Mục Lục

[TẠO MỘT DỰ ÁN NODEJS & EXPRESS 2](#_Toc162775572)

[I. Khởi tạo và cài đặt môi trường 2](#_Toc162775573)

[ Tải xuống NodeJs 2](#_Toc162775574)

[ Tạo cây thư mục chứa dự án 2](#_Toc162775575)

[ Sử dụng Terminal để khởi chạy 2](#_Toc162775576)

[II. Cài đặt Express và chạy chương trình. 3](#_Toc162775577)

[ Cài đặt express 3](#_Toc162775578)

[ Chạy chương trình 3](#_Toc162775579)

[III. Cài đặt một số thư viện và công cụ 4](#_Toc162775580)

[ Nodemon. Giúp hỗ trợ debug 4](#_Toc162775581)

[ Morgan. Giúp kiểm tra các yêu cầu HTTP 5](#_Toc162775582)

[IV. Bắt đầu xây dựng trang web 5](#_Toc162775583)

[ Câu lệnh nhanh. 5](#_Toc162775584)

[ Cấu trúc thư viện. 6](#_Toc162775585)

[ Mô hình MVC 7](#_Toc162775586)

[ Xây dựng routers 8](#_Toc162775587)

[ Xây dựng controllers 9](#_Toc162775588)

[ Static file(tập tin tĩnh) 14](#_Toc162775589)

[ Xây dựng views 15](#_Toc162775590)

[ Dùng template-engine 17](#_Toc162775591)

[ Sass và SCSS 23](#_Toc162775592)

[ Xây header và footer 27](#_Toc162775593)

[ Xây dựng modal 28](#_Toc162775594)

[V. Các chức năng chuyên sâu 37](#_Toc162775595)

[ Lấy một phần tử cụ thể. 37](#_Toc162775596)

[ Tạo một trang quản trị viên(ADMIN) 46](#_Toc162775597)

# TẠO MỘT DỰ ÁN NODEJS & EXPRESS

## Khởi tạo và cài đặt môi trường

### Tải xuống NodeJs

#### Sử dụng trình duyệt và nhập từ khoá NodeJs, bạn sẽ thấy nó thể hiện ngay kết quả đầu tiên. Tải xuống và cài đặt bình thường

#### 

### Tạo cây thư mục chứa dự án

#### Tạo một cây thư mục cho dự án của bạn. file main.js (tên\_tuỳ\_ý.js) được đặt trong thư mục src.

#### Src sẽ quản lí toàn bộ cấu trúc của chương trình.

#### 

### Sử dụng Terminal để khởi chạy

#### Gõ câu lệnh khởi tạo một chương trình NodeJs: npm init

#### 

#### Điền hoặc để trống các yêu cầu(không điền sẽ được đặt mặc định) sẽ thấy một file backage.json được tạo ra

#### 

#### Trong file đó, tại dòng main hãy sửa “index.js” (đặt mặc định) thành file Javascript vừa tạo bên trên “main.js”.



#### 



## Cài đặt Express và chạy chương trình.

### Cài đặt express

#### Mở terminal và thực hiện câu lệnh: npm i express.

#### Sau khi tải xong trong file package.json sẽ xuất hiện mục “dependencies” và express nằm bên trong đấy.

#### 

### Chạy chương trình

#### Trong file main.js hãy tích hợp express vừa cài vào

#### 

#### Phương thức “get” của express nhận và 2 đối số là đường dẫn và function.

#### Function chứa các đối số req(request) và res(respone). Res.send() sẽ in chữ được truyền vào lên trình duyệt.

#### Truy cập trang Express js trên trình duyệt để xem document và hiểu rõ hơn

#### 

#### ở terminal thực hiện câu lệnh: node src/main.js (node + đường dẫn đến file chính).

#### Có thể thấy app.listen đã hoạt động và khi ta truy cập vào localhost trên trình duyệt thì chương trình đã hoạt động.

#### 

#### 

## Cài đặt một số thư viện và công cụ

### Nodemon. Giúp hỗ trợ debug

#### Mở terminal và thực hiện dòng lệnh: npm i nodemon.

#### Nếu chỉ sử dụng cho mỗi một dự án cụ thể có thể tải bằng lệnh “npm I nodemon —save-dev” để chú thích gần nó chỉ cài đặt ở dự án này.

#### Có thể truy cập trang npm nodemon trên trình duyệt để biết cụ thể hơn.

#### 

#### Sau khi đã tải xong thì file package.json sẽ có thêm mục “devDependencies” và nodemon đã nằm ở đó.

#### Nodemon giúp bạn tự khởi động lại máy chủ khi nó được thay đổi thay vì phải khởi động thủ công, điều này giúp tiết kiệm thêm nhiều thời gian.

#### 

#### Bổ sung:

#### Có một điều thú vị ở Node v20. Bạn có một phương thức “—watch” và nó hoạt động giống như “—inspect” của nodemon.

#### 

#### Tuy nhiên ở đây chúng ta vẫn sẽ sử dụng thư viện.

### Morgan. Giúp kiểm tra các yêu cầu HTTP

#### Mở terminal và thực hiện dòng lệnh: npm I morgan.

#### Bạn cũng có thể cài và devDependencies nếu muốn.

#### Truy cập Morgan trên trình duyệt để tham khảo rõ hơn.

#### 

#### Morgan giúp bạn thể hiện các yêu cầu HTTP một cách rõ ràng và cụ thể. Thể hiện các phương thức HTTP được sử dụng(GET, PUT, POST, PATCH) thời gian yêu cầu….. từ đó giúp việc kiểm tra và sửa đổi trở nên thuận tiện hơn.

## Bắt đầu xây dựng trang web

### Câu lệnh nhanh.

#### Mỗi khi chúng ta muốn chạy chương trình có thể thấy sẽ rất dài dòng nếu cứ gõ lệnh: node + đường\_dẫn\_file

#### 

#### Hãy setup lại bằng một cách ngắn gọn hơn.

#### Trong file backage.json, bạn có thể thấy mục scripts

#### 

#### ở đây sẽ được chứa các lệnh debug được mặc định hoặc cài thêm từ người dùng. Trong ví dụ thì “test” sẽ in ra cầu “Eror: no test spectified” và đóng chương trình.

