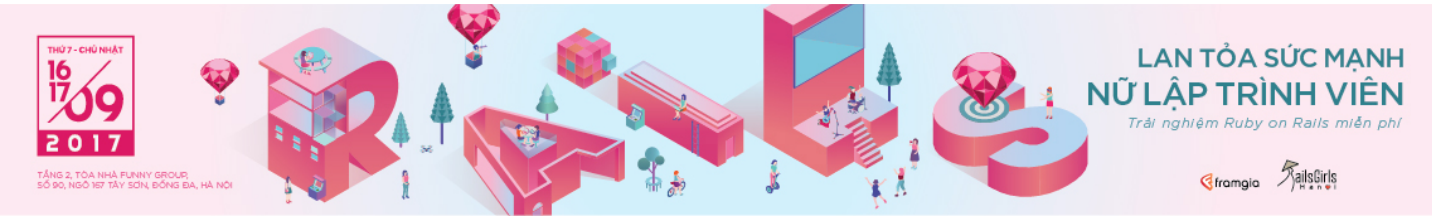


Nguyễn Quốc Đạt

Follow

Xây dựng ứng dụng chat bằng Nodejs, SocketIO, Angularjs (phần 2)

Node.js AngularJS socket.io JavaScript Framework

Nov 4th, 1:31 am

Tiếp nối phần 1 The best structure of Angular project mình sẽ viết tiếp phần 2: Tạo ứng dụng chat bằng `Nodejs` , `Socket.io` , `AngularJs`

Tổng quan

`Websoket` , `Nodejs` , `Socket.io` đã luôn là những hotkey trong giới công nghệ, đặc biệt là ngành lập trình trong những năm gần đây. Đặc biệt khi mà các ứng dụng realtime như mạng xã hội, chat, game, ứng dụng số ... phát triển như vũ bão thì công nghệ realtime luôn được ưa chuộng hơn bao giờ hết. Vậy những công nghệ này là gì? Ứng dụng nó như thế nào?

- `WebSocket` là gì?

  WebSoket là công nghệ hỗ trợ giao tiếp hai chiều giữa client và server bằng cách sử dụng một TCP socket để tạo một kết nối hiệu quả và ít tốn kém.

  Dữ liệu truyền tải thông qua giao thức HTTP (thường dùng với kĩ thuật Ajax) chứa nhiều dữ liệu không cần thiết trong phần header. Một header request/response của HTTP có kích thước khoảng 871 byte, trong khi với WebSocket, kích thước này bé hơn nhiều, chỉ là 2 byte (sau khi đã kết nối). Vì vậy Websoket có thể phục vụ được hàng ngàn, hàng triệu user tại một thời điểm. Quá tuyệt vời phải không nào ^^

- `Nodejs` là gì?

hình non-blocking I/O để tạo ra các ứng dụng nhẹ và hiệu quả cho các ứng dụng về dữ liệu thời gian thực chạy trên các thiết bị phân tán.

NodeJs là một mã nguồn mở, đa nền tảng cho phát triển các ứng dụng phía Server và các ứng dụng liên quan đến mạng. Ứng dụng Node.js được viết bằng Javascript và có thể chạy trong môi trường Node.js trên hệ điều hành Window, Linux

Đặc điểm của Nodejs.

- Không đồng bộ và phát sinh sự kiện (Event Driven): Tất các các APIs của thư viện Node.js đều không đồng bộ, nghĩa là non-blocking. Nó rất cần thiết vì Node.js không bao giờ đợi một API trả về dự liệu. Server chuyển sang một API sau khi gọi nó và có cơ chế thông báo về Sự kiện của Node.js giúp Server nhận đựa phản hồi từ các API gọi trước đó.
- Chạy rất nhanh: Dựa trên V8 Javascript Engine của Google Chrome, thư viện Node.js rất nhanh trong các quá trình thực hiện code.
- Các tiến trình đơn giản nhưng hiệu năng cao: Node.js sử dụng một mô hình luồng đơn (single thread) với các sự kiện lặp. Các cơ chế sự kiện giúp Server trả lại các phản hồi với một cách không khóa và tạo cho Server hiệu quả cao ngược lại với các cách truyền thống tạo ra một số lượng luồng hữu hạn để quản lý request. Nodejs sử dụng các chương trình đơn luồng và các chương trình này cung cấp các dịch vụ cho số lượng request nhiều hơn so với các Server truyền thống như Apache HTTP Server.
- Không đệm: Ứng dụng Node.js không lưu trữ các dữ liệu buffer.

