BOR3**BỘ CÔNG THƯƠNG**

**TRƯỜNG CAO ĐẲNG KỸ THUẬT CAO THẮNG**

**KHOA ĐIỆN TỬ - TIN HỌC**

…..  …..



ĐỒ ÁN TỐT NGHIỆP

*ĐỀ TÀI*

TÌM HIỂU FRAMEWORK SAILSJS VÀ XÂY DỰNG WEBSITE BÁN HÀNG

Giáo viên hướng dẫn: LƯU TUỆ HẢO

Sinh viên thực hiện:

1. TRẦN QUỐC LỘC **0306141150**

2. PHAN THÀNH NGHĨA **0306141157**

LỚP: CĐTH14B

KHÓA: 2014 - 2017

TP.HỒ CHÍ MINH, ngày …. tháng 07 năm 2017BOR3**BỘ CÔNG THƯƠNG**

**TRƯỜNG CAO ĐẲNG KỸ THUẬT CAO THẮNG**

**KHOA ĐIỆN TỬ - TIN HỌC**

…..  …..



ĐỒ ÁN TỐT NGHIỆP

*ĐỀ TÀI*

TÌM HIỂU FRAMEWORK SAILSJS VÀ XÂY DỰNG WEBSITE BÁN HÀNG

Giáo viên hướng dẫn: LƯU TUỆ HẢO

Sinh viên thực hiện:

1. TRẦN QUỐC LỘC **0306141150**

2. PHAN THÀNH NGHĨA **0306141157**

LỚP: CĐTH14B

KHÓA: 2014 - 2017

TP.HỒ CHÍ MINH, ngày …. tháng 07 năm 2017

**LỜI MỞ ĐẦU**

**LỜI CẢM ƠN**

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

TP.HỒ CHÍ MINH, ngày … tháng … năm 2017

**Giáo viên hướng dẫn**

(Ký và ghi rõ họ tên)

**NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

TP.HỒ CHÍ MINH, ngày … tháng … năm 2017

**Giáo viên phản biện**

(Ký và ghi rõ họ tên)

**NHẬN XÉT CỦA HỘI ĐỒNG BẢO VỆ LUẬN VĂN TỐT NGHIỆP**

TP.HCM, Ngày ……..Tháng…….Năm 2017

Thư kí hội đồng Chủ tịch hội đồng

**MỤC LỤC**

[Chương 1: GIỚI THIỆU NODE.JS 1](#_Toc486376151)

[1.1. Giới thiệu Node.js 1](#_Toc486376152)

[1.2. Cài đặt Node.js 2](#_Toc486376153)

[1.3. Tạo máy chủ 5](#_Toc486376154)

[1.4. Console.log() 7](#_Toc486376155)

[Chương 2: GIỚI THIỆU SAILS JS FRAMEWORK 8](#_Toc486376156)

[2.1. Giới thiệu Sails js framework 8](#_Toc486376157)

[2.2. Cài đặt và tạo project Sails js 8](#_Toc486376158)

[2.3. Tạo models và controllers 11](#_Toc486376159)

[2.4. Kết nối cơ sở dữ liệu mysql server 16](#_Toc486376160)

[2.5. Waterline 18](#_Toc486376161)

[2.5.1. Models 18](#_Toc486376162)

[2.5.2. Validation 21](#_Toc486376163)

[2.5.3. Truy vấn dữ liệu 23](#_Toc486376164)

[2.5.3.1. Create 23](#_Toc486376165)

[2.5.3.2. Find 24](#_Toc486376166)

[2.5.3.3. FindOne 25](#_Toc486376167)

[2.5.3.4. Update 25](#_Toc486376168)

[2.5.3.5. Destroy 26](#_Toc486376169)

[2.5.3.6. Sort 27](#_Toc486376170)

[2.5.3.7. Limit 29](#_Toc486376171)

[2.5.3.8. Skip 30](#_Toc486376172)

[2.5.3.9. Query 32](#_Toc486376173)

[2.6. Tổ chức thư mục của Sails js Framework 33](#_Toc486376174)

[2.6.1. Api 33](#_Toc486376175)

[2.6.1.1. Controllers 33](#_Toc486376176)

[2.6.1.2. Models 33](#_Toc486376177)

[2.6.1.3. Policies 34](#_Toc486376178)

[2.6.1.4. Services 34](#_Toc486376179)

[2.6.1.5. Respones 34](#_Toc486376180)

[2.6.2. Assets 34](#_Toc486376181)

[2.6.3. Config 35](#_Toc486376182)

[2.6.3.1. Blueprints.js 35](#_Toc486376183)

[2.6.3.2. Bootstrap.js 35](#_Toc486376184)

[2.6.3.3. Connections.js 35](#_Toc486376185)

[2.6.3.4. Cors.js 36](#_Toc486376186)

[2.6.3.5. Csrf.js 36](#_Toc486376187)

[2.6.3.6. Globals.js 36](#_Toc486376188)

[2.6.3.7. Htpp.js 36](#_Toc486376189)

[2.6.3.8. I18n.js 36](#_Toc486376190)

[2.6.3.9. Local.js 37](#_Toc486376191)

[2.6.3.10. Log.js 38](#_Toc486376192)

[2.6.3.11. Models.js 38](#_Toc486376193)

[2.6.3.12. Policies.js 38](#_Toc486376194)

[2.6.3.13. Routes.js 38](#_Toc486376195)

[2.6.3.14. Session.js 38](#_Toc486376196)

[2.6.3.15. Socket.js 38](#_Toc486376197)

[2.6.3.16. Views.js 38](#_Toc486376198)

[2.6.4. Node\_modules 39](#_Toc486376199)

[2.6.5. Tasks 39](#_Toc486376200)

[2.6.6. Views 40](#_Toc486376201)

[2.7. Route 40](#_Toc486376202)

[2.8. Controller 42](#_Toc486376203)

[2.9. View 43](#_Toc486376204)

[2.10. Locals 44](#_Toc486376205)

[2.11. Layout 46](#_Toc486376206)

[2.12. Session và cookie 46](#_Toc486376207)

[2.13. Req (request) 48](#_Toc486376208)

[2.13.1. Req.param() 48](#_Toc486376209)

[2.13.2. Req.allParams() 50](#_Toc486376210)

[2.13.3. Req.xhr 52](#_Toc486376211)

[2.13.4. Req.file() 52](#_Toc486376212)

[2.13.5. Req.url 54](#_Toc486376213)

[2.13.6. Req.method 54](#_Toc486376214)

[2.13.7. Req.query 54](#_Toc486376215)

[2.13.8. Req.path 54](#_Toc486376216)

[2.14. Req.flash 55](#_Toc486376217)

[2.15. Res (respone) 56](#_Toc486376218)

[2.15.1. Res.view() 56](#_Toc486376219)

[2.15.2. Res.json() 56](#_Toc486376220)

[2.15.3. Res.send() 57](#_Toc486376221)

[2.15.4. Res.serverError() 58](#_Toc486376222)

[2.15.5. Res.redirect() 58](#_Toc486376223)

[2.16. Policies 58](#_Toc486376224)

[2.17. Services 60](#_Toc486376225)

[Chương 3: TEMPALTE ENGINE 62](#_Toc486376226)

[3.1. Giới thiệu 62](#_Toc486376227)

[3.2. EJS 62](#_Toc486376228)

[3.3. Tags ejs 62](#_Toc486376229)

[3.3.1. Tag <%= %> 62](#_Toc486376230)

[3.3.2. Tag <% %> 63](#_Toc486376231)

[3.3.3. Tag <%- %> 63](#_Toc486376232)

[3.3.4. Tag <%/\* \*/%> 64](#_Toc486376233)

[Chương 4: GÓI MỞ RỘNG 65](#_Toc486376234)

[4.1. Giới thiệu npm 65](#_Toc486376235)

[4.2. Tải gói mở rộng 65](#_Toc486376236)

[4.3. Các gói đã sử dụng 66](#_Toc486376237)

[4.3.1. Momentjs 66](#_Toc486376238)

[4.3.2. Accountingjs 68](#_Toc486376239)

[4.3.3. url-slug 69](#_Toc486376240)

[Chương 5: GIỚI THIỆU ĐỀ TÀI 70](#_Toc486376241)

[5.1. Chức năng 70](#_Toc486376242)

[5.1.1. Quản lý sản phẩm 70](#_Toc486376243)

[5.1.1.1. Thêm sản phẩm 70](#_Toc486376244)

[5.1.1.2. Sửa thông tin sản phẩm 70](#_Toc486376245)

[5.1.1.3. Xóa thông tin sản phẩm 70](#_Toc486376246)

[5.1.1.4. Xem danh sản phẩm 70](#_Toc486376247)

[5.1.2. Quản lý danh mục thiết bị 71](#_Toc486376248)

[5.1.2.1. Thêm thông tin thiết bị mới 71](#_Toc486376249)

[5.1.2.2. Sửa thông tin thiết bị 71](#_Toc486376250)

[5.1.2.3. Xóa thông tin thiết bị 71](#_Toc486376251)

[5.1.2.4. Xem danh sách thiết bị 71](#_Toc486376252)

[5.1.3. Quản lý nhà sản xuất 72](#_Toc486376253)

[5.1.3.1. Thêm nhà sản xuất mới 72](#_Toc486376254)

[5.1.3.2. Sửa thông tin nhà sản xuất 72](#_Toc486376255)

[5.1.3.3. Xóa thông tin nhà sản xuất 72](#_Toc486376256)

[5.1.3.4. Xem danh sách nhà sản xuất 72](#_Toc486376257)

[5.1.4. Quản lý thông tin tài khoản 72](#_Toc486376258)

[5.1.4.1. Thêm thông tin thành viên mới 73](#_Toc486376259)

[5.1.4.2. Cấp lại mật khẩu 73](#_Toc486376260)

[5.1.4.3. Xóa thông tin thành viên 73](#_Toc486376261)

[5.1.4.4. Sửa thông tin tài khoản 73](#_Toc486376262)

[5.1.4.5. Xem danh sách tài khoản 73](#_Toc486376263)

[5.1.5. Quản lý đơn hàng 73](#_Toc486376264)

[5.1.5.1. Xem danh sách đơn hàng 74](#_Toc486376265)

[5.1.5.2. Xem chi tiết đơn hàng 74](#_Toc486376266)

[5.1.6. Thống kê doanh thu 74](#_Toc486376267)

**DANH MỤC HÌNH**

[Hình 1.1 Bản cài đặt Node.js theo từng hệ điều hành. 2](#_Toc486376268)

[Hình 1.2 Bản càn đặt node.js 2](#_Toc486376269)

[Hình 1.3 Điều khoản của node.js 3](#_Toc486376270)

[Hình 1.4 Nơi chứa cài đặt node.js 3](#_Toc486376271)

[Hình 1.5 Chọn cấu hình node.js 4](#_Toc486376272)

[Hình 1.6 Bắt đầu tiến trình cài đặt node.js 4](#_Toc486376273)

[Hình 1.7 Thông báo tiến trình cài đặt kết thúc 5](#_Toc486376274)

[Hình 1.8 Tạo một máy chủ với node.js 5](#_Toc486376275)

[Hình 1.9 Mở cửa sổ cmd 6](#_Toc486376276)

[Hình 1.10 Chạy tập tin node.js index 6](#_Toc486376277)

[Hình 1.11 Kết quả sau khi chạy node.js 7](#_Toc486376278)

[Hình 1.12 Hiển thị biến sieuxe bằng console.log. 7](#_Toc486376279)

[Hình 1.13 Kết quả của biến sieuxe 7](#_Toc486376280)

[Hình 2.1 Mở cửa sổ cmd 8](#_Toc486376281)

[Hình 2.2 Cửa sổ cmd của thư mục chứ project 9](#_Toc486376282)

[Hình 2.3 Câu lệnh cmd để cài đặt sails js 9](#_Toc486376283)

[Hình 2.4 Quá trình cài đặt thành công 10](#_Toc486376284)

[Hình 2.5 Câu lệnh để tạo một project sails 10](#_Toc486376285)

[Hình 2.6 Project demo tạo thành công 11](#_Toc486376286)

[Hình 2.7 Mở cmd trong thư mục chứa framework 11](#_Toc486376287)

[Hình 2.8 Thông báo đã tạo thành công controller demo 12](#_Toc486376288)

[Hình 2.9 Tập tin đã tạo thành công 12](#_Toc486376289)

[Hình 2.10 Thông báo đã tạo thành công api demo2 12](#_Toc486376290)

[Hình 2.11 Tập tin Demo2Controller đã tạo thành công 13](#_Toc486376291)

[Hình 2.12 Tập tin Demo2 đã tạo thành công 13](#_Toc486376292)

[Hình 2.13 Mở cửa sổ cmd 13](#_Toc486376293)

[Hình 2.14 Lệnh cmd để chạy Sails js 14](#_Toc486376294)

[Hình 2.15 Sails js đã chạy thành công 14](#_Toc486376295)

[Hình 2.16 Trình duyệt web hiển thị khi chạy Sails js thành công. 15](#_Toc486376296)

[Hình 2.17 Tắt Sails js 15](#_Toc486376297)

[Hình 2.18 Kết nối cơ sở dữ liệu mysql 16](#_Toc486376298)

[Hình 2.19 connection và migrate sau khi tắt chú thích 16](#_Toc486376299)

[Hình 2.20 Tải gói kết nối mysql hoàn tất 17](#_Toc486376300)

[Hình 2.21 Định nghĩa thuộc tính của model 18](#_Toc486376301)

[Hình 2.22 Bảng user vừa định nghĩa ở model user 19](#_Toc486376302)

[Hình 2.23 Tắt thuộc tính không cần thiết của model 21](#_Toc486376303)

[Hình 2.24 Kết quả sau khi tắt các thuộc tính không cần thiết của model 21](#_Toc486376304)

[Hình 2.25 Validation trong Sails js 22](#_Toc486376305)

[Hình 2.26 Hàm create để thêm một dòng mới vào cơ sỡ dữ liệu 23](#_Toc486376306)

[Hình 2.27 Kết quả sau khi thực thi hàm create thành công 24](#_Toc486376307)

[Hình 2.28 Hàm find để tìm kiếm dữ liệu 24](#_Toc486376308)

[Hình 2.29 Kết quả sau khi thực thi hàm find thành công 24](#_Toc486376309)

[Hình 2.30 Hàm find tìm kiếm dữ liệu có tham số. 25](#_Toc486376310)

[Hình 2.31 Kết quả sau khi thực thi hàm find có tham số 25](#_Toc486376311)

[Hình 2.32 Hàm update để cập nhật dữ liệu 26](#_Toc486376312)

[Hình 2.33 Kết quả sau khi thực thi hàm update thành công 26](#_Toc486376313)

[Hình 2.34 Hàm delete đẽ xóa dữ liệu 26](#_Toc486376314)

[Hình 2.35 Thông báo đã xóa thành công 27](#_Toc486376315)

[Hình 2.36 Kết quả bên trong cơ sỡ dữ liệu 27](#_Toc486376316)

[Hình 2.37 Bảng dữ liệu 27](#_Toc486376317)

[Hình 2.38 Sắp xếp giảm dần theo id 28](#_Toc486376318)

[Hình 2.39 Kết quả sau khi sắp xếp 28](#_Toc486376319)

[Hình 2.40 Bảng dữ liệu 29](#_Toc486376320)

[Hình 2.41 Hàm limit giới hạn dữ liệu hiển thị 29](#_Toc486376321)

[Hình 2.42 Kết quả sau khi giới hạn dữ liệu hiển thị 30](#_Toc486376322)

[Hình 2.43 Bảng dữ liệu. 30](#_Toc486376323)

[Hình 2.44 Hàm skip lấy kết quả từ dòng thứ 2 của cơ sở dữ liệu. 31](#_Toc486376324)

[Hình 2.45 Kết quả hiển thị sau khi dùng hàm skip 31](#_Toc486376325)

[Hình 2.46 Bảng dữ liệu 32](#_Toc486376326)

[Hình 2.47 32](#_Toc486376327)

[Hình 2.48 Kết quả của hàm query 32](#_Toc486376328)

[Hình 2.49 Tổ chức thư mục của sails js 33](#_Toc486376329)

[Hình 2.50 Bên trong thư mục api 33](#_Toc486376330)

[Hình 2.51 Bên trong thư mục assets 34](#_Toc486376331)

[Hình 2.52 Các tập tin bên trong thư mục config 35](#_Toc486376332)

[Hình 2.53 Bên trong thư mục tasks 40](#_Toc486376333)

[Hình 2.54 Bên trong thư mục views 40](#_Toc486376334)

[Hình 2.55 Định nghĩa 1 route 40](#_Toc486376335)

[Hình 2.56 Hành động demo trong DemoController. 41](#_Toc486376336)

[Hình 2.57 Kết quả thực hiện khi chạy route. 41](#_Toc486376337)

[Hình 2.58 Truyền tham số trên route 41](#_Toc486376338)

[Hình 2.59 Lấy giá trị của tham số id 41](#_Toc486376339)

[Hình 2.60 Giá trị của tham số id 42](#_Toc486376340)

[Hình 2.61 Controller. 42](#_Toc486376341)

[Hình 2.62 Tạo route 43](#_Toc486376342)

[Hình 2.63 Hành động index trong DemoController 43](#_Toc486376343)

[Hình 2.64 index.ejs 43](#_Toc486376344)

[Hình 2.65 Kết quả sau khi chạy 44](#_Toc486376345)

[Hình 2.66 Tạo route. 44](#_Toc486376346)