#### Để chạy các phương thức trong scripts ta chỉ cần gõ: npm + tên\_phương\_thức

#### 

#### Có thể thấy nó đã hoạt động như mêu tả.

#### Từ đây ta cũng có thể áp dụng nó cho việc chạy chương trình của mình

#### Tôi tạo thêm phương thức “start” trong scripts để thể hiện việc chạy dự án của mình.

#### Ngay bên trong hãy thêm cách chạy chương trình lúc đầu của chúng ta: node + đường\_dẫn\_file

#### 

#### Và đây là kết quả của việc đó.

#### 

#### Có thể thấy chương trình đã được chạy.

#### Điều này giúp bạn có thể dễ dàng chạy chương trình của mình mà không cần quan tâm đến đường dẫn của nó dài đến đâu.

### Cấu trúc thư viện.

#### Bạn còn nhớ Morgan đã được cài trên kia chứ. Mặc dù đã được cài nhưng Morgan sẽ cần được nạp vào nếu muốn sử dụng nó.

#### Tại file chính “main.js” hãy require nó vào.

#### 

#### Dùng phương thức use của Express để sử dụng log HTTP request của Morgan.

#### 

#### Cú pháp “combined” của Morgan sẽ ghi lại các thông tin như: IP của người dùng, thời gian yêu cầu, URL được yêu cầu….

#### Khi này nếu bạn thực hiện các request đến trình duyệt thì Morgan sẽ in ra cho bạn.

#### 

#### ở trên chúng ta cũng đã có cài Nodemon để việc reload lại sever trở nên thuận tiện hơn, nhưng nó cũng sẽ không hoạt động nếu bạn không cấu hình nó.

#### 

#### Có thể thấy mặc dù nội dung đã được thay đổi tuy nhiên phía trình duyệt vẫn giữ nội dung cũ

#### 

#### Cấu hình Nodemon ngay đoạn scripts chạy chương trình bạn đã tạo.

#### ở đây tôi cấu hình “node –inspect src/main.js” có nghĩa rằng yêu cầu tải lại sever khi tôi thay đổi cấu trúc của trình duyệt.

#### 

#### Tắt sever và chạy lại bạn sẽ thấy nodemon đã được áp dụng

#### 

#### Giờ thì mỗi khi bạn thay đổi code, phía trình duyệt cũng sẽ thay đổi theo.

### Mô hình MVC

#### MVC là viết tắt của Modal View Controller, đây là một cấu trúc tiêu chuẩn phổ biến cho các dạng mô hình web.

#### Code của bạn nếu được xây dưng theo như cách như chúng ta đa làm phía trên thì về lâu sẽ rất dài dòng và rối mắt, vì nó bị dồn nén và trong một file duy nhất cho nên việc quản lí cũng như thay đổi sẽ rất khó khăn. Mô hình MVC sẽ giúp bạn xây dựng một cấu trúc quản lí cụ thể.

#### Tìm hiểu chi tiết tại: [mô hình MVC là gì?](https://vietnix.vn/tim-hieu-mo-hinh-mvc-la-gi/)

#### Hoặc xem chi tiết tại F8: [mô hình MVC](https://www.youtube.com/watch?v=N8GhaR7K3tI)

### Xây dựng routers

#### Thông thường để truy cập các trang khác nhau thì ta sẽ phải tạo từng đường dẫn của file đó.

#### 

#### ở đây khi gõ dấu “/” trên sever thì chúng ta sẽ được di chuyển đến trang chính.

#### khi gõ “/menu” ta được chuyển đến trang menu.

#### 

#### Tuy nhiên việc một trang web sẽ chứa nhiều trang và cứ mỗi lần tạo đường dẫn như thế sẽ làm cho file gốc của chúng ta rất dài, chưa kể file này không chỉ chứa mỗi đường dẫn mà còn các công cụ khác. Vì thế việc chia router sẽ giúp code gọn hơn rất nhiều.

#### Tạo một thư mục routers bên trong src. Đây sẽ là nơi chứa các router của bạn.

#### Bên trong routers hãy tạo một file chính index.js

#### 

#### Cấu trúc nó thành một function route với tham số app và cuối cùng là xuất function đó ra.

#### 

#### Khi này tại main.js hãy nạp function route vừa được xuất ra.

#### 

#### Chạy route vừa nạp và gắn app vào làm đối số

#### 

#### Khi này ta đã có thể định tuyến các đường dẫn tại route

#### 

#### Trình duyệt vẫn sẽ hoạt động như cũ mà không có bất kì vấn đề nào. Ta đã có thể làm gọn file gốc main đi rất nhiều.

#### 

#### 

### Xây dựng controllers

#### Tuy nhiên việc chỉ có một router để định tuyến cũng sẽ gây khó kiểm soát khi chương trình của chúng ta trở nên phức tạp và có nhiều trang, vì thế hãy chia nhỏ nó ra. Ta sẽ xây các controller để quản lí từng trang cụ thể.

#### Bên trong src, tạo một thư mục và đặt tên là controllers.

#### Bên trong hãy tạo một file tên siteController.js, file này sẽ chịu trách nhiệm quản lí trang chủ và một số chức năng cụ thể.

#### 

#### Vì controller này sẽ quản lí nhiều chức năng mà mỗi chức năng sẽ là một hàm nên ta sẽ tạo một class để dễ dàng quản lí các hàm(phương thức) đó.

#### 

#### Tuy nhiên khi này cấu trúc sẽ chưa hoạt động do chưa được định tuyến router. Có thể bạn thấy nó vẫn hoạt động bình thường nhưng đó là do nó vẫn còn ăn đoạn code cũ, đoạn code mới này chưa hề được dùng đến.

#### Để nó hoạt động hãy định tuyến router cho controller đó. Chia nhỏ các router ra cho từng công việc cụ thể.

#### Tạo một file mới cùng tên với controller để dễ chỉnh sửa và tìm kiếm sau này, ở đây sẽ mang tên site.js

#### 

#### Express hỗ trợ cho ta một phương thức mang tên Router để dễ dàng định tuyến.

#### Nạp nó vào.

#### 

#### Logic hoạt động ở đây là site.js sẽ làm router định tuyến cho siteController.js và để làm điều đó ta cần kết nối chúng.

#### Do bên phía siteController đã xuất đối tượng(class) chứa các phương thức ra nên ta chỉ cần nhận chúng ở nơi ta muốn thức hiện, ở đây là tại site.js.

