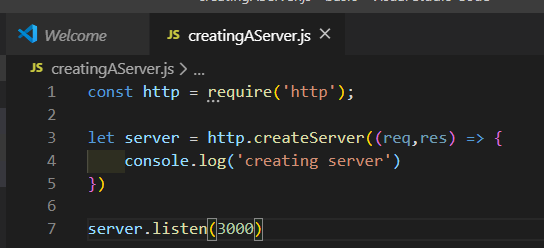
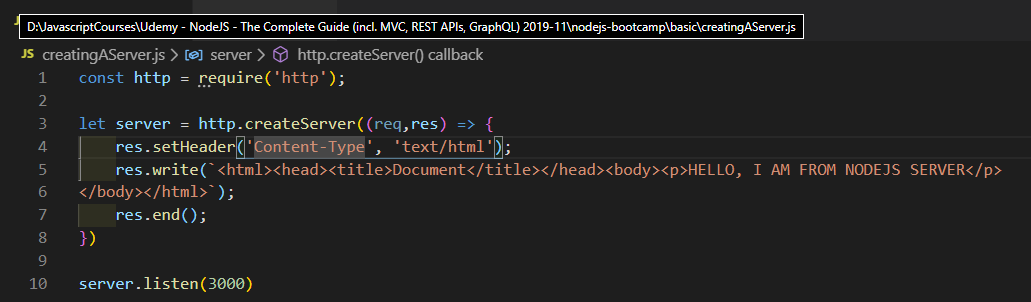
- Creating a Node Server:

1. Dùng ‘require’ để import các core modules của nodeJs hoặc các file nodejs của chính mình:
   1. Import các module, các file JS do mình tự viết thì dùng ‘./’: 
   2. Import các core module của nodejs như http, https,fs thì không dùng ‘./’:
2. Tạo và chạy 1 server với port 3000:
   1. 

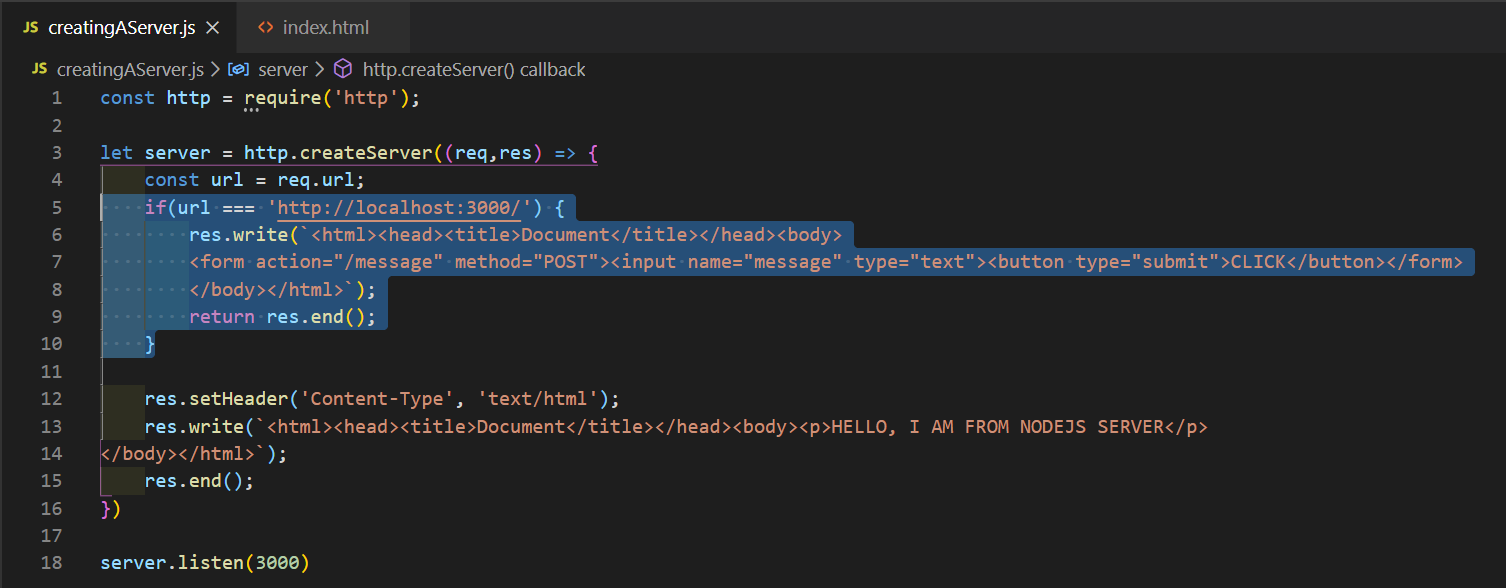
- The Node Lifecycle & Event Loop:

1. Cách shut down 1 chương trình: 

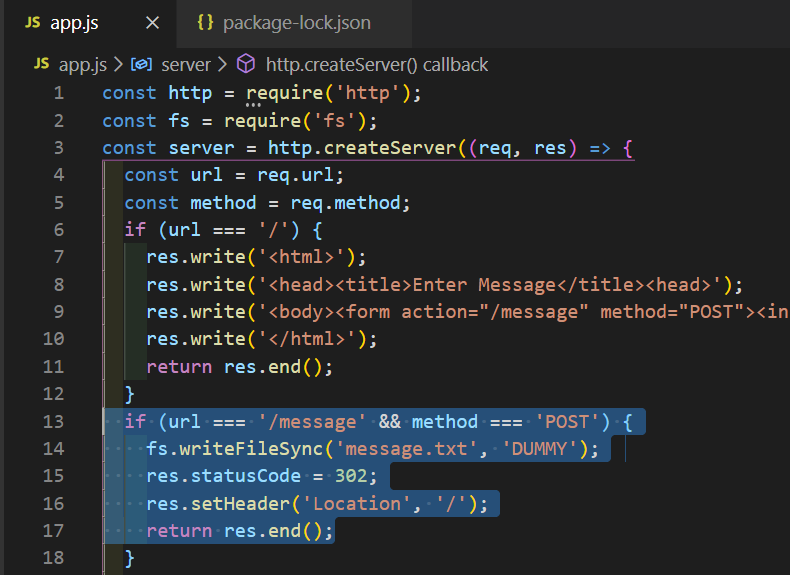
- Sending Responses: Set các header, gửi về file html …

1. Gửi về file HTML: trên localhost:3000 sẽ hiện ra file HTML.

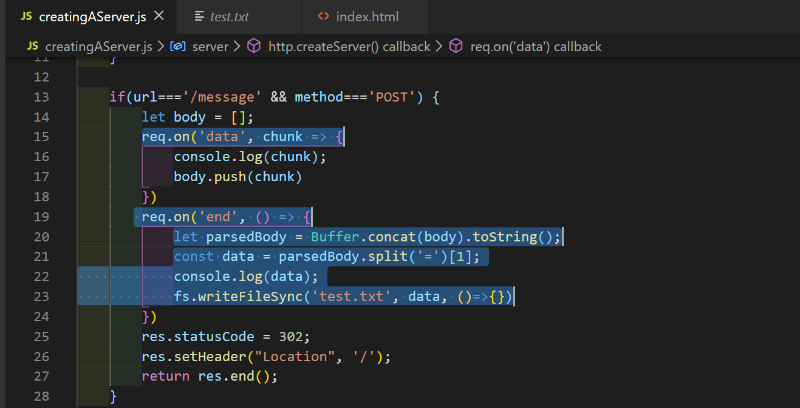
- Routing Requests: Xác định URL được gửi lên trong request, với mỗi URL khác nhau, sẽ trả về các file html, css, js hoặc các tác vụ khác nhau từ server.