[Hình 2.67 Hành động index trả về view index với dòng chữ xin chào. 45](#_Toc486376347)

[Hình 2.68 Khai báo để nhận biến hello. 45](#_Toc486376348)

[Hình 2.69 Hiển thị giá trị của biến hello. 45](#_Toc486376349)

[Hình 2.70 Cú pháp layout. 46](#_Toc486376350)

[Hình 2.71 Phần bị thay đổi. 46](#_Toc486376351)

[Hình 2.72 Thời gian sống của session. 47](#_Toc486376352)

[Hình 2.73 Session và cookie. 47](#_Toc486376353)

[Hình 2.74 Session hello với nội dung là hello. 47](#_Toc486376354)

[Hình 2.75 Giá trị của session hello. 48](#_Toc486376355)

[Hình 2.76 Truyền tham số trên route. 48](#_Toc486376356)

[Hình 2.77 Lấy giá trị của tham số id. 48](#_Toc486376357)

[Hình 2.78 Giá trị của tham số id. 49](#_Toc486376358)

[Hình 2.79 form nhập liệu. 49](#_Toc486376359)

[Hình 2.80 Route để lấy giá trị của form. 49](#_Toc486376360)

[Hình 2.81 Hành động getForm và postForm. 50](#_Toc486376361)

[Hình 2.82 Giá trị của ô nhập liệu. 50](#_Toc486376362)

[Hình 2.83 form với 2 ô nhập liệu. 51](#_Toc486376363)

[Hình 2.84 Hành động postForm trong DemoContorller. 51](#_Toc486376364)

[Hình 2.85 Kết quả hiển thị của req.allParams(). 51](#_Toc486376365)

[Hình 2.86 Minh họa req.xhr. 52](#_Toc486376366)

[Hình 2.87 Form tải tập tin lên máy chủ. 52](#_Toc486376367)

[Hình 2.88 Route tải tập tin. 52](#_Toc486376368)

[Hình 2.89 Hành đọng getUpload và postUpload. 53](#_Toc486376369)

[Hình 2.90 Kết quả sau khi tải tập tin lên thành công. 53](#_Toc486376370)

[Hình 2.91 Kiểm tra có phương thức gữi lên máy chủ xử lý. 54](#_Toc486376371)

[Hình 2.92 Dùng req.flash xuất thông báo mong muốn. 55](#_Toc486376372)

[Hình 2.93 Hiển thị flash 55](#_Toc486376373)

[Hình 2.94 Thông báo hiển thị khi tải tập tin thành công. 56](#_Toc486376374)

[Hình 2.95 Hành động trả về 1 JSON. 56](#_Toc486376375)

[Hình 2.96 Kết quả của res.json. 57](#_Toc486376376)

[Hình 2.97 Gửi JSON với trạng thái 200. 57](#_Toc486376377)

[Hình 2.98 Kết quả của res.send(). 57](#_Toc486376378)

[Hình 2.99 Áp dụng chính sách sessionAuth cho DemoController. 58](#_Toc486376379)

[Hình 2.100 Chứng thực tài khoản. 59](#_Toc486376380)

[Hình 2.101 Chứng thực tài khoản ở sessionAuth.js. 59](#_Toc486376381)

[Hình 2.102 Áp dụng chính sách sessionAuth cho AdminController. 60](#_Toc486376382)

[Hình 2.103 Tạo một services date. 60](#_Toc486376383)

[Hình 2.104 Sử dụng services date ở DemoController. 61](#_Toc486376384)

[Hình 2.105 Kết quả sau khi thực thi services date thành công. 61](#_Toc486376385)

[Hình 3.1 Cú pháp in ra kết quả của dòng lập for. 62](#_Toc486376386)

[Hình 3.2 Kết quả hiển thị của dòng lập for 62](#_Toc486376387)

[Hình 3.3 Cú pháp của điều kiện if. 63](#_Toc486376388)

[Hình 3.4 Kết quả hiển thị của điều kiện if. 63](#_Toc486376389)

[Hình 3.5 Cú pháp include v2.ejs vào demo.ejs 63](#_Toc486376390)

[Hình 3.6 Nội dung của v2.ejs. 64](#_Toc486376391)

[Hình 3.7 Kết quả sau khi include v2.ejs vào demo.ejs. 64](#_Toc486376392)

[Hình 3.8 Chú thích trong ejs. 64](#_Toc486376393)

[Hình 3.9 Kết quả hiển thị của chú thích. 64](#_Toc486376394)

[Hình 4.1 Cài đặt gói MD5 thành công 65](#_Toc486376395)

[Hình 4.2 Gói md5 trong thưc mục node\_modules. 66](#_Toc486376396)

[Hình 4.3 Sử dụng gói mở rộng md5 66](#_Toc486376397)

[Hình 4.4 Chuỗi ký tự đã được mã hóa 66](#_Toc486376398)

**DANH MỤC BẢNG**

[Bảng 2.1 Bảng định nghĩa các thuộc tính thêm của model 20](#_Toc486376399)

[Bảng 2.2 Bảng quy tắc validation 22](#_Toc486376400)

[Bảng 4.1 Bảng các loại format của Momentjs. 67](#_Toc486376401)

[Bảng 4.2 Quan hệ thời gian của Momentjs 68](#_Toc486376402)

[Bảng 4.3 Bảng format tiền tệ của accountingjs. 69](#_Toc486376403)

[Bảng 4.4 Bảng format số của accountingjs. 69](#_Toc486376404)

**CÁC TỪ VIẾT TẮT**

ĐATN: Đồ án tốt nghiệp

GVHD: Giảng viên hướng dẫn

SVTH: Sinh viên thực hiện

# GIỚI THIỆU NODE.JS

## Giới thiệu Node.js

Node.js là một nền tảng lập trình với mã ngồn mở.

Node.js miễn phí.

Node.js chạy trên nhiều nền tảng khác nhau (Windows, Linux, Unix, Mac OS).

Node.js dùng Javascript ở máy chủ.

Node.js nhập xuất dữ liệu không đồng bộ để giảm thiểu chi phí.

Node.js sử dụng Google V8 JavaScript engine để thực thi mã.

Node.js rất hiệu quả khi xây dựng các ứng dụng chạy thời gian thực ( real-time applications ), như các ứng dụng chat trực tuyến.

Node.js có thể thể tạo, mở, đọc, viết, xóa, và đóng các tệp trên máy chủ.

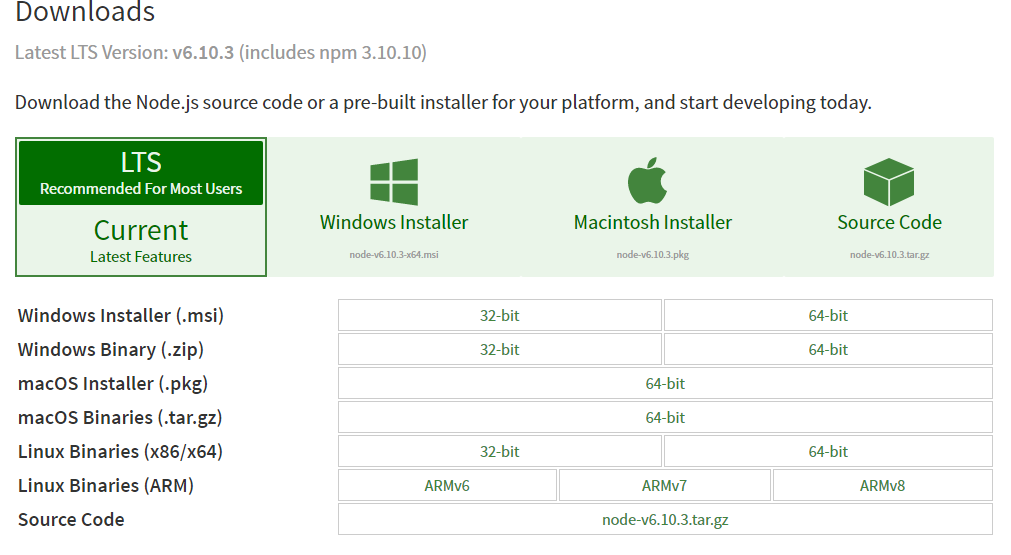
Node.js có thể thêm, xóa, sửa đổi dữ liệu trong cơ sở dữ liệu của bạn.

Node.js có thể tạo ra nội dung trang động.

Các tệp Node.js có phần mở rộng là \*.js.

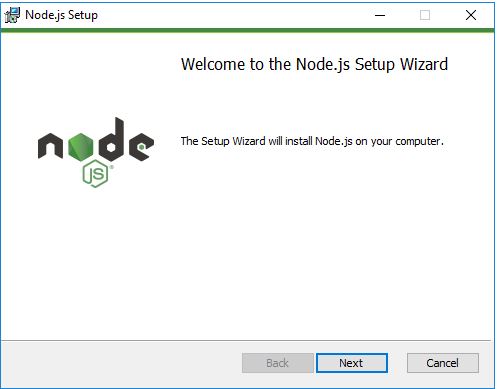
## Cài đặt Node.js

Để cài đặt Node.js truy cập trang [*https://nodejs.org/en/download/*](https://nodejs.org/en/download/) tải bản cài đặt theo hệ điều hành của máy.



Hình . Bản cài đặt Node.js theo từng hệ điều hành.

Cài đặt node.js



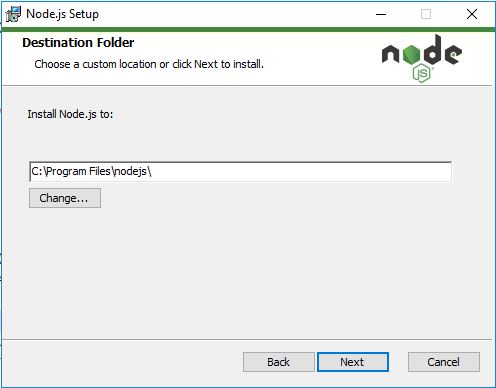
Hình . Bản càn đặt node.js

Chấp nhận điều khoản của node.js và nhấn next để tiếp tục.



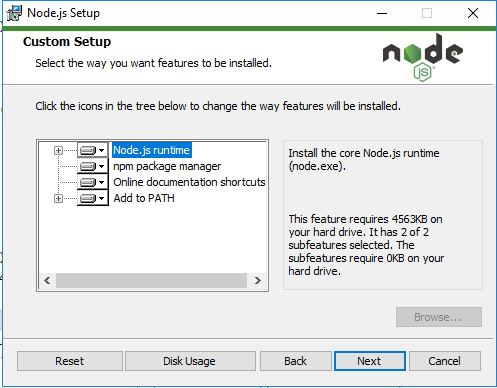
Hình . Điều khoản của node.js

Chọn nơi chứa cài đặt node.js và nhấn next để tiếp tục.



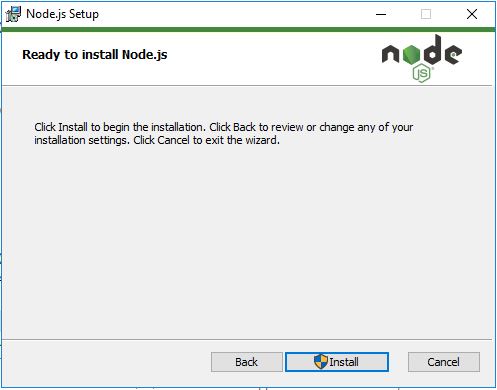
Hình . Nơi chứa cài đặt node.js

Chọn cài đặt theo ý khách hàng hoặc để mặc định rồi nhấn next để tiếp tục.



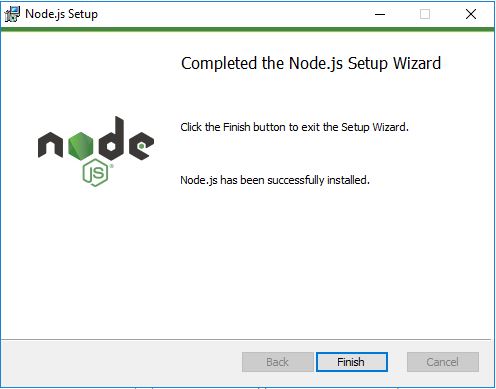
Hình . Chọn cấu hình node.js

Để bắt đầu tiến trình cài đặt node.js nhấn Install để cài đặt.



Hình . Bắt đầu tiến trình cài đặt node.js

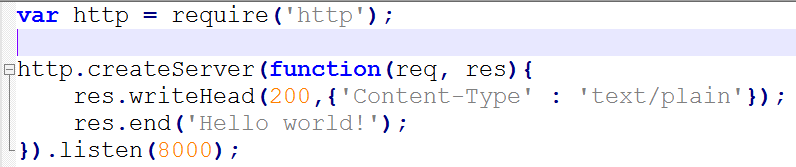
Thông báo khi quá trình cài đặt kết thúc.



Hình . Thông báo tiến trình cài đặt kết thúc

## Tạo máy chủ

Để tạo một máy chủ với node.js tao dùng module http hổ trợ sẵn trong node.js.



Hình . Tạo một máy chủ với node.js

res.writeHead(200,{'Content-Type' : 'text/plain'}); là dung sẽ hiển thị. Ngoài ra còn các Content-Type khác như: ‘html/plain’ để hiển thị html, ‘application/json’ để hiển thị một JSON, ngoài ra còn nhiều kiểu khác. Tham khảo thêm tại: <https://www.iana.org/assignments/media-types/media-types.xhtml>.

.listen(8000) là port ta sẽ chạy máy chủ này.

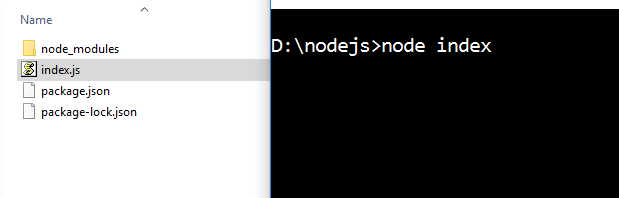
Res.end() là kết thúc việc trả dữ liệu về và cho khách hàng.

Để chạy được tập tin này ta mở của sổ cmd của thư mục chứ tập tin, nhấn tổ hợp phím **Ctrl + Shift + Chuột phải,** chọn **Open command window here** để mở của sổ cmd của thư mục.



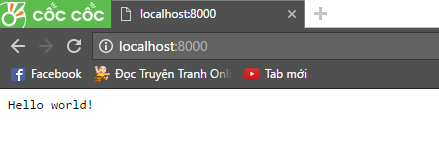
Hình . Mở cửa sổ cmd

Để chạy được node.js ta bâm lệnh: **node [tên tập tin].**



Hình . Chạy tập tin node.js index

Để kiểm tra kết quả ta gõ địa chỉ localhost:8000 để kiểm tra kết quả.

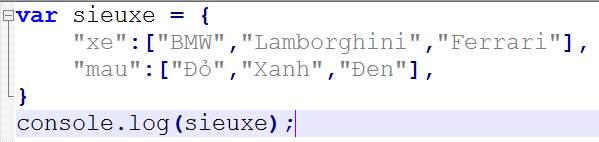


Hình . Kết quả sau khi chạy node.js

## Console.log()

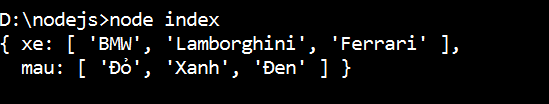
**Console.log()** là hàm thường được sử dụng để debug trong node.js, nó có nhiệm vụ hiển thị ra các kiểu dữ liệu như: số, chuỗi, đối tượng, mảng,… Kết quả sẽ được hiển thị ở của sổ cmd chương trình đang chạy.

Ví dụ:



Hình . Hiển thị biến sieuxe bằng console.log.

Kết quả sau khi chạy.



Hình . Kết quả của biến sieuxe

# GIỚI THIỆU SAILS JS FRAMEWORK

## Giới thiệu Sails js framework

Saisls js là một framework nodejs được xây dựng theo framework Expressjs, Giúp người dùng dễ dàng xây dựng ứng dụng web.

Sails js xây dựng website theo chuẩn MVC.

Sails js sử dụng sức mạnh của ORM Waterline để truy cập dữ liệu, có thể sử dụng bất kỳ cơ sở dữ liệu nào.

Sails tương thích với bất kỳ giao diện người dùng nào: Angular, React, iOS, Android, Windows Phone,…

Sails js miễn phí.

## Cài đặt và tạo project Sails js

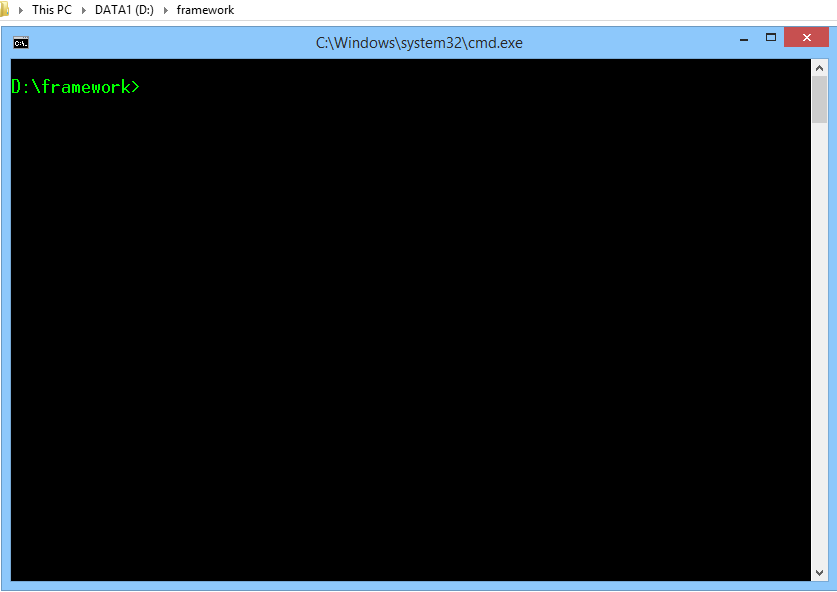
Để cài đặt sails js framework đầu tiên phải cài đặt Node.js trước

Để cài đặt Sails js tạo thư mục chứ project và nhấn tổ hộp phím **Shift + chuột phải** chọn **Open command window here** để mở cửa sổ cmd.