#### 

#### Một trang sẽ có nhiều thành phần, mỗi thành phần sẽ được phân chia bằng các đường dẫn.

#### Khi bạn truy cập đến một trang web ví dụ: <https://trangweb.com>.

#### Lúc này bạn sẽ được dẫn đến trang chủ theo địa chỉ “<https://trangweb.com>/” dấu “/” sẽ thường không được thể hiện do nó mặc định dấu đấy là đường dẫn đến trang chính.

#### Nhưng nếu địa chỉ là: <https://trangweb.com/search>.

#### Thì lúc này đường dẫn đưa bạn đến thanh tìm kiếm của trang.

#### Dựa trên quy tắc đó, ta hãy cùng định tuyến đường dẫn của mình.

#### 

#### Dùng đối tượng router vừa nhận từ Express và sử dụng phương thức get với 2 tham số. Tham số đầu tiên là đường dẫn bạn muốn đặt, tham số thứ hai chính là phương thức được sử dụng cho đường dẫn đó.

#### ở đây khi ta gõ dấu “/” nó sẽ đưa ta đến trang chủ ở trang hiện tại.

#### tuy nhiên chưa hoạt động đâu vì chúng ta vẫn chưa hoàn thành việc định tuyến mà.

#### Bạn còn nhớ file index.js trong routers chứ, bây giờ ta sẽ chỉnh sửa để nó sẽ là router chính điều khiển mọi router khác sau này.

#### 

#### Tôi nhận router ở site.js bằng biến siteRouter(hãy chắc chắn rằng bạn đã xuất nó ra ở file site.js)

#### Do các router site hoặc controller site là do chúng ta tự biên soạn mà không phải mặc định của Express nên ta phải sử dụng phương thức use() của Express để sử dụng chúng.

#### Use nhận 2 tham số, thứ nhất là đường dẫn của trang, thứ hai chính router hay định tuyến của trang đó.

#### 

#### Chạy sever lên và bạn sẽ thấy kết quả.

#### 

#### Đến đây chắc sẽ có nhiều thắc mắc được đặt ra, “tại sao use cũng sử dụng dấu “/” trong khi bên site router đã định tuyến dâu “/” rồi”.

#### Giải thích một cách đơn giản là dấu “/” ở router site có nghĩa là địa chỉ của một thành phần trong trang đó (dấu “/” ở đây là trang chủ của trang đó).

#### Còn dấu “/” ở file index khi sử dụng phương thức use có nghĩa là địa chỉ trang cần đến (ở đây dấu “/” la trang chủ của trình duyệt).

#### Lấy ví dụ nếu bạn định tuyến ở site là “/search” và use đến “/” thì địa chỉ URL của bạn sẽ có dang:

#### <https://trangweb/search>

#### lúc này thứ bạn truy cập là thanh tìm kiếm(search) trong trang chủ.

#### Nếu địa chỉ là: <https://trangweb/produce/search>

#### Thì bây giờ là thanh tìm kiếm(search) trong trang sản phẩm(produce).

#### 

#### Do thanh định tuyến “/search” ở trang chủ chưa được đặt nên đã bị lỗi. “cannot GET” có nghĩa không thể nhận dữ liệu từ định tuyến đó.

#### Hãy cùng định tuyến cho nó nào, làm như cách của phương thức home. Trở lại siteController của chúng ta.

#### Sau khi đã thêm phương thức thì đừng quên định tuyến cho nó. Quay lại định tuyến ở site router.

#### 

#### Reload lại trình duyệt và bạn nhận được kết quả

#### Từ đây bạn có thể tự do tuỳ chỉnh các router cho từng trang cụ thể bạn muốn.

### Static file(tập tin tĩnh)

#### Static file là một các thức giúp bạn có thể chi sẻ những đoạn mã hay hình ảnh của bạn lên chính trình duyệt mà không phải thông qua phí sever. Điều này áp dụng cho các đoạn mã css hoặc hình ảnh hay các tập tin mà không bị ảnh hưởng bởi tính bảo mật dữ liệu.

#### Hãy tạo một thư mục public, nơi sẽ chứa các file được static.

#### 

#### Quay trở lại file chính(main.js) và static thư mục đó bằng Express.

#### 

#### Khi này toàn bộ các thư mục thay tập tin bên trong public đã được tạo một đường dẫn từ phía ngoài.

#### Lấy ví dụ hiện giờ trong public của tôi đang có 1 file tên “theme3-flower.png”

#### 

#### Vậy để truy cập file này tôi chỉ cần “/theme3-flower.png” trên thanh URL là được.

#### 

### Xây dựng views

#### Một trang web được xây từ nhiều đoạn mã và chúng được chia ra từ file khác nhau tuỳ theo từng trang hay chức năng cụ thể.

#### Việc xây dựng views sẽ giúp ta quản lí các trang riêng lẻ một các cụ thể.

#### Express hỗ trợ chúng ta một phương thức tên sendFile() để nhận dữ liệu từ các file và thể hiện chúng qua trình duyệt.

#### Đầu tiên hãy tạo một thư mục views để chứa các giao diện của bạn.

#### 

#### Bên trong views hãy tạo 1 file html và đặt cho nó là home

#### Bên trong home.html, hãy setup theo cách bạn muốn, tôi có một ví dụ như sau:

#### 

#### Bây giờ thay vì chúng ta “send” lên dòng chữ mặc định ở file siteController

#### 

#### Ta đã có thể thiết lập giao diện theo cách tuỳ thích bằng views.

#### Sử dụng đối tượng path để dễ dàng tạo các đường dẫn.

#### 

#### Sử dụng “sendFile” nói trên để nhận đoạn mã html từ thư mục views.

#### 

#### Đến đây ta sẽ được kết quả.

#### 

#### Lưu ý: sendFile không nhận file theo kiểu liên kết tương đối, nó nhận đường dẫn tuyệt từ chính sever để lấy file(sever ở đây đang là localhost) cho nên bạn cần truy cập đường dẫn từ chính nơi chứa chương trình của bạn.

