, 這裡我們置 $tv_sec 引數為null，讓它一直阻塞，直到有可操作的 socket 返回。

下面是伺服器的主要邏輯:

```
$write = $except = NULL;
$sockets = array_column($this->sockets, 'resource'); // 獲取到全部的 socket 資源
$read_num = socket_select($sockets, $write, $except, NULL);

foreach ($sockets as $socket) {
        // 如果可讀的是伺服器 socket, 則處理連線邏輯;
        if ($socket == $this->master) {
            socket_accept($this->master);
            // socket_accept() 接受 請求 "正在 listen 的 socket（像我們的伺服器 socket ）" 的連線, 並一個客戶端 socket, 錯誤時返回 false;
             self::connect($client);
             continue;
            }
        // 如果可讀的是其他已連線 socket ,則讀取其資料,並處理應答邏輯
        } else {
```

```
            // 函式 socket_recv() 從 socket 中接受長度為 len 位元組的資料，並儲存在 $buffer 中。
            $bytes = @socket_recv($socket, $buffer, 2048, 0);

            if ($bytes < 9) {
                // 當客戶端忽然中斷時，伺服器會接收到一個 8 位元組長度的訊息（由於其資料幀機制，8位元組的訊息我們認為它是客戶端異常中斷訊息），伺服器處理下線邏輯，並將其封裝為訊息廣播出去
                $recv_msg = $this->disconnect($socket);
            } else {
                // 如果此客戶端還未握手，執行握手邏輯
                if (!$this->sockets[(int)$socket]['handshake']) {
                    self::handShake($socket, $buffer);
                    continue;
                } else {
                    $recv_msg = self::parse($buffer);
                }
            }

            // 廣播訊息
            $this->broadcast($msg);
        }
    }
}
```

這裡只是伺服器處理訊息的基礎程式碼，日誌記錄和異常處理都略過了，而且還有些資料幀解析和封裝的方法，各位也不一定看愛，有興趣的可以去 github 上支援一下我的原始碼~~

此外，為了便於伺服器與客戶端的互動，我自己定義了 json 型別的訊息格式，形似：

```
$msg = [
    'type' => $msg_type, // 有普通訊息，上下線訊息，伺服器訊息
    'from' => $msg_resource, // 訊息來源
    'content' => $msg_content, // 訊息內容
    'user_list' => $uname_list, // 便於同步當前線上人數與姓名
    ];
```

客戶端

建立客戶端

前端我們使用 js 呼叫 Websocket 方法很簡單就能建立一個 websocket 連線，伺服器會為幫我們完成連線、握手的操作，js 使用事件機制來處理瀏覽器與伺服器的互動：

```
// 建立一個 websocket 連線
var ws = new WebSocket("ws://127.0.0.1:8080");
```

```
// websocket 建立成功事件
ws.onopen = function () {
};

// websocket 接收到訊息事件
ws.onmessage = function (e) {
    var msg = JSON.parse(e.data);
}

// websocket 錯誤事件
ws.onerror = function () {
};
```

傳送訊息也很簡單，直接呼叫 `ws.send(msg)` 方法就行了。

頁面功能

頁面部分主要是讓使用者使用起來方便，這裡給訊息框 textarea 添加了一個鍵盤監控事件，當用戶按下回車鍵時直接傳送訊息；

```
function confirm(event) {
    var key_num = event.keyCode;
    if (13 == key_num) {
        send();
    } else {
        return false;
    }
}
```

還有使用者開啟客戶端時生成一個預設唯一使用者名稱；

然後是一些對資料的解析構造，對客戶端頁面的更新，這裡就不再囉嗦了，感興趣的可以看原始碼。

使用者名稱非同步處理

這裡不得不提一下使用者登陸時確定使用者名稱時的一個小問題，我原來是想在客戶端建立一個連線後直接傳送使用者名稱到伺服器，可是控制檯裡報出了 “websocket 仍在連線中或已關閉” 的錯誤資訊。

Uncaught DOMException: Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.

考慮到連線可能還沒處理好，我就實現了 sleep 方法等了一秒再發送使用者名稱，可是錯誤仍然存在。

後來忽然想到 js 的單執行緒阻塞機制，才明白使用 sleep 一直阻塞也是沒有用的，利用好 js 的事件機制才是正道：於是在伺服器端新增邏輯，在握手成功後，向客戶端傳送握手已成功的訊息；客戶

端先將使用者名稱存入一個全域性變數，接收到伺服器的握手成功的提醒訊息後再發送使用者名稱，於是成功在第一時間更新使用者名稱。

小結

聊天室擴充套件方向

簡易聊天室已經完成，當然還要給它帶有希望的美好未來，希望有人去實現：

頁面美化（資訊新增顏色等）

伺服器識別 '@' 字元而只向某一個 socket 寫資料實現聊天室的私聊；

多程序（使用 redis 等快取資料庫來實現資源的共享），可參考我以前的一篇文章： 初探PHP多程序

訊息記錄資料庫持久化（log 日誌還是不方便分析）

...

總結

多讀些經典書籍還是很有用的，有些東西真的是觸類旁通，APUE/UNP 還是要再多翻幾遍。此外網際網路技術日新月異，挑一些自己喜歡的學習一下，跟大家分享一下也是挺舒服的（雖然程式和部落格加一塊用了至少10個小時...）。

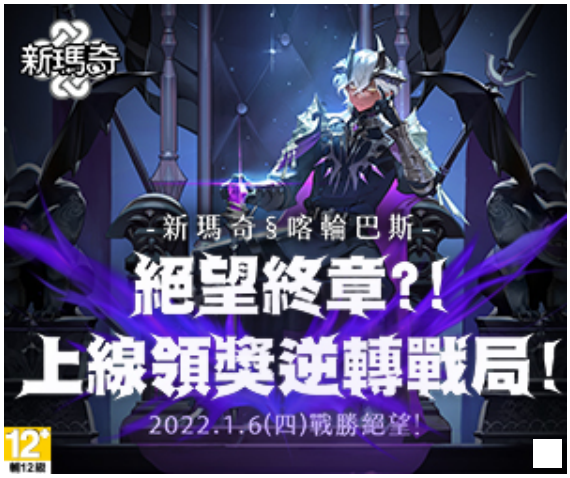
參考：

websocket協議翻譯

刨根問底 HTTP 和 WebSocket 協議（下）

學習WebSocket協議—從頂層到底層的實現原理（修訂版）

嗯，持續更新。喜歡的可以點個推薦或關注，有錯漏之處，請指正，謝謝。



相關文章

- 網頁實時聊天之PHP實現websocket
- 使用PHP+Swoole實現的網頁即時聊天工具：PHPWebIM（轉）
- 網頁實時聊天之js和jQuery實現ajax長輪詢
- 三大框架之SpringMVC ：一個小的登入註冊專案的網頁實現
- Odoo產品分析 (四) -- 工具板塊(11) -- 網站即時聊天(1)
- android------引導頁兩種實現方式（原生和WebView網頁實現）
- 移動端網頁實現撥打電話功能的幾種方法
- 微信掃碼登入網頁實現原理
- 分頁查詢 原理以及網頁實現固定頁碼數 搜尋的實現
- c#模擬網頁實現12306登陸、自動刷票、自動搶票完全篇
- Android WebView載入網頁，實現前進 後退 重新整理 超連結
- Android WebView載入網頁，實現前進、後退、重新整理、超連結
- 用程式來控制一個網頁，實現自動輸入等操作
- 通過CDN引用jQuery庫+jQuery的使用+網頁實現計算器的功能
- HTML5的靜態網頁實現（一）