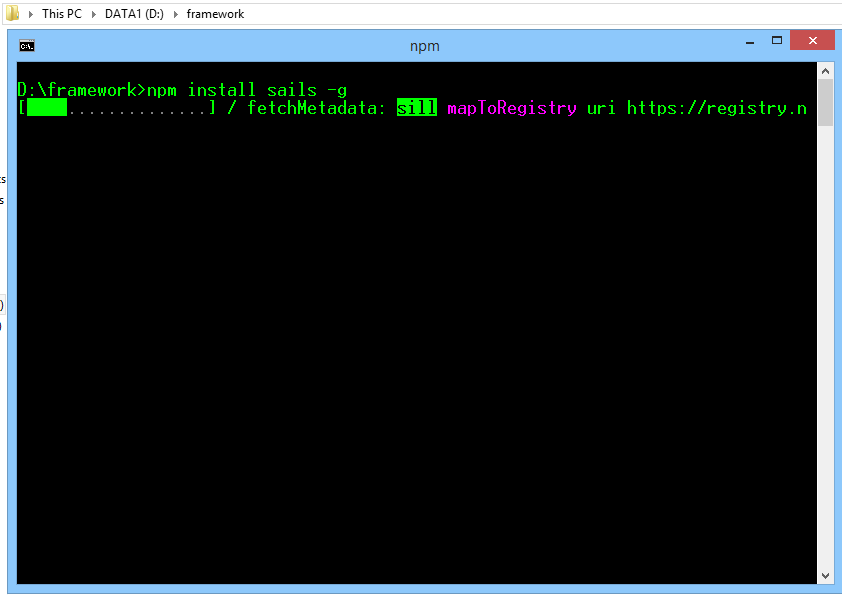
Hình . Mở cửa sổ cmd

Cửa sổ cmd cửa thưc mục chứa Sails js



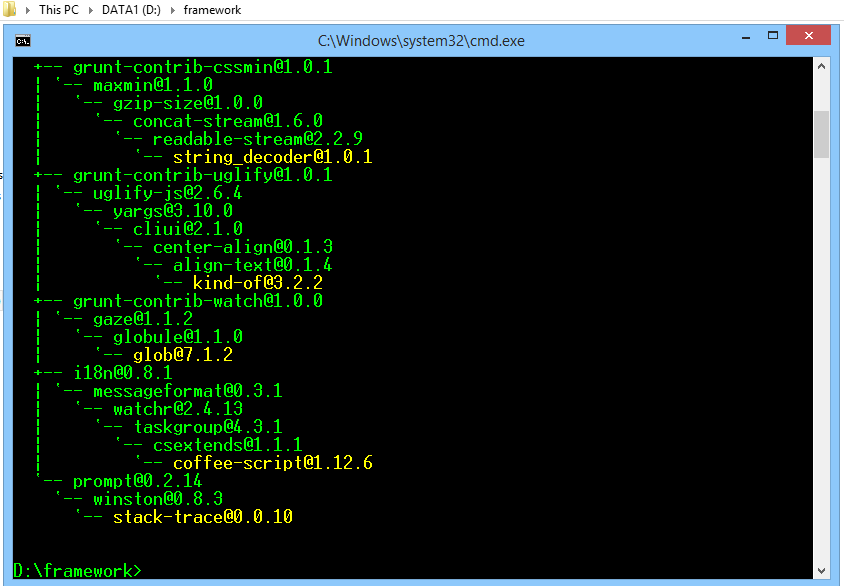
Hình . Cửa sổ cmd của thư mục chứ project

Để cài đặt sails js nhấn lệnh **npm install sails –g** trong cửa sổ cmd vừa bật lên để cho quá trình cài đặt và tải framework diễn ra.



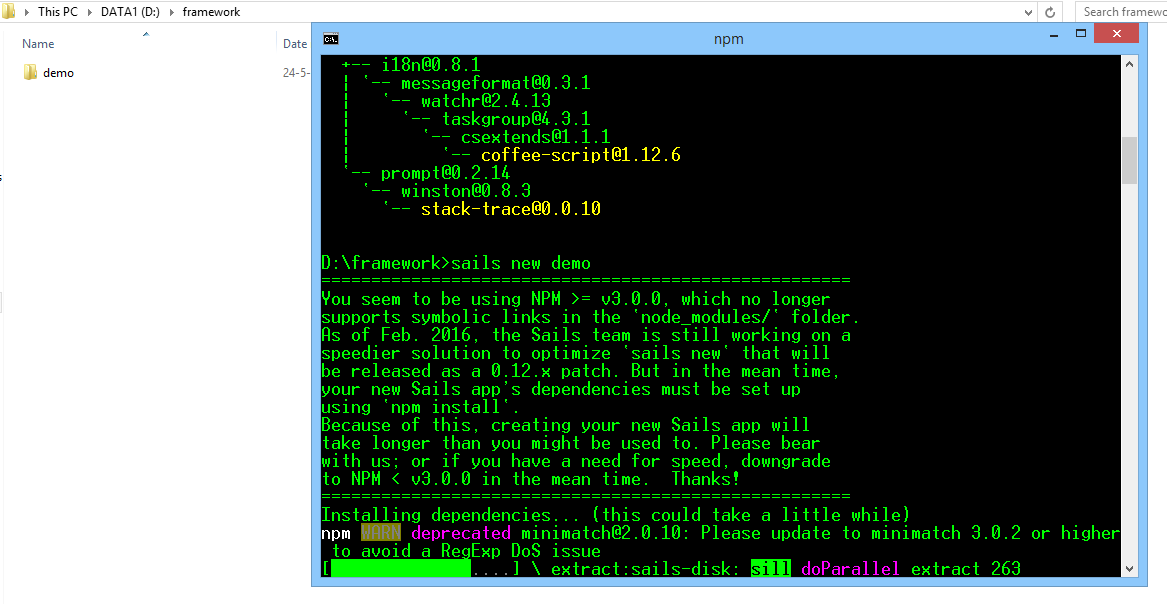
Hình . Câu lệnh cmd để cài đặt sails js

Khi cài đặt thành công sails js framework sẽ có kết quả như hình dưới



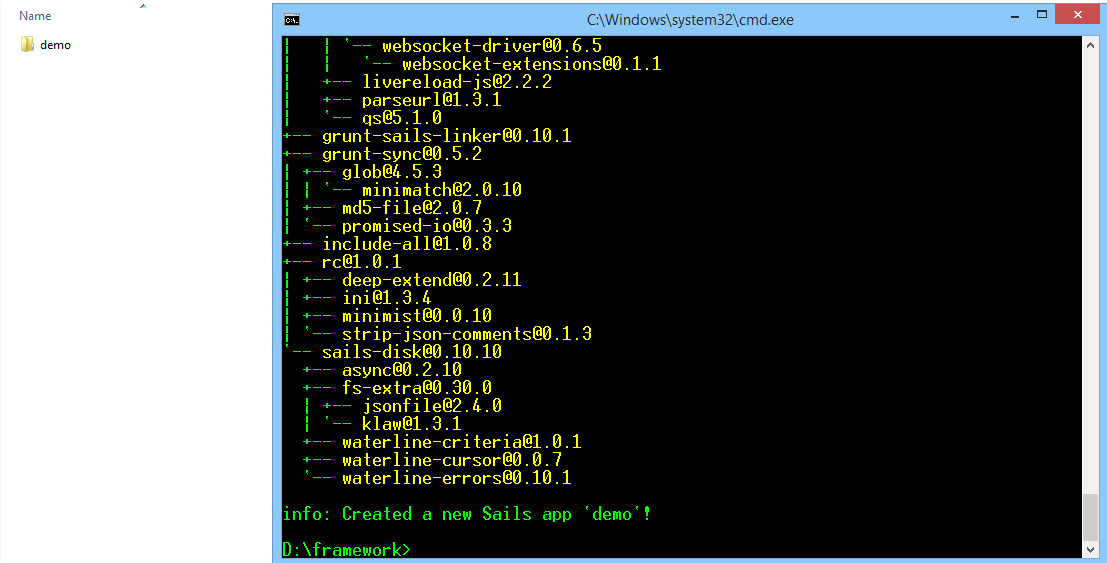
Hình . Quá trình cài đặt thành công

Để tạo project ta nhấn lệnh **sails new [tên project]** ở cửa sổ cmd thư mục



Hình . Câu lệnh để tạo một project sails

Quá trình tạo project thành công



Hình . Project demo tạo thành công

## Tạo models và controllers

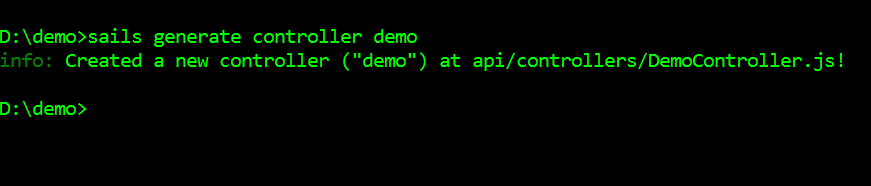
Để tạo một controller hoặc model vô thư mục chứa framework sails, nhấn tổ hợp phím **Ctrl + Shift + Chuột phải**, chọn **Open command window here.**



Hình . Mở cmd trong thư mục chứa framework

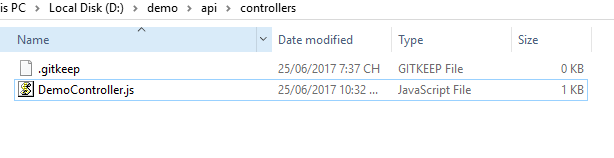
Bấm lệnh **sails generate controller/model [ tên controller hoặc model ].**

Ví dụ tạo controller với tên demo.



Hình . Thông báo đã tạo thành công controller demo

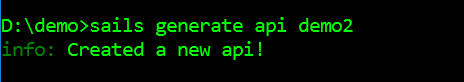
Tập tin DemoController.js đã tạo thành công và nằm trong thư mục api/controller/.



Hình . Tập tin đã tạo thành công

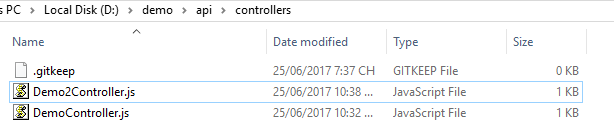
Để tạo Model cũng tương tự như tạo controller.

Để tạo cùng một lúc controller và model Sails js hổ trợ tạo 2 tập tin cùng một lúc, đê tạo ta cũng mở cửa sổ cmd như lúc tạo controller, thay vì bấm lệnh tạo riêng lẽ từng cái một ta bấm lênh: **sails generate api [ tên api ].**

****

Hình . Thông báo đã tạo thành công api demo2

Contrller demo2 đã tạo thành công



Hình . Tập tin Demo2Controller đã tạo thành công

Model demo2 đã tạo thành công

Hình . Tập tin Demo2 đã tạo thành công

Để chạy Sails js ta vào thư mục đã cài đặt framework Sails js, nhấn tổ hợp phím **Ctrl + Shift + Chuột phải**, chọn **Open command window here.**



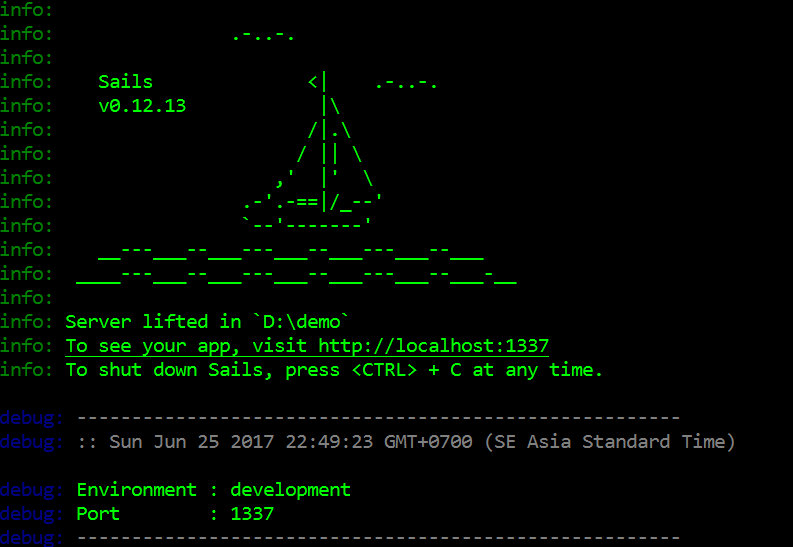
Hình . Mở cửa sổ cmd

Bấm lệnh **sails lift** để chạy Sails js.



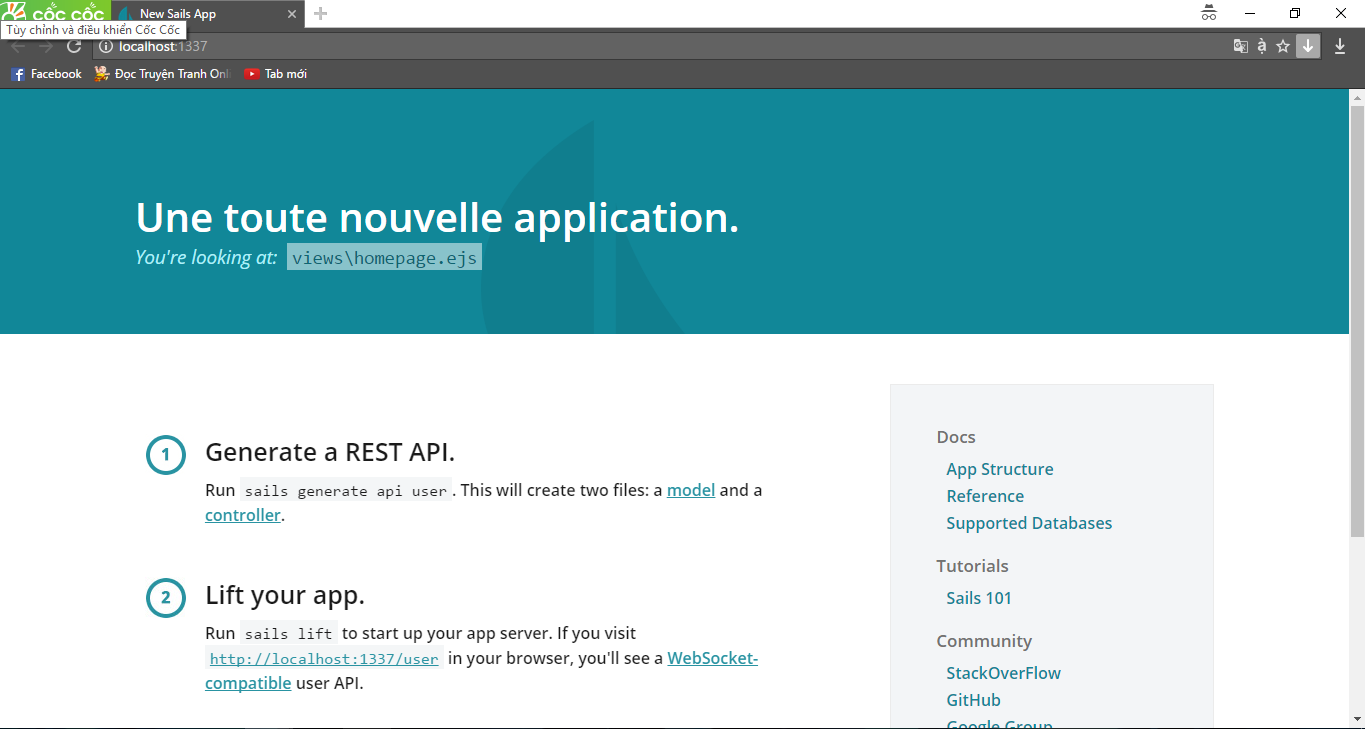
Hình . Lệnh cmd để chạy Sails js

Màn hình hiển thị Sails js đã chạy thành công.



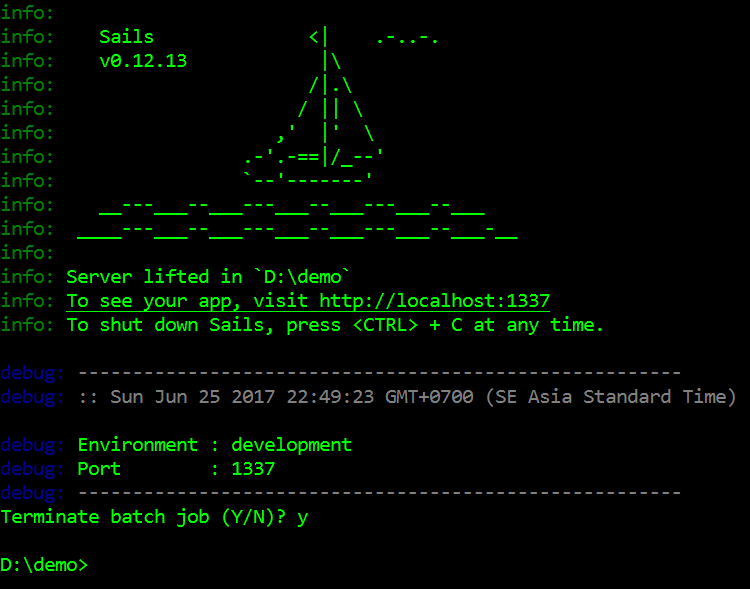
Hình . Sails js đã chạy thành công

Sau khi chạy thành công sails js báo sẽ chạy framework này ở port 1337, để kiểm tra ta vào trình duyệt web và gõ địa chỉ localhost:1337 để kiểm tra ( lưu ý là phải tải các chương trình chạy localhost dưới máy như: Wampserver hoặc XAMPP).



