

嵌入式系統實驗

實驗一報告

聊天室

電機三 童 寬 B03901039

電機三 李元顯 B03901093

教授：鄭振牟

繳交日期：2017/3/31

壹、功能

1. 即時訊息

此為聊天室的最基礎功能。所有已註冊的使用者登入後，便可選取「公共聊天室」並發送訊息至所有其它已登入的使用者。訊息發出後，所有使用者的聊天室皆會即時接收並顯示於「公共聊天室」的訊息區。


2. 使用者認證

當聊天室開啟時，伺服器將提示使用者輸入帳號名稱以及密碼，並點選「登入」或「註冊」。已事先註冊者輸入正確的帳號密碼後便可「登入」至訊息區開始傳送與接收訊息，第一次使用者則可選擇自訂的帳號密碼並「註冊」以使用服務。

當使用者在輸入錯誤的帳號密碼下點選「登入」，或以已被使用的帳號名稱進行「註冊」時，伺服器皆會發出錯誤訊息提示使用者重試。

3. 記錄登入歷史

當使用者在同一個瀏覽器下登入後，聊天室將儲存其登入狀態（即使用者名稱和密碼）；如此一來，若使用者將瀏覽器關閉或使用重新整理功能後，將不須再重新輸入帳號密碼即可繼續使用聊天室。

如需切換帳號或是登出聊天室，則可以點選訊息區右下角的登出按鈕（），便會返回2.所提到的使用者認證畫面。

4. 上線名單

成功登入聊天室後，訊息區的右側會列出伺服器上已經註冊的所有使用者名單，包括自己以及並未上線的成員。其中，自己的使用者名稱將以粗體顯示並以括號加註。此外，使用者亦可從這個動態名單判斷目前有哪些成員正在使用聊天室：離線的成員為灰色之斜字體、上線的成員則以黑色之正常字體標示，並且會實時更新。

5. 私密訊息

伺服器提供兩種傳訊方式：其一如1.所述，可在「公共聊天室」中接受並傳送所有已註冊的使用者皆看得到的訊息、其二則是選定單一使用者對象的「私密聊天室」，所傳送的訊息只有對方可以閱讀。

如欲使用私密聊天功能，需先於訊息區右側的使用者名單點選所欲傳送的對象，訊息區便會切換至與該使用者之間的私密聊天室。如想切換回公共聊天室，則需點選訊息區右上角的「ALL」。

6. 離線訊息

除了可以即時和正在線上的會員傳送與接收訊息外，使用者也可以將訊息留在「公共聊天室」或任意的「私密聊天室」中，當下未在線上的會員可以在稍後登入時前往相應的

聊天室查看離線期間他人所留下的訊息。

如欲向特定的離線使用者傳送私密離線訊息，可先從右側的使用者名單中點選相應的灰色斜線體帳號名後再行傳送。

7. 新訊息通知

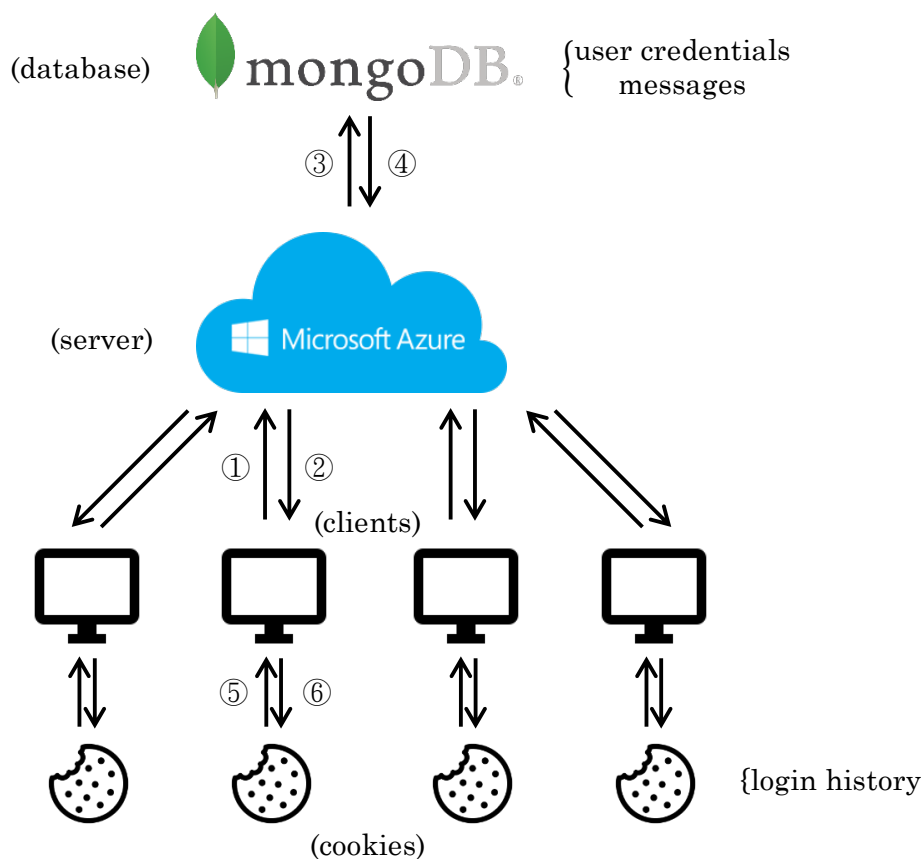
如 6. 所述，若使用者在離線期間曾收到他人傳送的私密訊息，則會在登入後收到通知：在右側的使用者名單中，有新訊息的私密聊天室使用者名稱左方將出現一個小燈泡圖示（💡），點選該聊天室並成功讀取離線訊息後，小燈泡便會消失。