- `Socket.io` là gi?
  - Socket.io là một thư viện WebSocket nổi tiếng. Nó rất mạnh mẽ, đơn giản, dễ ứng dụng và được sự hỗ trợ của nhiều nhà phát triển (từ Microsoft Office, Yammer, Zendesk, Trello… tới những đội hackathon, những start up trẻ).
  - Socket.io gồm 2 phần:
    1. Client: gồm bộ thư viện viết cho web(JavaScript), iOS, Android
    2. Server: gồm bộ thư viện viết cho Nodejs

Như vậy chúng ta có thể thấy được tính năng vượt trội và tính ứng dụng cao của `Nodejs` , `Socket.io` . Tiếp theo chúng ta sẽ kết hợp chúng với `AngularJs` để xây dựng ứng dụng web realtime.

Xây dựng ứng dụng

Yêu cầu: máy của bạn đã cài đặt `bower` , `Nodejs` . Nếu chưa cài đặt bao giờ bạn có thể tham khảo tại đây

Chuẩn bị

- Clone ứng dụng từ github
- Run command
    - `bower install`

Bắt đầu

Ứng dụng sẽ gồm 2 phần:

- Server: nhận thông tin gửi lên từ client, sau đó gửi dữ liệu cho tất cả client đang kết nối socket.
- Client: gửi thông tin (emit) lên server, chờ nhận (broadcast) thông tin từ server gửi về

Server

1. Cài đặt các module cần thiết
    - `npm install --save-dev http express socket.io`
2. Tạo file `server.js` ở thư mục root project như sau:

```
// load các module
var express = require('express');
var app = express();
var server = require('http').createServer(app);
var io = require('socket.io')(server);
var port = process.env.PORT || 9000;
// khởi tạo server listen cổng 9000
server.listen(port, function () {
  console.log('Server listening at port '  + port);
});

var numUsers = 0;
// khởi tạo kết nối socket
io.on('connection', function(socket) {
    console.log('a socket connected');
    var addedUser = false;
    // socket nhận tin nhắn từ 1 client
    socket.on('send_message', function(text) {
        console.log(socket.username);
        // gửi tin nhắn tới các client đang kết nối socket
        // ngoại trừ client đang kết nối (gửi tin nhắn)
        socket.broadcast.emit('receive_message', {
            username: socket.username,
            text: text
        });
    });
    // socket client  join vào room chát
    socket.on('user_join', function(username) {
        if (addedUser)
            return false;
        socket.username = username;
        console.log('user_join: '+ socket.username);

        ++ numUsers;
        addedUser = true;
        // báo cho client đang join phòng thành công
        socket.emit('login', {
            numberUsers: numUsers
        });
        // báo cho client khác biết có người mới join vào phòng
        socket.broadcast.emit('new_user_join', {
            username: socket.username
```

```
        socket.on('typing', function(data) {
            socket.broadcast.emit('typing', {
                username: socket.username
            });
        });

        socket.on('disconnect', function() {
                if (addedUser)
                    -- numUsers;
                socket.broadcast.emit('user_left', {
                    username: socket.username,
                    numUsers: numUsers
                });
        })
    });
```

Giải thích:

- Load module `express` , `http` , `socket.io` và khởi tạo tạo server listen cổng 9000.
- Khởi tạo kết nối socket sử dụng phương thức `io.on('connection')`
- Trong socket connection sẽ tạo ra các `socket.on` để lắng nghe chờ nhận dữ liệu được `emit` từ client lên như:
- `send_mesage` : gửi tin nhắn
- `user_join` : join phòng
- `typing` : đang chát
- `disconnect` : sự kiện ủe ngắt kết nối socket

Socket.io có một số phương thức:

- `socket.on` : lắng nghe sự kiện từ client gửi lên
- `socket.emit` : gửi sự kiện, dữ liệu tới duy nhất client đang kết nối hiện tại (gửi cho mình)
- `socket.broadcast.emit` : gửi sự kiện, dữ liệu tới tất cả client đang kết nối ngoại trừ client hiện tại (gửi cho thằng khác ngoại trừ mình)

```
    Như vậy chúng ta có thể thấy `socket.io` hoạt động theo hướng sự kiện. Tức mỗi khi client có sự kiện sẽ gửi thông báo lên server, sau đó server sẽ gửi thông báo cho các thằng khác (có thể có cả bản thân mình); các client bắt được sự kiện gửi từ server, nó sẽ xử lý dữ liệu dc nhận về đó.
```