1. 

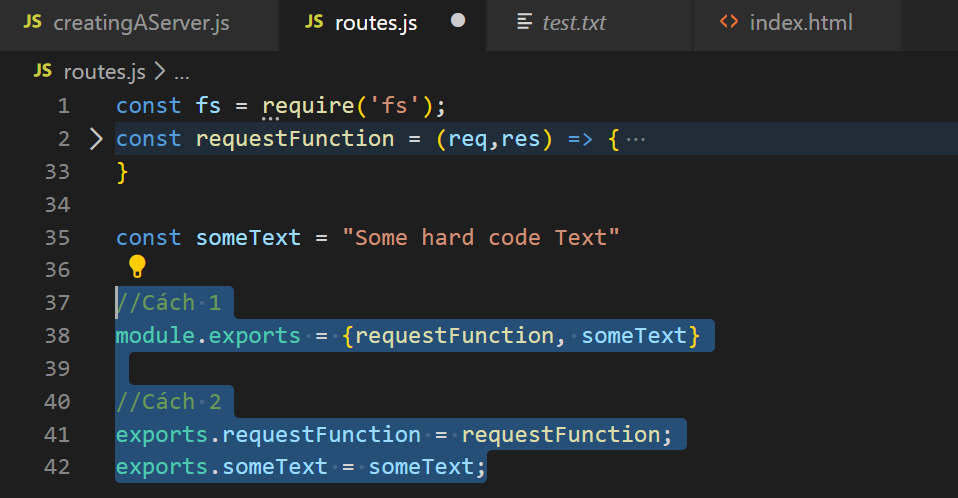
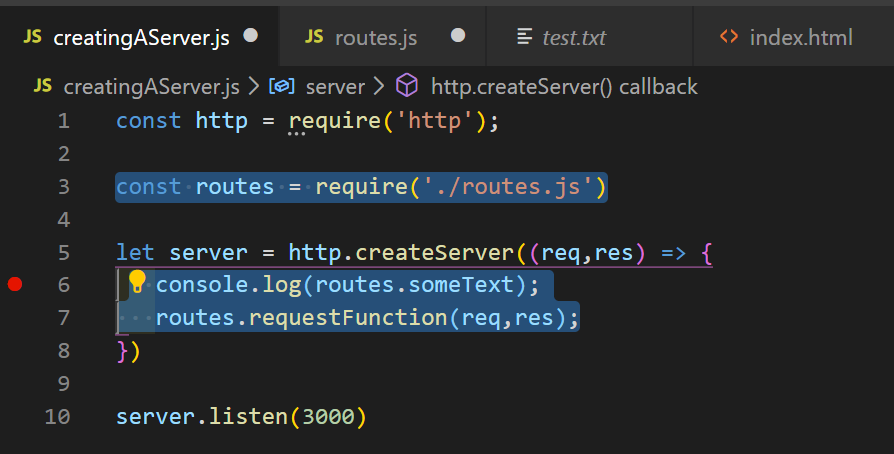
- Redirecting Requests : Xác định URL, rồi ghi file trên server, rồi chuyển trang voiws **statusCode = 302**:

1. 

-Parsing Request Bodies: Parse data được gửi từ client về server, sau đó ghi data ra file test.txt.

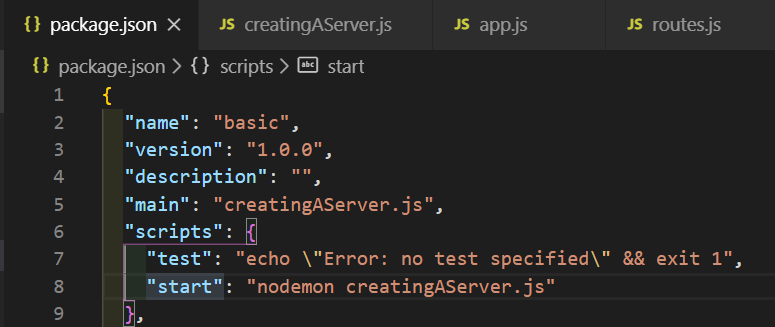
1. 
   1. Hàm on(): dùng để lắng nghe các sự kiện, listenerEvent.

-Using the Node Modules System: hướng dẫn cách export cũng như import các hàm, biến giữa các file js với nhau:

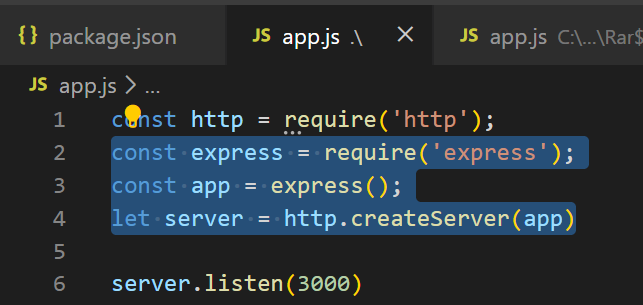
1. Export hàm requestFunction() và biến someText ở file Routes.js:gồm 2 cách:
2. Import hàm requestFunction() và biến someText :

-Understanding NPM Scripts: tạo file package.json với **‘npm init’**.

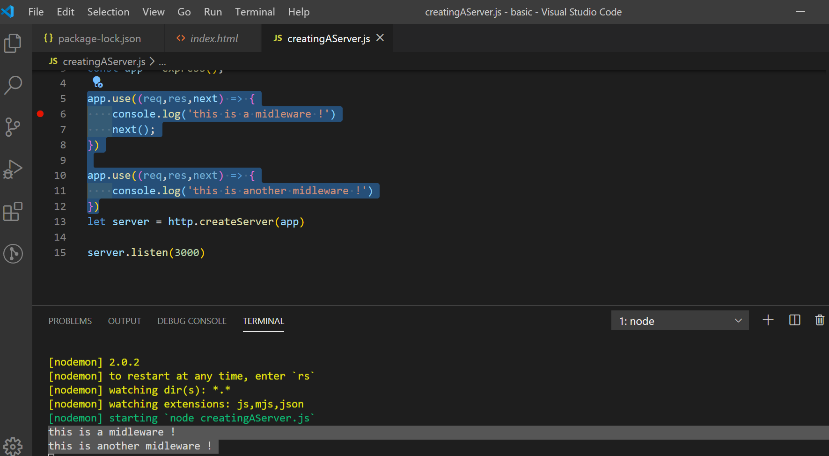
-Installing 3rd Party Packages: cài đặt gói “nodemon” package, *có chức năng tự động restart lại server sau mỗi lần thay đổi code*:

1. 
2. -Thay thế ‘node’ thành ‘nodemon’ ở line 8.

