目录

[1. SQL 1](#_Toc15201)

[2. Node & Express without the views 4](#_Toc9536)

[2.1 init the project 4](#_Toc28517)

[2.2 step 1&2:setting up express and adding middleware 5](#_Toc19165)

[2.3 config debugger to test if the program works 6](#_Toc22315)

[2.4 step 3: add data models 6](#_Toc12494)

[2.5 Step 4: Add router and controller 9](#_Toc22578)

[3 Use postman to test signup and sign in 10](#_Toc30026)

[1 Sign up 10](#_Toc5028)

[2 Verify using JWT sign in 12](#_Toc2405)

[4 Add views to node 13](#_Toc24743)

[4.1 Integrating html 13](#_Toc28800)

[4.2 link views to controller 16](#_Toc652)

[4.3 test form data from the login page 18](#_Toc16468)

[4.4 add auth to all routes except login page 20](#_Toc16678)

[4.5 connect login frontpage with api 20](#_Toc3683)

[4.6 add fileupload function: routes and controller 22](#_Toc32003)

[4.7 Node upload to IPFS 22](#_Toc7627)

# SQL

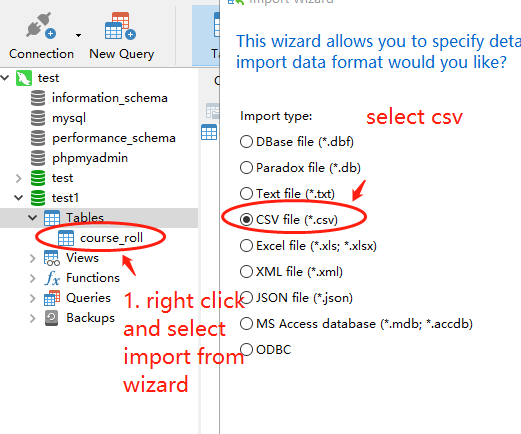
* 1. Prerequisite:

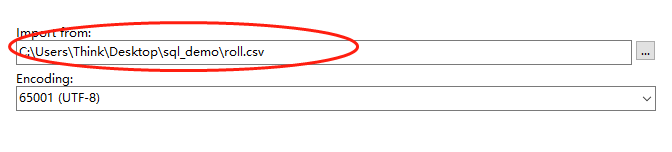
install xampp: <https://www.apachefriends.org/index.html>

Install Navicat free trail: <https://www.navicat.com/>

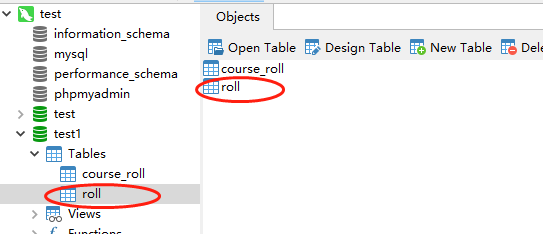
Data sets: <https://github.com/onebit256/sql_demo>

* 1. import data sets



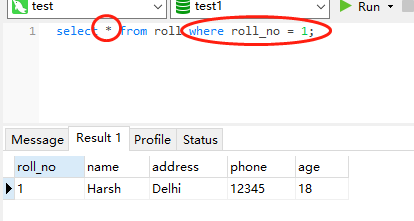


Select the right path, and keep clicking next til the end, then you will see the imported data base



* 1. insert, select, delete, joins

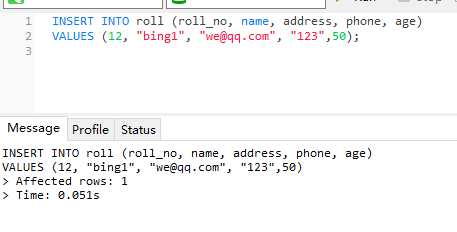
Select

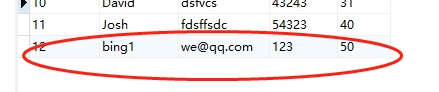


- \* means display all fields in the selected table

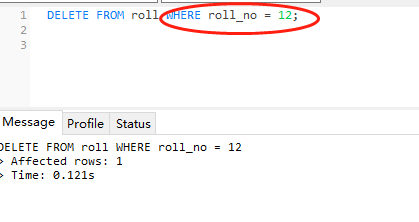
- “Where” is the condition to filter the records