3. Cuối cùng mở terminal khởi tạo server bằng câu lệnh:

Client

Cài đặt các thư viện cần thiết từ `bower`

- `bower install angular-socket-io`

Khởi tạo module `app.chat` tương tự phần 1

- Tích hợp module `app.chat` vào ứng dụng

```
    // app.module.js
    (function() {
    "use strict";
    angular.module('app', ['app.core', 'app.header', 'app.landing',
'app.login', 'app.chat']);
    })();
```

- Inject module `'btford.socket-io'` vào module `app.core`

```
    // core.module.js
     (function() {
    'use strict';
        angular.module('app.core', ['ui.router', 'ngStorage',
'btford.socket-io']);
    })();
```

Tạo thư mục `chat` trong thư mục `app` gồm các file

1. `chat.module.js` , `chat.config.js`

- Khởi tạo module `app.chat` và config route

```
    'use strict';

    angular.module('app.chat', [
        'app.core'
        ]);
})();
```

```
// chat.config.js
(function () {
    'use strict';

    angular
        .module('app.chat')
        .config(config);

    function config($stateProvider, $urlRouterProvider) {
        $urlRouterProvider.otherwise("/landing");

        $stateProvider
            .state('chat', {
                url: "/chat",
                templateUrl: "app/chat/chat.html",
                controller: "ChatController",
                controllerAs: "vm"
        })
    }
})();
```

2. `Chat.html`

- Tạo giao diện khung chát như sau:

```
vm">
            <div class="panel-heading">
                <h3 class="panel-title">
                    Chat mesage
                </h3>
            </div>
            <div class="panel-body">
                <ul>
                    <li ng-repeat="msg in vm.mesages">
                        <b>
                            {msg.username}}
                        </b>
                        : {msg.tết}}
                    </li>
                </ul>
                <form class="form-horizontal">
                    <div class="form-group">
                        <textarea class="form-control chat-bõ" my-
enter="vm.sendMsg();" ng-model="vm.tết" placeholder="Enter your mesage"
ró="3">
                        </textarea>
                    </div>
                </form>
            </div>
        </div>
```

3. `Core.directive.js`
    - Directive kiểm tra sự kiện người dùng nhấn `enter` lúc chát. Chú ý là file này lưu vào thư mục `app/core` .

```
(function() {
    'use strict';

    angular
        .module('app.core')
        .directive('myEnter', function () {
            return function (scope, element, attrs) {
                element.bind("keydown keypress", function (event) {
                    if(event.which === 13) {
                        scope.$apply(function (){
                            scope.$eval(attrs.myEnter);
                        });
                        console.log('enter enter');
                        event.preventDefault();
                    }
                });
            };
        });

})();
```

4. `Socket.service.js`

- Là service kết nối socket. Chúng ta sẽ sử dụng đối tượng `socketFactory` của `angular-socket-io` để kết nối lên server nodejs cổng 9000.

```
(function () {
    'use strict';

    angular
        .module('app.chat')
        .factory('socket', socket);

    function socket(socketFactory) {
        var socketConnection = io.connect('http://localhost:9000');
        var socket = socketFactory({
            ioSocket: socketConnection
        });

        return socket;
    }
})();
```

5. `Chat.controller.js`

- Là controller chính của ứng dụng chat. Nó sẽ inject `socket` service. Có 2 cặp sự kiện chúng ta quan tâm.
- Khi một người dùng join phòng chát
  - Khi mình tham gia phòng chát thì `socket.emit` lên server:
    `socket.emit('user_join', $localStorage.currentUser.username);`
  - Khi người khác tham gia phòng chát thì mình sẽ `soket.on` để nhận dữ liệu:
    `socket.on('new_user_join', fuction() {})`
- Gủi và nhận tin nhắn:
  - Gửi tin nhắn: `socket.emit('send_message', vm.text);`
  - Nhận tin nhắn: `socket.on('receive_message', function(data) {}`

```
(function() {
    'use strict';

    angular
        .module('app.chat')
        .controller('ChatController', ChatController)

    ChatController.$inject = ['$state', '$http', '$localStorage',
'socket'];
    function ChatController ($state, $http,  $localStorage, socket) {
        var vm = this;
        vm.messages  = [];
        vm.text = '';
        vm.sendMsg  = sendMsg;

        joinChat();
        onNewUser();
        // tham gia phòng chát
        function joinChat() {
            if ($localStorage.currentUser) {
                socket.emit('user_join',
$localStorage.currentUser.username);
            } else {
                return false;
            }
        }
        // có người tham gia phòng chát
        function onNewUser() {
            socket.on('new_user_join', function (data) {
                console.log('new_user');
                console.log(data);
            });
        }
        // gửi tin nhắn
        function sendMsg() {
            socket.emit('send_message', vm.text);
            vm.messages.push({username:
$localStorage.currentUser.username, text: vm.text});
            vm.text = '';
        };
        // nhận tin nhắn
        socket.on('receive_message', function(data) {
```

```
    })();
```