Hình . Trình duyệt web hiển thị khi chạy Sails js thành công.

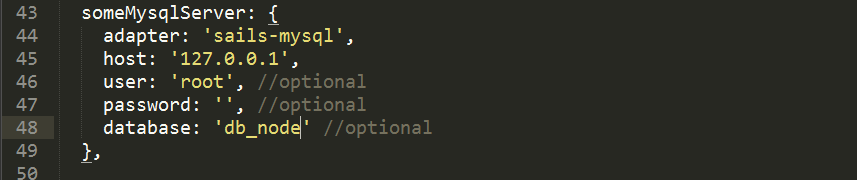
Để tắt Sails js ta bấm tổ hộp phím **Ctrl + C** và gõ **Y** để tắt Sails js



Hình . Tắt Sails js

## Kết nối cơ sở dữ liệu mysql server

Để kết nối với cở sở dữ liệu ta vào mục **config/connections.js**, mở tập tin connections.js, tắt phần chú thích kết nối mysql.



Hình . Kết nối cơ sở dữ liệu mysql

someMysqlServer: tên cài đặt của mình, có thể thay đổi.

Adapter: mặc định sử dụng adapter sails-mysql của framework.

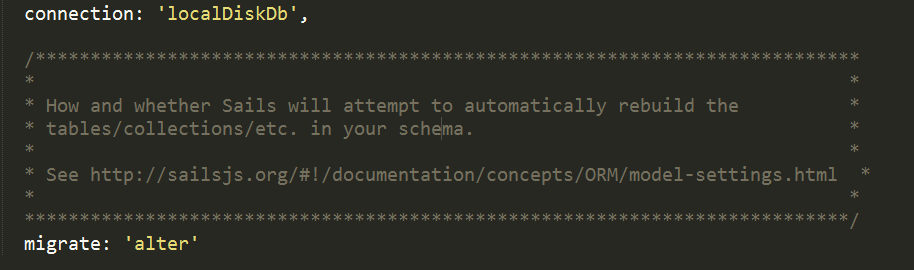
Host: điền host của mình vào, ở đây dùng localhost nên điền là localhost hoặc 127.0.0.1 đều được.

User: tên tài khoản để truy cập mysql server.

Password: mật khẩu để truy cập mysql server.

Database: cơ sở dữ liệu cần dùng để kết nối.

Sau khi thiết lập toàn bộ các thông số cài đặt tiếp tục vào **config/models.js**, mở tập tin models.js để cấu hình kết nối với mysql server. Trong models.js ta thấy connection và migrate đang ở dạng chú thích, ta tắt chú thích của 2 cái.



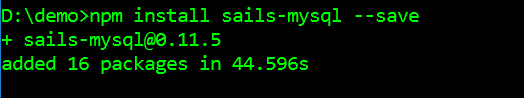
Hình . connection và migrate sau khi tắt chú thích

Connection: dùng để kết nối với cơ sở dữ liệu, ta thây thế “localDiskDb” bằng tên cài đặt “someMysqlServer” mà ta làm ở bên tập tin **config/connections.js.**

Migrate: là quá trình chuyển đổi dữ liệu, mặc định là “alter”. Có 4 kiểu chuyển đổi dữ liệu migrate:

1. Safe: Nếu đã có dữ liệu và bạn không muốn tác động đến cơ sở dữ liệu như: xóa bảng, đổi tên bảng, tên cột,… sau mỗi lần chạy thì safe sẽ không có tương tác giữa các định nghĩa mà bạn làm ở các model api/models, dữ liệu ở các bảng sẽ không có sự thay đổi
2. Alter: Nếu bạn muốn thay đổi như: tên bảng, tên cột, thêm một cột mới, xóa cột đó đi,… thì alter sẽ làm việc đó, các chỉnh sửa của dữ liệu chỉ thay đổi những thứ mà bạn định nghĩa thêm hoặc xóa bớt đi ở các model trong api/models và giữ lại dữ liệu cho bạn.
3. Create: Nếu tạo chưa có bảng nào trong cơ sở dữ liệu và đã định nghĩa các model trong api/models thì migrate sẽ tự động tạo các các bảng theo các định nghĩa mà bạn đã làm.
4. Drop: Xóa tất cả các dữ liệu của bạn sau mỗi lần chạy, và xây dựng lại cơ sở dữ liệu của bạn theo định nghĩa ở các model trong api/models.

Sau khi hoàn thành các cấu hình ở config/connections.js và config/models.js ta bắt đầu tải gói kết nối cơ sở dữ liệu mysql server về để hoàn tất kết nối, mở cửa sổ cmd ở thư mục chứa framework và bấm lệnh **npm install sails-mysql –save** và đợi quá trình hoàn tất.



Hình . Tải gói kết nối mysql hoàn tất

## Waterline

Sails được cài đặt với ORM / ODM mạnh mẽ được gọi là Waterline, một công cụ không đồng bộ dữ liệu mà làm đơn giản hoá đáng kể sự tương tác với một hoặc nhiều cơ sở dữ liệu. Nó cung cấp một lớp trừu tượng trên cơ sở dữ liệu cơ sở, cho phép bạn dễ dàng truy vấn và thao tác dữ liệu của bạn mà không cần viết mã tích hợp cụ thể của nhà cung cấp.

### Models

Model là đại diện cho việc xây dựng cấu trúc cho cơ sỡ dữ liệu, các models được định nghĩa trong thư mục api/models.

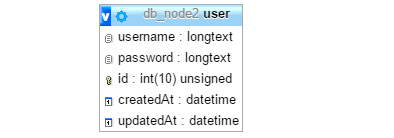
Định nghĩa các thuộc tính của models ta định nghĩa bên trong attributes của model đó.

Ví dụ: định nghĩa model user với các cột username, password với kiểu dữ liệu là text.



Hình . Định nghĩa thuộc tính của model

Sau khi chạy Sails trong cơ sở dữ liệu sẽ hiện ra các bảng mà ta vừa định nghĩa



Hình . Bảng user vừa định nghĩa ở model user

Lưu ý: tên bảng sẽ trùng với tên tập tin model.

**Username** và **password** vừa mới tạo có kiểu dữ liệu trùng với định nghĩa ở model, ngoài ra Warterline tự động phát sinh cột **id** làm khóa chính với thuộc tính mã tăng tự động và **createdAt**, **updatedAt** làm thời gian khi thêm một dòng dữ liệu mới hoặc cập nhật lại dòng dữ liệu đó.

Type có nhiều khác nhau như:

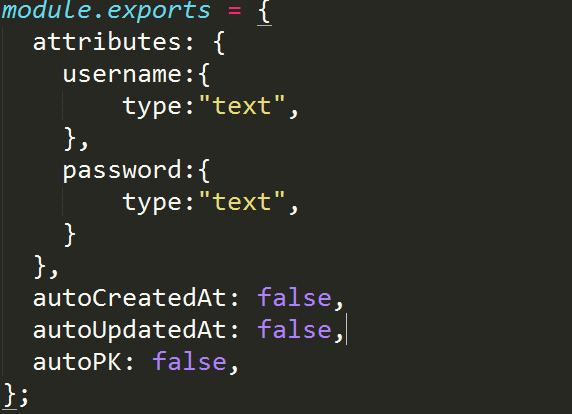
* string
* text
* integer
* float
* date
* datetime
* boolean
* binary
* array
* json
* mediumtext
* longtext
* objectid

Còn có các thuộc tính khác trong việc định nghĩa các models

Bảng 2.1 Bảng định nghĩa các thuộc tính thêm của model

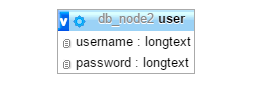
|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| defaultsTo | Chuỗi, số, function | Mặc định một giá trị khi thêm một dòng mới vào trong cơ sở dữ liệu. |
| autoIncrement | True hoặc False | Số tăng tự động, lưu ý chỉ sử dụng được khi kiểu dữ liệu (type) là integer. |
| primaryKey | True hoặc False | Khóa chính, nếu khóa chính có kiểu dữ liều (type) là chuỗi thì phải thêm thuộc tính **size.** |
| Size | Số nguyên | Kích thước bao nhiêu ký tự, chỉ dùng được cho kiểu dữ liệu là String giống như SQL varchar (15). |
| Unique | True hoặc False | Không muốn trùng lập dữ liệu khi thêm vào cơ sỡ dữ liệu. |

Để tắt chức năng tự tạo khóa chính, createdAt, updatedAt ta thêm các thuộc tính bên ngoài **attributes.**



Hình . Tắt thuộc tính không cần thiết của model

Kết quả sau khi chạy lại Sails

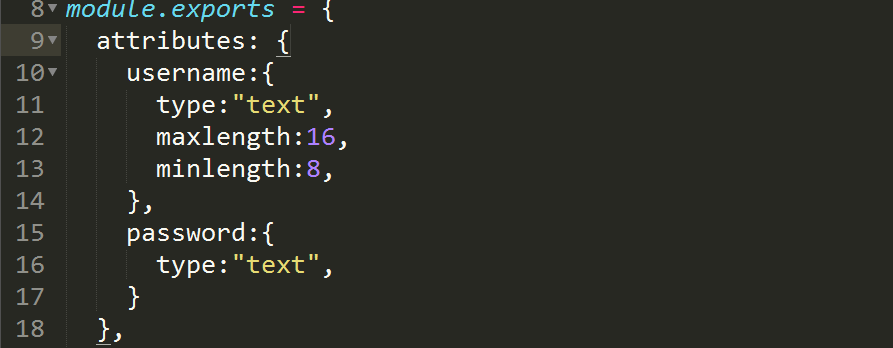


Hình . Kết quả sau khi tắt các thuộc tính không cần thiết của model

### Validation

Sails hổ trợ ràng buộc validation dễ dàng sử dụng bằng cách khai báo các validate trong từng thuộc tính.

Ví dụ ràng buộc username chiều dài 8 tới 16 ký tự.



Hình . Validation trong Sails js

Lưu ý: Sails js không hổ trợ tùy chỉnh thông báo lỗi theo ngôn ngữ, nếu muốn tùy chỉnh thông báo lỗi theo ngôn ngử có thể cài gói mở rộng **sails-hook-validation.**

Bảng . Bảng quy tắc validation

|  |  |  |
| --- | --- | --- |
| Quy tắc | Kiểu dữ liệu | Mô tả |
| maxLength | Số nguyên | Chiều dài tối đa của một chuỗi |
| minLength | Số nguyên | Chiều dài tối thiểu của một chuỗi |
| required | True hoặc False | Trường dữ liệu có bỏ trống hay không |
| unique | True hoặc False | Kiểm tra trùng dữ liệu |
| uppercase | True hoặc False | Kiểm có phải là tra chữ in hoa |
| lowercase | True hoặc False | Kiểm tra có phải là chữ thường |
| email | True hoặc False | Kiểm tra có phải là Email |
| url | True hoặc False | Kiểm tra có phải là dường dẫn |
| max | Số nguyên | Giá trị lớn nhất |
| Min | Số nguyên | Giá trị nhỏ nhất |

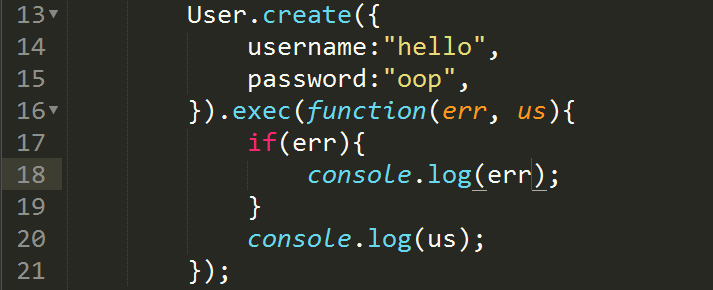
Xem thêm <http://sailsjs.com/documentation/concepts/models-and-orm/validations>

### Truy vấn dữ liệu

Truy vấn dữ liệu với Waterline là một cú pháp dựa trên đối tượng được sử dụng để lấy các bản ghi từ bất kỳ cơ sở dữ liệu nào được hỗ trợ, kết quả trả về thường là một JSON.

#### Create

Để thêm một dòng dữ liệu mới vào cơ sỡ dữ liệu ta dùng hàm **.create()** để thêm dữ liệu vào cơ sỡ dữ liệu. Ví dụ thêm một dòng dữ liệu vào bảng user với username là hello và password là oop.



Hình . Hàm create để thêm một dòng mới vào cơ sỡ dữ liệu

User: tên model mà ta tạo trong thư mục api/models.

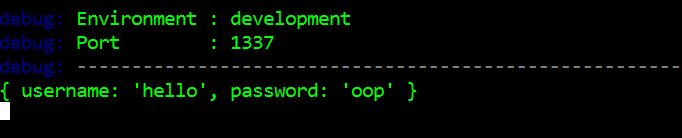
Username, password là các thuộc tính mà ta đã định nghĩa ở User models.

**.exec** là hàm dùng để thực thi truy vấn.

Err: Nếu như bị lỗi err sẽ giữ vai trò lưu lại lỗi phát sinh.

Us: là kết quả thực thi thành công nếu không bị lỗi.

Kết quả sau khi thực thi thành công.

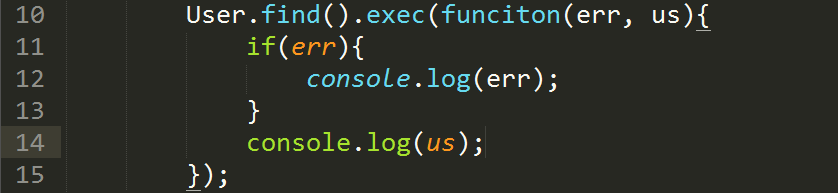


Hình . Kết quả sau khi thực thi hàm create thành công

Nếu viết theo SQL bình thường thì câu truy vấn sẽ là : “INSERT INTO user (username, password) VALUES (“hello”, “oop”)”.

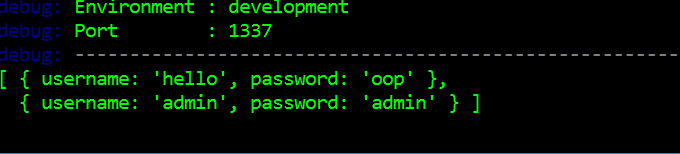
#### Find

**.find()** là hàm dùng để tìm kiếm dữ liệu trong cơ sỡ dữ liệu, nó giống như câu truy vấn SELECT của SQL, ví dụ: tìm kiếm tất cả kết quả trong bảng user.



Hình . Hàm find để tìm kiếm dữ liệu

Kết quả trả về sau khi thực thi thành công câu truy vấn.

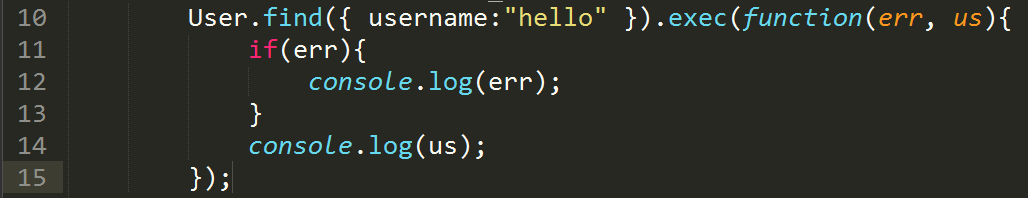


Hình . Kết quả sau khi thực thi hàm find thành công

Nếu viết theo SQL bình thường thì câu truy vấn sẽ là : “SELECT \* FROM user”.

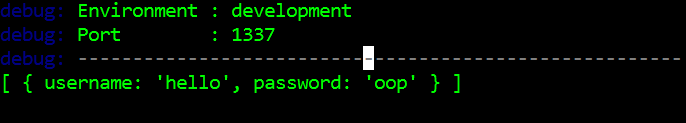
Như vậy hàm find dùng để tìm kiếm kết quả của cơ sỡ dữ liệu, nếu không có tham số được truyền vào trong hàm **.find()** thì hàm sẽ tìm tất cả các kết quả có trong cơ sỡ dữ liệu.

Để tìm những dữ liệu cần thiết trong cơ sỡ dữ liệu ta chỉ cần thêm các tham số vào bên trong hàm **.find()**, tham số ở đây là một JSON. Ví dụ: tìm trong bảng user với username là hello.



Hình . Hàm find tìm kiếm dữ liệu có tham số.

Kết quả sau khi thực thi thành công.



Hình . Kết quả sau khi thực thi hàm find có tham số

Như vậy để truyền tham số vào hàm **.find()** ta truyền vào một JSON với các điều kiện tìm kiếm phải phải giống với các thuộc tính được định nghĩa trong model cần tìm.

Nếu viết theo SQL bình thường thì câu truy vấn sẽ là : “ SELECT \* FROM user WHERE username = “hello” ”.