#### Tôi đã viết sai “views” để bạn có thể thấy đường dẫn cần nhận hoàn chỉnh.

#### 

#### Các đường dẫn tương đối kiểu này sẽ sinh ra lỗi. hãy chú ý.

#### 

### Dùng template-engine

#### Nhìn chung quy thì cấu trúc view-controllers đã hình thành khá ổn nhưng ta vẫn có một vài vấn đề.

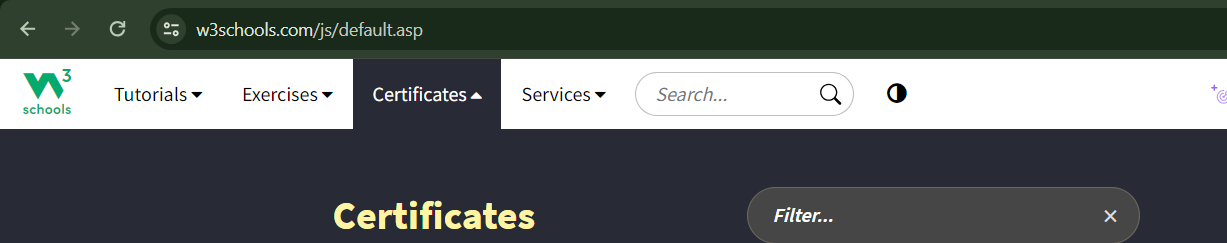
#### Nếu chỉ sử dụng các file html thông thường thì các mã của ta sẽ khá là lộn sộn và chúng sẽ lập đi lập lại một vài phần.

#### Lấy ví dụ từ trang W3schools, có thể thấy những điểm chung của toàn bộ trang là chúng đã sở hữu cùng một loại navbar.

#### 

#### 

#### Đây cũng vậy



#### Nếu sử dụng html thuần thì ta sẽ luôn phải code lại điểm chung này và trên hết sẽ không tận dụng tối đa các thành phần chung khác. Lúc này sử dụng một template-engine là lựa chọn hoàn hảo.

#### Các template-engine sẽ giúp chúng ta phân chia từng thành phần trong trang và bố trí chúng theo các quy luật. các thành phần chúng như header và footer có thể được đặt mặc định mà không cần phải code lại. những thành giống nhau dù không cùng một trang vẫn có thể được tận dụng hiệu quả.

#### Cuối cùng template-engine hỗ trợ trong việc nhận và xuất dữ liệu một cách dễ dàng và đầy hiệu quả.

#### Template engine chúng ta sử dụng ở đây là “express-handlebars’

#### Mở terminal và tải xuống thư viện: npm i express-handlebars

#### 

#### Sau khi cài đặt hãy cùng cấu hình nó.

#### Trong src hãy tạo một thư mục mới với tên resources, đây sẽ là nơi chúng ra lưu các file giao diện như html, css,….

#### Đưa thư mục views vào trong resources.

#### 

#### Giờ hãy cùng cấu hình cho template engine khi nảy.

#### Truy cập trang [express-handlebars](https://www.npmjs.com/package/express-handlebars). Bạn có thể thấy họ cấu hình nó trông như này.

#### 

#### 

#### Bạn sẽ thấy cấu hình của thư mục views như sau.

#### 

#### Hãy cấu hình theo đó.

#### 

#### Do cấu trúc của chúng ta đã khác một vài chổ cho nên ta sẽ phải tự cấu hình lại theo cách của mình. Điều này cũng sẽ giúp nắm rõ hơn trong việc chỉnh sửa trong tương lai.

#### Trở lại file chính. Hãy nạp các phương thức như họ.

#### 

#### Dùng const thay import để thống nhất code với các phương thức trước.

#### 

#### Phương thức app.engine() cho phép định một template engine mới cho express của bạn. ở đây là file có đuôi ‘.hbs’.

#### App.set(‘view engine’, ‘.hbs’) cho biết những file được render sẽ có dạng đuôi là ‘.hbs’. mà không cần phải thiết lập khi gọi. ví dụ nếu bạn render một file ‘index.hbs’ thì chỉ cần viết là ‘index’ express sẽ tự hiểu và thêm đuôi ‘.hbs’ vào file.

#### App.set(‘views’, ‘./views’) sẽ cho biết thư mục chứa các file cần render, ở đây là views.

#### Tuy nhiên đường dẫn của chúng ta sẽ phải là.

#### 

#### Sử dụng đường dẫn tuyệt đối.

#### Sau đó quay lại controller site và dùng phương thức render.

#### 

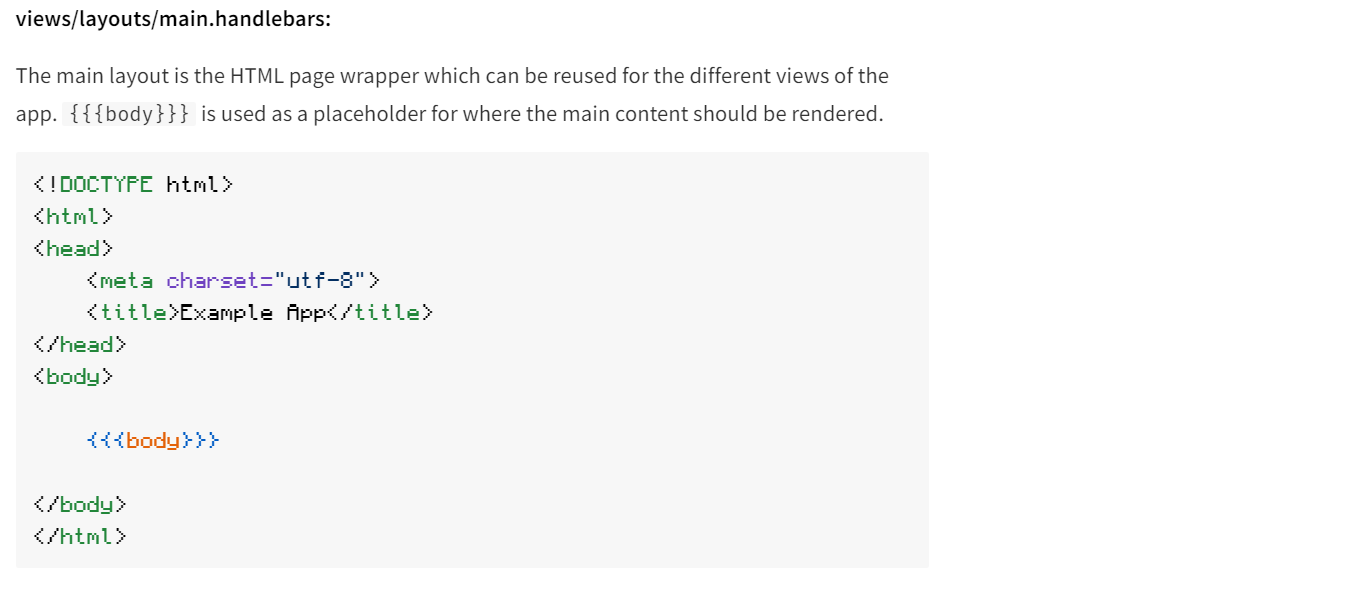
#### Render() cho phép bạn nhận dữ liệu từ cấu trúc template engine đã thiết lập, hoạt động tương tự sendFile().