Insert

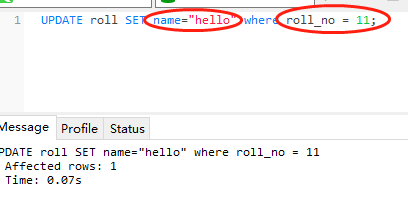


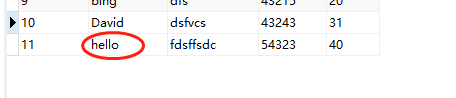


Delete

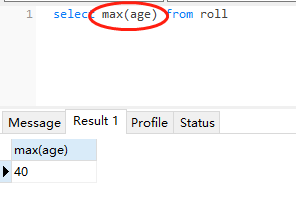


Update

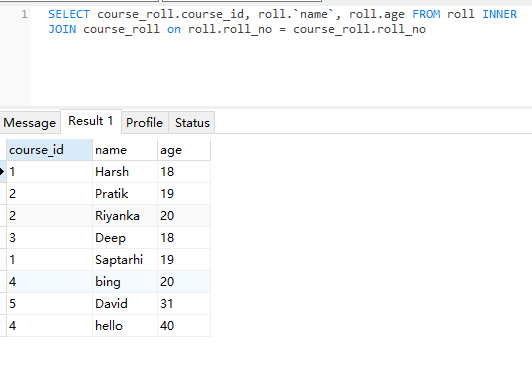




Max



Inner join



# Node & Express without the views

Use squelizer from project one: <https://github.com/onebit256/nodejs-express-sequelize-mysql>

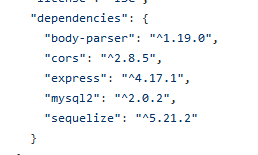
Use auth from prject 2: <https://github.com/onebit256/Node-js-JWT-auth---Medium>

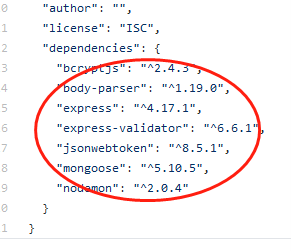
Final Code: <https://github.com/onebit256/batch2-lab4-node-without-views>