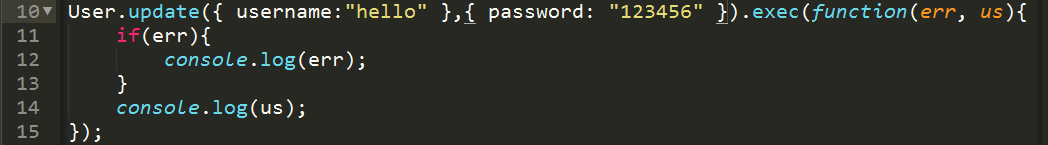
#### FindOne

**.findOne()** là hàm dùng để tìm kiếm kết quả giống với hàm **.find()** nhưng kết quả trả về chỉ có dòng đầu tiên mà hàm tìm kiếm. Giống với truy vấn trong SQL là “SELECT \* FROM user WHERE username = “hello” limit 1”.

#### Update

**.update()** là hàm dũng để cập nhật lại dòng dữ liệu trong cơ sở dữ liệu, giống với hàm UPDATE của SQL.

Ví dụ: cập lại lại password của username hello bằng 123456.



Hình . Hàm update để cập nhật dữ liệu

Lưu ý: model.update({ (1) },{ (2) })

(1): là tìm dòng dữ liệu cần cập nhật.

(2): kết quả cập cần cập nhật.

Kết quả sau khi thực thi thành công.

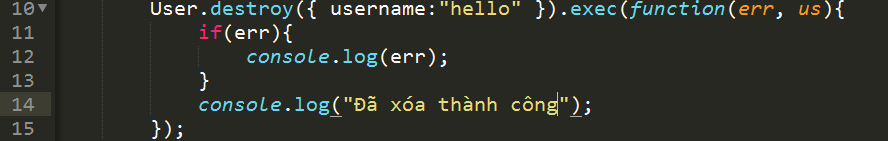


Hình . Kết quả sau khi thực thi hàm update thành công

#### Destroy

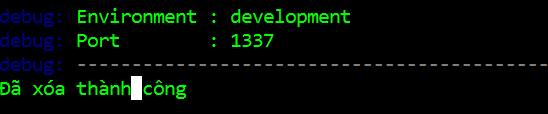
**.destroy()** là hàm dùng để xóa dữ liệu trong cơ sỡ dữ liệu, tương tự như câu truy vấn DELETE của SQL.

**.destroy()** có tham số điền vào là một JSON điều kiện để xóa dữ liệu, ví dụ: xóa dòng dữ liệu với username là hello trong bảng user.



Hình . Hàm delete đẽ xóa dữ liệu

Kết quả sau khi thực hiện.



Hình . Thông báo đã xóa thành công

Kết quả trong cơ sở dữ liệu.



Hình . Kết quả bên trong cơ sỡ dữ liệu

#### Sort

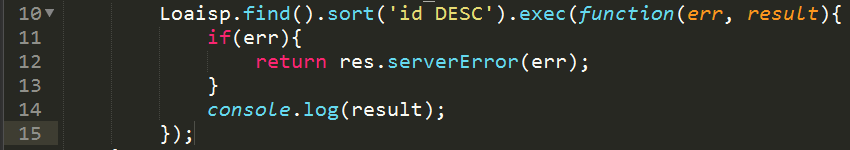
**.sort()** là sắp xếp lại kết quả được hiển thị ra, tham số của **.sort()** là chuổi SQL, ví dụ sắp xếp giảm dần theo id .sort(“id DESC”);

Ví dụ: có bảng cơ sở dữ liệu với 9 dòng dữ liệu, hãy sắp xếp dữ liệu giảm dần theo id.



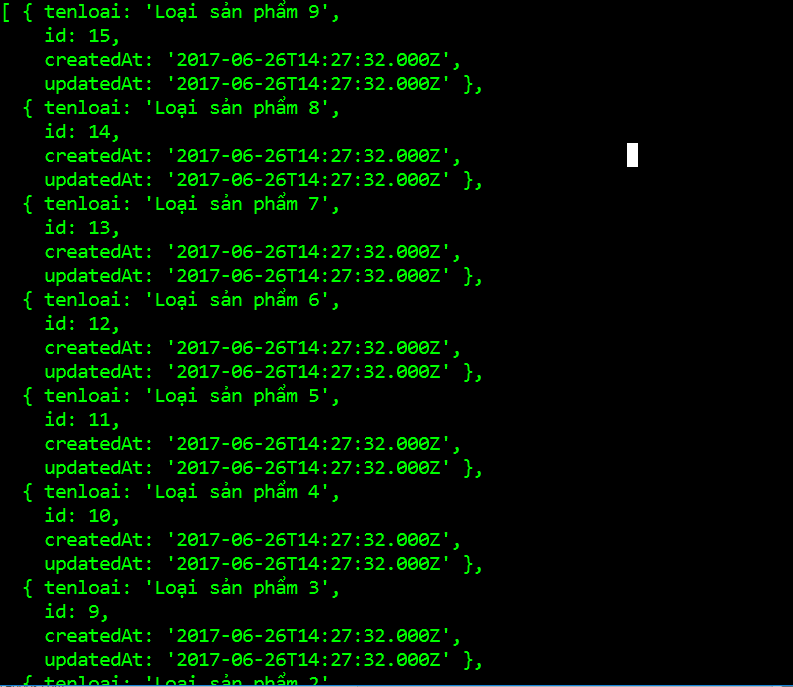
Hình . Bảng dữ liệu

Hàm .**sort()** sắp xếp id giảm dần.



Hình . Sắp xếp giảm dần theo id

Kết quả sau khi sắp xếp.



Hình . Kết quả sau khi sắp xếp

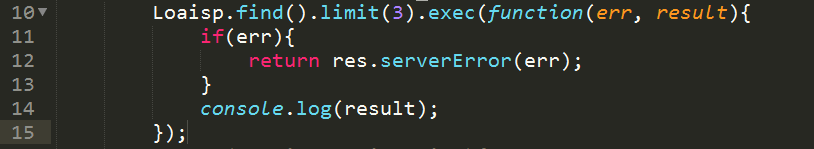
#### Limit

**.limit()** là hàm để giới hạn kết quả cần lấy sau khi truy vấn cơ sở dữ liệu, tham số của **.limit()** là một số nguyên. Ví dụ: hiển thị 3 dòng đầu tiên khi truy vấn bảng dữ liệu loaisp bên dưới.



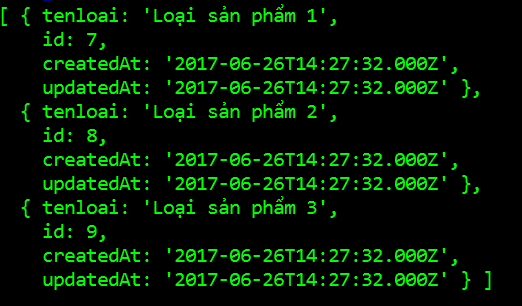
Hình . Bảng dữ liệu

Hàm **.limit()** lấy giới hạn dữ liệu.



Hình . Hàm limit giới hạn dữ liệu hiển thị

Kết quả sau khi thực hiện.



Hình . Kết quả sau khi giới hạn dữ liệu hiển thị

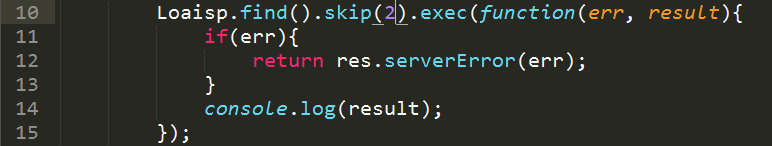
#### Skip

**.skip()** lấy từ dòng thứ bao nhiêu trong cơ sỡ dữ liệu, tham số của **.skip()** là một số nguyên , ví dụ: có bảng cơ sở dữ liệu với 9 dòng dữ liệu hãy lấy dòng dữ liệu thứ 2 trở đi



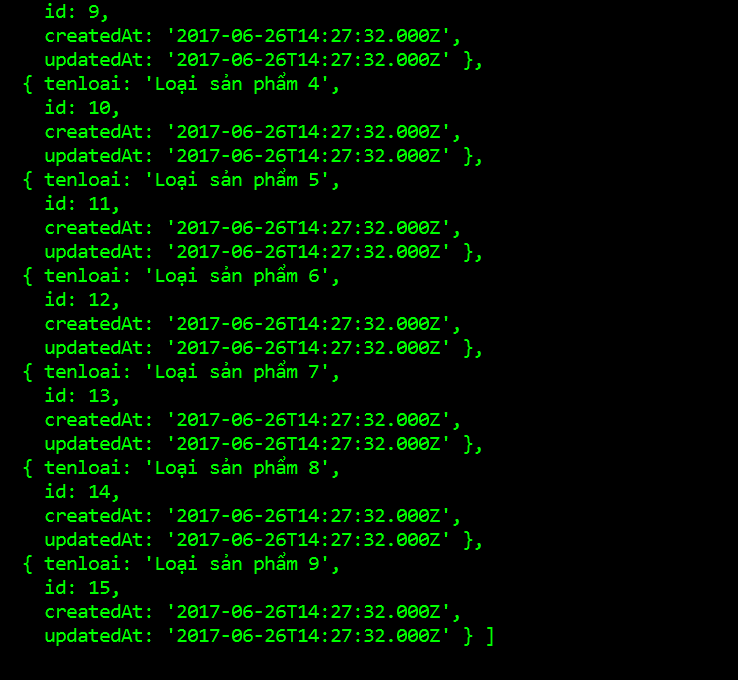
Hình . Bảng dữ liệu.

Hàm **.skip()** để lấy dữ liệu từ dòng thứ 2 trở đi của cơ sở dữ liệu.



Hình . Hàm skip lấy kết quả từ dòng thứ 2 của cơ sở dữ liệu.

Kết quả hiển thị.



Hình . Kết quả hiển thị sau khi dùng hàm skip

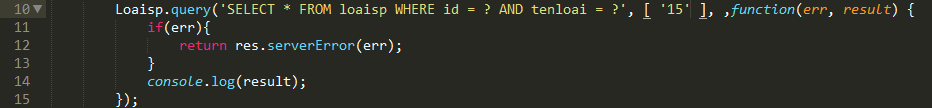
#### Query

Đôi khi có những truy vấn mà các hàm của Warterline không đáp ứng được buộc chúng ta phải ghi truy vấn SQL bằng tay. Ví dụ: xuất kết quả của bảng dữ liệu bên dưới với id = 15.



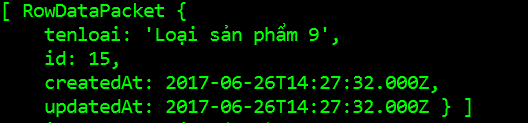
Hình . Bảng dữ liệu

Hàm **.query().**

****

Hình .

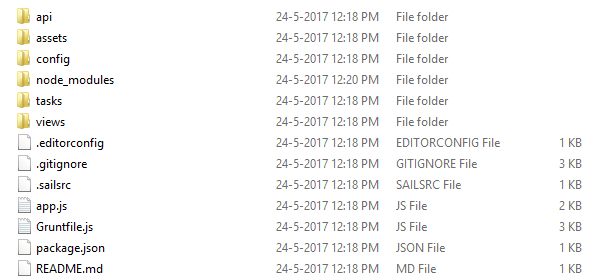
Kết quả



Hình . Kết quả của hàm query

## Tổ chức thư mục của Sails js Framework

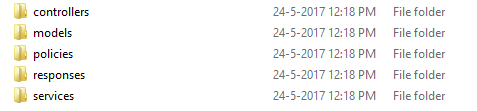
Tổ chức thư mục của sails js



Hình . Tổ chức thư mục của sails js

### Api

Api là thư mục chứa cái xử lý của trang web



Hình . Bên trong thư mục api

#### Controllers

Controllers là thư mục chứa điểu khiển xử lý giữa người dùng với cơ sở dữ liệu, truyền dữ liệu đến giao diện, thay đổi giao diện. Trong sails các tập tin controller là cái tập tin javascript.

#### Models

Models là thư mục chứa các định nghĩa cấu trúc dữ liệu.

#### Policies

Policies chứa định nghĩa các chính sách, ví dụ: chứng thực tài khoản theo session khi đăng nhập để hạn chế truy cập.

#### Services

Chứa các dịch vụ mà người dung tự định nghĩa, các dịch vụ được định nghĩa có thể sử dụng ở bất kỳ nơi đâu.

#### Respones

Thư mục này chứa các định nghĩa để gữi từ máy chủ về cho người dung như: serverError, badRequest,…

### Assets

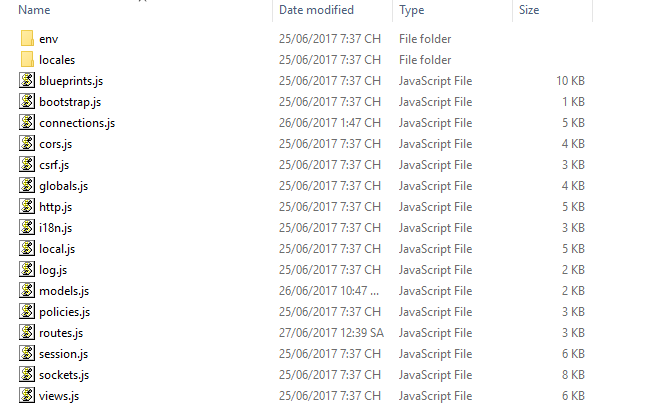
Assets nơi chứa các tập tin css, javascript,... của trang web.



Hình . Bên trong thư mục assets

### Config

Chứa các thư mục cấu hình của framework Sails js, ngoài ra người dung có thể tạo các tập tin định nghĩa khác theo mục đích của mình.

****

Hình . Các tập tin bên trong thư mục config

#### Blueprints.js

Sails nhằm mục đích giảm cả số lượng mã bạn viết và thời gian cần để có được một ứng dụng chức năng lên và chạy. Blueprints là cách Sails tạo ra các tuyến đường và hành động API một cách nhanh chóng dựa trên thiết kế ứng dụng của bạn.

#### Bootstrap.js

Là một chức năng khởi động không đồng bộ có thể tùy chỉnh chạy được trước khi ứng dụng Sails của bạn được nâng lên (tức là khởi động).

#### Connections.js

Là tập tin chứa cấu hình các kết nối đến cở sở dữ liệu.

#### Cors.js

CORS là một cơ chế cho phép tập lệnh trình duyệt trên các trang được phân phối từ các tên miền khác (ví dụ: myothersite.com) để nói chuyện với máy chủ của bạn (ví dụ: api.mysite.com). Giống như JSONP, mục tiêu của CORS là phá vỡ chính sách cùng nguồn gốc; Cho phép máy chủ Sails của bạn đáp ứng thành công các yêu cầu từ mã JavaScript phía khách hàng chạy trên một trang được lưu trữ từ một số miền khác. Nhưng không giống như JSONP, nó hoạt động với nhiều hơn là chỉ yêu cầu GET. Và nó cho phép bạn đưa vào danh sách trắng các nguồn gốc cụ thể (staging.yoursite.com hoặc yourothersite.net) và ngăn chặn các yêu cầu từ những người khác (evil.com).

Sails có thể được cấu hình để cho phép các yêu cầu chéo nguồn gốc từ một danh sách các tên miền mà bạn chỉ định hoặc từ mọi miền. Việc này có thể được thực hiện trên cơ sở mỗi tuyến đường hoặc trên toàn cầu cho mọi tuyến đường trong ứng dụng của bạn.

#### Csrf.js

Thẩm định yêu cầu cross-site (CSRF) là một loại tấn công buộc người dùng cuối thực hiện những hành động không mong muốn trên một ứng dụng web mà hiện tại anh ta đã xác thực. Nói cách khác, không được bảo vệ, cookie được lưu trữ trong trình duyệt như Google Chrome có thể được sử dụng để gửi yêu cầu tới Chase.com từ máy tính của người dùng cho dù người dùng đó hiện đang truy cập Chase.com hay Horrible-Hacker-Site.com.

Để bảo vệ trang web ta đổi thuộc tính mặc định false sang true.

#### Globals.js

Tập tin này dùng để cấu hình cho các biến toàn cục trên sails.

#### Htpp.js

Tập tin này thường dùng để cấu hình cho máy chủ http.

#### I18n.js

Nếu ứng dụng của bạn tiếp cận đến người hoặc hệ thống từ khắp nơi trên thế giới, quốc tế hóa và nội địa hóa (i18n) có thể là một phần quan trọng trong chiến lược quốc tế của bạn. Sails cung cấp sự hỗ trợ tích hợp cho việc phát hiện các tùy chọn ngôn ngữ người dùng và dịch các từ / câu tĩnh nhờ i18n-node. (NPM).

#### Local.js

Tệp config / local.js rất hữu ích cho việc định cấu hình ứng dụng Sails cho môi trường địa phương của bạn (ví dụ máy tính xách tay của bạn). Cài đặt trong tệp này được ưu tiên hơn tất cả các tệp cấu hình khác, ngoại trừ .sailsrc. Vì chúng chỉ được sử dụng cho mục đích sử dụng cục bộ nên chúng không được đặt dưới điều khiển phiên bản (và được bao gồm trong tệp .gitignore mặc định vì lý do đó). Sử dụng local.js để lưu trữ cài đặt cơ sở dữ liệu cục bộ, thay đổi cổng được sử dụng khi nâng ứng dụng trên máy tính của bạn.