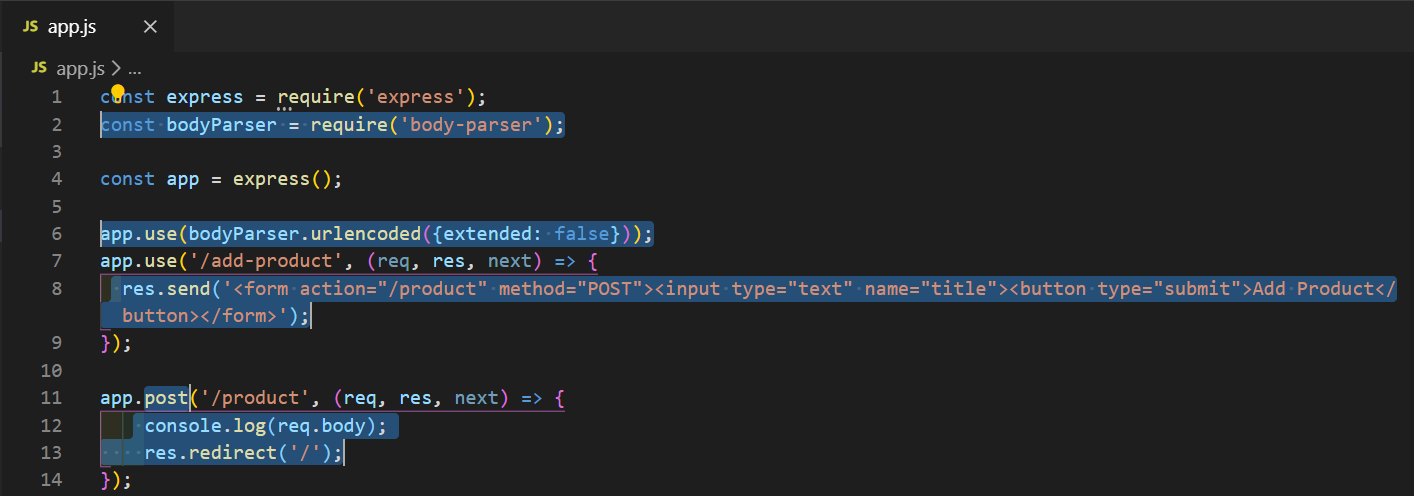
-Installing Express.js:

1. 
2. 

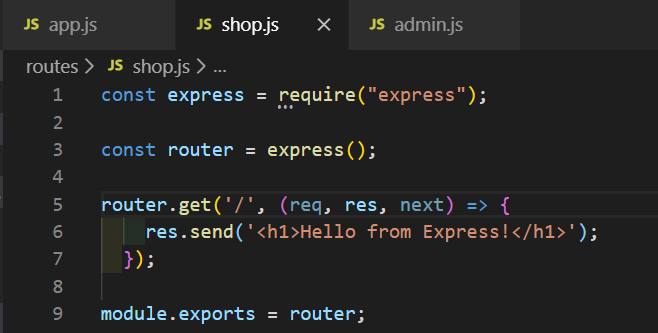
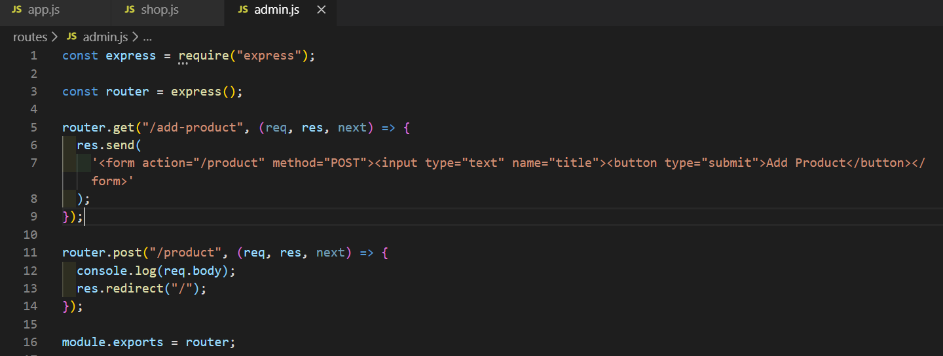
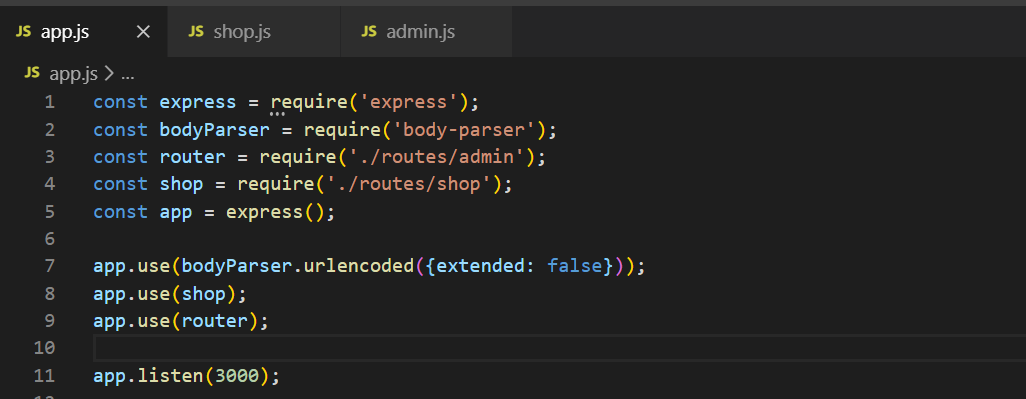
-Adding Middleware:

1. 
2. Có hàm Next() thì nó mới thực hiện middleware tiếp theo.

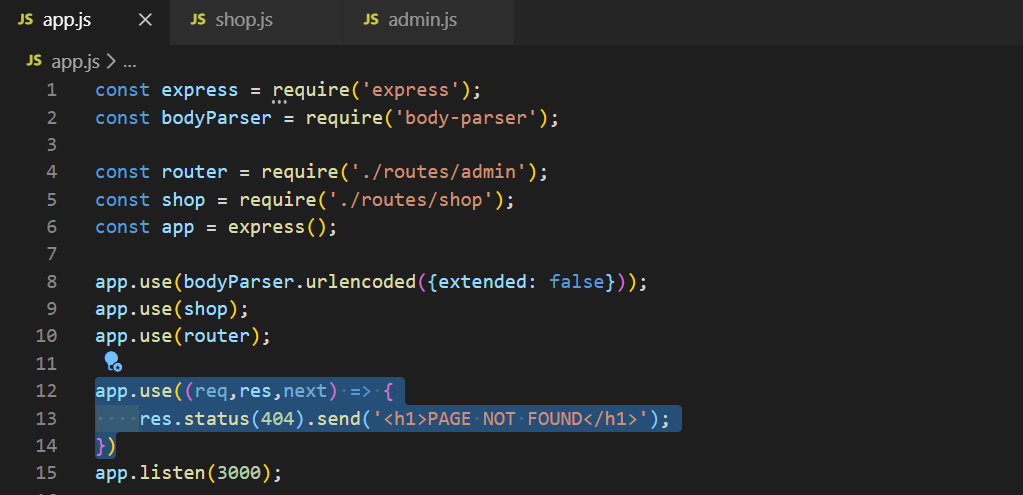
-Handling Different Routes và Limiting Middleware Execution to POST Requests: Với các routes, đường dẫn khác nhau, thì sẽ trả về dữ liệu khác nhau với res.send():

1. 
   1. Cài đặt npm install bodyParser để parse data từ req ra thành object cho nodejs xử lý, nếu không có thì req.body sẽ là undefined.
   2. Khi user truy cập vào url “/add-product”, thì sẽ gửi 1 request lên server, server sẽ trả về 1 response với thẻ html <Form> như line 8.
   3. Khi user truy cập vào url ”/product”, res sẽ redirect sang trang có url là “/” như line 13.
   4. Line 11: sẽ có các methods trong http như Post, Get, Put, …. Mỗi khi User truy cập vào url ‘/product’, nếu method của request là “POST” thì nó sẽ thực thì hàm callBack, không thì thui. Còn dùng Use, cứ truy cập đúng URL là nó sẽ thực hiện.