## init the project

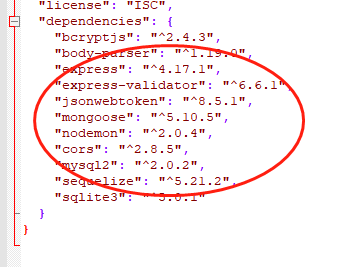
Npm init

Copy and post dependencies from project 1 and 2 to this new project



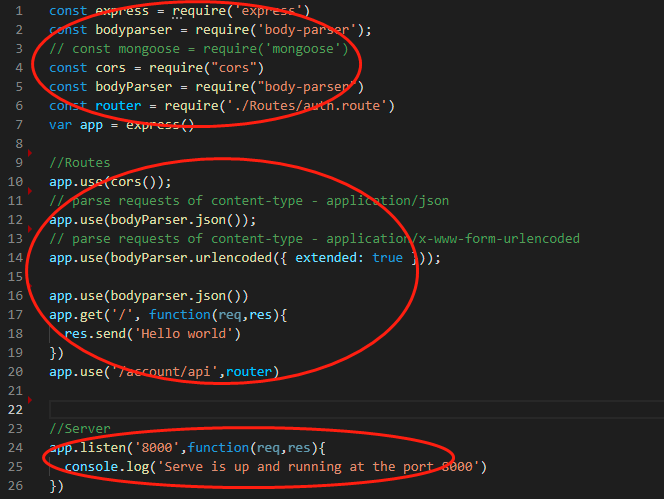


To the package.json in the new project folder



## step 1&2:setting up express and adding middleware

Copy index.js from project 2 to app.js

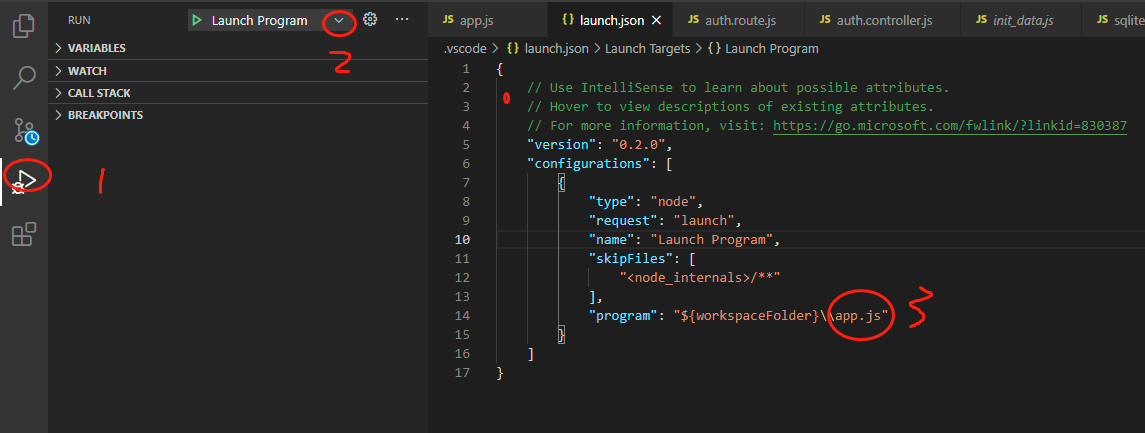


1 create express app

2 add middle ware

3 add port listener

## config debugger to test if the program works



Click 1

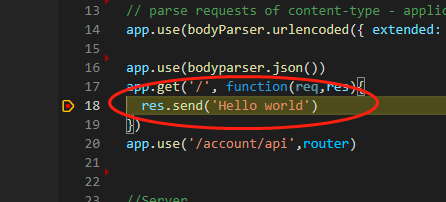
Click 2 select nodejs

Replace index.js with app.js

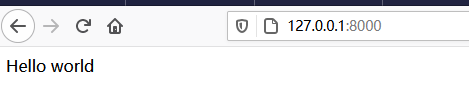
Set break point on app.js

When type 127.0.0.1:8000 on the browser

The break points stops



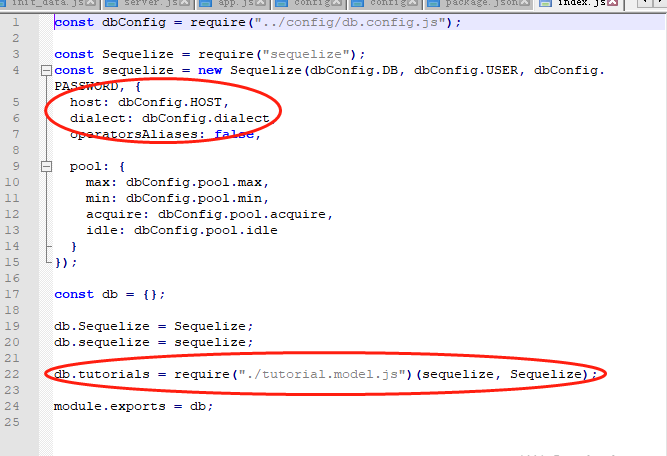
So the router works



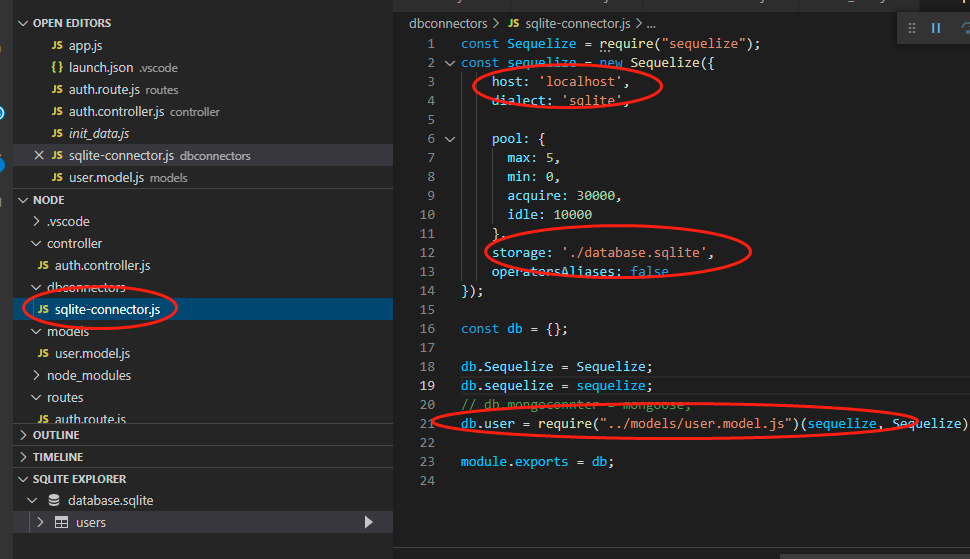
## step 3: add data models

Copy index.js under model folder from project 2 to this new project under dbconnector folder