Trong khi bạn đang phát triển ứng dụng của mình, tệp cấu hình này phải bao gồm bất kỳ cài đặt nào đặc biệt cho máy tính hoặc máy chủ phát triển của bạn (mật khẩu db, v.v.) Nếu bạn đang sử dụng git, lưu ý rằng config / local.js được bao gồm trong .gitignore in Các ứng dụng mới Sails theo mặc định, và do đó nó sẽ không được kiểm tra vào kho của bạn khi bạn cam kết.

Khi bạn đã sẵn sàng triển khai ứng dụng của mình trong quá trình sản xuất, bạn cũng có thể sử dụng tệp này để có các tùy chọn cấu hình trên máy chủ nơi nó sẽ được triển khai. Tuy nhiên, đối với việc triển khai máy chủ, các biến môi trường thường thích hợp hơn. Bạn cũng có thể sử dụng các đối số dòng lệnh và tệp .sailsrc làm các thay thế cho config / local.js cho cấu hình phát triển địa phương của bạn. Xem tổng quan về thông tin chung về cấu hình Sails.

Lưu ý: tệp này được bao gồm trong tệp .gitignore của bạn, vì vậy nếu bạn đang sử dụng git làm giải pháp kiểm soát phiên bản cho ứng dụng Sails của mình, hãy nhớ rằng tệp này sẽ không được cam kết với kho của bạn! Tin tốt lành là, có nghĩa là bạn có thể chỉ định cấu hình cho máy địa phương của bạn trong tệp tin này mà không thực hiện vô ý thông tin cá nhân (như mật khẩu cơ sở dữ liệu) vào repo. Thêm vào đó, điều này ngăn cản các thành viên khác trong nhóm của bạn thay đổi cấu hình địa phương của bạn trên đầu trang của bạn.

#### Log.js

Sails đi kèm với một logger đơn giản, có sẵn được gọi là captains-log. Cách sử dụng của nó là cố ý rất giống với console.log của Node, nhưng với một số ít các tính năng bổ sung; Cụ thể là hỗ trợ cho các mức đăng nhập nhiều với colorized, prefixed console sản lượng

#### Models.js

Là là thư mục định nghĩa về việc tự động tạo các bảng của cơ sở của bạn, kết nối đến database thông qua các cấu hình ở config/connections.js

#### Policies.js

Tập tin chứa các định nghĩa về các chính sách mà bạn đã định nghĩa ở **api/policies.** Bạn có thể thực hiện một hoặc nhiều chính sách cho ứng dụng của bạn.

#### Routes.js

Tập tin này nơi định nghĩa các route của ứng dụng web của bạn bạn, nó định nghĩa các route rõ ràng để kết nối từ khách hàng đến tài nguyên của mà họ yêu cầu.

#### Session.js

Là tập tin dùng đễ cấu hình session cho ứng dụng của bạn. Tệp này cũng chứa 'Secret Session' của bạn được tạo ra bởi Sails khi bạn tạo ứng dụng của bạn. Đừng thay đổi hoặc loại bỏ điều này trừ khi bạn thực sự biết bạn đang làm gì.

#### Socket.js

Là tập tin chứa cấu hình khi sử dụng gói Socket.io.

#### Views.js

Tập tin này chứa cấu hình template engine mà Sails sử dụng khi render các HTML từ máy chủ trả về cho khách hàng.

### Node\_modules

Là thư mục chứa các module, các gói mở rộng của framework, khi tải một gói mở rộng thì gói đó sẽ ở trong thư mục node\_modules.

### Tasks

Grunt là 1 công cụ dòng lệnh dựa trên các task dùng để hỗ trợ các project javascript. Chúng ta có thể coi Grunt là 1 công cụ để build code cũng như tự động các task lặp đi lặp lại (task runner). Công việc của task runner có thể kể đến như nén, biên dịch các file css, javascript, chạy unit test.

Một Task đại diện cho một số công việc cần phải hoàn thành. Các công việc này có thể chạy trong 1 luồng riêng hoặc có thể thực thi một cách đồng bộ, và kết quả được chờ đợi trả về ở luồng gọi nó.

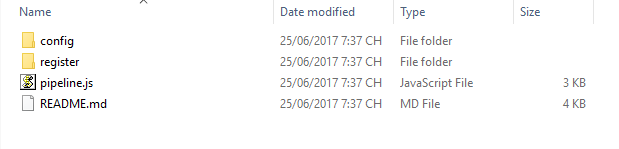
Task là một lớp trừu tượng, bạn có nhiều kiểm soát hơn là việc sử dụng luồng.

Task cho phép bạn tổ chức công việc một cách linh hoạt.

Sails sử dụng Grunt để quản lý tài sản. Tập tin này chứa thông tin cấu hình cho các task GRUNT mà Sails cho mục đích này.

Nếu bạn đã tạo ứng dụng Sails với “sails new foo –linker”, đây cũng là nơi bạn có thể xác định vị trí của các tệp được tự động thu nhỏ và được tiêm theo thứ tự vào các chế độ xem bởi ứng dụng của bạn.

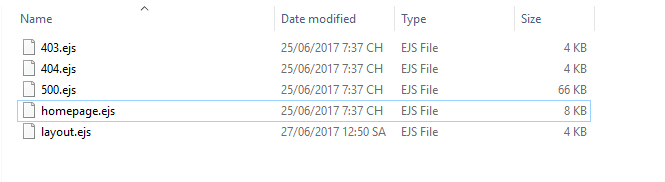
Ở phần đầu của tập tin này, bạn sẽ tìm thấy một vài trong số các tùy chọn được cấu hình thường xuyên nhất, nhưng Sails tích hợp với Grunt cũng hoàn toàn tùy biến được. Nếu bạn muốn làm việc với tài sản của mình một cách khác, bạn có thể thay đổi tệp này để làm bất cứ điều gì bạn thích.



Hình . Bên trong thư mục tasks

### Views

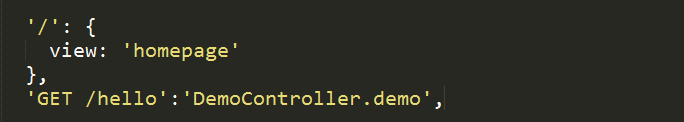
Là nơi chứa các tập tin thư mục hiển thị ra cho người dùng xem.



Hình . Bên trong thư mục views

## Route

Route giữ vai trò chỉ đường cho các request từ phía người dùng lên máy chủ. Định nghĩa một route ta vào **config/routes.js**.

****

Hình . Định nghĩa 1 route

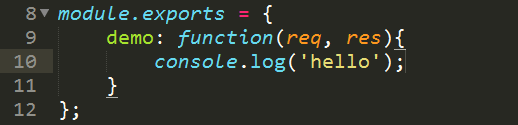
GET: phương thức của route ngoài ra còn có các phương thức khác như POST, PUT, DELETE,…

/hello: Là đường dẫn.

DemoController: là Controller.

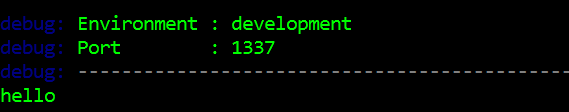
Demo: là hành động trong Controller.

Hành động demo trong DemoController



Hình . Hành động demo trong DemoController.

Để chạy route tao lift Sails và bấm locahost:1337/hello để chạy route.



Hình . Kết quả thực hiện khi chạy route.

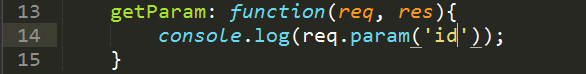
Để truyền tham số trên route ta dùng cú pháp [:tên tham số].

Ví dụ: Tạo 1 route truyền tham số với tên tham số là id.



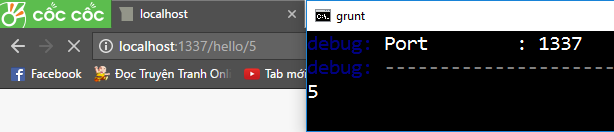
Hình . Truyền tham số trên route

Lấy giá trị của tham số id.



Hình . Lấy giá trị của tham số id

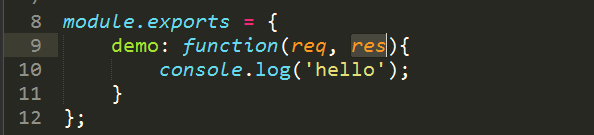
Giá trị của tham số id.



Hình . Giá trị của tham số id

## Controller

Controller dùng để điều hướng dữ liệu.



Hình . Controller.

Demo : hành động của controller.

Req: request là yêu cầu từ gữi yêu cầu từ phía người dùng lên máy chủ.

Res: respone là máy chủ sẽ gữi lại yêu cầu khi nhận được resquet.

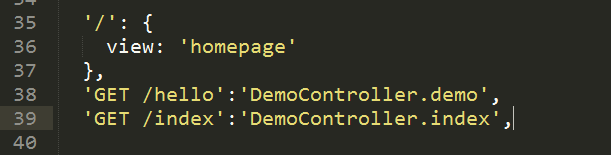
## View

Sails sử mặc định sử dụng template engine EJS.

Để trả về một khung nhìn cho khách hàng khi đăng nhập vào trang web ta dùng **res.view();**

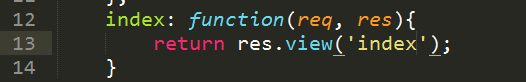
Ví dụ: khi khách hàng truy cập đường dẫn /index sẽ hiển thị trang web với dòng chữ xin chào.

Tạo một route với đường dẫn là /index và phương thức là GET, sử dụng controller demo với hành động là index.



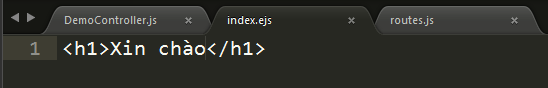
Hình . Tạo route

Tạo hành động index trong DemoController và trả về một trang web với nội dung xin chào.



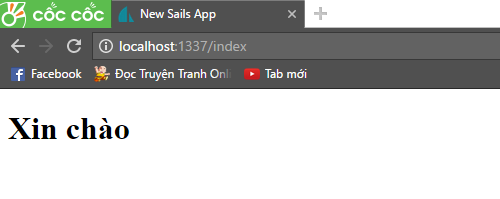
Hình . Hành động index trong DemoController

Tạo tập tin index.ejs trong thư mục view với nội dung là xin chào



Hình . index.ejs

Chạy Sails và truy cập đường dẫn localhost:1337/index



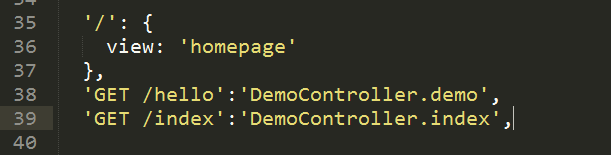
Hình . Kết quả sau khi chạy

## Locals

Locals dùng để truyền dữ liệu từ một hành động của controller sang một view nào đó được chỉ định.

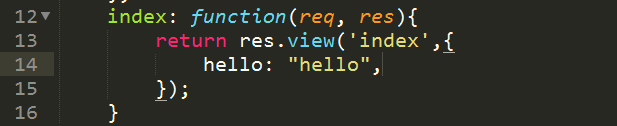
Ví dụ: truyền dòng chữ hello sang view index.

Tạo một route với đường dẫn là /index và phương thức là GET, sử dụng controller demo với hành động là index.



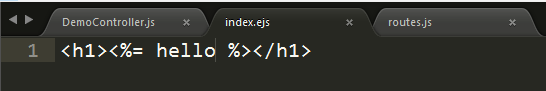
Hình . Tạo route.

Tạo hành động index trong DemoController và trả về một trang web kèm theo một biến chứa dòng chữ hello.



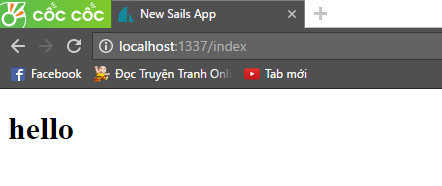
Hình . Hành động index trả về view index với dòng chữ xin chào.

Bên view/index.ejs để nhận được biến hello ta khai báo <%= hello %> (xem thêm về chương 3 template engine).



Hình . Khai báo để nhận biến hello.

Kết quả hiển thị sau khi chạy đường dẫn localhost:1337/index

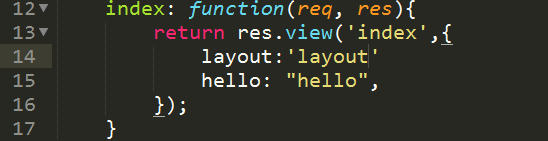


Hình . Hiển thị giá trị của biến hello.

## Layout

Nếu như có những phần không bị thay đổi của trang web khi chạy còn những phần còn lại khác bị thay đổi ta có thể dùng biến layout để cố định những phần không bị thay đổi khi chạy trang web.

Cú pháp:



Hình . Cú pháp layout.

Layout: là đường dẫn tới tập tin cố định nằm trong view.

Bên trang layout muốn hiển thị kết quả bị thay ta thêm **<- body >** vào nơi đó.



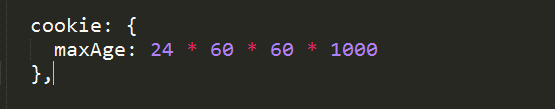
Hình . Phần bị thay đổi.

Nếu khi **return** **res.view()** không có chỉ định layout nào mặc định sẽ sử dụng layout mặc định mà Sails tạo view/layout.ejs.

## Session và cookie

Để sử dụng session trong sails bạn vào config/session.js để cấu hình session cho ứng dụng.

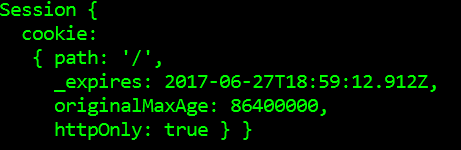
Để cài đặt thời gian sống cho session bạn tắt chú thích cho cookie.



Hình . Thời gian sống của session.

Thời gian sống của session mặc định là 24 giờ, bạn có thể tùy chỉnh lại thời gian sống của session, thời gian được tính bằng milli giây.

Session sẽ có dạng.



Hình . Session và cookie.

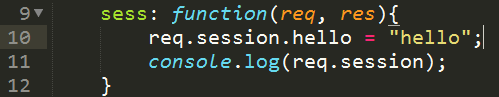
\_expires: thời gian mà session lẫn cookie sẽ hết thời gian sống

originalMaxxAge: thời gian mà session lẫn cookie sống.

path: đường dẫn gốc của tên miền.

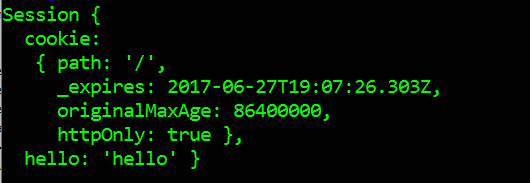
Để sử dụng session bạn dùng lệnh **req.session**, nếu muốn tạo một biến chứa dữ liệu trong session bạn dùng lệnh **req.session.[tên session].**

Ví dụ: tạo một session với tên là hello chứa nội dung là hello



Hình . Session hello với nội dung là hello.

Giá trị của session hello.



Hình . Giá trị của session hello.

## Req (request)

Request là những yêu cầu từ máy của khách hàng gữi lên máy chủ để lấy tài nguyên của ứng dụng.

### Req.param()

Trả về giá trị tham số với tên của tham số đó.

**Req.param()** tìm kiếm trên đường dẫn url, query string. Nếu không tồn tại giá trị bất kỳ nơi đâu của request với tên của nó, nó sẽ trả về undefinded.

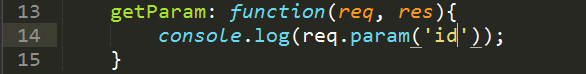
**Req.param()** có thể lấy giá trị của Form với phương thức POST, lấy giá trị của tham số trên đường dẫn route.

Ví dụ: lấy giá trị của tham số id trên route.



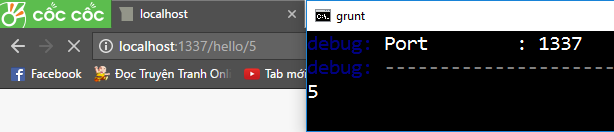
Hình . Truyền tham số trên route.

Lấy giá trị của tham số id.



Hình . Lấy giá trị của tham số id.

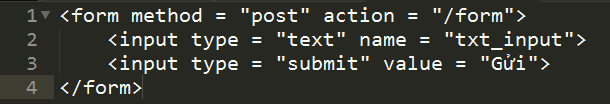
Giá trị của tham số id.



Hình . Giá trị của tham số id.

Ví dụ: lấy giá trị của ô nhập liệu trên form với phương thức POST.

Tạo một form với ô nhập nhiệu tên “txt\_input”.



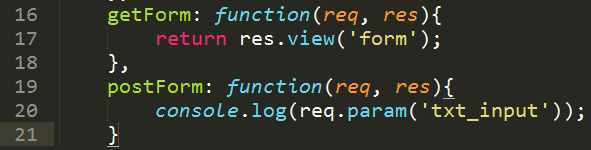
Hình . form nhập liệu.

Tạo 2 route, 1 route để hiển thị form nhập liệu, 1 route để lấy giá trị của ô nhập liệu.



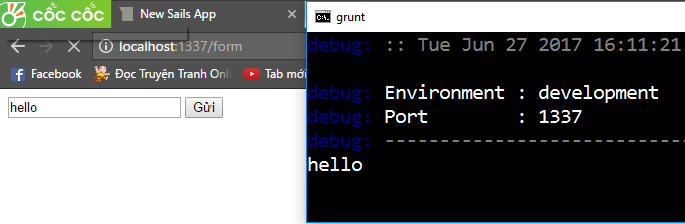
Hình . Route để lấy giá trị của form.