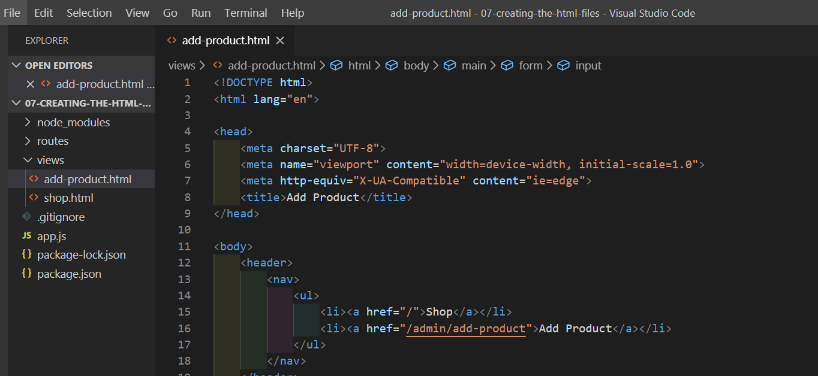
-Using Express Router: Tách mỗi đường dẫn ra thành các file khác nhau để dễ quản lý, mỗi file chỉ phụ trách 1 url, sau đó import lại trong App.js:

1. 
2. 
3. 

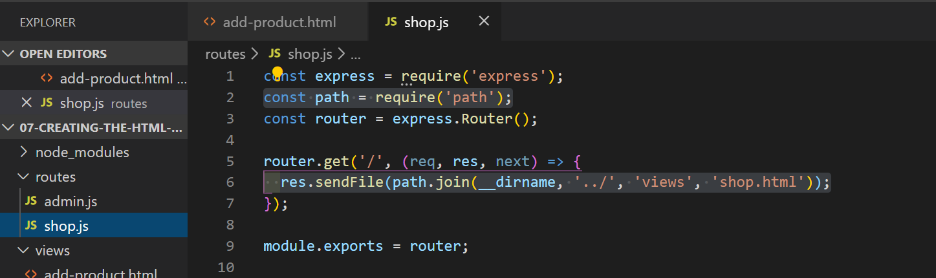
- Adding a 404 Error Page:Hiển thị trang 404 khi không đúng đường dẫn URL:

1. 
   1. Phải đặt ở cuối.

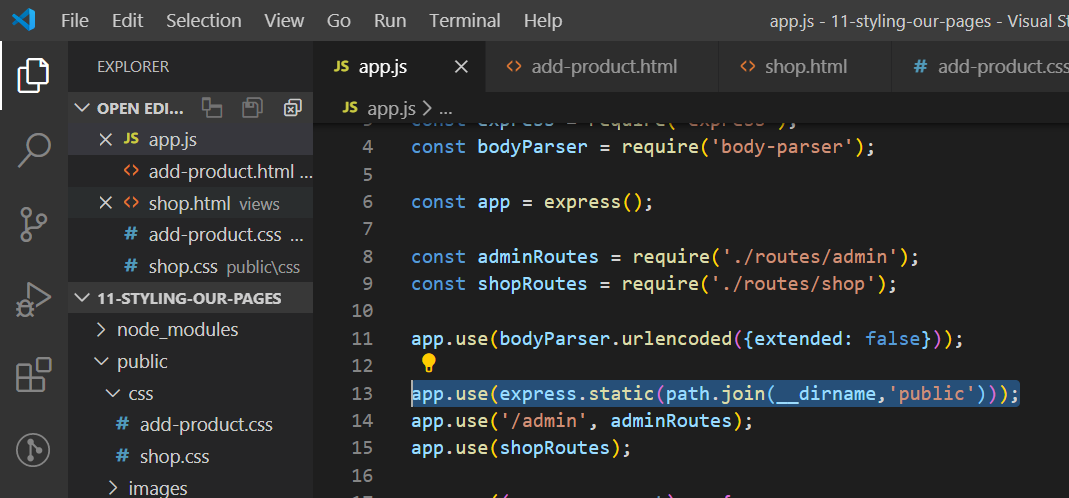
- Creating HTML Pages: Tạo folder View (theo mô hình MVC) để chứa các file HTML, tạo 2 file shop, add-product.html:

1. 

- Serving HTML Pages: Trả về 1 file html shop.html khi user gửi request url “/” lên server:

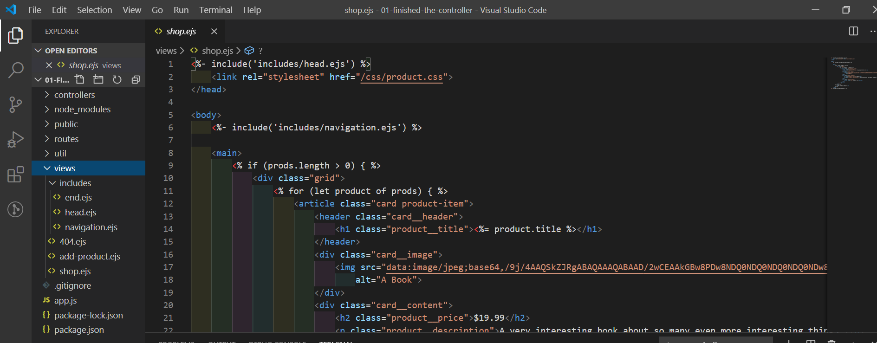
1. 
   1. Dùng core module để truy xuất đến thư mục, file còn tìm chính xác.
   2. “\_\_dirname” chính là trỏ tới file hiện tại của nó (đang nằm trong folder Routes).
   3. Senfile() là dùng để trả về 1 file HTML.