Bước cuối

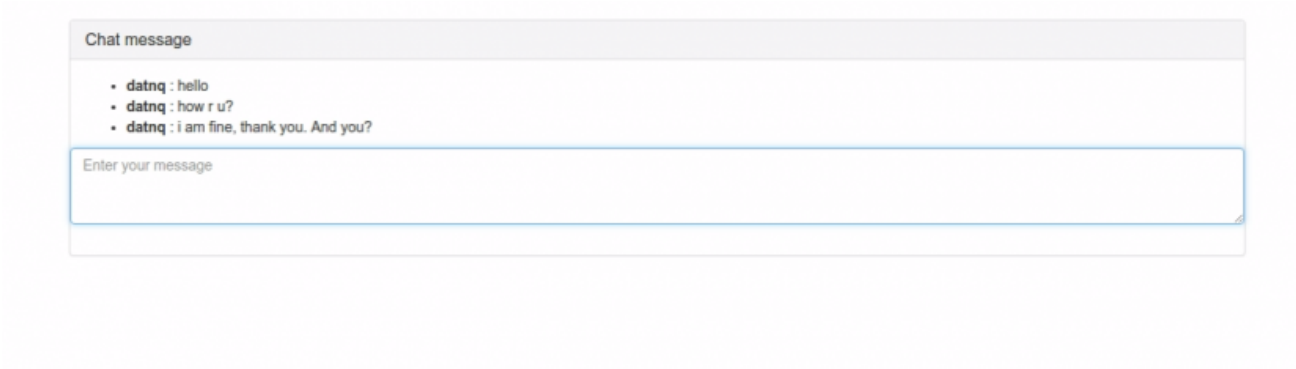
- Chúng ta load các thư viện `socket.io` , `angular-socket-io` và chat js file vào `index.html`

```
    <head>
        <base href="/">
        <script src="http://localhost:9000/socket.io/socket.io.js">
</script>
    </head>
    ....
    <script src="bower_components/ngstorage/ngStorage.min.js"></script>
    <script src="bower_components/angular-socket-io/socket.min.js">
</script>

    <script src="app/core/core.directive.js"></script>
    ...
    <script src="app/chat/chat.module.js"></script>
    <script src="app/chat/chat.config.js"></script>
    <script src="app/chat/socket.service.js"></script>
    <script src="app/chat/chat.controller.js"></script>

    <script src="app/app.module.js"></script>
    <script src="app/app.config.js"></script>
```

- Chú ý: để sử dụng được `socket.io` chúng ta phải load thư viện này từ server nodejs. Chúng ta đang chạy server nodejs ở cổng 9000 nên link sẽ là: `http://localhost:9000/socket.io/socket.io.js` .
- Mở trình 2 duyệt lên và thưởng thức kết quả của mình nào: `http://viblo.dev/chat`



Kết luận

- Qua ví dụ trên các bạn có thể thấy rất dễ dàng để tạo được một ứng dụng chát bằng `Nodejs` , `Socket.io` và `AngularJs` . Các bạn có thể phát triển hơn ứng dụng này để phát triển ứng dụng chát riêng tư, chát nhóm và bảo mật hơn kênh truyền.
- Source code bạn có thể tải tại đây

Tài liệu tham khảo

- https://nodejs.org/en/
- http://socket.io/
- http://socket.io/

Related

Từ hàng tỉ phép so sánh đến 10 giây

Luc Duong

1249 16 7 18

Mình đã làm bể cá thông minh như thế nào (phần 1)

Hoàng Hữu Hợi

1088 10 13 25

Unit test cho Nodejs RESTful API với Mocha và Chai

Dinh Van Hoang

Z)

Hoàng Hữu Hợi

471 2 10 6

More from Nguyễn Quốc Đạt

Processes vs Threads

Nguyễn Quốc Đạt

50 0 0 0

Gearman - Multi workers parallel

Nguyễn Quốc Đạt

153 4 0 6

https và SSL

Nguyễn Quốc Đạt

337 4 0 3

Dependency Injection trong PHP

Nguyễn Quốc Đạt

343 2 0 3

Comments

No comments yet.

FACEBOOK

+2