Tạo 2 hành động của DemoController với hành động getForm và postForm. getForm là hành động để hiển thị form nhập liệu, hành động postForm để lấy giá trị của ô nhập liệu trong form.



Hình . Hành động getForm và postForm.

Hiển thị giá trị của ô nhập liệu trong form

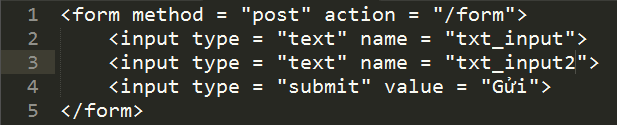


Hình . Giá trị của ô nhập liệu.

### Req.allParams()

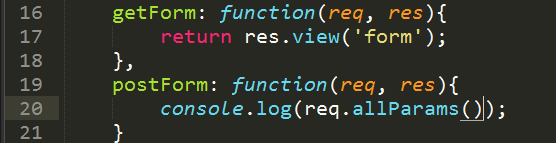
**Req.allParams()** dùng để lấy tất cả các giá trị tham số của form, của đường dẫn url.

Lấy lại ví dụ ở **req.param()** nhưng form bây giờ có 2 ô nhập liệu là “txt\_input” và “txt\_input2”.



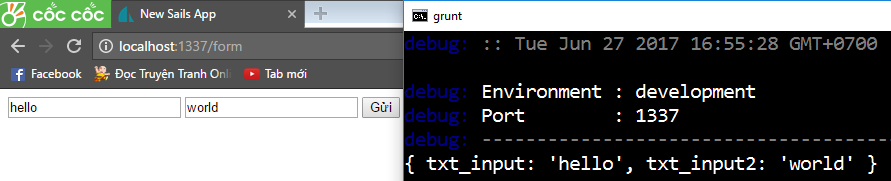
Hình . form với 2 ô nhập liệu.

Hành động postForm trong DemoController.



Hình . Hành động postForm trong DemoContorller.

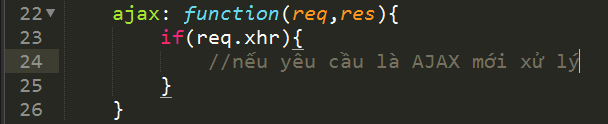
Kết quả hiển thị.



Hình . Kết quả hiển thị của req.allParams().

### Req.xhr

Cho biết yêu cầu hiện tại có phải là AJAX hay không (XMLHttpRequest).



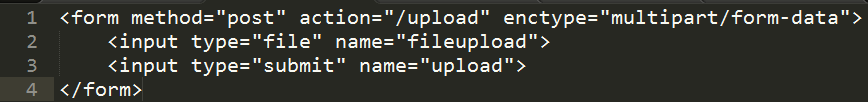
Hình . Minh họa req.xhr.

### Req.file()

**Req.file()** dùng để tải tập tin lên máy chủ sử dụng gói mở rộng Skipper mặc định trong Sails.

Ví dụ: tải 1 tập tin hình ảnh lên máy chủ và nằm trong thư mục assets/images.

Tạo một form dùng để tải tập tin lên máy chủ.



Hình . Form tải tập tin lên máy chủ.

Tạo 2 route, 1 route dùng để hiển thị khung form tải tập tin. 1 dùng để tải tập tin lên.



Hình . Route tải tập tin.

Tạo 2 hành động trong DemoController với tên các hành động là getUpload và postUpload, getUpload dùng để hiển thị form tải tập tin, postUpload dùng để tải tập tin.



Hình . Hành đọng getUpload và postUpload.

Req.file(‘fileupload’) là tên của thẻ input với type là file trên form.

Các tham số của **.upload()**:

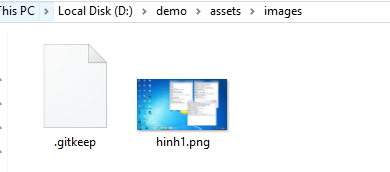
maxBytes: Dung lượng tối đa có thể tải lên tính bằng byte.

Dirname: Nơi lưu tập tin tải lên.

saveAs: tên mới khi tải tập tin lên, nếu không có saveAs thì tập tin vẫn giữ tên củ.

Nếu tải lên nhiều tập tin thì không cần saveAs, tên các tập tin sẽ là các chuỗi ngẫu nhiên.

Kết quả sau khi tải tập tin vào thư mục assets/images với tên là hinh1.png.



Hình . Kết quả sau khi tải tập tin lên thành công.

### Req.url

**Req.url** cho biết đường tên dẫn hiện tại, nó khá giống với req.path nhưng req.url bao gồm các query string .

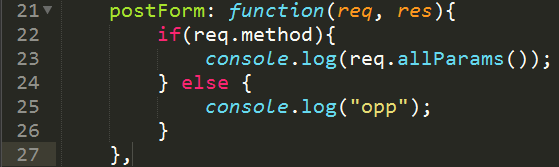
Ví dụ: Đường dẫn localhost:1337/form/2?name=hello

Kết quả của **req.url** sẽ là: /form/2?name=hello

### Req.method

Cho biết có phương thức sẽ gữi lên máy chủ xử lý.

Ví dụ: Khách hàng gữi một phương thức POST với yêu cầu tới /product.



Hình . Kiểm tra có phương thức gữi lên máy chủ xử lý.

### Req.query

Req.query để lấy query string ở trên đường dẫn.

Ví dụ: localhost:1337/form?q=search.

Sử dụng req.query kết quả hiển thị sẽ là { q: ‘search’ }.

Nếu muốn lấy kết quả của q thì sử dụng req.query.q, kết quả hiển thị ‘search’.

### Req.path

Req.path cho biết tên đường dẫn hiện tại bao gồm cả dấu /

Ví dụ: Đường dẫn localhost:1337/form/2?name=hello

Kết quả của **req.path** sẽ là: form/2

## Req.flash

Req.flash sẽ trả về cho chúng ta một thông báo theo mong muốn của chúng ta. **Req.flash()** được dưới dạng session, khi tải lại trạng nó sẽ mất.

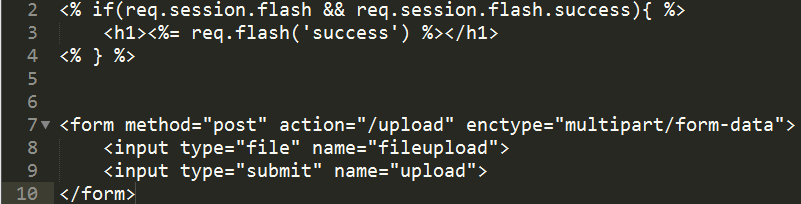
Cú pháp: req.flash(“tên của flash”, “nội dung”);

Ví dụ: trả về thông báo thành công khi tải tập tin lên thành công.



Hình . Dùng req.flash xuất thông báo mong muốn.

Hiển thị flash.



Hình . Hiển thị flash

Kết quả sau khi tải tập tin thành công.



Hình . Thông báo hiển thị khi tải tập tin thành công.

## Res (respone)

Respone là những tài nguyên mà máy chủ trả về cho máy khách khi nhận được một yêu cầu request.

### Res.view()

**Res.view()** trả về khung nhìn cho khách hàng với cú pháp:

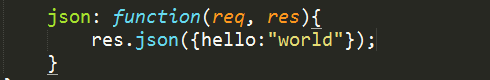
res.view(“đường dẫn của khung nhìn”, locals);

### Res.json()

**Res.json()** trả về cho người dùng một JSON kết quả mà họ mong muốn.

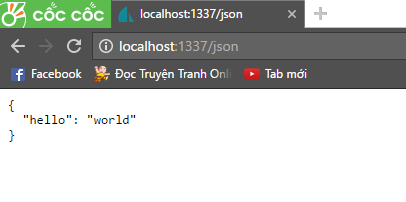
Ví dụ:

Trả về một JSON với keys là hello và nội dung là world.



Hình . Hành động trả về 1 JSON.

Kết quả hiển thị.



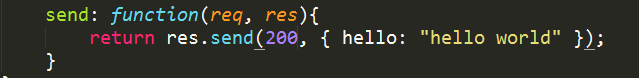
Hình . Kết quả của res.json.

### Res.send()

Gửi giá trị, với trạng thái mặc định 200 OK.

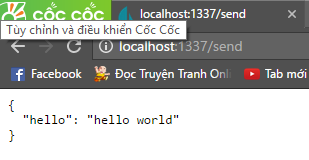
Cú pháp : return res.send(trạng thái , nội dung);

Ví dụ:



Hình . Gửi JSON với trạng thái 200.

Kết quả của **res.send()**



Hình . Kết quả của res.send().

### Res.serverError()

**Res.serverError()** sẽ thông báo cho chúng ta lỗi khi trong quá trình chạy bị lỗi.

Cú pháp : return res.serverError(“lỗi”);

### Res.redirect()

**Res.redirect()** sẽ chuyển chúng đường dẫn khác.

Cú pháp: return res.redirect(“đường dẫn”);

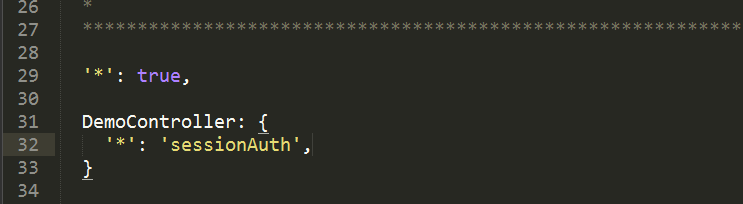
## Policies

Policies là những chính sách mà chúng ta muốn áp dụng vào ứng dụng, điển hình như đăng nhập để hạn chế người truy cập vào ứng dụng web của chúng ta.

Các tập tin để định nghĩa các chính sách của chúng ta nằm ở **api/policies**.

Để áp dụng các chính sách mà ta áp dụng ta vào **config/policies.js**.

Ví dụ: áp dụng chính sách sessionAuth cho tất cả các hành động của DemoController.



Hình . Áp dụng chính sách sessionAuth cho DemoController.

Với ‘\*’: true cho phép ta công khai dữ liệu của mình.

DemoController: là tên Controller áp dụng chính sách.

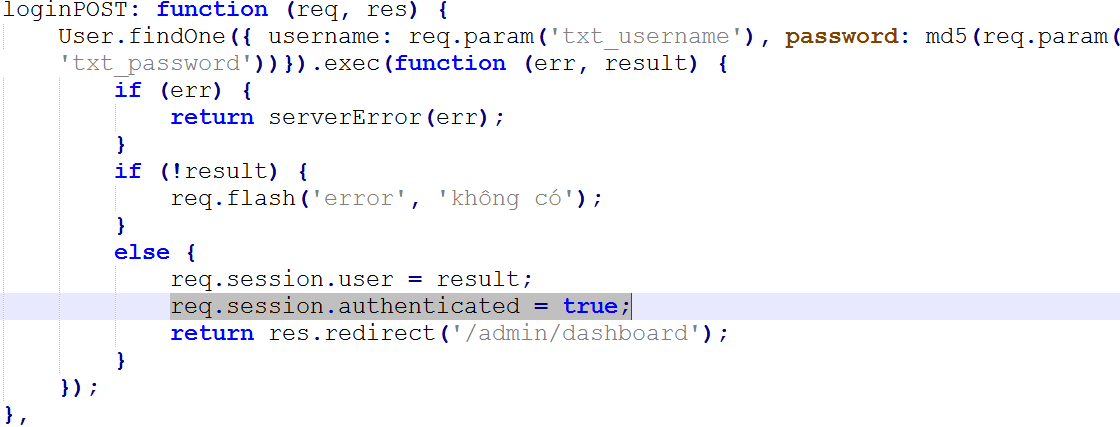
‘\*’: Tất cả các hành động của DemoController để áp dụng chính sách sessionAuth.

sessionAuth: là chính sách ta định nghĩa ở tập tin **api/policies/sessionAuth.js.**

Chứng thực người dùng bằng **req.session.authenticated.**

Ví dụ: tất cả các Controller ở trang quản trị admin đều phải đăng nhập mới truy cập được.

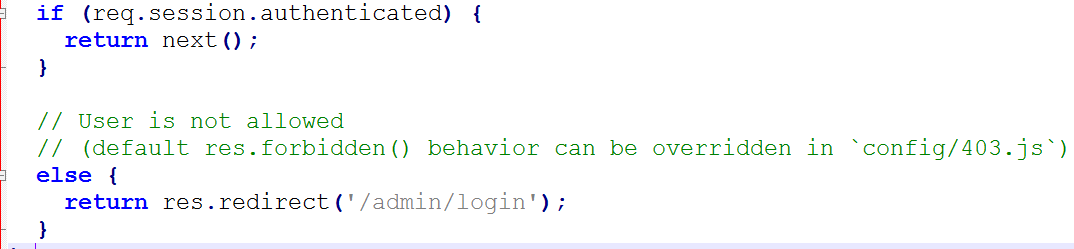
Ta tạo một Controller với action là login để chứng thực nếu tồn tại tài khoản và mật khẩu.



Hình . Chứng thực tài khoản.

Nếu tồn tại tài khoản thì ta bật **req.session.authenticated = true** để lưu lại là người dùng này đã được chứng thực.

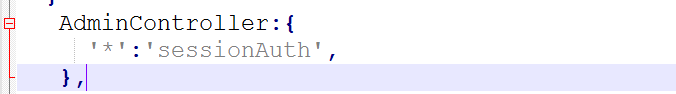
Tập tin chứa chính sách chứng thực tài khoản **api/policies/sessionAuth.js**



Hình . Chứng thực tài khoản ở sessionAuth.js.

Nếu đã tồn tài req.session.authenticated thì chuyển sang bước tiếp theo, nếu không sẽ trả ngươi dùng về lại với trang đăng nhập.

Để áp dụng chính sách sessionAuth cho các Controller mà ta muốn bảo vệ, ta cấu hình trong config/policies.js.



Hình . Áp dụng chính sách sessionAuth cho AdminController.

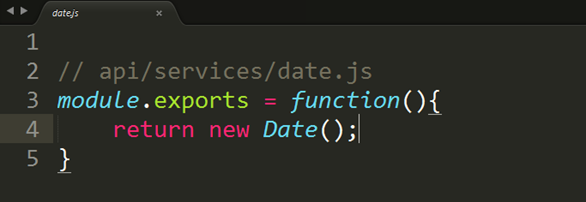
Tất cả hành động của AdminController khi muốn truy cập đều phải chứng thực thành công mới được thao tác.

## Services

Chứa các dịch vụ mà người dung tự định nghĩa, các dịch vụ được định nghĩa có thể sử dụng ở bất kỳ nơi đâu.

ví dụ: Tạo 1 Services trả về ngày tháng hiện tại và sử dụng nó.

Tạo một services date



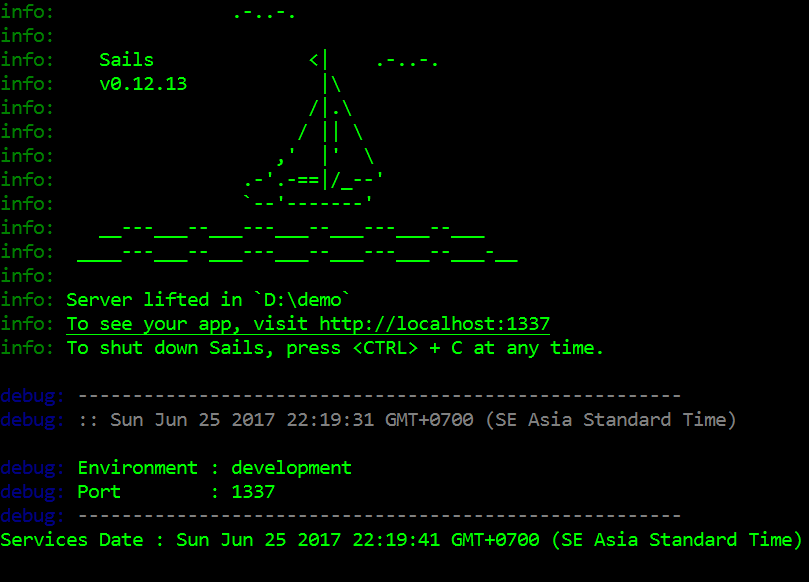
Hình . Tạo một services date.

Sử dụng services date ở controller Demo.



Hình . Sử dụng services date ở DemoController.

Kết quả hiển thị khi sử dụng services date.



Hình . Kết quả sau khi thực thi services date thành công.

# TEMPALTE ENGINE

## Giới thiệu

Tempalte engine là một ngôn ngữ trình bày dữ liệu, giúp phân tách code dữ liệu ra khỏi code html. Có nhiều template engine hổ trợ Node js như: ejs, pug, dust, ect,…

## EJS

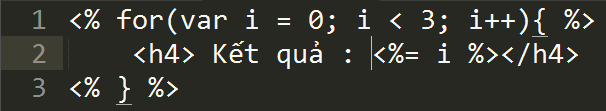
Template engine ejs cũng là một ngôn ngữ trình bày dữ liệu, giúp phân tách code dữ liệu ra khỏi code html.

## Tags ejs

Tags của ejs bắt đầu bằng **<%** , và kết thúc bằng **%>**.

### Tag <%= %>

Để in ra kết quả thì ta dùng tag **<%=** **%>**, ví dụ: in ra kết quả của dòng lập for.



Hình . Cú pháp in ra kết quả của dòng lập for.