此外，若使用者在上線期間收到來自他人的私密訊息，但目前並未開啟該聊天室時，瀏覽器將發出系統通知並告知新訊息寄送人及訊息內容；當使用者點選該通知後，瀏覽器會將訊息區自動切換至接收到新訊息的私密聊天室。（此功能需先允許 Google Chrome 或 Microsoft Edge 的系統通知功能，目前暫未支援其它瀏覽器）

8. 查看訊息發送來源與時間

在各個聊天室中，所有接收的訊息左側皆會標註傳送者的使用者名稱，並以不同顯著色彩便利視覺上的區分。此外，每則訊息的傳送時間皆會被記錄，若需查看則可將滑鼠移至該訊息上方，文字頂部便會出現資訊框顯示該訊息發送的日期及時間。

貳、架構



如上頁圖所示，本聊天室的架構大致上可分為三層：用戶端（client）、伺服器（server）以及資料庫（database），而用戶端本身亦與瀏覽器內的 cookie 功能互動。其中，我們將伺服器部署於微軟提供的 Microsoft Azure 雲端服務平台，負責處理用戶端的要求以及作為資料庫與用戶端之間的資料傳輸中介站。我們使用的資料庫服務為 MongoDB，並負責儲存所有已註冊使用者的帳號名稱和密碼，以及在聊天室中被傳出的所有訊息之內容、發送時間和寄送人與收件人。此外，使用者在用戶端進入聊天室後，為避免需要時常重新登入的繁瑣，我們將已登入的記錄儲存於瀏覽器的 cookie 中，並於使用者登出後清除。

以下，我們將舉數個使用本聊天室時將遇到的情境為例，並依其解說上述的架構實際上會如何運作。下頁的架構圖中已將共六條箭頭標號①至⑥，分別代表六種伺服器、用戶端與資料庫互動的方式；我們將於接下來的事例說明中以代號方式註明聊天室運作的過程。為方便起見，我們假設共有五位使用者 A、B、C、D、E 正要使用本聊天室。

1. A 使用者首次使用欲註冊帳戶

當 A 進入網站後，用戶端將先自瀏覽器的 cookie 提取資料，並檢查 A 是否有先前登入過的記錄（⑤）。以本例而言由於 A 為首次使用沒有記錄，登入頁面便會出現並提示使用者輸入帳號名稱與密碼。

A 將輸入一組新的帳號名稱與密碼並點選註冊（Register）按鈕。此時，用戶端將輸入的資料傳送給伺服器（①），再由伺服器向資料庫聲索（④）已存在的使用者名單並由伺服器檢查帳號名稱是否已被使用：若否，則代表成功註冊，伺服器會將註冊資訊送至資料庫儲存（③），再用 socket.io 的函數 join 建立一個屬於 A 的 room，並告知用戶端成功登入令其進入聊天室（②），同時將 A 在用戶端已登入的資訊存入其瀏覽器的 cookie 中（⑥）以避免後續重複登入的需要。反之，則通知用戶端發出錯誤訊息並拒絕註冊。送至資料庫的使用者註冊資訊則將以下列 JSON 格式儲存：

```
1. {  
2.   username: 'A',  
3.   password: '*****',  
4. };
```

而我們同樣將登入資訊以下述方法存入 cookie：

```
1. document.cookie = 'loggedIn=${1};path=/';  
2. document.cookie = 'userName=${A};path=/';  
3. document.cookie = 'userPass=${*****};path=/';
```