#### Tuy nhiên có thể ta sẽ gặp phải lỗi này.

#### 

#### Trình duyệt trống chơn và không thấy bất kì giao diện nào của file home được render ra.

#### Tiếp tục đọc hướng dẫn ta sẽ thấy.



#### Ta cần cấu trúc file main.handlebars thành thành phần html chính và sử dụng {{{body}}} để nhận nội dung render.

#### Nói đơn giản là main.handlebars lúc này sẽ làm khung và các file render chỉ cần chứa các thẻ là được mà không cần phải viết lại khung html hoàn chỉnh.

#### 

#### Cấu trúc main.handlebars như hướng dẫn.

#### Tại home.handlebars chỉ cần chứa nội dung.

#### 

#### Lúc này sẽ thấy mọi thứ đã bình thường trở lại.

#### 

#### Tuy nhiên ta sẽ sửa đổi một chút, cảm thấy đuôi ‘.handlebars’ quá dài dòng, tôi muốn đổi thành đuôi ‘.hbs’. ta sẽ thiết lập nó như sau.

#### 

#### Thiết lập trong đối tượng phương thức engine(‘.hbs’, engine()) nhận đối số đầu tiên chính là module template engine được thiết lập. đối tượng engine() được tạo bởi express-handlebars là một object sẽ nhận các tham số thiết lập(cụ thể xem trên trang express-handlebars). ‘extname’ sẽ thay đổi tên đuôi file handlebars mặc định.

#### Giờ đổi tên các file thành đôi ‘.hbs’ là được.

#### 

### Sass và SCSS

#### Không chỉ các file HTML mà các file CSS cũng phải được sắp xếp và tối ưu một cách cụ thể nhất để dễ dàng chỉnh sửa và tìm kiếm trong tương lai.

#### Trong resources hãy tạo một thư mục css để tiến hành lưu các css của bạn.

#### 

#### Lúc này ta sẽ thử liên kết css cho file main.hbs.

#### Trong file main.css tôi có một vài css như sau.

#### 

#### 

#### Tuy nhiên lúc này điều đó sẽ không hoạt động.

#### 

#### Các đường dẫn tương đối lúc này sẽ không hoạt động nếu chúng ta nạp css, nhưng ta cũng sẽ không thể gán một đường dẫn tuyệt đối trong file HTML.

#### Để sử lí vấn đề này ta sẽ public css và nhận nó từ trình duyệt.

#### Cắt file css từ resources và đưa nó vào public.

#### 

#### Đến đây ta đã có thể nhận css từ trình duyệt.

#### 

#### Đừng quên gắn đường dẫn.

#### 

#### Nó đã hoạt động trở lại.

#### ở đây ta sẽ lại có một vấn đề. Nếu ta viết tất cả css trong một file thì về lâu dài sẽ rất khó quản lí nhưng nếu chia nhỏ chúng thì ta sẽ cần phải có rất nhiều thẻ “link” trong “head”.

#### 

#### 

#### Xét về tối ưu và hiệu quả thì điều này không hề khả thi. Lúc này ta cần một cách có thể cho ta vừa giảm số lượng link nạp mà cũng vừa có thể dễ dàng quản lí các css.

#### Sass chính là một lựa chọn. Sass hoạt động dựa trên SCSS một ngôn ngữ được xây dựng trên CSS nhưng mạnh hơn. Thông qua đó ta cũng sẽ tận dụng SCSS để dễ dàng CSS hơn.

#### Trên terminal tải thư viện node-sass như thông thường: npm I sass

#### 

#### Sau khi đã cài đặt hãy cùng setup cho sass để nó có thể dịch từ scss sang css cho chúng ta vì trình duyệt chỉ hiểu css chứ không hiểu scss.

#### ở package.json hãy tạo một script mới.

#### 

#### Cú pháp để sử dụng là: sass --flag <input> <output>

#### --flag là các chức năng muốn sử dụng. ở đây tôi dùng –watch để lắng nghe sự thay đổi css mới và áp dụng cho trình duyệt.

#### <input> chính là nơi chứa file scss.

#### <output> chính là nơi chứa file css.

#### Để chạy chức năng script dạng này hãy gõ vào terminal: npm + run + tên\_script

#### 

#### Sau đó bạn sẽ thấy có 2 file được tạo ra ở thư mục css.

#### 

#### File “.map” kia khá dư thừa cho nên nếu bạn muốn loại bỏ nó có thể thêm một flag khác như thế này.

#### --embed-source-map sẽ giúp bạn loại bỏ file “.map” kia.

#### Nhưng ở đây để tối ưu css tôi sẽ compile theo thư mục. compile nhiều file scss và chuyển về thành một file css duy nhất, vì việc code của ta sẽ được làm trên scss nên không phải lo nếu file css duy nhất kia quá dài.

#### Để compile theo folder: sass –flag <folder input>:<folder output>

#### 

### Xây header và footer

#### Header và footer là 2 thành phần không thể thiếu của một trang web, hai thành phần thường sẽ không đổi ở từng trang con. Ta sẽ xây dựng cấu trúc mặc định cho 2 dạng này đối với những trang web của ta như sau

#### 

#### Tại partials, tạo 2 file là footer.hbs và header.hbs. bên trong 2 file đấy hãy xây dựng header và footer theo cách của bạn, ở đây tôi sử dụng boostrap.