- Serving Files Statically: Đối với các file tĩnh như css, image, nên tạo folder public để chứa.Chúng ta dùng express.static(*url)* để download các file tĩnh ấy về cho user:

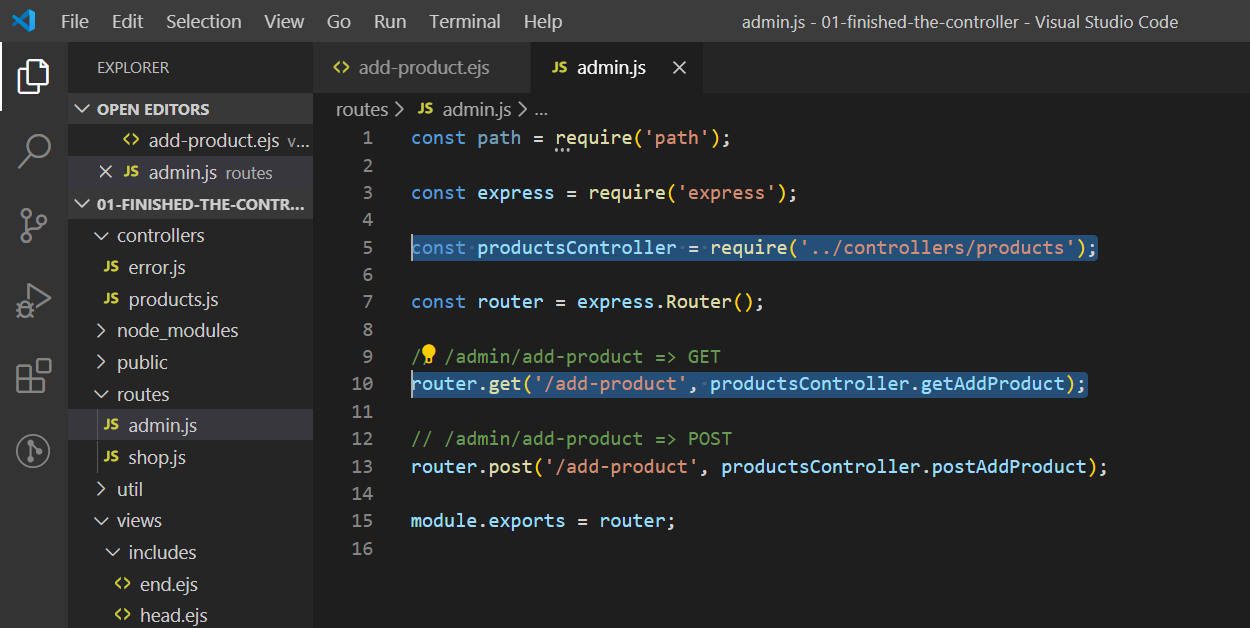
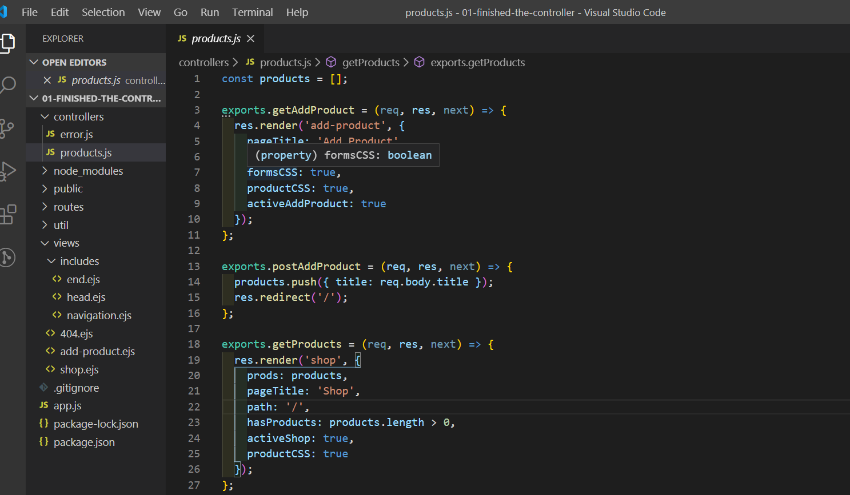
1. 

**MVC:**

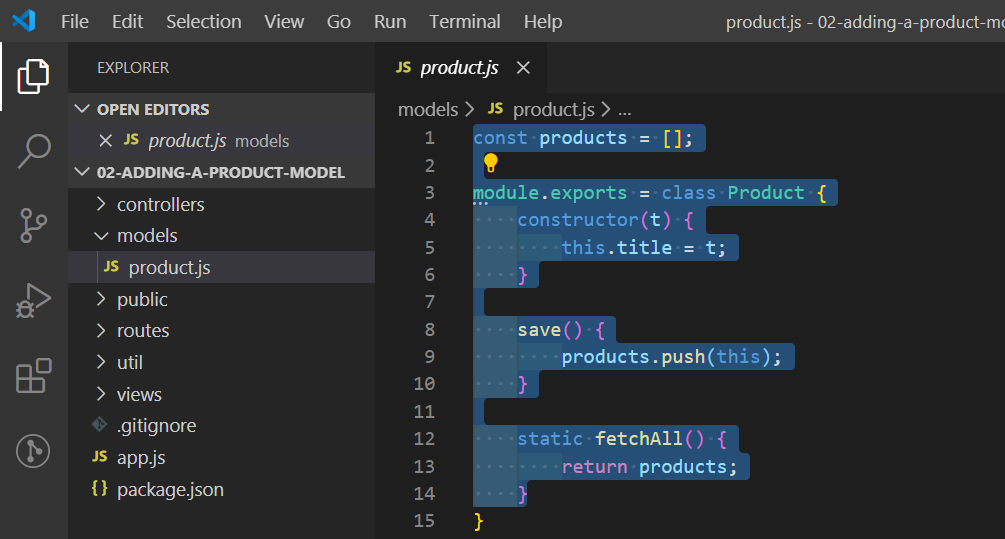
**-**Tạo folder views để chứa các file html, hoặc .ejs, hoặc .pug, giống .twig bên PHP:

1. 

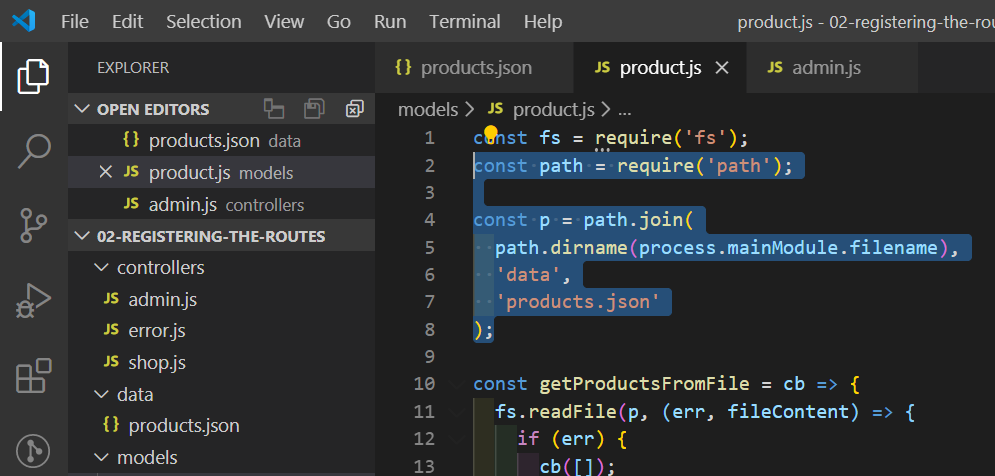
**-**Adding Controllers: Controller là điểm nối giữa views và models, response trả về users như thế nào sẽ do Controller xử lý:

1. 
   1. Tạo productsController để xử lý các data liên quan đến product trước khi trả về view.

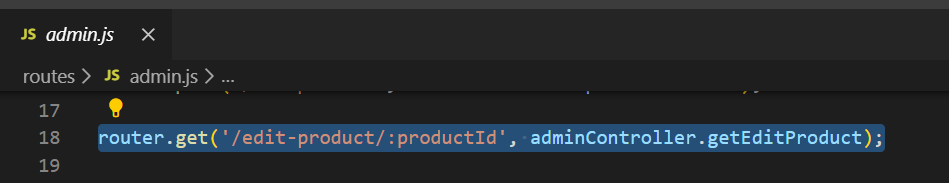
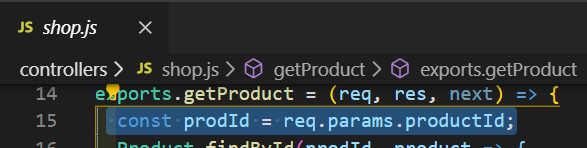
- Adding a Product Model: là tạo khai báo đối tượng, thuộc tính, hàm như bên FE.