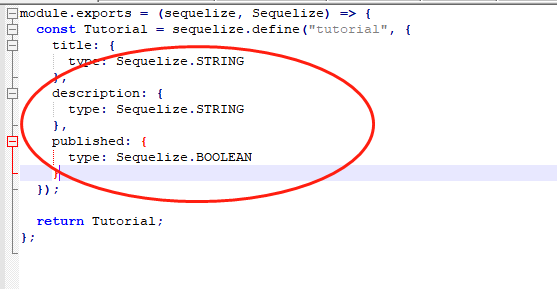
Replace mysql config to sqlite config



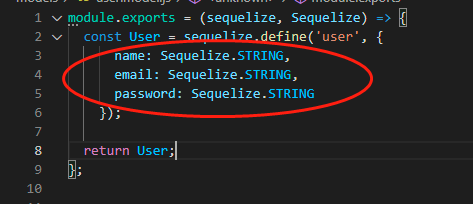
To



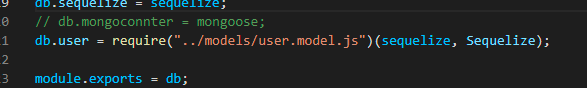
Then copy tutorial.model.js under model folder from project 1 to the new project under the model folder as well. And set user model



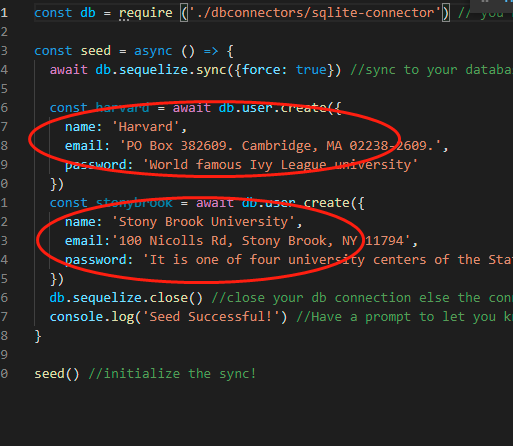
To



Then add user element to the db in dbconnetor

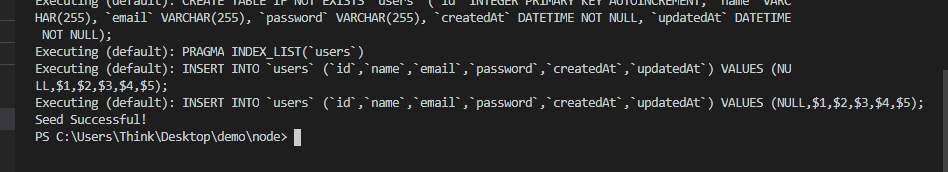


Create a init\_data.js to test if the dbconnetor works

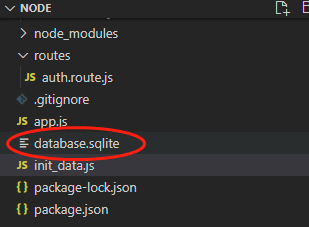


Node init\_data.js

Console prints

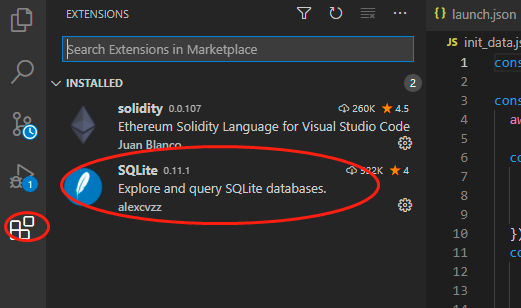


And



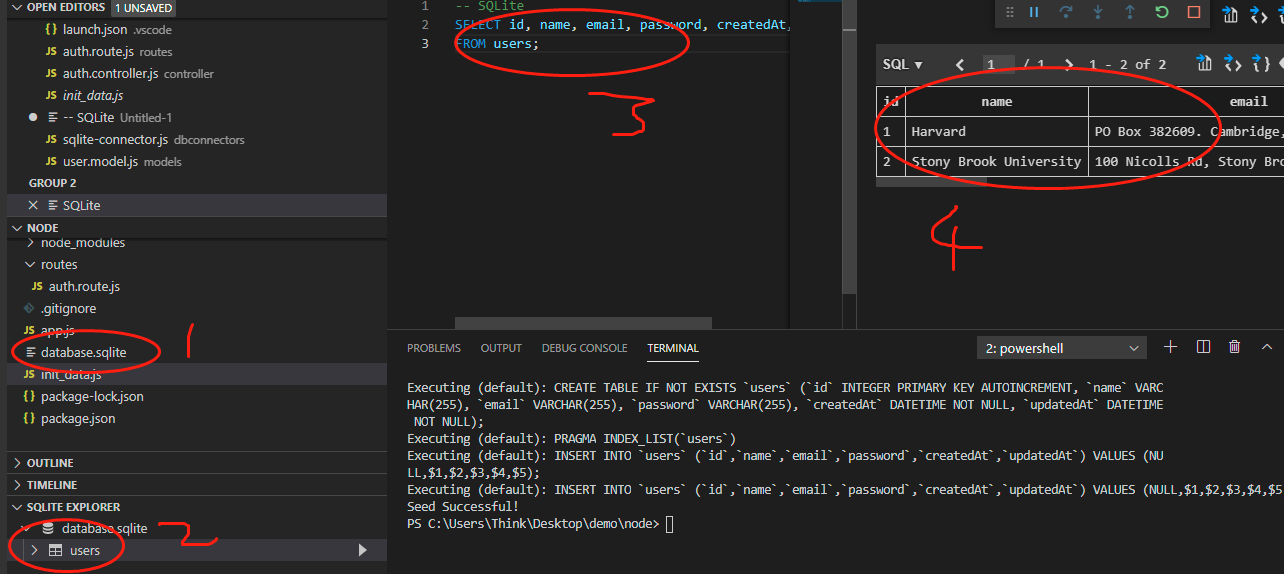
Database.sqlite is generated

Install Sqlite add-on: click squares button, and install SQLite