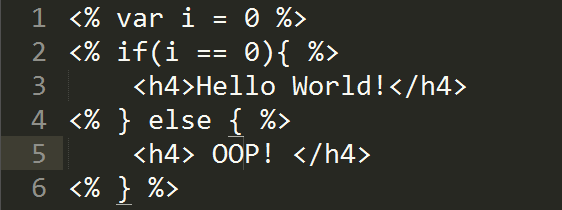
Kết quả hiển thị sau khi chạy dòng lập for.



Hình . Kết quả hiển thị của dòng lập for

### Tag <% %>

Tag **<% %>** dùng để biểu diễn các thuật toán logic như **if else**, các dòng lập **for, while**, khai báo biến theo kiểu javascript,… Ví dụ: nếu i = 0 thì in ra dòng chữ “Hello World!”, nếu không phải thì in ra dòng chữ “OOP!”



Hình . Cú pháp của điều kiện if.

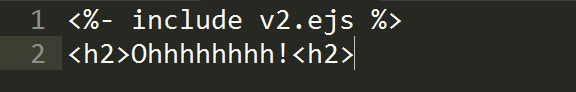
Kết quả hiển thị



Hình . Kết quả hiển thị của điều kiện if.

### Tag <%- %>

Ví dụ: include nội dung v2.ejs vào demo.ejs



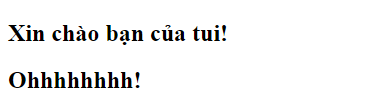
Hình . Cú pháp include v2.ejs vào demo.ejs

Nội dung của v2.ejs.



Hình . Nội dung của v2.ejs.

Kết quả.

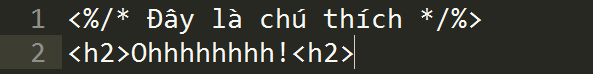


Hình . Kết quả sau khi include v2.ejs vào demo.ejs.

### Tag <%/\* \*/%>

Tag **<%/\* \*/%>** dùng để để chú thích, khi chạy sẽ không hiển thị ra bên ngoài.

Ví dụ: Tạo 1 dòng chú thích trong ejs.



Hình . Chú thích trong ejs.

Kết quả hiển thị.



Hình . Kết quả hiển thị của chú thích.

# GÓI MỞ RỘNG

## Giới thiệu npm

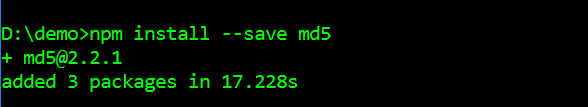
Sử dụng NPM để cài đặt, chia sẻ và phân phối mã nguồn; quản lý vào các dự án của bạn; và chia sẻ và nhận thông tin phản hồi với những nhà phát triển khác.

Nếu bạn đã làm việc với Javascript cho một thời gian, bạn có thể đã nghe nói về NPM: NPM làm cho nó dễ dàng cho các nhà phát triển Javascript để chia sẻ mã mà họ đã tạo ra để giải quyết vấn đề cụ thể, và cho các nhà phát triển khác để sử dụng lại mã đó trong họ các ứng dụng riêng.

Một khi bạn đang tùy thuộc vào mã này từ các nhà phát triển khác, NPM làm cho nó rất dễ dàng để kiểm tra xem họ đã thực hiện bất kỳ bản cập nhật cho nó, và để tải về những bản cập nhật khi họ đang thực hiện.

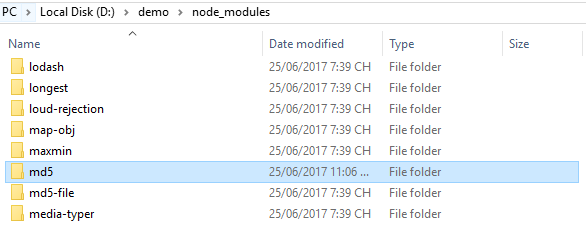
## Tải gói mở rộng

Để cài đặt các gói mở rộng trong quá trình sử dụng Sails js ta cũng mở cửa sổ cmd chứa framework Sails js, và bấm lệnh **npm install –save [ tên gói ].** Ví dụ: cài đặt gói MD5 vào framework Sails js.



Hình . Cài đặt gói MD5 thành công

Sau khi cài đặt thành công gói MD5 sẽ nằm trong thư mục node\_modules.



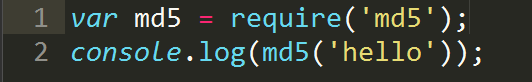
Hình . Gói md5 trong thưc mục node\_modules.

## Các gói đã sử dụng

Để sử dụng gói mở rộng trong tập tin \*.js

Var x = require(‘tên gói’);.

Ví dụ: sử dụng gói mở rộng MD5 để mã hóa chuỗi ký tự hello.



Hình . Sử dụng gói mở rộng md5

Kết quả sau khi mã hóa md5 chuỗi ký tự hello.



Hình . Chuỗi ký tự đã được mã hóa

### Momentjs

Momentjs là gói mở rộng hổ trợ định dạng thời gian theo các format mà gói cung cấp sẵn, ngoài ra còn có thể chọn ngôn ngữ để hiển thị theo từng quốc gia. Momentjs có thể tính thời gian từ một thời gian của quá khứ tới hiện tại. Để cài đặt ngôn ngữ tiếng Việt cho moment: **moment().locale(‘vi’);**

Bảng . Bảng các loại format của Momentjs.

|  |  |
| --- | --- |
| **Kiểu format** | **Kết quả hiển thị** |
| moment().format(); | 2017-06-26T01 :29 :30+7 :00 |
| moment().format('LT'); | 01 :29 |
| moment().format('LTS'); | 01 :29 :00 |
| moment().format('L'); | 26/06/2017 |
| moment().format('l'); | 26/6/2017 |
| moment().format('LL'); | 26 tháng 6 năm 2017 |
| moment().format('ll'); | 26 Th06 2017 |
| moment().format('LLL'); | 26 tháng 6 năm 2017 01:31 |
| moment().format('lll'); | 26 Th06 2017 01:31 |
| moment().format('LLLL'); | thứ hai, 26 tháng 6 năm 2017 01:32 |
| moment().format('llll'); | T2, 26 Th06 2017 01:32 |
| moment().format('MMMM Do YYYY, h:mm:ss a'); | tháng 6 26 2017, 1:33:01 sa |
| moment().format('dddd'); | thứ hai |
| moment().format("MMM Do YY"); | Th06 26 17 |
| moment().format('YYYY [-] YYYY'); | 2017 - 2017 |

Bảng . Quan hệ thời gian của Momentjs

|  |  |
| --- | --- |
| **Phép tính** | **Kết quả hiển thị** |
| moment().calendar(); | Hôm nay lúc 01:42 |
| moment("20111031", "YYYYMMDD").fromNow(); | 6 năm trước |
| moment("20120620", "YYYYMMDD").fromNow(); | 5 năm trước |
| moment().startOf('day').fromNow(); | 2 giờ trước |
| moment().endOf('day').fromNow(); | một ngày tới |
| moment().startOf('hour').fromNow(); | 43 phút trước |
| moment().subtract(10, 'days').calendar(); | 16/06/2017 |
| moment().subtract(6, 'days').calendar(); | thứ ba tuần rồi lúc 01:43 |
| moment().subtract(3, 'days').calendar(); | thứ sáu tuần rồi lúc 01:43 |
| moment().subtract(1, 'days').calendar(); | Hôm qua lúc 01:43 |
| moment().add(1, 'days').calendar(); | Ngày mai lúc 01:43 |
| moment().add(3, 'days').calendar(); | thứ năm tuần tới lúc 01:44 |
| moment().add(10, 'days').calendar(); | 06/07/2017 |

### Accountingjs

Accountingjs dùng để format lại giá lại số theo các định dạng mình tự định nghĩa hoặc dùng sẵn của gói, accountingjs thường dùng để định nghĩa lại giá trị của tiền tệ.

Bảng . Bảng format tiền tệ của accountingjs.

|  |  |
| --- | --- |
| **Kiểu format** | **Kết quả hiển thị** |
| accounting.formatMoney(12345678); | $12,345,678.00 |
| accounting.formatMoney(4999.99, "€", 2, ".", ","); | €4.999,99 |
| accounting.formatMoney(5318008, { symbol: "GBP", format: "%v %s" }); | 5,318,008.00 GBP |

%v: Giá trị.

%s: Ký hiệu tiền tệ.

Bảng . Bảng format số của accountingjs.

|  |  |
| --- | --- |
| **Kiểu format** | **Kết quả hiển thị** |
| (0.615).toFixed(2); | 0.61 |
| accounting.toFixed(0.615, 2); | 0.62 |

### url-slug

Url-slug là gói mở rộng hổ trợ slug lại tên tiêu đề bài viết để tạo đường dẫn thân thiện cho SEO.

# GIỚI THIỆU ĐỀ TÀI

## Chức năng

### Quản lý sản phẩm

Cho phép quản lý tất cả các sản phẩm mà cửa hàng hiện đang cung cấp, gồm có các chức năng chính: thêm sản phẩm mới, sửa sản phẩm, xóa sản phẩm, xem danh sách sản phẩm còn tồn tại trong cửa hàng.

#### Thêm sản phẩm

Thêm một sản phẩm mới nhập về vào trong cơ sở dữ liệu trong cửa hàng, thông tin của sản phẩm bao gồm: Tên sản phẩm, thiết bị, nhà sản xuất, số lượng sản phẩm, giá, giá khuyến mãi, mô tả chi tiết, thông tin cấu hình của sản phẩm, hình ảnh đại diện của sản phẩm, hình ảnh liên qua của sản phẩm. Sản phẩm khi vừa mới thêm vào sẽ có trạng thái là 1 để cho biết sản phẩm đó vẫn còn kinh doanh trong cửa hàng, khi lưu lại sản phẩm vừa mới thêm vào sẽ lưu lại ngày thêm sản phẩm.

#### Sửa thông tin sản phẩm

Khi 1 sản phẩm có thông tin bị sai hoặc thay đổi về giá có thể sửa lại thông tin của sản phẩm như ở phần thêm sản phẩm, thông tin sau khi lưu sẽ cập nhật lại vào cơ sở dữ liệu và lưu ngày đã sửa.

#### Xóa thông tin sản phẩm

Sản phẩm mà không còn kinh doanh trong cửa hàng nữa, người dùng có thể xóa thông tin của sản phẩm đó, khi xóa sẽ cập nhật trạng thái của sản phẩm về 0 để cho biết sản phẩm đó đã không còn kinh doanh trong cửa hàng nữa.

#### Xem danh sản phẩm

Xem danh sách sản phẩm hiện đang có trong cửa hàng bao gồm sản phẩm còn kinh doanh, sản phẩm không còn kinh doanh, thông tin hiển thị sản phẩm gồm tên sản phẩm, giá sản phẩm, số lượng, trạng thái của sản phẩm (còn kinh doanh, không còn kinh doanh). Người dùng có thể xem thông tin chi tiết của sản phẩm khi chọn vào tên của sản phẩm, có thể sủa thông tin của sản phẩm khi xem, có thể xóa thông tin sản phẩm khi xem thông tin trên danh sách, tìm kiếm bất kỳ thông tin nào của sản phẩm hiện đang hiển thị trên danh sách, chọn số sản phẩm được hiển thị trên 1 trang.

### Quản lý danh mục thiết bị

Cho phép người dùng quản lý danh mục thiết bị hiện đang có trong cửa hàng, cửa hàng cung cấp các loại thiết bị như: điện thoại, laptop, máy tính bảng, vv, các quản lý bao gồm xem danh sách danh mục thiết bị, sửa thông tin danh mục thiết bị, thêm thiết bị mới, xóa thông tin thiết bị đó.

#### Thêm thông tin thiết bị mới

Người dùng thêm thông tin thiêt bị mới gồm những thông tin của thiết bị đó: tên thiết bị, khi thêm thiêt bị trạng thái của thiết bị là 1 cho biết cửa hàng còn kinh doanh thiết bị đó. Khi thêm thông tin thiết bị mới sẽ lưu lại ngày đã thêm thông tin của thiết bị.

#### Sửa thông tin thiết bị

Thông tin thiết bị sai có thể sửa thông tin của thiết bị đó: tên của thiết bị. Sau khi cập nhật thông tin thiết bị vừa sửa sẽ cập nhật ngày đã sửa thông tin thiết bị đó.

#### Xóa thông tin thiết bị

Khi một thiết bị không còn kinh doanh trong cửa hàng, người dùng có thể xóa thông tin của thiết bị đó, khi xóa sẽ cập nhật lại trạng thái của thiết bị đó là 0 để cho biết không còn kinh doanh thiết bị đó nữa.

#### Xem danh sách thiết bị

Xem được danh sách thiết bị mà cửa hàng hiện đang kinh doanh, có thể xóa thông tin của thiết bị đó tên danh sách được.

### Quản lý nhà sản xuất

Mỗi một thiết bị có thể do nhiều nhà sản xuất làm ra, cửa hàng kinh doanh nhiều sản phẩm của nhiều nhà sản xuất làm ra thiết bị, quản lý nhà sản xuất gồm: thêm nhà sản xuất mới, xóa nhà sản xuất, sửa thông tin nhà sản xuất, xem danh sách nhà sản xuất.

#### Thêm nhà sản xuất mới

Khi cửa hàng kinh doanh một thiết bị của một nhà sản xuất, người dùng phải thêm thông tin nhà sản xuất đó, thông tin bao gồm: tên nhà sản xuất, tên thiết bị mà nhà sản xuất đó làm ra. Khi thêm mới thông tin của nhà sản xuất trạng thái của nhà sản xuất đó bằng 1

#### Sửa thông tin nhà sản xuất

Khi thông tin của nhà sản xuất bị sai có thể sửa thông tin của nhà sản xuất, thông tin sửa bao gồm: tên nhà sản xuất, có thể thay đổi thiết bị mà nhà sản xuất đó làm ra.

#### Xóa thông tin nhà sản xuất

Khi cửa hàng không kinh doanh sản phẩm của nhà sản xuất nữa thì có thể xóa thông tin của nhà sản xuất đó đi, khi xóa trạng thái của nhà sản xuất bằng 0.

#### Xem danh sách nhà sản xuất

Xem được danh sách nhà sản xuất, thông tin được hiển thị gồm tên nhà sản xuất, tên thiết bị, có thể xóa thông tin của nhà sản xuất trực tiếp trên danh sách, tìm kiếm thông tin của nhà sản xuất trên danh sách.

### Quản lý thông tin tài khoản

Muốn truy cập vào trang quản trị cần phải có một tài khoản do admin cấp mới có thể truy cập vào trang quản trị trang để thực hiện các chức năng quản trị. Quản lý thông tin tài khoản bao gồm: quản lý việc đăng ký tài khoản thành viên, cấp lại mật khẩu cho thành viên, xóa thông tin thành viên, xem danh sách thông tin thành viên, sửa thông tin thành viên.

#### Thêm thông tin thành viên mới

Khi một người muốn làm thành viên để quản trị trang thì phải đăng nhập thành công vào trang quản trị, muốn làm điều đó cần phải có tài khoản do admin đăng ký, thông tin đăng ký tài khoản bao gồm: Họ tên, tài khoản, mật khẩu, email. Mật khẩu sẽ được mã hóa. Khi đăng ký thành công chủ tài khoản đó có thể đăng nhập vào trang quản trị.

#### Cấp lại mật khẩu

Khi một tài khoản quên mật khẩu, admin có thể cấp lại tài khoản mới cho người đó

#### Xóa thông tin thành viên

Khi một thành viên không còn là thành viên của quản trị nữa thì có thể xóa hẵn thông tin của tài khoản thành viên đó đi, tài khoản đó sẽ không còn có thể truy cập vào trang quản trị nữa

#### Sửa thông tin tài khoản

Khi thông tin tài khoản bị sai có thể sửa lại và cập nhật thông tin tài khoản lại, thông tin sửa có thể là: họ tên, email, mật khẩu.

#### Xem danh sách tài khoản

Xem được danh sách tài khoản thành viên hiện đang có trong của hàng, thông tin hiển thị gồm: Họ tên, tên tài khoản, có thể xóa một tài khoản trực tiếp trên danh sách, tìm kiếm thông tin của tài khoản.

### Quản lý đơn hàng

Mỗi khi khách hàng đặt hàng và mua hàng thành công sẽ lưu lại một đơn hàng cho biết khách hàng nào đã mua sản phẩm nào, thông tin của khách hàng, quản lý đơn hàng gồm: xem danh sách đơn hàng, xem thông tin đơn hàng, duyệt đơn hàng sẽ giao.

#### Xem danh sách đơn hàng

Xem được danh sách đơn hàng mà khách hàng đã mua của của hàng, thông tin hiển thị gồm mã đơn hàng, trạng thái đơn hàng: đã giao hoặc chưa giao, hoặc đã hũy. Khi khách hàng đặt hàng thành công trạng thái của đơn hàng là chưa giao. Khi chọn vào 1 đơn hàng có thể xem chi tiết của đơn hàng đó.

#### Xem chi tiết đơn hàng

Khi xem chi tiết đơn hàng thông tin hiển thị gồm: danh sách sản phẩm của đơn hàng, tổng giá trị đơn hàng, thông tin khách hàng đặt đơn hàng, thông tin khách hàng gồm: tên khách hàng, địa chỉ, số điện thoại, email (nếu có).

### Thống kê doanh thu

Cho phép xem doanh thu của cửa hàng hàng tháng, người dùng có thể chọn thời gian để xem doanh thu, vẽ biểu đồ thống kê doanh thu của cửa hàng theo thời gian hiện tại về 1 tháng.

# KẾT LUẬN

**TÀI LIỆU THAM KHẢO**