#### Tại main.hbs hãy cấu trúc như thế này.

#### 

#### Bạn có thể thay đổi tên 1 cách tuỳ thích tuy nhiên phải giữ cấu trúc {{>Tên\_file}}

#### Sau khi hoàn thành bạn sẽ có một trang web với 3 thành phần cơ bản.

#### 

#### Lưu ý gần việc xây dựng giao diện sẽ không được đề cập quá nhiều ở chủ đề này.

### Xây dựng modal

#### Dữ liệu là một phần quan trọng tất yếu của một hệ thống. để quản lí một dữ liệu ta thường lưu vào các database. Có nhiều chương trình giúp quản lí các dữ liệu tuy nhiên ở đây chúng ta sẽ xử dụng noSQL cụ thể là mongodb.

#### Tải mongodb và cài đặt. sau khi cài đặt hãy vào mongodb compass.

#### Kết nối đến URI của bạn. mặc định sẽ tự kết nối đến localhost.

#### ở bên trái bạn sẽ thấy một cửa sổ với nhiều các database mặc định.

#### 

#### ở phần Databases, ấn dấu cộng và tạo một database mới với collection tên groups(tuỳ ý).

#### 

#### 

#### Lưu lại và bạn sẽ thấy một database mới bên trái

#### 

#### Truy cập và groups và chọn add data.

#### 

#### Hãy chọn insert documents(tuỳ dạng dữ liệu của bạn) thêm một số dữ liệu thử.

#### Tôi tạo một vài data có dạng này

#### 

#### Bây giờ hay cùng kết nối đến database vừa tạo.

#### Tạo thư mục config ở src. Bên trong config hãy tạo một file chứ phương dùng để kết nối đến database.

#### 

#### Phương thức kết nối là một async function thực hiện tác vụ bất đồng bộ.

#### 

#### ở phương thức connect của mongo hãy đưa địa chỉ URI của bạn vào.

#### Function sẽ kết nối với database. Nếu thành công sẽ trả về then, thất bại trả về catch.

#### ở file main, ta sẽ nhận đối tượng vừa xuất ra.

#### 

#### Dùng nó và gọi đến phương thức connect

#### 

#### Nếu hiện thế này thì có nghĩa đã thành công.

#### Nếu hiện lỗi, hãy thử kiểm tra các phương thức hay đường dẫn đã chính xác hay chưa.

#### Công việc tiếp theo để có thể lấy data từ mongo thì đầu tiên ta phải cài đặt nó vào dự án.

#### Tại terminal tải thư viện mongoose. Thư viện giúp hỗ trợ các chức năng khi làm việc với mongodb.

#### Tạo một thư mục tên “app” bên trong sẽ chứa controllers và modals(tạo thêm modals).

#### 

#### Trong modals tạo một file trùng tên với collection vừa tạo ở trên.

#### 

#### Trên trong hãy tạo một Schema mới với cấu trúc như này.

#### 

#### Tên xuất ra phải trùng tên với collection, không cần thêm “s” vì mongo sẽ tự thêm vào.

#### Đặt những thuộc tính cho đối tượng giống với những gì bên collection ở mongo. Nếu bạn đặt thiếu, dữ liệu nhận được hoặc truyền đi sẽ bị sai.

#### 

#### Trở về controller của site.

#### Mongoose cung cấp cho ta các chức năng để nhận dữ liệu. “find” dùng để nhận dữ liệu của một collection cụ thể.

#### Tạo một biến để gán modal muốn nhận.

#### 

#### Sau đó dùng phương thức find để tìm dữ liệu của modal đó(ở đây là groups) bên trong mongodb.

#### 

#### Do “find” là một promise nên ta sẽ nhận đầu ra thông qua .then() và .catch().

#### Khi nhận được dữ liệu(nghĩa là then).

#### Ta sẽ trả về một object cho file ta render. ở đây là home.hbs

#### 

#### Khi không nhận được (nghĩa là catch). Ta sẽ trả về một điều gì đó để nhận biết không nhận được dữ liệu. bạn có thể tạo một trang error để báo lỗi lên. ở đây tôi “send” một dòng text cho gọn.

#### 

#### Tuy nhiên khi chạy chương trình bạn sẽ không thấy bất kì dữ liệu nào cả do ta vẫn chưa tạo giao diện cho chúng.

#### Tôi dùng kiểu gia diện này của boostrap.

#### 

#### Truy cập [vào đây](https://getbootstrap.com/docs/4.6/components/card/) nếu bạn cũng muốn dùng nó.

#### Giờ thì hãy xem ta cần làm gì để lấy dữ liệu truyền vào một file handlebars [tại đây](https://handlebarsjs.com/guide/#what-is-handlebars).

#### Đối với object, ta có dạng như sau:

#### 

#### Tuy nhiên chúng ta không chỉ có duy nhất một đối tượng như trong ví dụ.

#### Bên dưới phần đó nữa bạn sẽ thấy một kiểu liệt kê khác mà trang đề cập cho một mảng nhiều thành phần.

#### 

#### Cách này sẽ dùng “each” để lập qua toàn bộ phần tử tìm được và dùng “this” để chỉ phần tử đó.

#### Áp dụng điều này với file home.hbs

#### 

#### Do ở controller chúng ta truyền biến “groups” nên ở đây sẽ nhận lại “groups”.

#### Ta có 3 phần tử gồm img, name, và descripton cho nên sẽ nhận 3 phần tử đó cho từng phần giao diện.

#### Lưu lại vào chạy chương trình xem kết quả thế nào.

#### 

#### Có thể thấy đã có 2 phần tử được tạo đúng với số lượng trong database tuy nhiên vấn đề là ta không nhận được các dữ liệu mong muốn.