After install Sqlite add-on

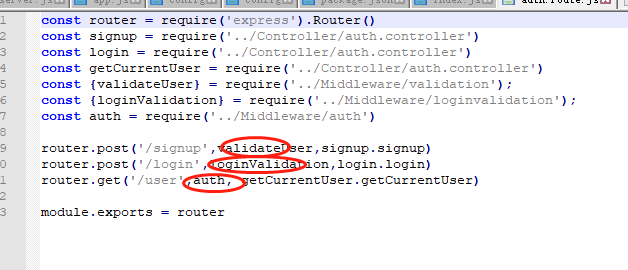
Right click database.sqlite, then open database, right click on users, and select new query [select]



Now as we can see the data has been written to the database

## Step 4: Add router and controller

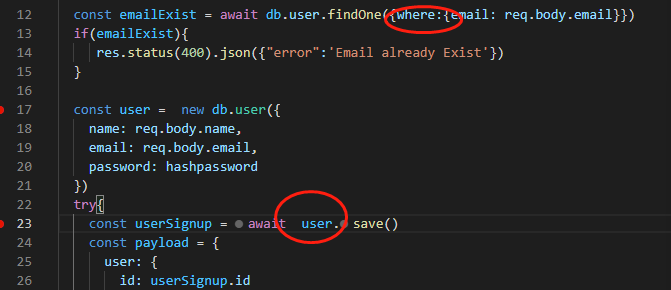
Create router folder and copy the router auth.router.js from project2 to this project under the router folder



Remove these customized middleware

create empty controller

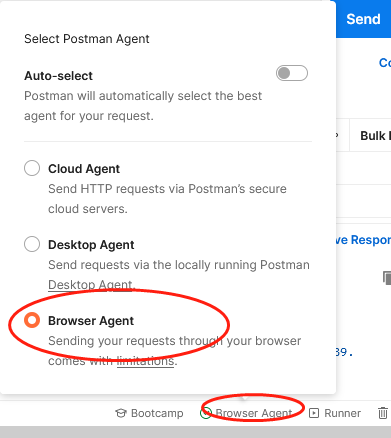
Copy auth.controller.js from project 2 to here



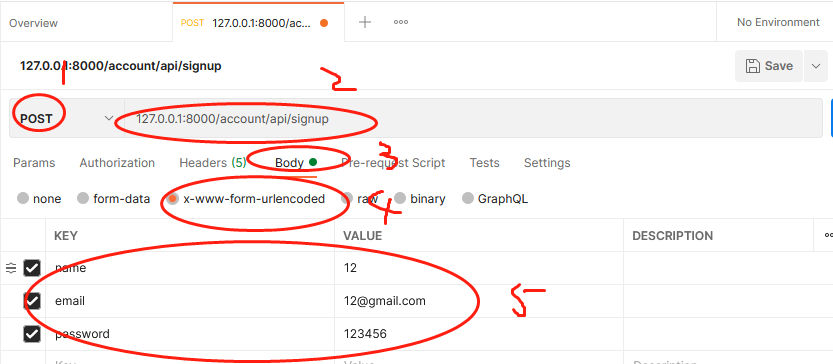
# 3 Use postman to test signup and sign in

## 1 Sign up

1 switch to Browser Agent otherwise the request would be sent from the postman cloud



