

嵌入式實驗一報告

Chat room

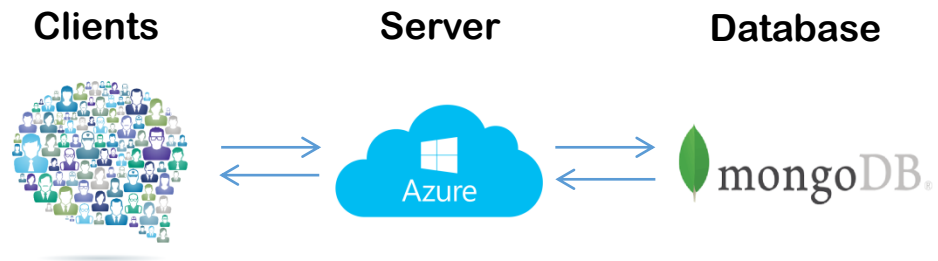
組員：電機三 林尚謙

電機三 楊耀程

指導教授：鄭振牟

繳交日期：2017/03/31

架構



我們聊天室的架構可分為 client 端、server 端與 database，client 端即為瀏覽器上的網頁，server 端部屬在微軟 Microsoft Azure 的平台上，database 使用 MongoDB 作為雲端資料庫。Client 會先與 server 建立 socket 連線，在互動發生後(送出訊息、點擊按鈕等)，會由 client 端向 server 發出請求，再由 server 判斷該對誰(全部或特定的 socket)做出對應的回應，而 database 則負責提供 server 存取資料。我們的 client 端與 server 端分別由一個檔案(index.html 與 server.js)構成。

特色與操作介面



1. 雲端的資料庫與 server

點開網址 <https://eslab2017dereknico.azurewebsites.net/>，或是在本機執行 `node server.js` 並輸入 `localhost:3000`，即可進入本聊天室。

我們的聊天室有雲端上的資料庫，`server` 也在雲端上執行，所以聊天並不限於本機，使用者可以用不同台電腦連上網路後，互相聊天。

```
var db;
dbClient.connect(url, function(err, tempdb) {
  db = tempdb;
});

io.on('connection', function(socket) {

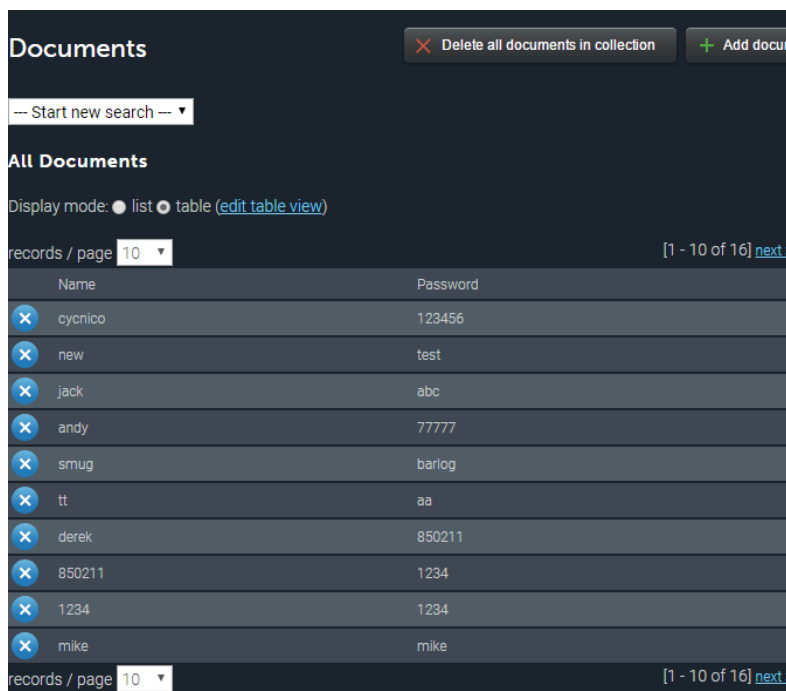
  var profiles = db.collection('userProfile'); //user profiles
  var records = db.collection('talkingRecord'); //talking records
```

在建立連線後可由 `server` 端存取 database，其中 `userProfile` 記錄使用者的帳號資料，`talkingRecord` 則是之前的歷史聊天紀錄。

2. 帳號密碼系統:

進入聊天室後，登入畫面為表單的形式，先輸入使用者名稱，在輸入使用者密碼。如果第一次登入，就會自動註冊，輸入的姓名與密碼會被記錄在資料庫中，供日後使用。

帳號密碼的資料庫由雲端的 `mongoDB` 管理，以 `json` 格式存放，只有資料庫的管理者能夠 `access`。



The screenshot shows the MongoDB Documents interface. At the top, there are buttons for 'Delete all documents in collection' and 'Add document'. Below that is a search bar. The main section is titled 'All Documents' and shows a table of documents. The table has two columns: 'Name' and 'Password'. There are 16 documents in total, and the first 10 are displayed. Each document has a delete icon (X) to its left. The table is paginated, showing 'records / page 10' and '[1 - 10 of 16] next >'. The data in the table is as follows:

| Name | Password |
|--------|----------|
| cynico | 123456 |
| new | test |
| jack | abc |
| andy | 77777 |
| smug | bariog |
| tt | aa |
| derek | 850211 |
| 850211 | 1234 |
| 1234 | 1234 |
| mike | mike |