#### Vấn đề này do dữ liệu từ mongo được chuyển qua JS sẽ không phải là một dạng dữ liệu mà JS có thể đọc được. ta cần chuyển nó về object thuần tí để JS có thể đọc. thực hiện như sau:

#### Tạo một thư mục util trong src. ở util sẽ dùng để viết các tác vụ chuyển đổi dữ liệu.

#### Tôi tạo file để chuyển đổi dữ liệu từ mongoose tên “setupMongoose.js”

#### 

#### Bên trong ta sẽ setup như sau:

#### ” multipMongooseToObject” dùng để chuyển đổi hàng loạt (một mảng) dữ liệu cùng lúc.

#### Trong khi “mongooseToObject” dùng để chuyển đổi 1 dữ liệu duy nhất.

#### Trở lại controller và nhận đối tượng vừa xuất ra bên trên.

#### 

#### Khi này trước khi gửi dữ liệu đi ta sẽ dùng 1 trong 2 phương thức ấy để chuẩn hoá lại dữ liệu.

#### 

#### Lưu lại và sẽ thấy kết quả.

#### 

#### Điều chỉnh lại một chút bằng boostrap.

#### 

#### Ta được kết quả cuối cùng.

## Các chức năng chuyên sâu

### Lấy một phần tử cụ thể.

#### ở trên, ta đã biết cách lấy hàng loạt dữ liệu bằng “find”. Tuy nhiên không phải lúc nào cũng cần lấy tất cả dữ liệu ra kiểu vậy. ví dụ để xem thông tin của một chủ đề nào đó thì ta chỉ cần lấy dữ liệu của chủ đề đó mà không phải của tất cả chủ đề.

#### Mongoose vẫn có thể giúp ta làm điều đó.

#### Trước mắt hãy chỉnh sửa lại một chút.

#### Tôi tách phần groups ra làm một phần riêng gồm router và controller của riêng nó.

#### 

#### Cấu trúc bên trong tương tự như việc làm ở site thôi, không có gì cả.

#### 

#### 

#### Cũng đừng quên liên kết vào bên router tổng nhá.

#### 

#### Lúc này đường dẫn đã chỉ đúng những gì ta cần

#### 

#### Mục tiêu ở đây là khi bấm vào một trong các nội dung trên nó sẽ chuyển ta đến một trang nội dung cụ thể và chi tiết hơn.

#### Khi này bấm xem thì nó chỉ đưa ta đến một link như này.

#### 

#### Dấu “#” có nghĩa là chính trang hiện tại nên nó không chuyển ta đi đâu cả.

#### Điều này là vì đường link của ta đã hướng đến “#”

#### 

#### Vậy hãy thay đổi nó.

#### Tôi muốn nó sẽ thay đổi theo đúng phần tử tôi chọn.

#### 

#### Vì ta đang làm việc tại router “group” nên đừng quên thêm “/groups” vào như ta đã cấu trúc router trước đó.

#### Lúc này khi ta bấm vào xem, URL sẽ đưa đúng đường dẫn mong muốn.

#### 

#### Tất nhiên sẽ có vấn đề xảy ra

#### 

#### Lỗi báo rằng nó không thể nhận được một cái router “/groups/tên\_phần\_tử”

#### Vậy tạo đường dẫn thôi.

#### Tại controller hãy tạo một phương thức mang tên “slug” kiểu như một cái đuôi động. vì ta sẽ không biết cụ thể phần tử đính vào URL là gì, không thể cứ mỗi phần tử lại tạo một router nên ta sẽ dùng dạng slug để làm một URL động.

#### 

#### Tôi sẽ send một đoạn text lên để xác định rằng đã làm đúng.

#### Đừng quên hãy định nghĩa router nữa nhé.

#### 

#### Chú ý một router động sẽ bao gồm “:” nó sẽ cho biết phần tử phía sau dấu “:” sẽ biến đổi chứ không cố định.

#### Bây giờ đường dẫn đã hoạt động đúng như mong đợi.

#### 

#### Vấn đề là làm sao để biết nó là cái nào trong database mà lấy ra. ở đây sẽ có nhiều cách để trích xuất một phần tử từ database. Ta có thể lấy qua tên của chúng hoặc id của chúng.

#### 

#### Vậy thì ta có gì nào.

#### Lúc nảy ta đã đính một thứ rất riêng biệt của mỗi phần tử lên URL đó là name.

#### 

#### Vậy thì giờ ta có thể dựa vào điều đó để trích dữ liệu ra.

#### Tôi tách một dữ liệu được truyền trên URL bằng “req.params” params ở đây chính là một trong những yêu cầu được truyền đi (khi nảy ta đã truyền đi name).

#### 

#### Tôi biến dữ liệu thành json cho dễ quan sát.

#### 

#### Ta nhận được một key có tên là slug và nó chưa phần tên khi nảy ta chuyền đi.

#### Có vẻ như key slug ở đây là mặc định và không liên quan đến phương thức slug của ta đã đặt.

#### Vì khi đổi tên phương thức thì nó cũng không thay đổi.

#### 

#### Vậy key đó ở đâu mà có.

#### Nếu chú ý thì nó chính là phần sau dấu “:” lúc nảy ở router. Nó được lấy làm key cho dữ liệu của ta

#### 

#### Vậy giờ ta đã có thể hiểu được cách để cấu trúc cho hợp lí.

#### Tôi đổi phương thức thành “getGroup” cho dễ hiểu.

#### 

#### Nhớ rằng đây là cấu trúc do bạn tạo, bạn có thể đặt bất kì thứ gì bạn muốn. nếu không muốn quá rờm rà cứ đặt là slug vì nó sẽ luôn hợp lí dù bạn có lọc theo bất cứ dữ liệu gì.

#### Từ dữ liệu json ở trên tôi có thể lấy phần name ra.

#### 

#### 

#### Mongoose hỗ trợ một phương thức tên là “findOne” nó sẽ tìm một đối tượng theo thông tin truyền vào.

#### Tôi làm như sau.

#### 

#### Tôi dùng finOne để tìm phần tử có tên như chính cái tôi đã đính trên URL. Nếu tìm được thì tôi in nó ra dạng json để kiểm tra, không thấy thì tôi bỏ qua không làm gì cả.

#### Mẹo: có thể tải extension này để dễ làm việc hơn.

#### 

#### Đây chính là dữ liệu được trả về từ database

#### 

#### Rồi giờ chỉ cần cho một chút giao diện vào là được.

#### Tạo một folder để chứa những giao diện liên quan đến group.

#### 

#### Giờ truyền dữ liệu nhận được cho group-info.hbs

#### Lưu ý cấu trúc nhận dữ liệu đơn của handlebars.

#### 

#### Dữ liệu nhận được cũng cần được chuyển về object thuần cho javascript đọc nên hãy dùng đến những phương thức chuyển đổi.

#### Lưu ý gần hãy render đúng file nhé.

#### 

#### Tuỳ theo việc tạo giao diện của bạn mà kết quả sẽ cho ra những giao diện phù hợp. ở đây tôi chỉ ví dụ đơn giản bằng boostrap.