2 set request params



- set to post request

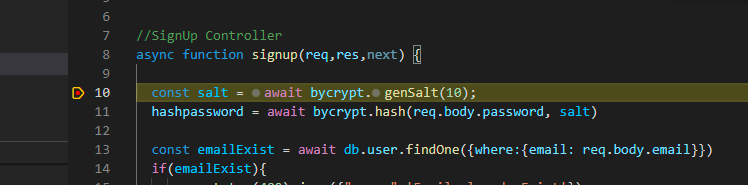
- type api url

- select body

- select x-www-form-urlencode which is form data format

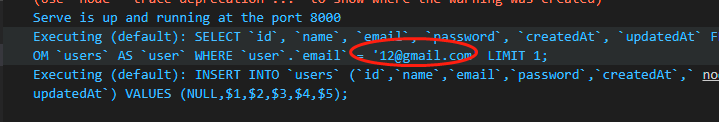
- add post data

Then put a break point on signup functionin the controller, and click send

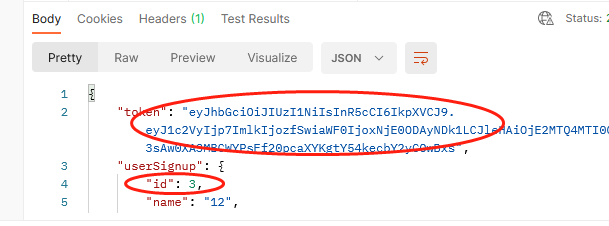


The break point stops, which means router is working, then continue

Console prints

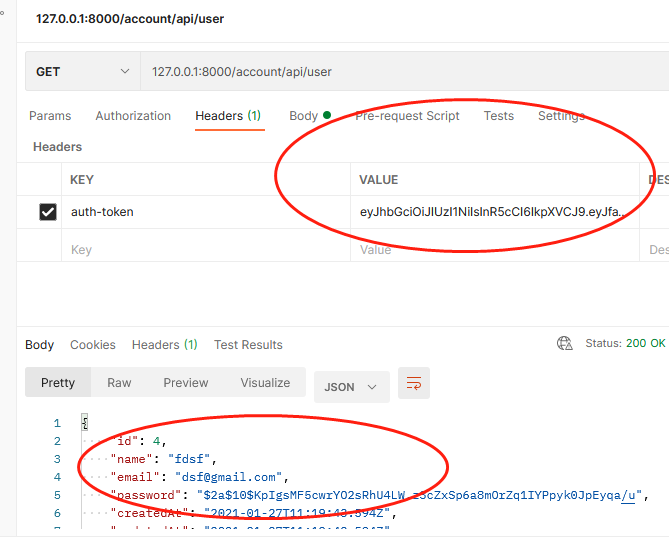