可由 mlab 的網站
介面直接管理
MongoDB 的資料

登入介面：

eslab2017dereknico.azurewebsites.net 顯示：

請輸入帳號 (或輸入guest以訪客登入)
第一次登入的帳號會自動註冊

guest

確定 取消

eslab2017dereknico.azurewebsites.net 顯示：

請輸入密碼

nico

☐ 防止此網頁產生其他對話方塊。

確定 取消

eslab2017dereknico.azurewebsites.net 顯示：

密碼錯誤，請重新登入！

☐ 防止此網頁產生其他對話方塊。

確定

eslab2017dereknico.azurewebsites.net 顯示：

此帳號已從別處登入，請用其他帳號重新登入！

☐ 防止此網頁產生其他對話方塊。

確定

2. 即時聊天系統

即時聊天系統是本聊天室最主要的功能，能夠多人聊天並即時反應，不用重新整理就能看到新的訊息跳出來，自己的訊息會在畫面右側，其他人的訊息在左側。訊息的發送者，訊息內容，發送時間，存在雲端上的 **mongodb**，以 **json** 格式存放。

| client | server |
|---|---|
| <pre>\$('#textbox').submit(function() { if (\$('#m').val() != '') { socket.emit('chat message', \$('#m').val()); \$('#m').val(''); // clean input } return false; });</pre> | <pre>socket.on('chat message', function(msg) { //receive msg from client console.log(socket.username + ":" + msg); var mytime = mygetTime(); io.emit('chat message', { u_name: socket.username, u_word: msg, u_time: mytime }); //when 'chat message' records.insert({ u_name: socket.username, u_word: msg, u_time: mytime }); });</pre> |

3. 對話記錄保存

每當 **guest** 之外的使用者登入時，此聊天室會連結 **mongodb** 中的聊天記錄，把先前的聊天訊息載入，代表即使沒在線上，也能看到其他人的對話。因此此聊天室支援離線留言的功能，即使對方不在線上，在此多人對話框發出訊息，對方上線後依然能夠看到，而且可以看到發送者，訊息內容，訊息發送時間。

4. 在線名單

在對話框的右側有欄在線名單，此在線名單也有即時性，不必重新整理就會更新，在線名單會顯示在線上的人，同時在線名單也會標明使用者本人，會在使用者後面加上(You)。



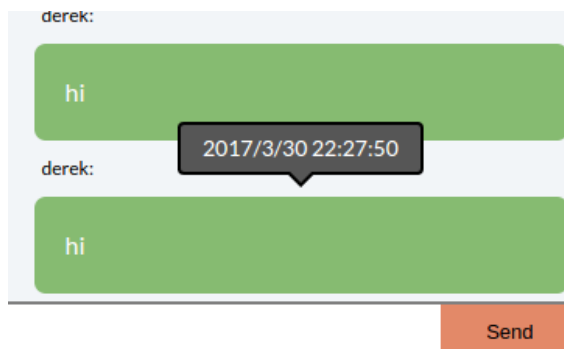
```
var getuserlist = function() {  
  const userList = [];  
  for (let i = 0; i < clientlist.length; i += 1)  
    userList[i] = clientlist[i].username;  
  return userList;  
};
```

在有人登入/登出時，server 重新更新
userlist 後傳給所有 clients

5. 查看訊息時間

同 3 所敘述，聊天的記錄會被存放在雲端的 mongodb，因此聊天時間是有被保存的，即使是歷史記錄的訊息，也都有辦法顯示時間。顯示時間的方式，是利用 tooltipster，採用滑鼠互動模式。每當滑鼠移到訊息上方，就會有個小視窗顯示訊息的時間，當滑鼠移開，小視窗又消失。

Tool tipster 效果：



6. 單人即時聊天系統

單人私訊聊天的部分，我們在每次更新在線名單時，同時附加一個 click 的監聽事件在每個 username 上，當 client 點擊時就會對 server 發出一個 private chat 的請求，server 會同時回應給要求者與被要求的目標，在兩個 client 都開啟一個私訊視窗，視窗上會顯示你目前私訊對象的 username，使用者便可透過這個視窗與其他使用者進行單人私訊聊天。

client

server

```
socket.emit('private chat', userlist[i]);
```

Client 要求開啟私訊, Server 回應
在兩邊都開啟視窗

```
socket.on('onetoone chat', function(user2) {=
```

```
socket.emit('private chat message', {
```

Client 傳出私訊, Server 接收後顯
示於兩邊的私訊視窗中

```
socket.on('private send', function(data) {  
    AppendPriMessage(data);  
});
```

```
socket.on('private chat', function(user2) {  
    socket.emit('onetoone chat', user2);  
    target.emit('onetoone chat', socket.username);  
});
```

```
socket.on('private chat message', function(data) {  
    socket.emit('private send', {=  
    target.emit('private send', {=  
});
```