#### 

#### Kết quả cuối cùng

#### 

### Tạo một trang quản trị viên(ADMIN)

#### Bạn có thể tách trang quản trị viên của mình ra làm một thành phần riêng biệt nếu nó nhiều tính năng và phức tạp về cấu trúc bằng các chia cây thư mục thành dạng user và admin. Và cấu trúc cho trang admin tương tự lúc trước là được.

#### 

#### Tuy nhiên ở đây tôi sẽ gộp chung chúng lại là một mà không cần phải tách ra.

#### Tôi tạo một đường dẫn mới cho trang quản trị của mình thế này.

#### 

#### Việc này tương tự như cách tạo các router phía trên nên tôi sẽ không nói lại nữa.

#### Tuy nhiên có một điểm khác biệt. trang quản trị dùng để quản lí các dữ liệu trong trang nên ta sẽ không cần đến các header và footer của trang người dùng nữa.

#### Tôi tạo một layout riêng cho admin để tái cấu trúc của nó lại.

#### 

#### Bên trong hãy quăng header và footer đi.

#### Điều này không có nghĩa ta không cần header và footer, những trang admin phức tạp vẫn cần thanh điều hướng, tuy nhiên ở đây ta sẽ tạo một form đăng nhập cho admin trước nên tạm thời sẽ không cần những thành phần đó.

#### 

#### Để thay đổi layout hãy render theo cấu trúc như sau:

#### 

#### Đối số thứ 2 lúc này của phương thức render nhận một object với các định dạng bạn cần thay đổi. ở đây ta chỉ định layout là admin ta vừa tạo.

#### Xây dựng router như bình thường và ta được kết quả.

#### 

#### Hãy tạo một form đăng nhập theo ý của bạn nhưng phải gồm tên đăng nhập và mật khẩu để ta có thể làm việc tiếp theo.

#### Tô điểm cho nó một chút và ta có kết quả sau:

#### 

#### Tiếp theo hãy cùng cấu hình cho việc đăng nhập

#### Trong database, tạo một collection mới với tên là admins. Nơi này sẽ dùng để quản lí các tài khoản admin.

#### 

#### Bên trong tôi tạo một tài khoản admin theo cấu trúc sau:

#### 

#### ở modals, hãy tạo một cấu trúc mới cho loại dữ liệu này.

#### 

#### Thế là t đã có được một dạng cấu trúc để xác định dữ liệu trong database phục vụ cho việc đăng nhập.

#### Đầu tiên, tôi tạo một phương thức login trong controller.

#### 

#### Vai trò của nó sẽ gửi dữ liệu từ form đăng nhập lên sever và trả về kết quả trùng khớp.

#### Việc tạo router sẽ có một chút thay đổi. vì đây là gửi dữ liệu đi nên ta sẽ không sử dụng GET mà thay vào đó là sử dụng POST để đảm bảo tính bảo mật dữ liệu.

#### 

#### Tuy nhiên nếu lúc này ta bấm nút đăng nhập ở form thì sẽ xảy ra lỗi. do cấu trúc form của ta chưa đúng.

#### Bình thường form sẽ có method mặc định là GET và action = “#”

#### 

#### Action = “#” có nghĩa là yêu cầu sẽ được gửi tại chính trang hiện tại nên sẽ chẳng có gì xảy ra cả.

#### 

#### URL chỉ thêm “#” và nó không hướng về đâu cả.

#### Việc đầu tiên ta cần thay đổi action đến router mà ta muốn thực hiện đăng nhập đó là /admin/login

#### 

#### Thay đổi method cho trùng vớ method của router nếu không bạn sẽ bị lỗi do không tồn tại phương thức GET cho router /admin/login.

#### ở trình duyệt hãy mở devtool lên và chọn vào network.

#### Nhập dữ liệu vào các trường input và bấm đăng nhập, bạn sẽ thấy có một form login được gửi đi.

#### 

#### ấn vào và ta sẽ thấy nó chính là dữ liệu mà ta nhập ở input.

#### Có một điều lưu ý ở đây là dữ liệu được submit trong form sẽ chỉ được gửi khi thẻ đó có name, nếu bạn không đặt name cho thẻ, dữ liệu có nó sẽ không được gửi. và để gửi dữ liệu thì type của nút đăng nhập phải là submit.

#### Tôi nhận dữ liệu được gửi đi bằng “req.body”.

#### Nếu “req.param” lấy dữ liệu của phương thức GET thì “req.body” sẽ áp dụng cho POST.

#### 

#### Tôi sẽ thử chuyển dữ liệu đó thành json.

#### 

#### Có thể thấy mặc dù chúng ta đã đến đúng đường dẫn đã cài đặt nhưng không thấy bất cứ dữ liệu nào.

#### 



#### Hãy thử console.log ra xem.

#### 

#### Và ta thấy vấn đề ở đây.

#### 

#### Body không nhận được bất cứ dữ liệu nào được truyền đi. Điều này xảy ra do một thứ được gọi là middleware không mặc định gắn dữ liệu từ form vào body cho ta. Bạn có thể tìm hiểu kĩ hơn việc này ở đây.

#### Để giải quyết vấn đề này ta có thể cài một thư viện tên là boby-parser. Tuy nhiên ở những bản express mới thì nó đã được tích hợp sẵn.

#### Trở lại main.js và gắn nó vào.

#### 

#### Với điều này ta đã thật sự có thể nhận được dữ liệu từ form.

#### 

#### 

#### Tiếp theo ta sẽ kiểm tra dữ liệu đó bên trong database. Dùng phương thức findOne để tìm theo yêu cầu.

#### 

#### Tôi tìm theo 2 dạng thông tin trong form. Nếu truyền thành công thì tôi nhận về một kết quả từ database nếu không thì tôi “next” nó đi.

#### 

#### Kiểm tra xem kết quả nhận lại được có tồn tại không, nếu không có đối tượng trong database thì result sẽ là “null” còn có nó sẽ trả về thông tin của tài khoản ấy.

#### Áp dụng if else cơ bản để kiểm tra.

#### 

#### Với điều này ta đã có thể hoàn thành việc xử lí đăng nhập cơ bản. bạn có thể cấu trúc nó phức tạp hơn trong tương lai bằng việc xác thực email hoặc xác thực khác để đảm bảo bảo mật tốt hơn.