1. 
   1. Tạo Model product với thuộc tính ‘title’, phương thức ‘save()’, ‘fecthAll()’.

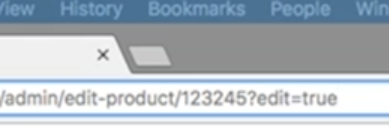
-Get project file path: truy cập đến folder chứa project, folder chạy file index.js:

1. 
   1. Truy xuất vào folder “02-REGISTERING-THE-ROUTES” với line 5.

-Add Params ProductId into url and Extracting Dynamic Params: Truyền biến vào URL và lấy biến ấy về để xử lý dữ liệu:

1. 
2. 

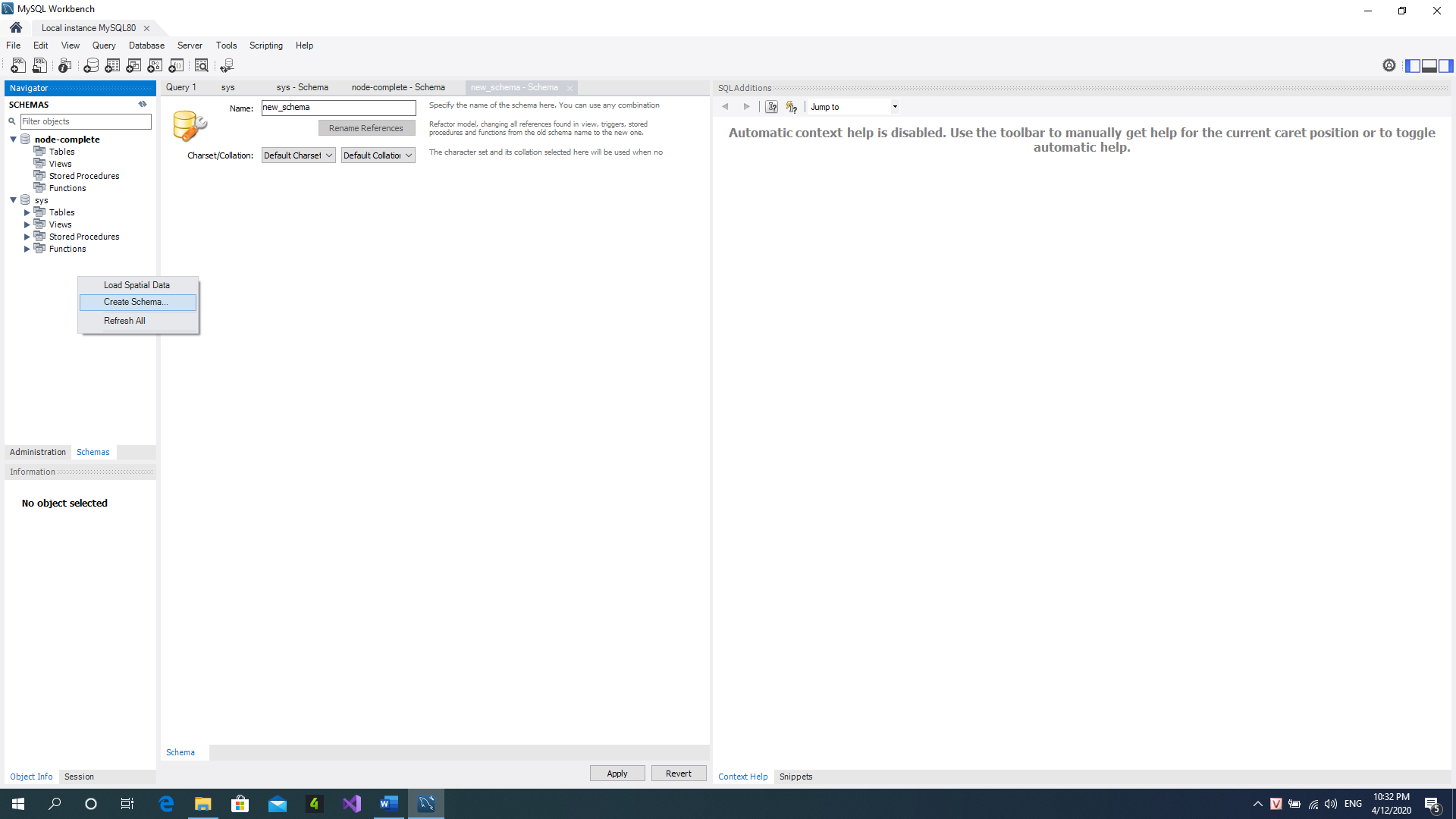
-Add Query Params into URL: them dấu “?” vào cuối URL và 1 hoặc nhiều cặp key value:

1. 
2. 
3. 
   1. Cách truy xuất biến query params, luôn trả về kiểu string.

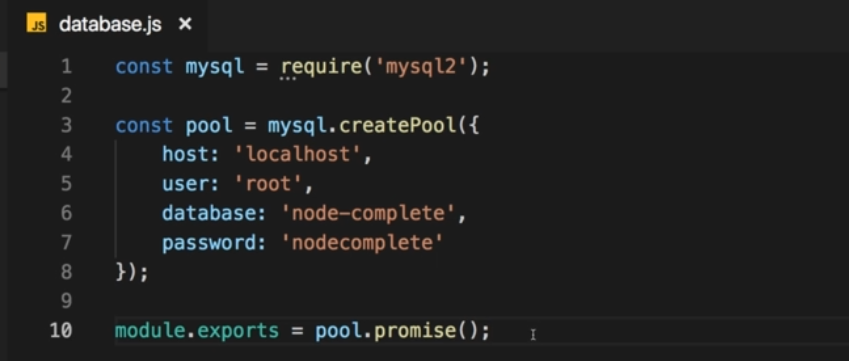
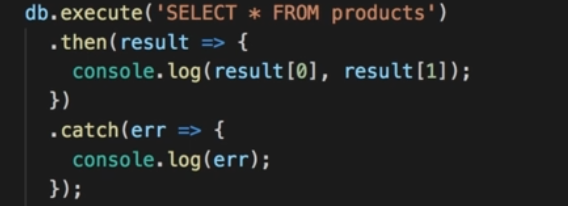
-TÓM TẮT MVC và CẤU TRÚC ĐƠN GIẢN, FLOW ĐƠN GIẢN CỦA 1 PROJECT NODEJS, EXPRESS:

1. MVC:
   1. Thư mục Views:
      1. **Khái niệm**: là những teamplate giống twig của PHP mà server trả về cho FE, có các biến động trong template ấy.
      2. **Nhiệm vụ:** dùng để trả về view cho Client để hiện thị cho user.
   2. Thư mục Models:
      1. **Khái niệm:** lập trình theo hướng đối tượng OOP giống JAVA đã học, mỗi 1 model là 1 đối tượng với các chức năng như saveData(), deleteData(), addData(), updateData() …..
      2. **Nhiệm vụ:** định nghĩa các phương thức, hành vi để xử lý đối tượng ấy vào database, vào files, vào Json …
   3. Thư mục Controller:
      1. **Khái niệm:** là điểm giữa của Models và Views.
      2. **Nhiệm vụ:** lấy dữ liệu từ Request gửi lên server, xử lý các dữ liệu ấy. Sau đó gọi các hàm trong model để xử lý các dữ liệu ấy với database như lưu vào database, updateDatabase… Sau đó lấy các dữ liệu tự database lên để trả về cho Views để hiển thị ra cho người dùng.