其中，**loggedIn** 儲存 0 或 1 的布林值代表是否曾經成功登入、**userName** 儲存帳號名稱、**userPass** 儲存密碼。

在儲存密碼時，我們使用了加鹽的雜湊函數（salted hash）將其加密，並以 `sha512(password, salt)` 作為處理加密的函數：

```

1. const sha512 = (password, salt) => {
2.   const hash = crypto.createHmac('sha512', salt); /** Hashing algorithm sha512 */
3.   hash.update(password);
4.   const value = hash.digest('hex');
5.   return {
6.     salt,
7.     passwordHash: value,
8.   };
9. };

```

2. A 使用者非首次使用欲登入帳戶

在此例中我們假設 A 使用了全新的電腦且並未在該裝置上登入過。本情境運作方式同 1.，惟按下登入 (Login) 按鈕並將資料送至伺服器後，伺服器將同時檢查輸入的帳號名稱與密碼是否皆與其中一筆資料完全相符：若否則通知用戶端發出錯誤訊息並拒絕登入，反之則 A 成功進入聊天室，並將已登入的資訊存入 cookie。

在上兩例中，用戶端皆會在使用者成功登入後通知伺服器更新在線使用者名單 (①)、及透過伺服器向資料庫聲索公共聊天室歷史訊息並下載顯示於訊息區 (④→②)。

3. A 使用者在公共聊天室中傳送訊息、B 使用者在公共聊天室中接收

當 A 發出訊息時，用戶端將以下列 JSON 格式將訊息送至伺服器 (①)：

```

1. {
2.   username: 'A',
3.   message: 'example message',
4.   time: dateTime(),
5.   room: '',
6. };

```

其中 **username** 儲存發送人的帳號名稱、**message** 儲存訊息本文、**time** 將呼叫 **dateTime** 函數產生訊息發送時的時間字串、**room** 則儲存收件人的帳號名稱 (由於此例中傳送的為公開訊息故本欄儲存空字串)。

伺服器收到 A 使用者傳送的訊息後，先後將該訊息送至資料庫儲存 (③) 並將其透過 **socket.io** 提供的 **broadcast** 功能推送至所有正在線上的使用者 (②)，包括 B。當 B 使用者於用戶端收到伺服器推送的 JSON 格式訊息後，便會將其解碼並顯示於公共聊天室的訊息區中。

4. A 使用者向 B 使用者傳送私密訊息

當 A 使用者按下右側使用者名單中 B 的名稱時，用戶端將透過伺服器向資料庫聲索 A 與 B 之間的私密聊天室的所有歷史訊息，並下載顯示於訊息區 (④→②)。此例和 3. 運作方式類似，當 A 發出訊息時，用戶端一樣會將一個 JSON 格式訊息送到伺服器 (①)，差別在於其中的 **room** 會是 B 的名稱：

```
1. {  
2.   username: 'A',  
3.   message: 'example message',  
4.   time: dateTime(),  
5.   room: 'B',  
6. };
```

接著，伺服器會先後將該訊息送至資料庫儲存 (③)，並將其推送至 **room** 所記錄的使用者之用戶端 (②)，此例中即 B。當 B 的用戶端收到伺服器推送的訊息並解碼後，將先檢查目前 B 是否位於與 A 之間的私密聊天室頁面：若然，則直接將該訊息顯示於訊息區；若否，則向 B 發出瀏覽器內建的通知功能，同時在 B 用戶端右側的使用者名單中 A 的帳號名稱左側加上未讀訊息的標註 (💡)。當 B 按下通知後，訊息區便會切換至與 A 之間的私密聊天室，而訊息讀取成功後通知便會消失。

參、參考資料

1. “socket.io docs”, <https://socket.io/docs/>, 2017/03
2. 「[Node.js]mongolab & mongodb 初探」, <http://howard10335.blogspot.tw/2015/08/nodejsmongolab-mongodb.html>, 2017/03
3. “Create a Node.js chat application with Socket.IO in Azure App Service”, <https://docs.microsoft.com/en-us/azure/app-service-web/web-sites-nodejs-chat-app-socketio>, 2017/03
4. “Tooltipster”, <http://iamceege.github.io/tooltipster/>, 2017/03
5. “jQuery API Documentation”, <https://api.jquery.com/>, 2017/03
6. “Set Hash passwords using NodeJS crypto”, <https://ciphertrick.com/2016/01/18/salt-hash-passwords-using-nodejs-crypto/>, 2017/03