Postman returns JWT and user info



## 2 Verify using JWT sign in

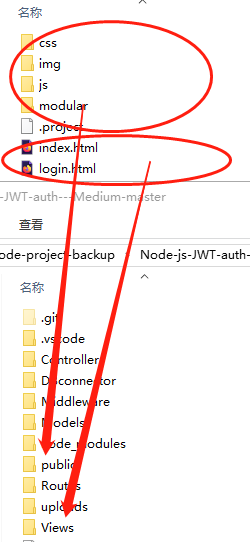
Add this token to get header



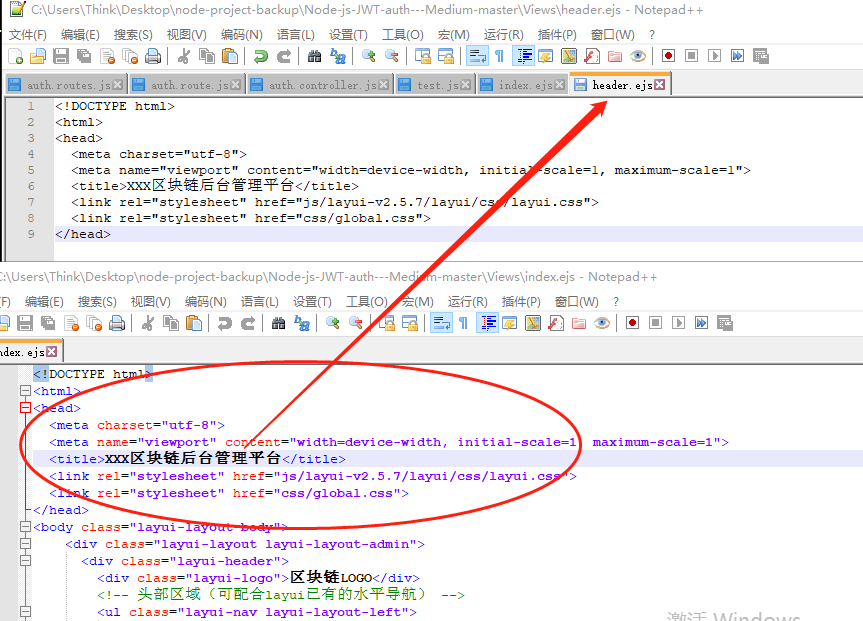
# 4 Add views to node

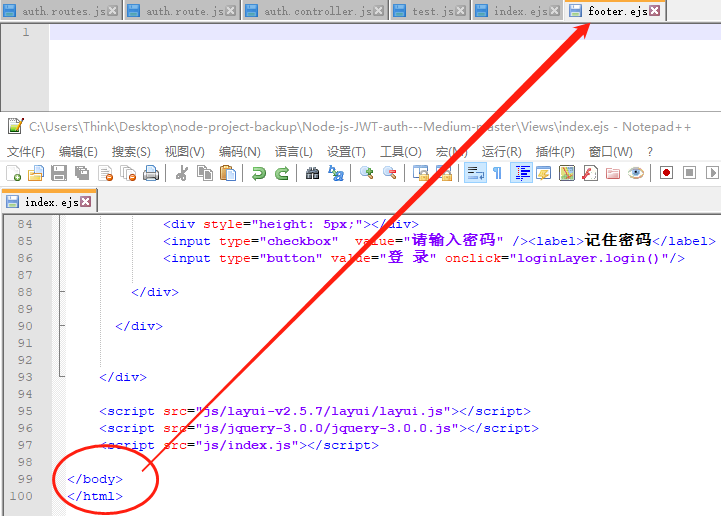
## 4.1 Integrating html

1. Move css js to public folder, and change html to ejs and move them to the folder

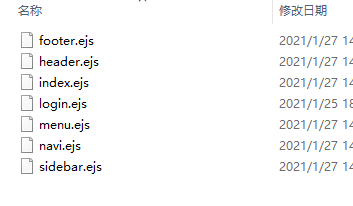


2 split html into parts

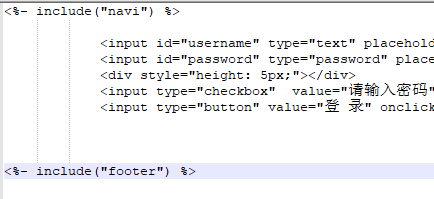




Eventually:



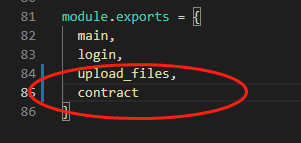
Index.ejs

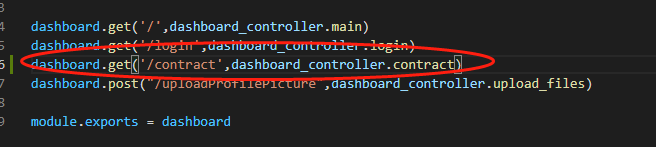


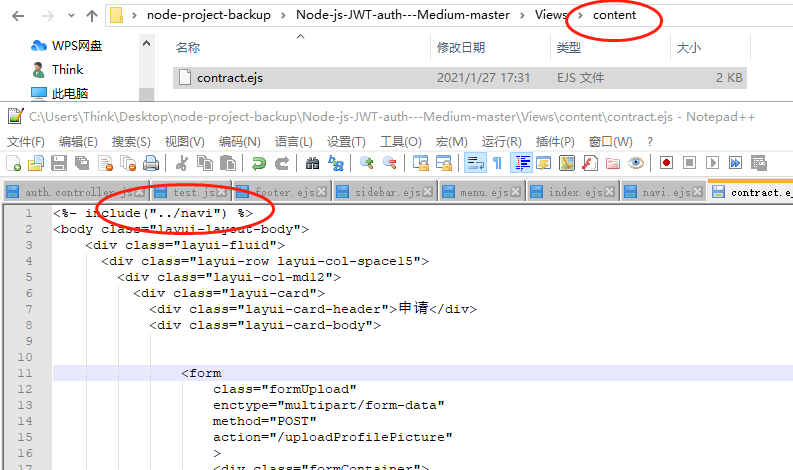
## 4.2 link views to controller

1. Add controller to apply\_for

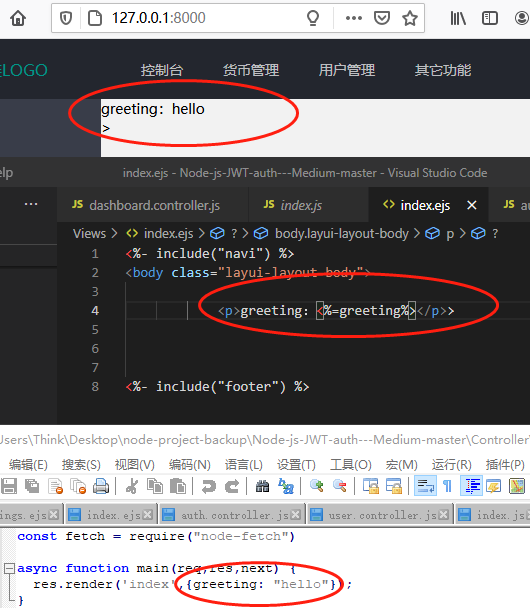






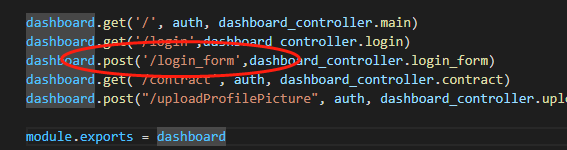


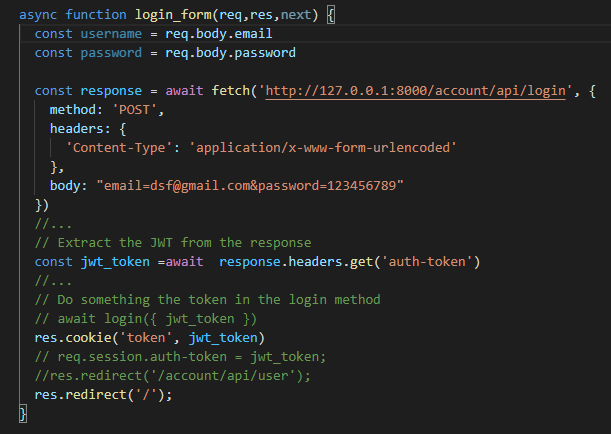
2 pass data from controller to the views



## 4.3 test form data from the login page

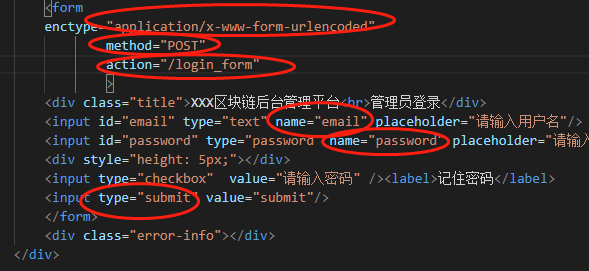
1 create a route to handle the form data, and create its corresponding controller



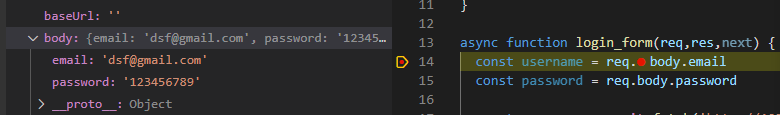


1. Create form

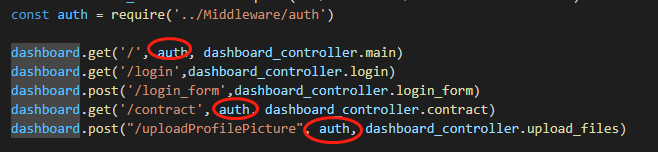
Set form data to www-form-urlencode, set names and set type to be submit, set action to be the route



1. Test

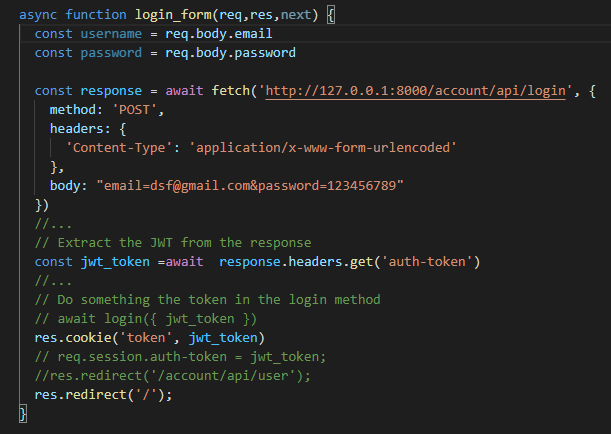


## 4.4 add auth to all routes except login page



## 4.5 connect login frontpage with api

1 add login logic



2 use cookie to store token



1. Test

It will return user info in json format on the browser

## 4.6 add fileupload function: routes and controller

## 4.7 Node upload to IPFS