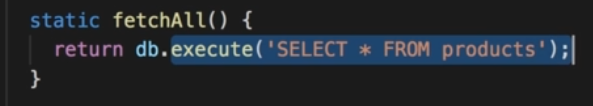
-SQL Introduction:

1. Sử dụng MySQL: <https://www.mysql.com/>
2. Connecting SQL to our App: <https://www.npmjs.com/package/mssql#asyncawait>
3. Cài đặt bình thường như hướng dẫn trên mạng.
4. Tạo 1 database (schema):
   1. 

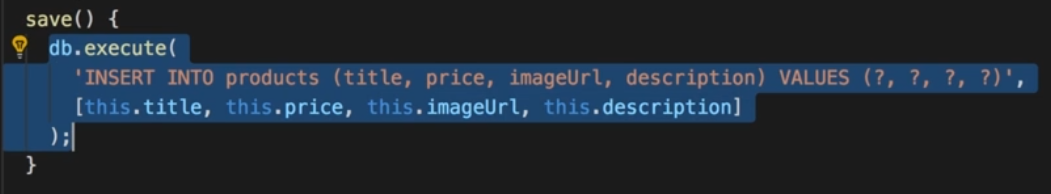
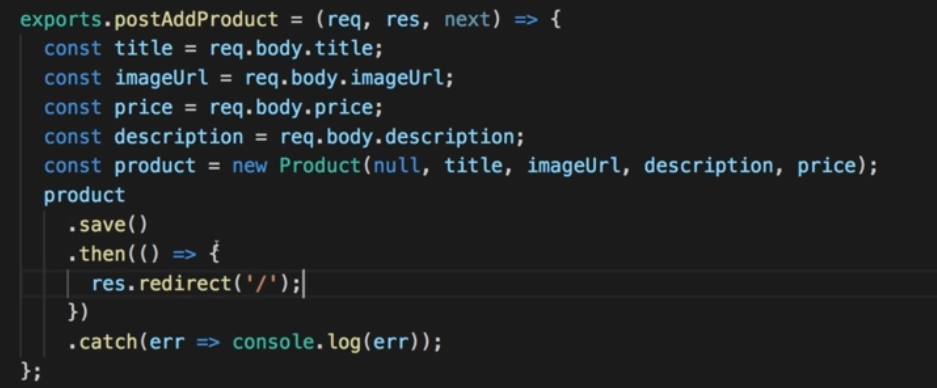
-Connecting App to MySQL Database: kết nối database với chương trình

1. Cài đặt: .
2. Tạo 1 file *database.js* trong folder Util để tạo đối tượng kết nối database, kết nối database.
3. Tham khảo thêm ở documents “*mysql2*”: <https://www.npmjs.com/package/mysql2>.
   1. Nên sử dụng Promise() và CreatePool().
4. File ‘*database.js*’:
   1.  => Đã tạo đối tượng Connection, giờ khi nào cần thực thi thì import module này vào.
5. Khi cần thực thi ở 1 file bất kì:
   1. 
   2. 
   3. => Lấy dữ liệu về rồi console.log ra.

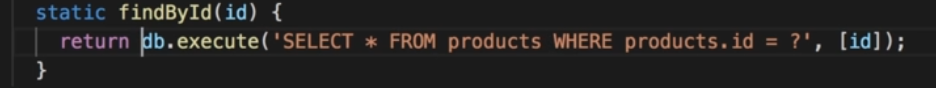
-Fetching Products from Database, display to Views:

1. Trong file Models, thực thi câu lệnh query, lấy các sản phẩm Products từ database sau đó trả về cho client
   1. => thao tác với database ở lớp Model trong MVC.
   2.  => xử lý logic ở lớp Controller khi nhận về database, rồi trả dữ liệu về cho Client. Trả về “rows” và “fieldData”.

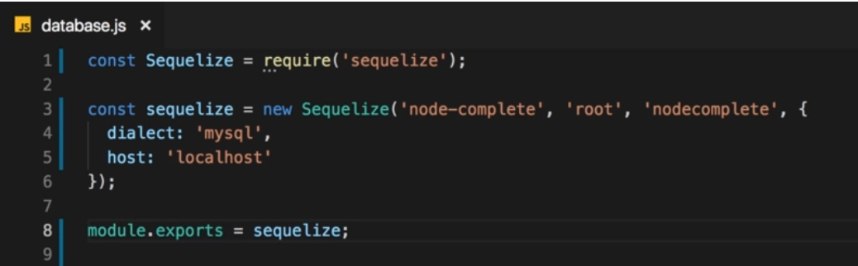
- Inserting Data Into the Database:

1. Trong file Models, thực thi câu lệnh query, lưu product xuống database
   1.  => xử lý ở Product Model.
   2.  => xử lý ở Controller.

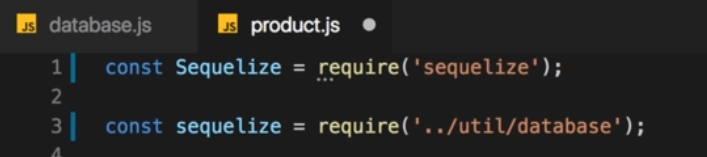
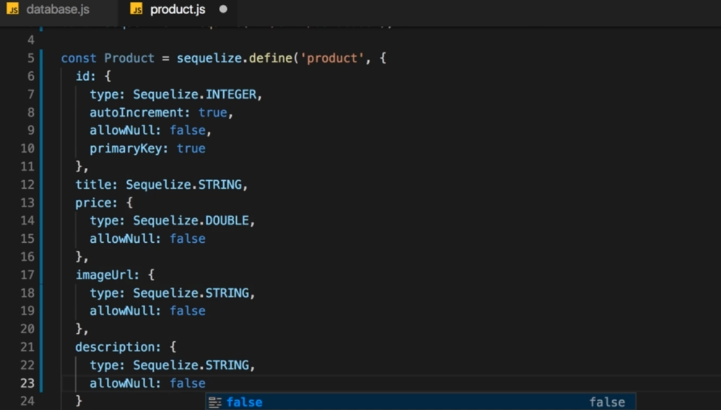
- Fetching a Single Product with the where Condition:

1. Lấy 1 sản phẩm bất kì dựa trên Id:
   1.  => thao tác ở model.
   2.  => Xử lý ở controller.

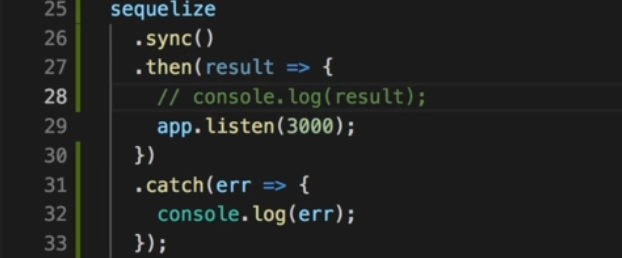
-Connecting Sequelize to the Database:

1. Kết nối Sequelize với database:
   1.  => Yêu cầu phải cài đặt package ‘npm install –save mysql2’ trước.
   2.  => Khởi tạo trong database.js

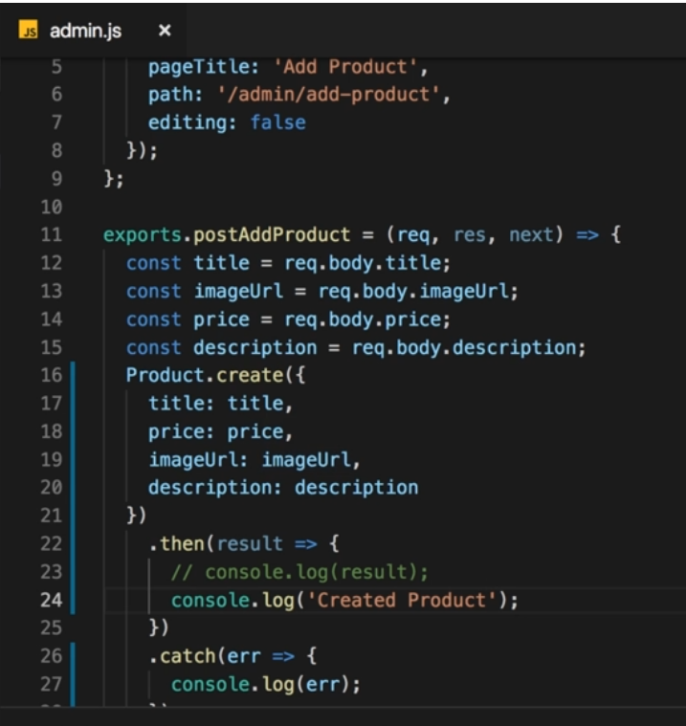
- Defining a Model: Tạo 1 model có kiểu dữ liệu giống 1 bản trong Database:

1. Tạo model Product.js:
   1.  => Line 1 là 1 Class, function. Line 3 là 1 object được trả về như phía trên.
   2.  => Tạo 1 Model ‘Product’ có kiểu dữ liệu đơn giản giống như Table ‘Product’ trong Database.
   3.  => Export Product Model.

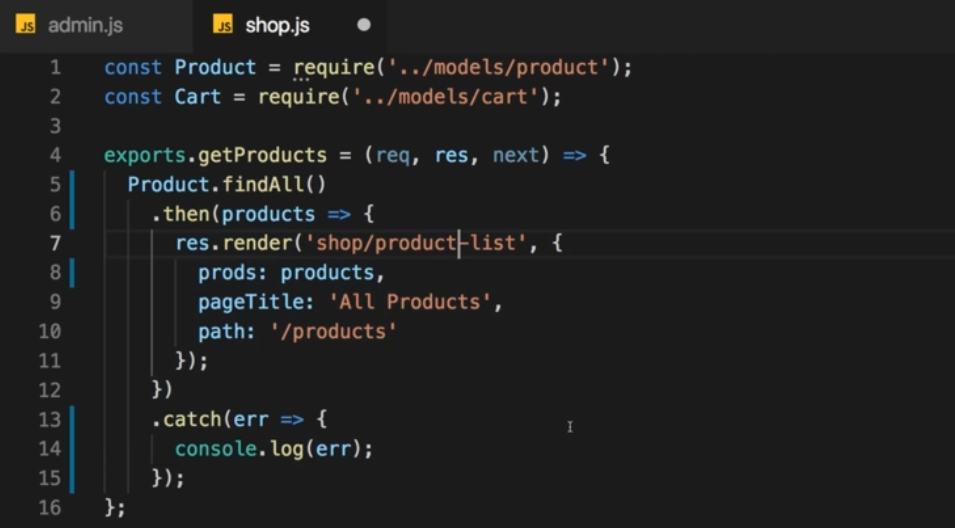
-Syncing JS Definitions to the Database: Đồng bộ hóa database với sequelize trong project:

1. Ở App.js
   1.  => import
   2.  => Đồng bộ hóa với database, sau đó mới chạy app.

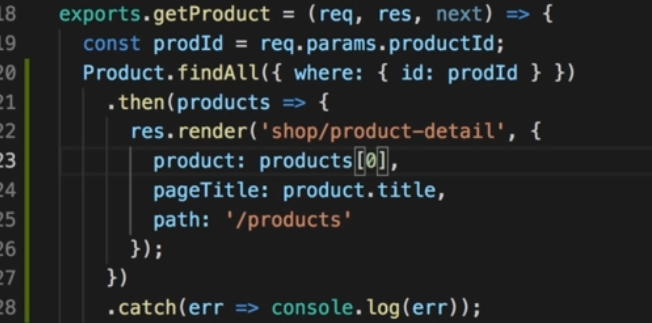
-Inserting Data & Creating a Product: Tạo 1 product và insert Product ấy vào database từ controller:

1.  => line 16 dùng create() để tạo 1 function và import vào database. Nó gọi thẳng thằng Create() trong controller luôn, ko cần thông qua thằng model.

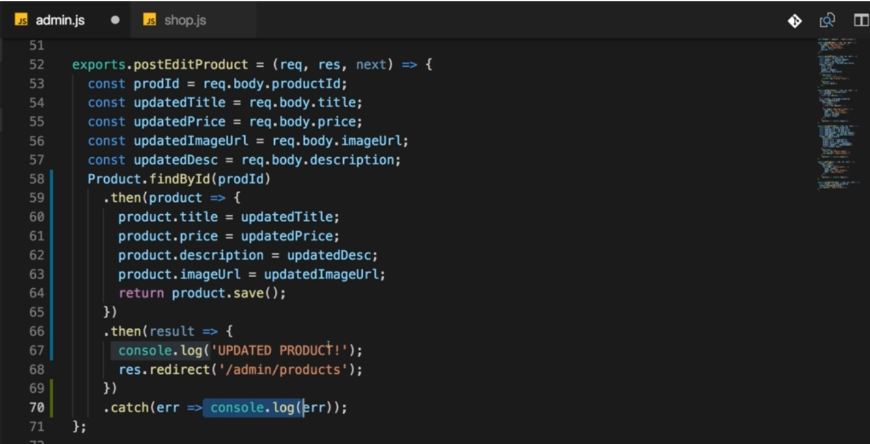
-Retrieving Data & Finding Products: Nhận dữ liệu từ bảng table:

1.  => hàm findAll() sẽ tự động lấy các cột từ bảng Product.

-Getting a Single Product with the where Condition: Lấy ra 1 sản phẩm Product với mệnh đề WHERE:

1.  => Dùng findAll() để lấy tất cả, rồi dùng WHERE để lấy rows cần tìm.

-Updating Products: Update giá trị mới cho product trong bảng product

1.  => Tìm kiếm phần tử rồi save() lại.

- Deleting Products: Xóa phần tử trong bảng Product

1.  => Dùng hàm destroy() để delete phần tử ấy trong bảng.

- Creating a User Model:

1. 